NASA Conference Publication 3268

1994 Goddard Conference on Space Applications of Artificial Intelligence

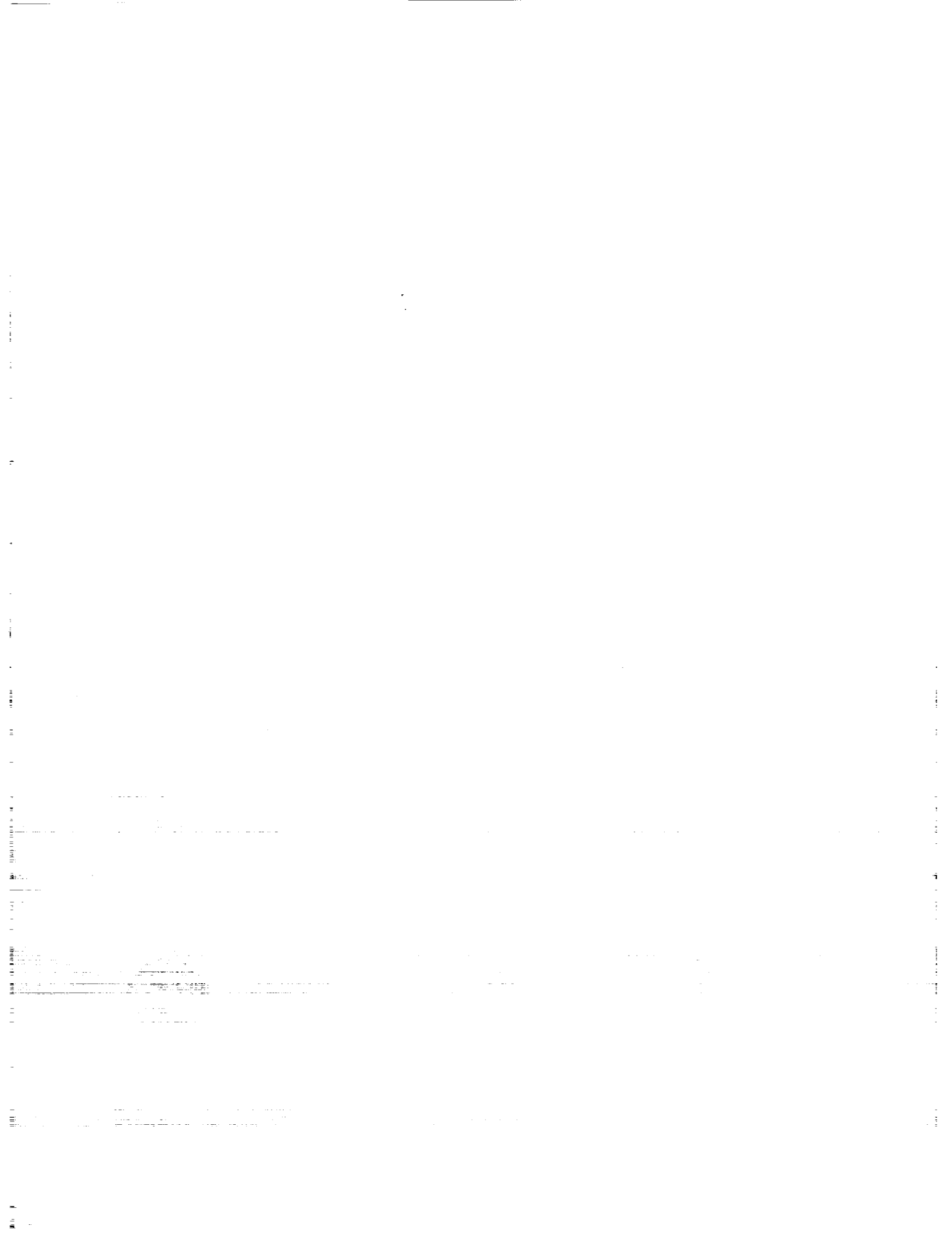
Edited by
Carl F. Hostetter
Goddard Space Flight Center
Greenbelt, Maryland

Proceedings of a conference held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 10 - 12, 1993

National Aeronautics and
Space Administration

**Scientific and Technical
Information Branch**

1993



1994 Goddard Conference on
Space Applications of
Artificial Intelligence

Conference Committee

Nicholas Short, Jr., GSFC (Chairman)
David Beyer, AlliedSignal Technical Services Corp.
Robert Cromp, GSFC
Carl Hostetter, GSFC
Peter Hughes, GSFC
Nikki Johnson, GSFC
J. Michael Moore, GSFC
James Rash, GSFC

Foreword

The ninth annual Goddard Conference on Space Applications of Artificial Intelligence (AI) is sponsored by the Mission Operations and Data Systems Directorate (Code 500) in cooperation with the Space Data and Computing Division (Code 930). The mission of this conference is to provide an opportunity for researchers and practitioners of AI in the aerospace industry to share the results of their applied work.

During the 1980s, AI moved out of the theoretical laboratory and into the applications arena. In NASA, challenging project requirements and the need to cut operations costs under flat budgets are fueling the demand for automated technology that resulted from applied research in the '80s. In the late '90s, for example, NASA's Mission to Planet Earth program will manage numerous satellites and process over one terabyte of data per day, requiring automation of mission operations as well as of data processing.

This year's conference reflects the greatest challenges to NASA and, in particular, Goddard's role in the Mission to Planet Earth. Several of the papers cover topics such as automatic interpretation, management, and processing of satellite data as well as automation of mission control centers. More than ever, NASA requires help from academia, industry, and other government agencies to achieve its upcoming missions. As in past Goddard AI conferences, I hope this conference, with its participation from various sectors, will make a contribution the achievement of these goals.

This conference would not have been possible without the help of many contributors. I would like to especially thank the authors for their outstanding technical papers. Second, I would like to thank the invited speakers for taking the time to prepare their talks and tutorials. Finally, I would like give special thanks to the Conference Planning Committee for their support during the numerous difficult times in planning the conference. I would also like to extend special thanks to last year's chair, Mike Moore, for all of his invaluable help with logistics.

Nick Short
Chair
1994 Goddard Conference on
Space Applications of Artificial Intelligence

Table of Contents

Information Management

Planning, Scheduling, and Robotics

Image Processing and Data Classification

Monitoring, Control, and Diagnosis

Knowledge Engineering

Information Management

N94-35044

Interactive Archives of Scientific Data

Lloyd A. Treinish
IBM Thomas J. Watson Research Center
Yorktown Heights, NY
lloydt@watson.ibm.com

Abstract

Data generators in many disciplines are rapidly improving, typically much faster than the techniques available to manage and use the data they produce. Appropriate data management techniques coupled with the process or methods of scientific visualization, or at least the technologies that support them show promise in helping to address some of these problems. Consider browsing, for example, in a role for feature identification by the scientist/user to serve as guide in the data selection process. To date, most efforts associated with data browsing have focused on simple images with image data. Unfortunately, these techniques are not applicable to many classes of data or when more than one data set is to be considered. By recalling that browsing is more of a subjective process involving the human visual system and that this is one of the origins of the notion of scientific visualization as a method of computing, then the utilization of visualization strategies for qualitative presentation of data becomes a viable approach. For browsing to be effective it must be interactive with near-real-time system response. With data sets of interesting size, e.g., ≥ O(1 GB), immediate interaction cannot take place on current conventional systems (i.e., high-end graphics workstations). Even though a 1 GB data set is admittedly modest by today's standards, the access and visualization of the entire data set or even a large fraction of it place significant burdens on the floating point and bandwidth capacities of the computer system being employed. This idea can also be extended to environments without high-bandwidth access to an interactive system by considering the distribution of compressed visualizations instead of data for predefined access and browsing scenarios.

Introduction

The instrumentation technology behind data generators in a myriad of disciplines, whether observational or computational, is rapidly improving, especially in the earth and space sciences. The capability to produce data is typically growing much faster than the techniques available to manage and use them. Traditional bulk access to data archives do not scale well to the volume and complexity of current or planned data streams nor the demanding application of such data, such as analysis and modelling. A fundamental change in the access of archives of these data from static, batch systems to dynamic, interactive systems is required. As a first step in enabling this paradigm shift, consider appropriate data management techniques coupled with the process or methods of scientific visualization. The technologies that support visualization show promise in helping to address some of these problems.

Data management

Data management relevant to the access and utilization of large scientific data sets (e.g., locating, retrieving and using data of interest) may be classified into four levels:

I. At the back-end are warehousing issues related to problems of media, bandwidth, protocols, formats.

II. Above that are data models, access techniques, data base systems, query languages and programming interfaces.

III. As a medium between the access of data and the user, consider browsing as an aid for feature identification, which serves as a guide in the data selection process.

IV. At the front-end are human factors issues (e.g., system ergonomics, human perception) and the incorporation of domain-specific knowledge to permit a system to be usable and useful (e.g., domain-driven task-based interfaces and tools).

At level I, high-speed and high-capacity storage and networking are either available now or will be common in the near future. A chief concern is not the volume of data nor the signaling speeds of these devices, but the effective capacity and bandwidth that can be utilized by applications that require warehousing (i.e., above level I). Typical storage and communications protocols are not a match for GB archives, yet current archives are in the TB range,

while those being planned are measured in PB. In addition, accepted data distribution mechanisms such as CD-ROMs hardly scale to data rates planned for projects like NASA's Earth Observing System, which are measured in TB/day (i.e., 1 TB/day is about 2000 CD-ROMs/day).

At level II and to a limited extent level I, consider what is stored and how it is stored. Self-describing physical storage formats and structures should be used. There are many interfaces and structures developed by or used in the scientific community [Treinish, 1992b; Brown et al, 1993], which are driven by application, politics, tradition and requirements. For the kind of data rates that need to be supported for specific computational codes, applications, etc., it is NOT sufficient to provide only bulk access. The accessing software must be able to do so in a meaningful way. This means that the self-describing formats must have associated disk-based structures and software interfaces (e.g., at least abstract data types for programmers and applications, simple utilities for those do not wish to program). They must be transparently accessible at the desk of an end-user. These structures and interfaces must be consistent with the high-speed access to be provided. This must also include task-to-task communication (e.g., transparent workstation access to high-speed storage).

Such requirements demand the utilization of a generalized data model as a mechanism to classify and access data as well as map data to operations. Further, it defines and organizes a set of data *objects* [Treinish, 1992b]. The model (e.g., Haber et al, [1991]) must provide a self-describing

- representation of the physical storage structure (e.g., format)
- structural representation of the data (e.g., data base schema)
- higher-level logical structure

Since the behavior of access to data organized via such a model is in terms of a logical structure, a generalized data model provides a foundation of a data access server as shown in **figure 1**. Otherwise, performance of high-speed devices will not be realized. High-data-rate computations like signal processing, visualization and some classes of modelling also require such support.

Level II also requires enabling tools for data/information systems. Having efficient storage and access are critical for driving applications, but will not be directly useful for helping find data of interest. At the very least, there is a need for low-level metadata management that enables both content and context for the warehousing information. Traditionally, a RDBMS could be used for its implementation. Although, this concept was first prototyped over 10 years ago, a RDBMS cannot handle semantics, spatial information or the bulk [Treinish and Ray, 1985]. The RDBMS would not have any data per se, but would have pointers to the bulk storage enabling simple querying of what is there and where it is stored by characteristics such as spacecraft, instrument, mission, code, date/time, investigator, owner, etc. It would have to supplemented by a non-relational system to adequately support spatial metadata (e.g., Fekete [1990]), as shown in **figure 1.**

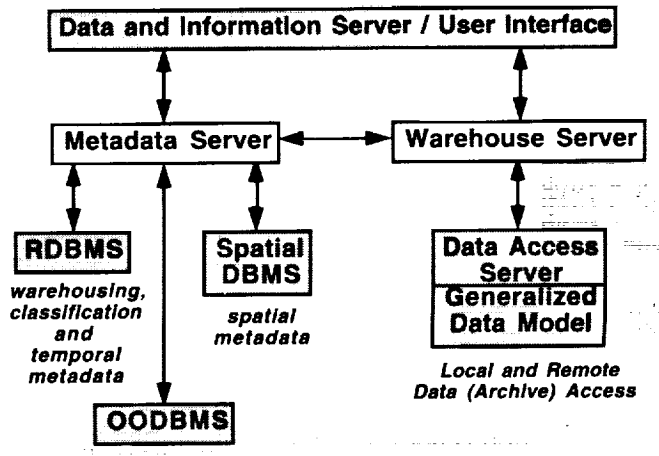


Figure 1. Data management for interactive archives.

There are many challenges in the implementation of an effective data system at level II -- to provide complete access to data of interest simply and easily. A key role for data systems is a means of efficient and intelligent searching and querying for relevant data based upon an assumption that the data volume and complexity is sufficiently large that practical examination of more than a tiny fraction of archive contents is prohibitively expensive. Therefore, the implementation of searches as *meta*-operations on abstractions of the archive (e.g., contextual, domain-driven, spatial, user-driven and visual), though difficult, are highly beneficial. Visual searches or browsing imply the perusal of pictorial representations of data or their abstractions with sufficient content for a user to determine the utility for further, more detailed examination. The advent of practical

methods of scientific visualization show promise in the implementation of visual searches. Hence, consider a very simple notion: *looking* at data of interest in an "appropriate" fashion to determine merit of access for further study. Thus, visualization can be an adjunct to archive management.

History

The idea of visually browsing data is hardly new. Scientists have always used visual abstractions (on paper) of their experimental results for communication. Often these same visualizations en masse were filed and later perused to help recall findings from earlier work. The mechanism for storing, scanning and distribution of such visualizations was the same as for text -- initially shelves and files of paper followed by boxes of microfiche. With the advent of digital data collection instrumentation (e.g., seismic soundings for oil prospecting, study of the earth's atmosphere from spacecraft), this same paradigm was adopted with computer-generated paper- and later microfiche-based visualizations. These visualizations were based upon the graphics technology of the era and the traditions in these fields (e.g., line drawings of iso-contours). Monochromatic microfiche became an effective means of storing visual information compactly that were relatively inexpensive to generate after the cost of the production equipment was amoritized. Further, they were easily distributed, cheap to examine remotely and archiveable. These are important virtues to consider in a modern visualization approach to browsing that goes beyond the relatively limited medium of microfiche.

For example, **figure 2** is a photographic reproduction of a microfiche (approximately three inches x five inches in size) showing contour maps of daily total column ozone observed by a NASA spacecraft over Antarctica during two months of 1986. Although the system that generated the microfiche is primarily intended for interactive use in the analysis of many kinds of earth and space science data [Treinish, 1989], the needs of a low-cost browsing medium could only be met by employing its tools in the batch production of microfiche until fairly recently.

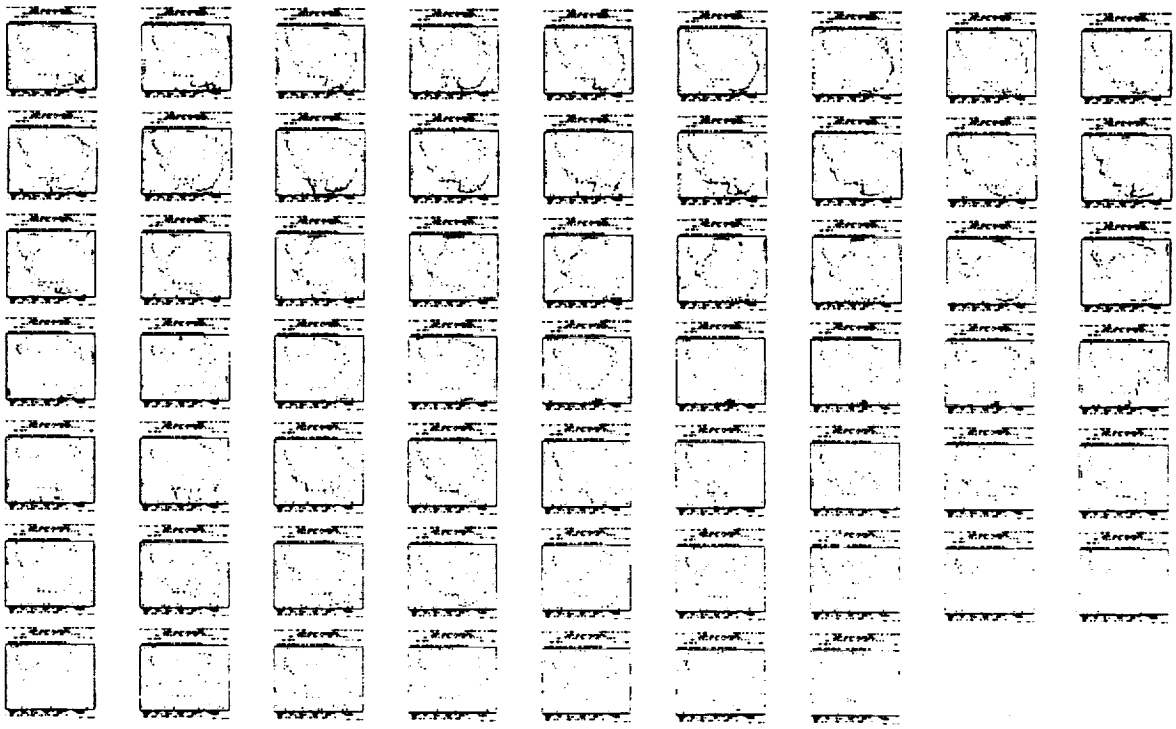


Figure 2. Azimuthal equidistant contours maps of Antarctic daily column ozone for September 24, 1986 through November 23, 1986 (from microfiche).

The advent of digital instrumentation, especially in remote sensing, created another role for browsing that is based upon the assumption that there is insufficient computing resources to support the analysis of all acquired data from all experiments of a specific project or mission. Hence, visual abstractions were created to help identify what subset of data should actually be fully processed to support scientific investigations. These graphical representations, known as *summary plots*, were generated on microfiche and distributed to all participants in a project. They usually contained a time sequence of simple visualizations of instrument outputs in specific modes at sufficient resolution to suitably identify a set of "events" as interesting, and thus warrant further processing. The particular presentations were chosen to highlight the signatures of such events within the limited graphical techniques that could be supported on monochromatic microfiche (line drawing, simple gray-scale polygon fill). Users of such a mechanism would receive a stack of microfiche each week, for example, and visually browse them, looking for patterns in the plots that would be indicative of something interesting (e.g., [Treinish, 1982]).

With improvements in technology to support interactive computing, most efforts associated with data browsing as level III in the aforementioned hierarchy of archive data management and access, have focused on simple images with image data. Furthermore, implementations are generally confined to one data set or a small number of similar data sets [cf, Simpson and Harkins, 1993; Oleson, 1992]. This has been a conscious choice based upon the need for supporting only a limited domain and/or driving interactivity. Unfortunately, these techniques are not applicable to many classes of data or when multiple disparate data sets are to be considered simultaneously, which are the characteristics of most current or planned archives.

Approach

There are four issues associated with interactive data browsing using visualization from archives of scientific data:

A. What are the visual abstractions for data presentation and interaction?

B. How are the browsing products distributed?

C. How is interactivity achieved?

D. What is the mechanism for integrating browsing into a data system?

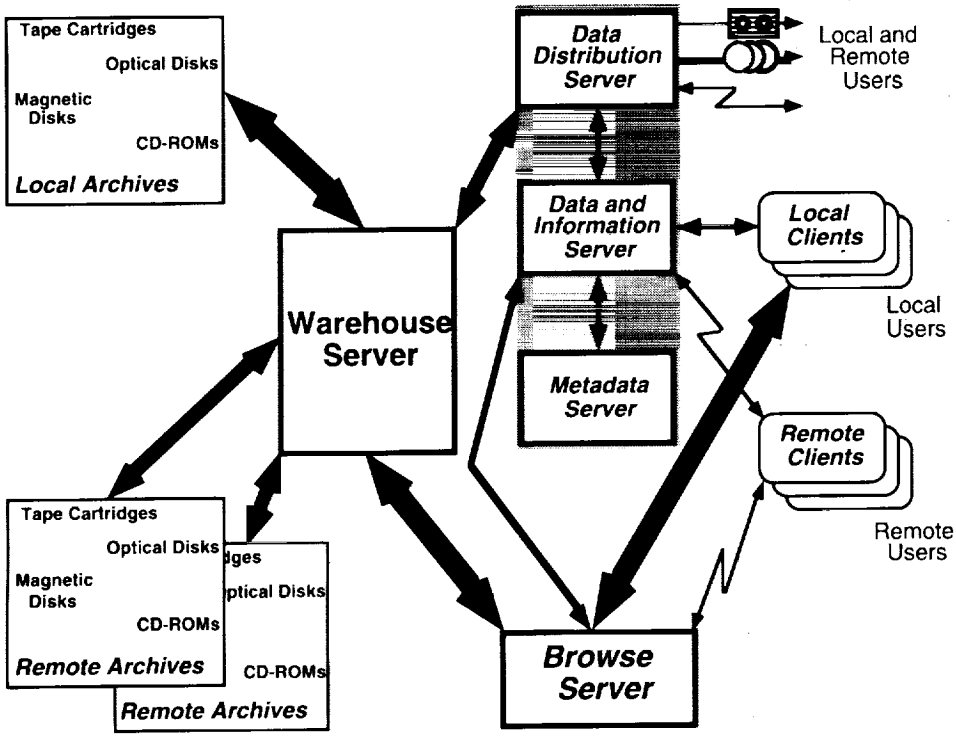


Figure 3 is a schematic of a simple interactive archive system. Each of the aforementioned levels of data management are shown, such that level I services are provided via archive, warehouse and data distribution servers. Level II services are within the gray box: data and information, and metadata servers. **Figure 1** illustrates the relationship between these components and specific data management technology.

Figure 3. Architecture for an interactive archive system utilizing visualization browsing.

Browsing (level III) is provided via the browse server, and local and remote clients. At this level of detail, any level IV services would be embedded within the local and remote clients. The schematic only indicates the major interaction paths between each component as arrows. The arrow thickness corresponds to relative bandwidth of communications, where thickest arrows would imply the greatest bandwidth (e.g., ANSI HiPPI), and the thinnest arrow would imply the least (e.g., Ethernet). A jagged arrow refers to traditional remote network communications. In addition, remote communications could be disconnected by distribution of products to a remote site via an alternate medium (e.g., CD-ROM) to be utilized with a local client at the remote site.

Abstraction

Efforts in scientific visualization typically focus on the analysis of data in a post-processing mode. This is still the emphasis even with the potential advent of computational steering or telescience, the remote sensing equivalent. Browsing is a subjective process involving the human visual system, which is consistent with one of the origins of the notion of scientific visualization as a method of computing. Consider the process by which an individual as an observer walks into a room. That person scans the room, and identifies and categorizes the contents (objects) in the room. From the classification or browsing process, further interaction with those contents may take place. Hence, the requirements for qualitative browsing presentation are not the same as for analysis.

As with primitive summary plots, browsing visualization methods should intentionally transform the structure of the data to highlight features to attract attention, scale dimensions to exaggerate details, or segment into regions [Rogowitz and Treinish, 1993a]. Traditional image browsing is very limited in this regard. Highlighting can only be achieved by modifying a pseudo-color or false-color presentation, which may not be sufficient even for image data or data than can be represented by an image. Of course, this method cannot even be considered for data that cannot be visualized as a simple image (e.g., any non-scalar data, any data of more than two dimensions).

Visual highlighting is often ill-advised for analysis because of the intentional distortion. In general, the browsing visualization should show data in the proper context (e.g., spatial or temporal). This is especially true in the earth and space sciences, where acquired data need to be registered in some (geographic) coordinate system for viewing. If the browsing visualization is to show multiple parameters, then the highlighting techniques must preserve the fidelity of the data (e.g., regions of missing data, original mesh structure) well enough, so that the artifacts in the presentation are not erroneously interpreted as potential (correlative) features in the data. Although the focus of such techniques herein is not for quantitative study, they may have a role in such tasks [Treinish, 1993a].

Distribution

One potential problem with the visual browsing of large data archives is the need to be in relatively close proximity to either the data archives or a facility that can generate the browse products (i.e., high-bandwidth access to an interactive system). In general, high-bandwidth communications between an archive and its users is not always practical, given typical geographic dispersal of scientists and their need for utilizing more than one archive.

Since interactive browsing has usually considered only image data, the distribution of browse products has focused on the distribution of these same images in some form, usually lossy compressed because of the typical size of image archives [e.g., Simpson and Harkins, 1993]. In this sense, the browsing and visualization media and the data are all the same. Hence, the compression is of the original data, where the compressed data will be easier to distribute compared to the original data, but they only apply to a limited class of data (i.e., images). Furthermore, the resultant compressed form may not be suitable for browsing and the quality may be poor.

Alternatively, consider the distribution of compressed visualizations instead of compressed data. These visualizations would be represented as images, so that available image compression technology could be easily utilized. As with image data, uncompressed visualizations would be of high quality but of sufficient volume to be impractical to distribute to remote sites. The compressed visualization images, would be easy to distribute as with compressed image data. The key difference is that the compressed visualization could apply to any collection of data and would be of high quality. Hence, one could consider the distribution of compressed visualizations as the *summary plots* of the 1990s because the lack of resources to access or distribute all archived data or their abstractions is similar to the lack of resources to

process or analyze all acquired bits, which motivated the generation of summary plots in the past.

Such an approach could be extended for predefined access and browsing scenarios, where the compressed visualizations are available for on-line remote access or via CD-ROM. In this case, the viewing of such compressed visualizations would require relatively simple, low-cost display software and hardware, which is becoming more readily available in desktop environments. In general, the costs associated with the access and distributed of compressed visualizations will be similar to those of image data, but of course, be significantly less than those of data. There will be an additional cost of generating the visual browse products, which is justified given the added value and obviating the need for distributing the data themselves.

Interactivity

For visual browsing to be effective it must be interactive. Otherwise, it is little different than watching television or traditional image browsing. One aspect of the interaction is in terms of data management: selection of and access to data of potential interest and metadata to help in guiding that process. In terms of visualization, the ability to interact with the browse "objects" (e.g., spatially, temporally) in near-real-time is critical. This requires rapid access to the data and generation of the browse products.

Implementation

Abstraction

The requirements to create qualitative visualizations that are effective as browse products do have implications on the software used to create them. Registration of data into an appropriate coordinate system for viewing requires the support of user-defined coordinate systems. To be able to properly show more than one data set simultaneously requires the ability to operate on different grid structures simultaneously and transformation of grid geometry independent of data. Depending on the visualization strategies being used, rendered images may need to contain different geometries (e.g., points, lines, surfaces and volumes independent of color or opacity or of original grid structure). (See Treinish [1993a] for a discussion of these ideas with respect to data analysis.)

A commercial scientific visualization environment (IBM Visualization Data Explorer - DX) has been used to experimentally implement the aforementioned browsing techniques. DX is a general-purpose software package for scientific data visualization. It employs a data-flow-driven client-server execution model and is currently available on Unix workstation platforms (e.g., manufactured by Sun, SGI, IBM, HP and DG) as well as a parallel supercomputer, IBM POWER Visualization System [Lucas et al, 1992].

Distribution

Near-real-time browsing of visualizations at sufficient resolution to see relevant contents requires the distribution of a large number of images. Clearly, lossy compression is necessary to drive viewing with update rates near the refresh rate of display controllers. For utilization remote from the archive or browse server, low-cost, simple display software and hardware is needed as well. There is much literature on data compression strategies and algorithms (e.g., a few hundred citations reported over the last decade by the National Technical Information Service alone, [NTIS, 1992]), which will not be discussed herein. This notion of visualization distribution is built upon the extant and growing body of implementations of compression algorithms, which are being utilized in scientific, multimedia and entertainment applications.

The idea of distributing imagery for visualization is not new. For example, Johnston et al [1989] experimented with both block-truncation and Lempel-Ziv compression for the distribution of visualization animation. Rombach et al [1991] discussed the Joint Photographic Experts Group (JPEG) compression scheme for the distribution of cardiographic imagery from different sources like ultrasound, magnetic resonance imagery and angiography. In these and other cases, the authors considered a low-cost viewing environment on the desktop as being critical, especially if the expense of generating the images to be distributed is high. Therefore, in this initial implementation, block-truncation (lossy, i.e., reducing the number of colors to represent a full-color pixel), modified Lempel-Ziv (lossless, e.g., like Unix compress) and temporal coherence between animation frames (lossy, e.g., the Moving Picture Experts Group, MPEG [LeGall, 1991]) will be considered.

To illustrate the viability of this approach, consider a modest data set, composed of a rectilinear scalar field of 32-bit floating-point numbers (e.g., atmo-

spheric temperature) at one-degree of geographic resolution at seven levels in the earth's atmosphere. Therefore, each time step would require about 1.7 MB. If these are daily data then less than a year would fit on a single CD-ROM, uncompressed. This does not include ancillary data required for annotation such as coastline maps, topography or other reference material. Lossy compression would not be relevant since the data are not imagery. Lossy compression could be applied to each layer of the atmosphere individually. However, the results would be rather poor (i.e., the two-dimensional spatial resolution is already low, 180 x 360), and spatial coherence for the entire volume could not be maintained. If lossless compressed, decompression could be expensive (e.g., Lempel-Ziv) or inconvenient (e.g., scaled/encoded 12- or 16-bit integers). Either compression approach is highly sensitive to the contents of the data set.

Alternatively, visualization compression is independent of data characteristics and only the resolution of the visualization image(s) drive the compression/distribution/decompression cost. Of course, the distribution of uncompressed browse visualizations is expensive, potentially more than that of the uncompressed data. Lossless compression although cheaper to distribute would still require the decompression process, and could also be more expensive than that for the data themselves. Hence, the lossy compression of the visualization imagery is the best approach from both a cost perspective as well as from that of image quality. Hence, for sequence of 640x480 24-bit image representations of the simple volumetric data set, over 14 years worth of frames for such daily data could be stored using a simple 8:1 block truncation compression (i.e., each 24-bit pixel is represented by three bits) on a single CD-ROM. Using 32:1 JPEG compression, a sequence of over two years of these images at hourly resolution could be stored on a single CD-ROM.

Interactivity

For browsing to be effective it must be interactive with near-real-time system response. With data sets of interesting size, e.g., ≥ O(1 GB), immediate interaction cannot take place on current conventional systems (i.e., high-end graphics workstations). Even though a 1 GB data set is admittedly modest by today's standards for data generation, the access and visualization of an entire data set for browsing or even a large fraction of it place significant burdens on the floating point and bandwidth capacities of the computer system being employed. The bandwidth requirements are derived from the bulk access speeds of large data sets and the transmission of images sufficiently fast to be interactive. The floating point requirements stem from three classes of computation for visualization:

1. Transformation (e.g., warping, registration)

2. Realization (e.g., contouring, color mapping, surface deformation)

3. Rendering (i.e., creating images)

Although the visualization requirements are different, the computational needs of interactive browsing are very similar to those of visualization in a virtual world environment (e.g., [Bryson and Levit, 1992]).

Experimentation with a commercial parallel supercomputer (IBM POWER Visualization System) and the aforementioned DX environment has shown the viability of such interactive visual browsing, even with multiple data sets. In this effort, the PVS and DX combination has been used as the browse server with local and remote clients on workstations as indicated in **figure 3**. The PVS functions as an archive server in this context. **Figure 4** shows the relationship between the browse server, and the data and information server and its components in the interactive archive system shown in **figure 3**.

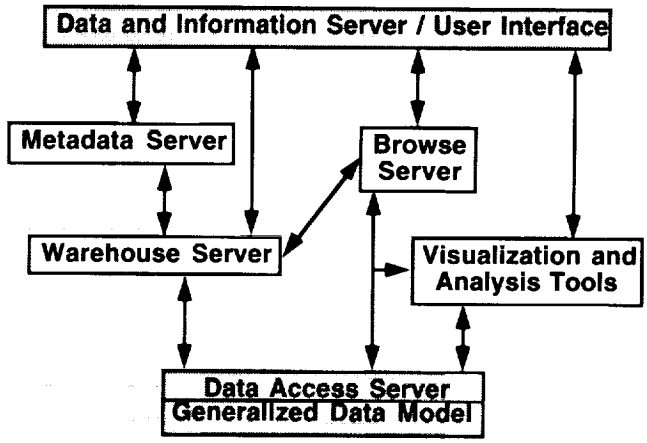


Figure 4. Architecture for a data and information system incorporating visualization browsing.

The IBM POWER Visualization System (PVS) is a medium-grain, coherent shared-memory parallel supercomputer with the interactivity of a workstation. This has been achieved via a programmable (general-purpose) approach instead of special-purpose hardware with balance among floating point

performance via moderate parallelism, large physical memory and high-speed external and internal bandwidth. The PVS consists of three major hardware components: server, disk array and video controller. The server is a symmetric multi-processor with up to 32 processors (40 MHz Intel i860XR or 44 MHz Intel i860XP), a 1.28 GB/sec (at 40 MHz) internal backplane supporting a hierarchical bus structure, hierarchical memory (16 MB local memory per processor and up to 2 GB global/shared), ANSI HiPPI communications, fast and wide SCSI-2, and an IBM RISC System/6000 support processor (for local area network and storage access). The server supports parallelized computations for visualization via DX. The disk array is a HiPPI-attached RAID-3 device with either 50 MB/sec or 95 MB/sec sustained access speeds, or a fast and wide SCSI-2 four-bank RAID-3 device with 76 MB/sec sustained access speeds. It provides access to archived data to be browsed. The video controller is a programmable 24-bit double-buffered with 8-bit alpha overlay (for custom XWindow server) frame buffer attached to an IBM RISC System/6000 workstation. It receives images from the PVS server via HiPPI (either compressed or uncompressed) at resolutions up to 1920x1536, including HDTV, for real-time image updates. The video controller provides an interface for interaction with and viewing of the browsing visualization at speeds up to 95 MB/sec.

Results

Abstraction

Figure 5 shows a traditional two-dimensional visualization of ozone data similar to those shown in **figure 2**. The data are realized with a pseudo-color map and iso-contour lines for September 30, 1992. The rectangular presentation of the data is consistent with the provided mesh in that it is torn at the poles and at a nominal International Date Line. The ozone data are overlaid with a map of world coastlines and national boundaries in magenta as well as fiducial lines (of latitude and longitude) in white.

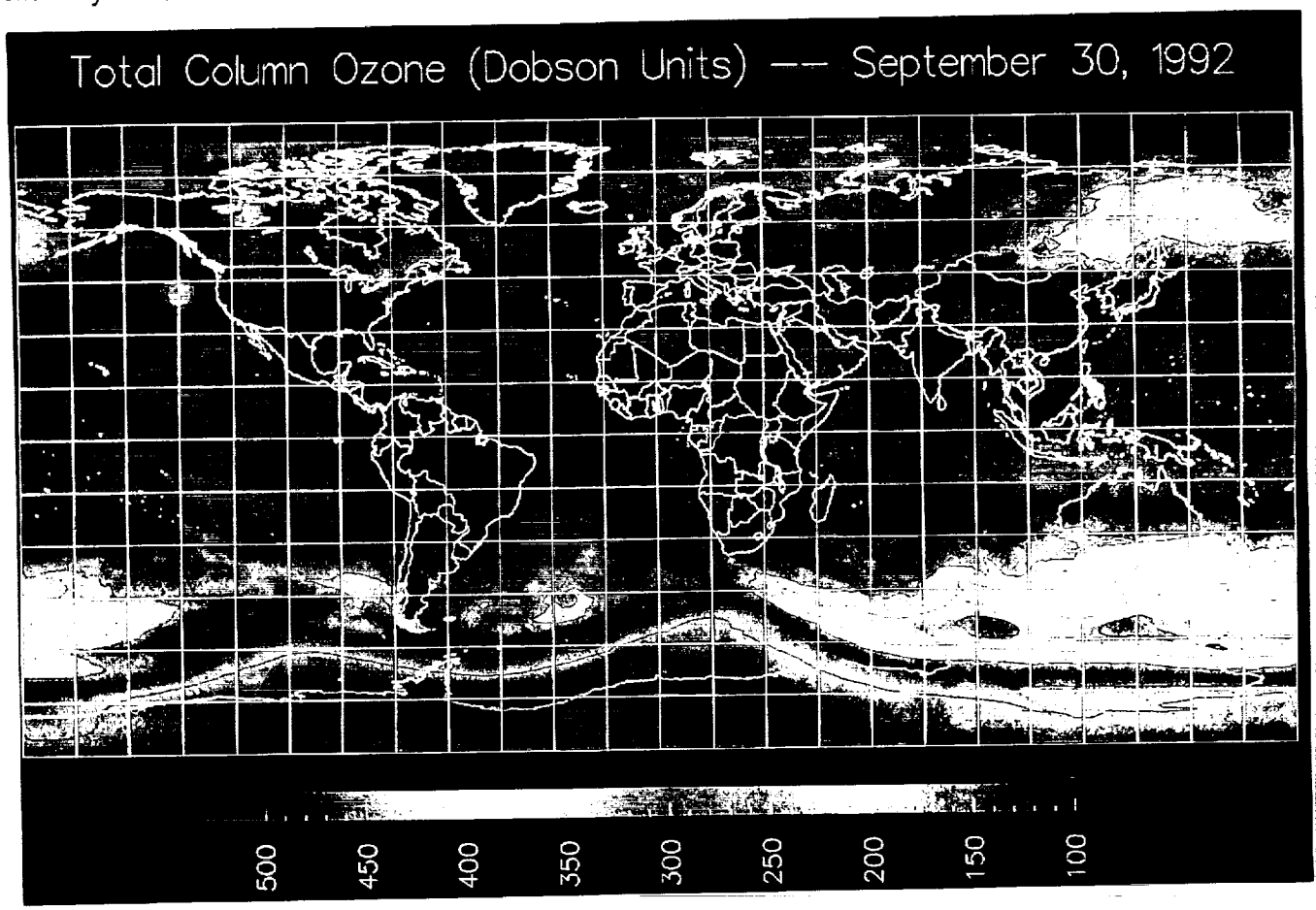


Figure 5. Pseudo-color-mapped global column ozone on September 30, 1992.

To provide a qualitative impression for browsing, the data are transformed to a three-dimensional continuous spherical surface in **figure 6**. The ozone is triply redundantly mapped to radial deformation, color and opacity so that high ozone values are thick, far from the earth and reddish while low ozone values are thin, close to the earth and bluish. Replacing the map for annotation is a globe in the center of this ozone surface. The use of three redundant realization techniques results in textures for qualitatively identifying regions of spatial or temporal interest. The gauges on the left illustrate the daily **total ozone** statistics. The pseudo-hour hand position ranges from 100 to 650 Dobson Units, while the color corresponds to that of the ozone surface. From the top they show the mean, minimum and maximum for each day. The value corresponding to geographic view for each frame is shown next. At the bottom is a bar chart indicating the standard deviation of the daily measurements. This approach to qualitative visualization is potentially applicable to a large variety of simulated or observed earth and planetary data on a large spatial scale, especially for two and three-dimensional scalar fields.

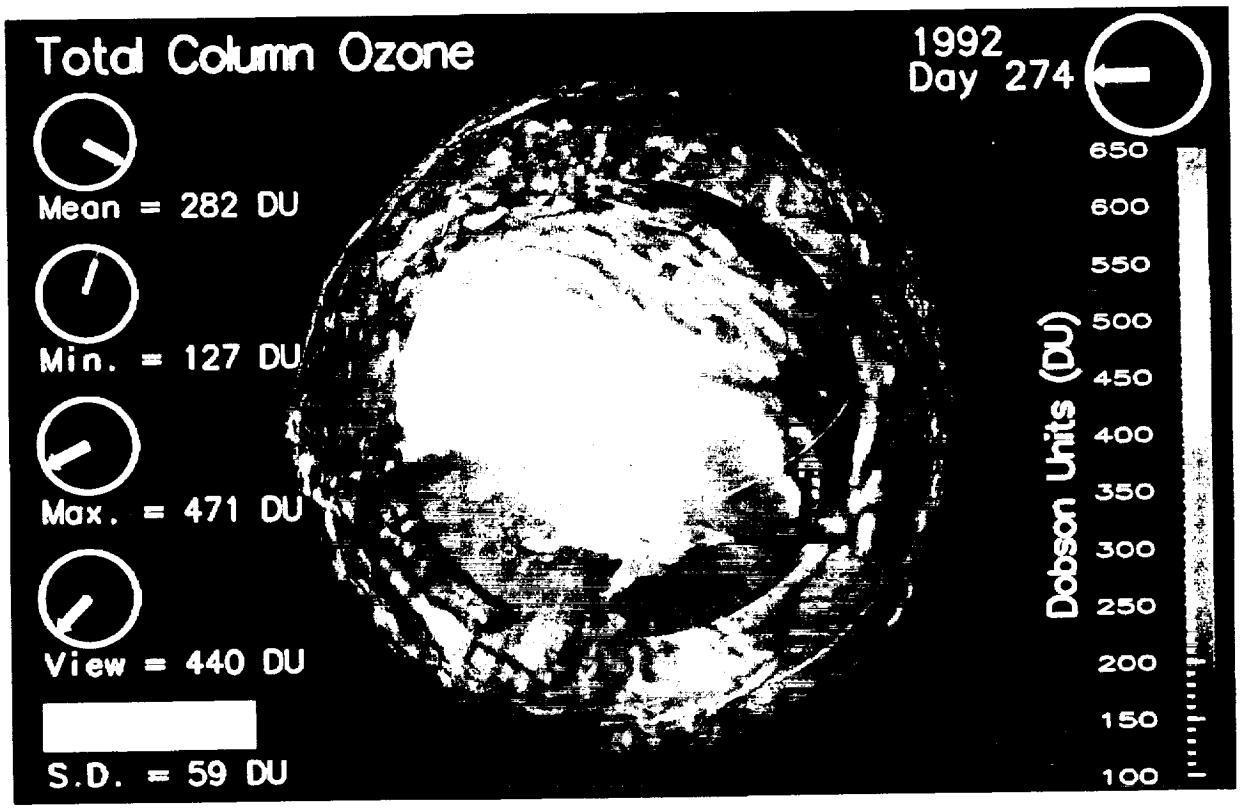


Figure 6. Radially deformed pseudo-color and opacity-mapped spherically warped surface of global column ozone on September 30, 1992 with annotation.

A browsing animation of the ozone would be one that illustrates the data on a daily basis as in **figure 6** for the entire archive of available data (i.e., late 1978 through early 1993). The geographic view of these data would change with each day to provide reasonable coverage of the entire globe over a complete year. The view would be chosen to concentrate on interesting regions such as the poles during appropriate seasons like spring. Treinish [1992a] and Treinish [1993a] are examples of such a browsing animation, which are useful a posteriori of its generation for identifying periods of time or geographic regions that warrant further study.

These browsing animation sequences are derived from about 1 GB of data, a two-dimensional scalar field

over a torn geographic mesh. For each of the over 4700 frames of the sequence, several calculations are required to create a visualization image. Each image is actually composed of two images, which have been blended. There is a background image, which is composed of frame-variant contents under a constant view: primarily opaque polygonal text, dials and bars. This annotation changes to summarize daily statistics. There is a foreground image, which is also composed of frame-variant contents, but with a frame-variant view. Each foreground image contains a static globe with the surrounding translucent ozone surface. For each day, the ozone data are transformed (irregularized to remove regions of missing data and warped onto a sphere), realized (color and opacity mapped and surface deformed) and full-color rendered (about 45,000 to 50,000 translucent quads for the ozone; 259,200 full-color-mapped opaque quads on a sphere with normals for the globe). The foreground and background images are brightened and blended to compose each final frame. Each frame at workstation resolution (about 1.2 million pixels) using DX on a 32-way (40 MHz) PVS required about 12 seconds of computing time. Hence, the entire animation took about 15 hours at that resolution.

Distribution

Current efforts on data distribution have focused on the application of compression techniques to three sample animation sequences on a 32-way (40 MHz) PVS equipped with a RAID-3 HiPPI disk array capable of 50 MB/sec sustained access speeds. The first example is a high-resolution sequence of 2040x1536 32-bit (8-bits of red, green, blue, alpha) images, 88 frames in length totalling about 1052 MB. An 8:1 block-truncation lossy compression (i.e., each 32-bit pixel is represented by 4 bits) required about 41 seconds, resulting in a rate of approximately 26 MB/second or 2.15 Hz disk to disk. Lempel-Ziv lossless compression was applied to the entire sequence as a whole, not on a frame-by-frame basis. As expected the results were considerably slower, requiring about 2 minutes, 2 seconds, yielding a rate of approximately 8.7 MB/second disk to disk to achieve 45.4% compression. In both cases, the compression algorithms were parallelized on the PVS.

The second example is with a 151-frame sequence of 640x480 32-bit images of about six months worth of animation similar to that illustrated in **figure 6**. Results with this considerably smaller collection (about 177 MB) are quite similar, pointing to the potential scalability of the shared-memory, symmetric multiprocessor systems like a PVS to this problem. For the 8:1 lossy compression, about eight seconds were required yielding a rate of approximately 23 MB/second or 18.9 Hz disk to disk. The lossless compression of the entire sequence required about 22 seconds to achieve 62.6% compression at 8.0 MB/second disk to disk.

The third example is with the 5853 frames of a digital video (D1) sequence [Treinish, 1993b]. Most of the sequence is composed of frames similar to **figure 6** -- one for each day from January 1, 1979 through December 31, 1991. Each D1 frame is composed of 10-bits each of YUV (a chrominance and intensity-based specification of color) at 720 x 486 for playback at 30 Hz. Hence, this 3 minute, 15 second sequence is 10.6 GB in size, which is maintained as a single file on a PVS. A PVS-based MPEG compression facility was used to create a 250:1, lossy-compressed MPEG-1 sequence. Approximately, 12 minutes, 5 seconds were required for this operation, yielding a rate of 14.7 MB/second or 8 Hz, disk to disk.

Interactivity

Figure 7 is a snapshot of a DX Motif-based interface for a prototype interactive browsing system. It provides very simple modes of interaction: selection of space and time (i.e., geographic regions or seasons of interest) for browsing via the spherically warped presentation shown in **figure 6**. There are dial widgets for the specification of geographic viewing centroid of the global "object", and slider widgets for selection of the year to examine and the viewing width. A VCR-like widget provides control over the choice of the portion of the year to browse by Julian day. Optionally, a zonal slice of the data being browsed at the specified longitude may be shown as a pseudo-colored line plot of the latitudinal distribution. The data illustrated in **figure 7** are the aforementioned global column ozone data derived from the same 14-year archive as shown in **figures 2, 5** and **6**.

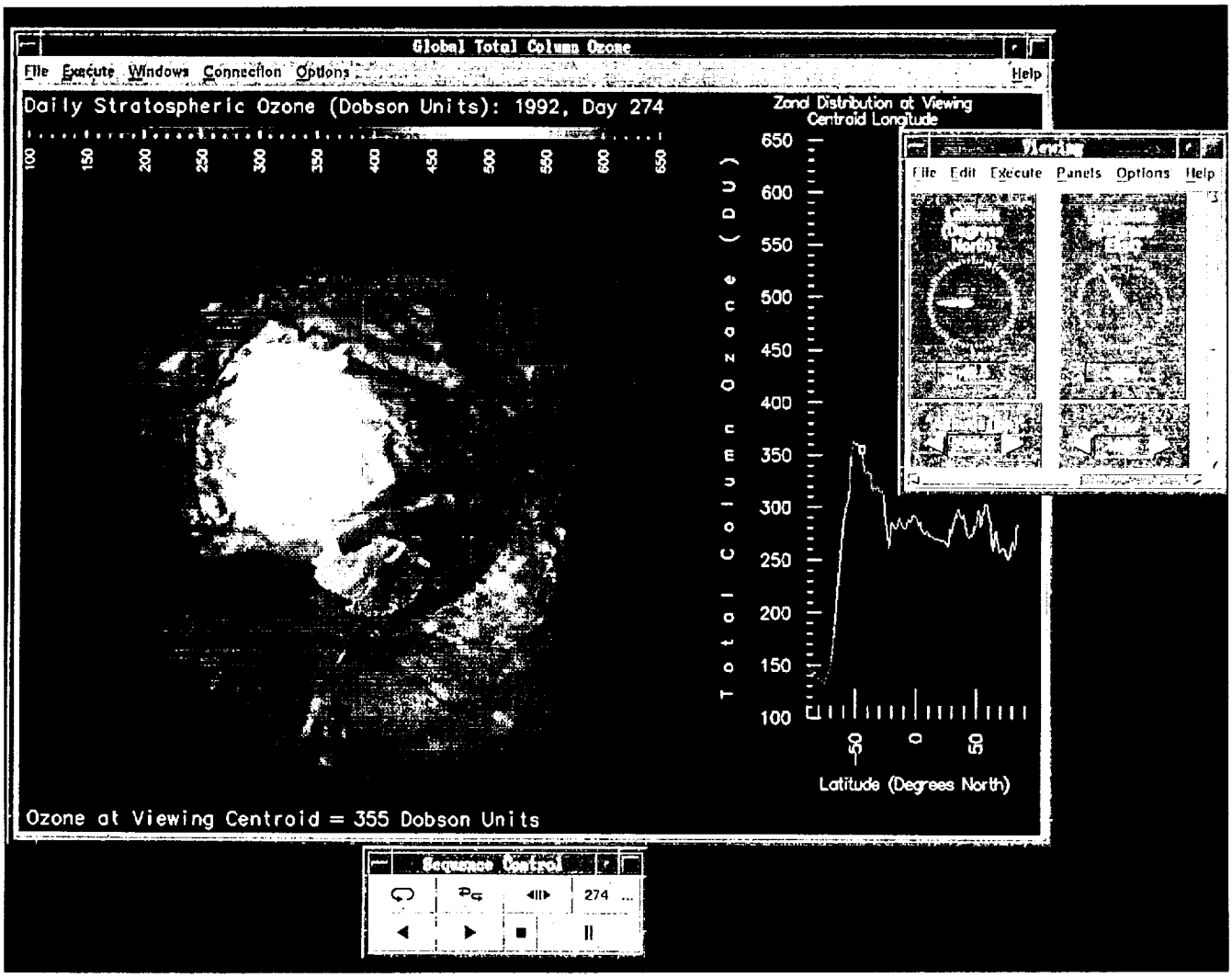


Figure 7. Example user interface for browsing showing global column ozone on October 1, 1987.

The prototype browsing system is built as a client-server, consistent with the architecture of DX, as shown in **figure 8**. A PVS functions as the browse server in this implementation. High-speed display of browsing visualizations is local to the PVS. Remote display is via standard XWindow services with update rates limited to what the network infrastructure can provide.

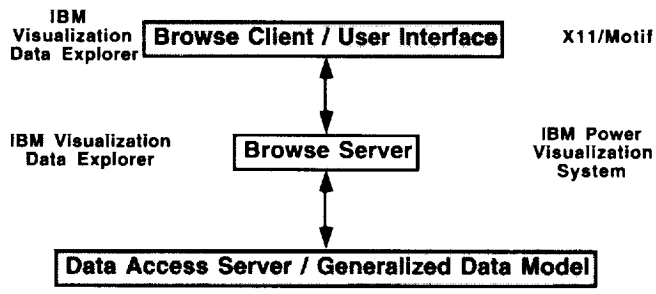


Figure 8. Architecture for a prototype interactive browsing system.

Conclusions

A focus on qualitative methods of presenting data shows that visualization provides a mechanism for browsing independent of the source of data and is an effective alternative to traditional image-based browsing of image data. To be generally applicable, such visualization methods, however, must be based upon an underlying data model with support for a broad class of data types and structures.

Interactive, near-real-time browsing for data sets of interesting size today requires a browse server of considerable power. A symmetric multi-processor with very high internal and external bandwidth demonstrates the feasibility of this concept. Although this technology is likely to be available on the desktop within a few years, the increase in the size and complexity of archived data will continue to exceed the capacity of "workstation" systems. Hence, a higher class of performance, especially in bandwidth, will generally be required for on-demand browsing.

A few experiments with differing digital compression techniques indicates that a MPEG-1 implementation within the context of a high-performance browse server (i.e., parallelized) is a practical method of converting a browse product to a form suitable for network or CD-ROM distribution.

Future Work

From this initial prototype implementation of an interactive data browser, there are several areas for future work. Since practical low-cost decompression of JPEG-compressed images is becoming available on the desktop, experimentation with JPEG is warranted [Pennebaker and Mitchell, 1993]. As with JPEG, the MPEG-1 motion video compression technique is becoming available for multimedia applications of video sequences on the desktop, whether the animation is distributed via network or CD-ROM. Additional testing with MPEG-1 and the higher quality, MPEG-2 as it becomes available is required within the browse server as well as on various desktop playback systems.

The second area of research would focus on fleshing out more of the interactive archive architecture schematically illustrated in **figures 1, 3, 4** and **8**. Specifically, the prototype interface and visualization could migrate to a data-driven one conceptually similar to the primitive implementation discussed by Treinish [1989], which could be integrated with metadata and data servers to achieve a browsing archive system (i.e., data management services at the aforementioned levels I, II and III). This would also imply the availability of integrated data and information services similar to those in the rudimentary system described by Treinish and Ray [1985]. Such an approach would be further enhanced by the integration of the prototype browsing system with tools for data analysis, which are already available.

The strategies for qualitative visualization have focused on only a few methods for spherically-oriented data with large spatial extent. Clearly, investigation of alternative approaches of highlighting features in such data are required, for which there are a number of potential issues [Rogowitz and Treinish, 1993b]. In addition, the extension of this browsing architecture to other classes of data is also warranted.

Acknowledgements

All of the data sets discussed above were provided courtesy of the National Space Science Data Center, NASA/Goddard Space Flight Center, Greenbelt, MD.

References

1. Brown, S. A., M. Folk, G. Goucher and R. Rew. *Software for Portable Scientific Data Management.* **Computers in Physics, 7**, n. 3, pp. 304-308, May/June 1993.

2. Bryson, S. and C. Levit. *The Virtual Wind Tunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows.* **Proceedings IEEE Visualization '91**, pp. 17-24, October 1991.

3. Fekete, G. *Rendering and Managing Spherical Data with Sphere Quadtrees.* **Proceedings IEEE Visualization '90**, pp. 176-186, October 1990.

4. Haber, R., B. Lucas and N. Collins. *A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids.* **Proceedings IEEE Visualization '91 Conference**, pp. 298-305, October 1991.

5. Johnston, W. E., D. W. Robertson, D. E. Hall, J. Huang, F. Renema, M. Rible, J. A. Sethian. *Video-based scientific visualization.* **Proceedings of a Workshop on Geometric Analysis and Computer**

Graphics, University of California at Berkeley, Berkeley, CA, pp. 89-102, May 1989.

6. LeGall, D. J. *The MPEG Video Compression Standard.* **Proceedings IEEE COMPCON Spring '91**, pp. 334-335, 1991.

7. Lucas, B., G. D. Abram, N. S. Collins, D. A. Epstein, D. L. Gresh and K. P. McAuliffe. *An Architecture for a Scientific Visualization System.* **Proceedings IEEE Visualization '92**, pp. 107-113, October 1992.

8. National Technical Information Service. **Data Compression (Citations from the NTIS Database).** NTIS Document PB92-802750, U. S. Department of Commerce, Springfield, VA, 1992.

9. Oleson, L. The Global Land Information System. Eros Data Center, U. S. Geological Survey, Sioux Falls, SD, 1992.

10. Pennebaker, W. B. and J. L. Mitchell. **JPEG: Still Image Data Compression Standard**. To be published by Van Nostrand Reinhold, 1993.

11. Rogowitz, B. E. and L. A. Treinish. *Data Structures and Perceptual Structures.* **Proceedings of the SPIE/SPSE Symposium on Electronic Imaging, 1913**, pp. 600-612, February 1993.

12. Rogowitz, B. E. and L. A. Treinish. *An Architecture for Perceptual Rule-Based Visualization.* **Proceedings IEEE Visualization '93**, pp. 236-243, October 1993.

13. Rombach, M. R., U. Solzbach, U. Tites, A. M. Zeiher, H. Wollschlager, H. Just. *PACS for cardiology - Perspective or Fiction?* **Proceedings IEEE Computers in Cardiology**, Venice, Italy, pp. 77-79, September 1991.

14. Simpson, J. J. and D. N. Harkins. *The SSable System: Automated Archive, Catalog, Browse and Distribution of Satellite Data in Near-Real Time.* **IEEE Transactions on Geoscience and Remote Sensing, 31**, n.2, pp. 515-525, March 1993.

15. Treinish, L. A. *The Dynamics Explorer Summary Plot Software System.* NASA/Goddard Space Flight Center, Internal Report, March 1982.

16. Treinish, L. A. *An Interactive, Discipline-Independent Data Visualization System.* **Computers in Physics, 3**, n. 4, July 1989.

17. Treinish, L. A. *Climatology of Global Stratospheric Ozone (1979 through 1991).* **Proceedings IEEE Visualization '92**, video supplement, October 1992.

18. Treinish, L. A. *Unifying Principles of Data Management for Scientific Visualization.* **Proceedings of the British Computer Society Conference on Animation and Scientific Visualization**, Winchester, UK, December 1992 and **Animation and Scientific Visualization Tools and Applications** (R. Earnshaw and D. Watson, editors), Academic Press, pp. 141-169, 1993.

19. Treinish, L. A. *Visualization of Stratospheric Ozone Depletion and the Polar Vortex.* **Proceedings IEEE Visualization '93**, pp. 391-396, October 1993.

20. Treinish, L. A. *Climatology of Global Stratospheric Ozone (1979 through 1991).* **ACM SIGGRAPH Video Review, 93**, August 1993.

21. Treinish, L. A. and S. N. Ray. *An Interactive Information System to Support Climate Research.* **Proceedings of the First International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology**, American Meteorology Society, pp 72-79, January 1985.

Planning Applications in Image Analysis

Mark Boddy Jim White Robert Goldman
{boddy | jwhite | goldman}@htc.honeywell.com
Honeywell Technology Center, MN65-2100
3660 Technology Drive
Minneapolis, MN 55418

Nick Short, Jr.
short@dunloggin.gsfc.nasa.gov
ISTO, Code 930.4
NASA Goddard Space Flight Ctr.
Greenbelt, MD 20771

Abstract

We describe two interim results from an ongoing effort to automate the acquisition, analysis, archiving, and distribution of satellite earth science data. Both results are applications of Artificial Intelligence planning research to the automatic generation of processing steps for image analysis tasks. First, we have constructed a linear conditional planner (CPed), used to generate conditional processing plans. Second, we have extended an existing hierarchical planning system to make use of durations, resources, and deadlines, thus supporting the automatic generation of processing steps in time and resource-constrained environments.

1 Introduction

The collection, analysis and distribution of data resulting from NASA science missions is an increasingly daunting task. The National Space Science Data Center (NSSDC) responds to more than 2500 data requests from remote users in a single year [8]. As of 1990, NSSDC's archives included more than 6000 Gigabytes of digital data and 91 million feet of film. By 1995, the NSSDC is expected to contain 40,000 Gigabytes of digital data. Shortly thereafter, the satellites of the Earth Observing System (EOS) will come online, eventually adding new data at a rate of nearly 2000 Gigabytes per day, over an expected mission duration of 15 years [12, 6].

The EOS Data Information System (EOSDIS) is being designed and built to support the storage, analysis, and retrieval of data from this immense archive. Dozier [6] offers the following characterization:

> EOSDIS must allow scientists to easily and quickly acquire usable, understandable, timely data. "Timely" means "a reasonable period following the measurements" – one to two days after the observations, or up to a week for higher-level products. "Quickly" means minutes, not hours. "Easily" means that the user should not have to jump through many hoops to request the data.

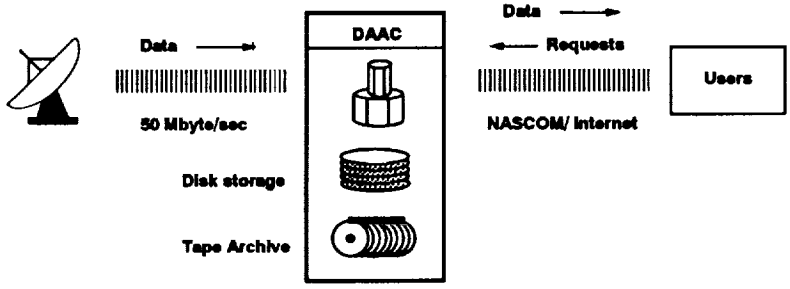


Figure 1: The EOSDIS domain

EOS data will be supplied by several different types of sensors and used by scientists in a variety of disciplines, most with no special knowledge of how EOS data is obtained or organized. The type of sensor from which the data was gathered will affect the processing necessary to render the data useful. The use to which the data is put will determine both the images retrieved (a geologist and an oceanographer will be interested in very different sets of data) and the analysis to which that data is subjected (e.g., topography vs. phytoplankton levels).

Raw and analyzed data will be stored in a distributed network of database sites, known as Distributed Active Archive Centers (DAACs). Also connected to the network will be a variety of special-purpose hardware that can be used for further analysis of either new or retrieved data. These analyses may consist of several steps (e.g., scan line removal, georegistration, or normalization for the incident angle of the sun), and will be run on a distributed network of heterogeneous machine types. Given the enormous amount of data involved, most of it will of necessity be stored off line. We anticipate hierarchical caches for data storage [2] with high-speed disks at the top of the hierarchy and tape archives at the bottom. Data will move up and down in this hierarchy for further analysis.

A high level concept of the resulting system is depicted in Figure 1. Data is received by any of several ground stations from any of a set of satellites, and transmitted to one or more of the archive centers, where it is analyzed as necessary (and as time permits), and then archived. Scientists interested in using the data may make requests that data be retrieved from one or more of the archives and analyzed further.

In both joint and separate work at NASA's Goddard Space Flight Center and the Honeywell Technology Center, we have been working on automating the acquisition, initial processing, indexing, archiving, analysis, and retrieval of satellite earth science data, with particular attention to the processing taking place at the DAACs.

In this paper, we present the results of ongoing work on planning for image process tasks in the EOSDIS *Product Generation System* (PGS). Section 2 presents the problem presented by PGS in additional detail. Section 4 describes the extension of a Nonlin style hierarchical planner to use information about deadlines, durations,

and resources. Section 3 is a dicussion of the use in image processing of *conditional plans:* plans including branching points dependent on the outcome of some earlier action (e.g., an observation of some type).

2 Data Management for Earth Science

NASA's role in the Mission to Planet Earth is the Earth Observing System (EOS) program and several smaller Earth science missions. These missions represent efforts to study the Earth's geosphere, biosphere, atmosphere, and cryosphere, as a system of interrelated processes by modelling surface temperature, ozone depletion and greenhouse effects, land vegetation and ocean productivity, and desert/vegetation patterns to name a few. With participation from the European Space Agency, Japan, Canada, and NASA, several platforms containing a multitude of sensors will be launched in the late 1990's, producing data that will be stored in geographically-oriented data systems such as the EOS Data and Information System (EOSDIS). In general, EOSDIS will manage the mission information, the data acquisition and distribution, the generation of scientific data products, and the interface to external systems.

Ensuring access to this information is a challenging task because of the daunting size of EOSDIS and the potential limitations of current technologies. Over its 15 year life, the Data Archival and Distribution System (DADS), a component of the EOSDIS, will eventually maintain around 11 petabytes [12].[1] While mass storage technology will solve some archiving problems [2], finding data will require new and innovative methods for users to effectively search the archives. The archives will include a variety data types including raster satellite images, ancillary vector/raster maps, derived spatial products from model simulations (e.g., output from global temperature models), and associated engineering and management textual data, suggesting that the archive and meta database will be both diverse and complex.

In current NASA scientific data systems, data are found by users who already know information related to the context of the satellite processing environment, such as the time of the satellite's observation, the satellite and sensor type, and location. This context-based metadata search forces the user to translate scientific needs into project specifications that often contain esoteric NASA nomenclature. A better solution, often called content-based metadata search, is to allow scientists to find data based upon their scientific interests within the imagery. Providing features based upon scientific interests for searching through a database assumes that a system can be created to interpret imagery with the skill of a scientist, yet with the speed of the computer. This automation has been the goal of many researchers in remote sensing, image processing, and computer vision for years; there is no known general solution to the problem.

[1] One petabyte is 10^15 bytes.

2.1 Opportunities for Automation

In this section, we describe the necessary functions for an automated planning system for image classification and indexing according to *browse products*. The entire range of functions described here are actively under development or investigation at this time. In the rest of the paper, we restrict ourselves to a discussion of the generation of plans for image analysis.

Despite the the lack of a general theory, computer-based photo interpretation operations for satellite/aerial imagery can be partially defined as file manipulation, calibration, reduction of the number of channels, image enhancement and correction, segmentation, and pattern classification (see figure 3). These operations often require an expert to "mix and match" the steps depending on the quality of the sensor, the format of the data, the properties of the sensing environment (e.g., atmospheric conditions, direction of sun illumination, etc.), availability of ancillary data such as topographic maps and ground truth observations, and the set of possible features within an image. Typically, the end result of this process is a map labelling pixels to classification categories from proven recognizable schemes for which the sensors were designed. Example schemes include: land use/land cover cloud cover type, vegetation cover, and soil type. While these examples refer to physical objects, properties such as temperature and aerosol content also constitute legitimate labels, only each label represents a range of continuous values. In EOS, much of the work of the PGS will be to recognize these features for processing at level 2 and above.

In the realm of automatic feature recognition, the planner is the component that optimizes accuracy as a function of the resource constraints. If there is a lot of available processing time due to a low incoming data rate, then the planner chooses the image processing sequence with the highest expected accuracy. If the data rate is high, then the planner constructs a sequence that either substitutes computationally cheaper, yet less accurate image processing steps for expensive operations, eliminates steps that can be deleted without a major loss, or uses a fixed-time default plan that implies an upper bound on the highest allowable data rate (e.g., ingest only file header information that comes with the raw data).

The planner must make choices regarding preprocessing steps and image classifiers as a function of the input image's header information, called ephemeris data. For example, suppose that an image from the Moderate-Resolution Imaging Spectrometer-Nadir (MODIS-N) sensor of EOS arrives with its areal extent over Washington D.C. Further suppose that after launch, MODIS-N produced scan lines such as LANDSAT MSS's "sixth-line striping," evidenced by horizontal banding within the images . Modis-N was designed to characterize surface temperature at 1-km resolution, ocean color, vegetation/land surface cover (e.g., leaf area index and land cover type, vegetation indices), cloud cover and properties, aerosol properties, and fire occurrence. Based upon this information, the planner constructs the sequence of steps by first stripping off the header file from the raw data, which indicates the time of observation, the sensor, sun angle and azimuth, location, and

file format. The planner then queries an online database to insert the new header information annotated with a unique image id and waits for known information to be returned related to the header, for a set of neural network weight files that have been created by training over similar conditions, and for any ancillary data files such as digital elevation data, ground truth, and hydrology maps. In this case, because the location of the image is over land, the set of recognizable features will include vegetation, cloud, and temperature classes, while it will exclude ocean related classes. Once the planner has the combined dynamic information of the header with the static knowledge about the sensor, it begins constructing the sequence or image processing plan.

Once the pixel labelling is completed, the planner must choose the form of browse product as a function of the amount of storage available and the importance of the image, as defined by priority. Ranging from low to high available storage, the browse product can be an image classification vector ICV, a "postage stamp" rendition of the classification map, a low resolution version of the classification map, or a classification map that is the size of the original image. Finally, the planner must ingest the browse product into the appropriate database with the associated header information and the sequence of processing steps used. If it is found later that a particular processing step was inadequate, then the meta database can be searched for all browse products containing that step in order to initiate reprocessing. Likewise, if a scientist, through his own analysis, determines that the classification accuracy was incorrect, then he can submit his changes, as well as methods, to the meta database administrators for update.

3 Conditional Analysis Plans

The automatic generation of plans for image analysis is a challenging problem. Preliminary processing (e.g., removal of sensor artifacts) and analysis (e.g., feature detection) involve a complex set of alternative strategies, depending in some cases on the results of previous processing. For example, detailed location of roads and rivers is only worth doing if there is evidence that those features are present in the image. Plans for image processing need to be *conditional*, in the sense that the course of action to be followed is dependent on the outcome of previous actions.

We have developed a conditional planner that advances the state of the art in several respects, including the use of regression in the generation of conditional plans and a careful treatment of the modelling of observations by permitting the specification of a proposition as true, false, or unknown. We have successfully applied our planner to the generation of conditional plans for image analysis in "EOS world" (named by analogy to the "blocks world"), a planning domain based on data analysis problems related to the Earth Observing System's Data and Information System (EOSDIS).

3.1 Motivation and Background

Classical planning has been criticized for its reliance on a complete model of actions [4]. Constructing an elaborate plan to achieve some set of goals makes little sense if the environment is sufficiently unpredictable that the plan is likely to fail at an early stage. There are several approaches to the problem of generating plans for use in a changing and uncertain world. These fall generally into three classes: making plans more robust in the face of changes in the environment [7], modifying plans as new information becomes available, [2] and conditional planning (more precisely, planning with conditional actions): planning which takes into account the uncertain outcomes of actions.

Conditional action planning is suitable for domains in which there is limited uncertainty and in which plans are constructed at a fairly high level of granularity. Preliminary indications are that planning for image analysis is eminently suitable. Robot planning is probably *not* such an application, unless it can be carried out at a level of abstraction sufficiently high that much of the uncertainty can be ignored.

Peot and Smith [11] have developed a non-linear planner for conditional planning. In conventional, "classical" planning applications, non-linear planning is usually an improvement over linear planning because fewer commitments yields a smaller search space, at a relatively minimal added cost to explore each element of that search space [10]. However, it is not clear that this tradeoff operates in the same way for conditional planners. Furthermore, the operation which is needed to properly construct branching plans — resolving clobberers through conditioning apart — is a very difficult operation to direct. Accordingly, a *linear* conditional planner may be a reasonable alternative.

We have developed a linear conditional planner, based on McDermott's regression planner PEDESTAL [9]. This planner has been implemented in Quintus Prolog, running on Sun SPARCstations. It has been tested on Peot and Smith's "Ski World" sample problem and on the simplified model of the EOSDIS image processing domain described above.

3.2 Action representation

Following McDermott, we represent actions in the plan library in terms of three predicates: `preconditions`, `add` lists and `delete` lists.[3] A precondition entry in the database looks as follows:

`precond(`*action*`,`*preconditions*`)`

This database entry specifies the facts which must hold in order that action be performable. These preconditions are necessary, but may *not* be sufficient for the action to achieve the ends we desire.

[2] "Reactive systems" [3, 1] are yet another approach to this issue, in which it is argued that we are better off not planning at all.

[3] In practice, we are free to use a more convenient notation in composing the plan library than the one the planner will use.

Entries describing the effects of actions look like this:

add(*formula*, *action*, *effect-preconditions*)
or
delete(*formula*, *action*, *effect-preconditions*)

These entries specify that if *action* is performed in a world in which *both effect-preconditions* **and** the preconditions for *action* hold, then *formula* will hold (not hold) at the end of *action*.

Here is a simple action from Peot and Smith's ski world example:

```
precond(go(?x, ?y), [at(?x), clear(?x, ?y)])
add(at(?z), go(?_,?z), [])
delete(at(?z), go(?z,?_), [])
```

We have used the underline as in Prolog, as an "anonymous" or don't-care variable.

We expand this representation to allow for conditional actions, like those of Peot and Smith [11]. Such conditional actions may have several different, mutually exclusive, sets of outcomes. We capture this by associating with every such outcome an integer. Integers can be added to the effect-preconditions of a postcondition entry to specify that one particular outcome must happen in order for the postcondition to hold. For example, in the Ski World, Peot and Smith have an operator for observing road conditions between two points. There are two possible outcomes to this operator: either the road will be found to be clear, or it will be seen to be closed. Here is how we represent this operator:

```
precond(observe(road(?x,?y)),[unknown(clear(?x,?y)),at(?x)])
add(clear(?x,?y),observe(road(?x,?y)),[bead(?act,1)])
postcond(not(clear(?x,?y)),observe(road(?x,?y)),[bead(?act,2)])
```

The variable `?act` is a special one, which will be bound to the name of the step — the actual instance of the operator — so that we may have more than one conditional action of the same type in our plan.

3.3 Pedestal

McDermott's PEDESTAL planner is a regression planner which represents its plan as a dense line segment, beginning at the initial conditions and ending at the goal. Steps are incrementally added to the plan by associating them with points on the line segment. In order to control this process, the planner will always have a set of active (not yet solved) goals and a set of protections which must be respected. PEDESTAL's goals are pairs $\langle g, v \rangle$: the first component, g being a proposition to be established, and the second being a step for whose benefit the proposition is to be established. The top-level goals are goals of the form $\langle g, \texttt{finish} \rangle$ for the distinguished final step.

At each point in the planning process, PEDESTAL will pick a goal out of its active set, and resolve it. PEDESTAL resolves its goals $\langle g, v \rangle$ in one of three ways, chosen nondeterministically:

1. **g holds in initial conditions:** In this case, the goal may be achieved without performing any action. PEDESTAL adds a protection which guards the goal from the beginning of the plan until step v and continues.

2. **g is established by existing step:** Call this step s. PEDESTAL does the following:

   (a) adds a protection of g from s until v.

   (b) PEDESTAL must ensure that s has the desired effect of establishing g. Let $\Sigma(g, s)$ be the *causation preconditions* for g with respect to s. Post new goal(s)[4] $\langle \Sigma(g, s), s \rangle$.

   (c) PEDESTAL must also ensure that no already-existing step between s and v negates g. This is done by posting additional goals:
   For all steps x such that $s < x < v$, let the *preservation preconditions* of g with respect to x be $\Pi(g, x)$. Post $\langle \Pi(g, x), x \rangle$ as a new goal.

3. **g is established by a new step:** Choose some point in the plan at which to insert a new step, s. Now proceed as per a preexisting step to achieve the goal. In addition, however, PEDESTAL must post as goals the preconditions for act s. Let those preconditions be $\Phi(s)$. Post goal(s) $\langle \Phi(s), s \rangle$.

3.4 Conditional Pedestal

PEDESTAL admits of a fairly straightforward adaptation to conditional planning. Essentially, one adapts the PEDESTAL algorithm by mapping steps onto a chronicle *tree*, instead of a line segment. When one adds conditional actions to the plan, one adds new branches to the tree, running from the conditional action to newly-created goal nodes. One then plans for each new goal node as well as the pre-existing goal node.

At each point in the planning process, pick a goal out of the active set, and resolve it. As before, goals are resolved either by finding that the goal holds in the initial conditions, is established by a pre-existing step, or by inserting a new step.

There is one (substantial) complication: handling the addition of conditional actions to the plan. Recall that conditional actions have multiple outcomes. When we add the conditional action to the plan, we will do so because one of the outcomes will achieve a goal. However, there will be other outcomes which will not, in general, achieve the same goal. One may think of these as "bad outcomes" for the action. For each bad outcome, we introduce a new goal node following the bad outcome. Informally, one might think of this goal node as causing us to plan a recovery from the bad outcome.

Consider a problem from the Ski World. One wants to get to a resort (Snowbird or Park City). One's plan so far might be as shown in Figure 2. One has planned

[4] Because we are assuming ground actions, we can blur the distinction between posting a single goal which is a conjunction and posting a conjunction of goals each of which is a literal.

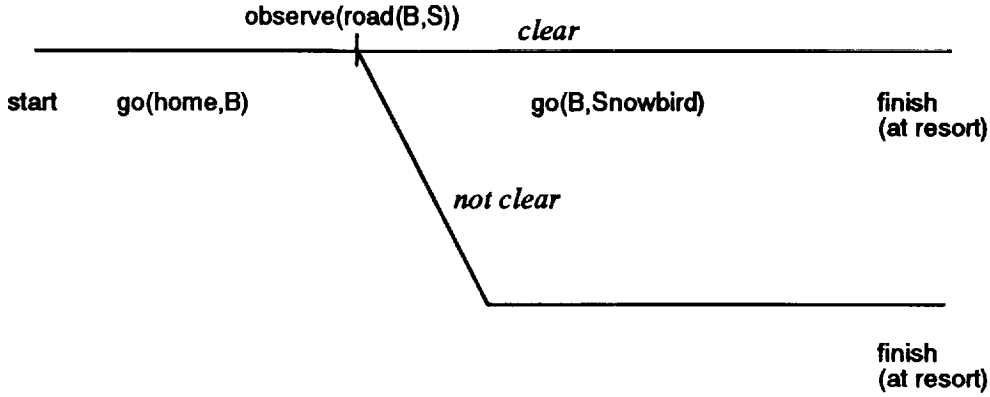


Figure 2: Initial plan to get to the resort.

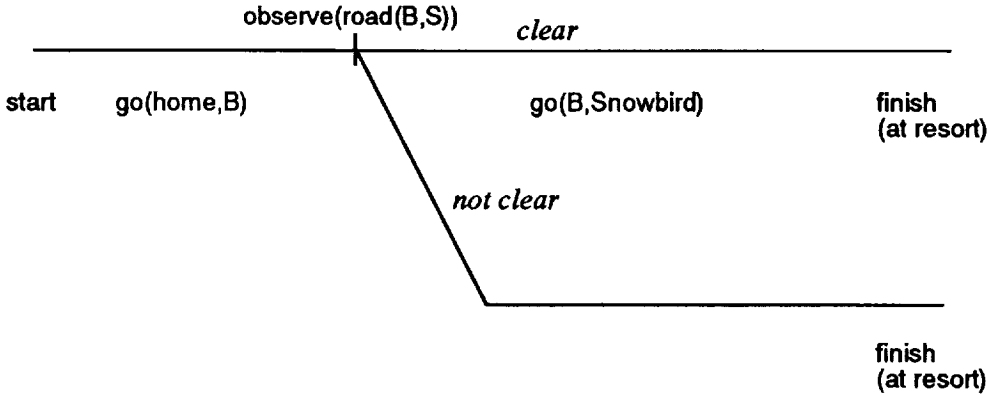


Figure 3: Plan to go to resort after the addition of the conditional action.

to go from home to position B and then from B to Snowbird. However, one has a remaining subgoal, which is to determine that the road from B to Snowbird is clear. Unfortunately, this is not a sure thing. The observation operator has two possible outcomes: either the road will be seen to be clear, or seen to be blocked. In the latter case, one will have to plan a new way to get to the resort. The planner's state after the addition of the observation action is shown in Figure 3. The planner will now have as goals whatever it had before *and* the goal to get to a resort when the road from B to S is not clear, represented by the new goal on the second branch of the plan. Notice that the two plans will share any actions which take place up until the time the status of the road is observed. Notice also that additional actions may be inserted into this shared prefix of the plan: for example, we might as the first step of the plan take some money, if there was a toll on the road from C to Park City. This would only be *necessary* in the event that the road from B to Snowbird is blocked, but would be done before the agent knows whether or not the road is blocked.

3.5 Future work

We are in the process of extending conditional planning to an approach we call *epsilon-safe planning*, in which probabilities are associated with the various outcomes of conditional actions (e.g., with the success or failure of a given classification routine). For any given branch of a conditional plan, we can determine a probability of success. The total probability of success is the sum of the branches which lead to the goal state.

4 Hierarchical Planning with Deadlines

Automated image processing within the *Product Generation System* (PGS) of the Earth Observing System's Data and Information System (EOSDIS) requires the automatic generation of complex analysis plans, detailing the processing steps to be taken to clean up, register, classify, and extract features from a given image. These plans will be executed in a resource-limited environment, competing for such resources as processing time, disk space, and the use of archive servers to retrieve data from long-term mass storage. To complicate matters, it is important that the results of these plans (the completed analysis products) be delivered in a timely fashion to the scientists requesting them.

In joint work at the Honeywell Technology Center (HTC) and NASA's Goddard Space Flight Center, we have developed a planner that generates hierarchical plans for PGS image processing. The schemas used by this planner (based on Nonlin's *Task Formalism* (TF))[13] have been extended to record information about the estimated and worst-case duration of a given task, and about the tasks' resource usage. This information is used during plan construction, for example in the rejection of an otherwise promising expansion for a given sub-task because it requires more time than is available, and in the construction of detailed schedules for image processing tasks.

Accurate estimates of the time required for image processing tasks are hard to come by, particularly for more abstract tasks (e.g., "identify features," rather than a detailed set of file manipulations). We have constructed a routine that traverses the set of tasks defined for the PGS planner, defining worst-case estimates for abstract tasks based on the time required for their subtasks. Use of this routine, coupled with a facility allowing primitive task estimates to be updated either manually or based on statistics gathered as the system runs over time, allows us to continually refine the initially somewhat undependable time estimates, resulting in increasingly effective management of scarce computational resources for the image processing task.

The choice of Nonlin as a starting place was driven by the fact that the TF can be used effectively to describe image processing tasks. Human users tend to break these tasks down into hierarchies of subtasks (e.g., "remove noise" may involve scan-line removal, smoothing, and despeckling, usually in that order) in a way very naturally expressible in TF. Nonlin's main drawbacks included the lack of any facilities for

reasoning about duration, deadlines, and resources. Deviser [14] adds durations, but is not sufficiently flexible and scales poorly. Eventually, the planning function will be integrated with scheduling and dispatch functions that will use the same representations.

Durations and deadlines were added to the TF through the addition of a :duration slot in task schemas. These specifications may be numbers, ranges, or a function of the schema variables evaluated when the schema is instantiated. The underlying representation of time is the TMM [5], in an implementation developed at the Honeywell Technology Center. Calculation of duration bounds during task expansion provides an additional constraint on search: if at any time the time needed for a given task expansion exceeds the time available, the system will backtrack, trying an alternative expansion at the current level or higher planning levels until a time-feasible schedule is found (or the system gives up).

5 Summary and Conclusions

Automating the processing of satellite earth science data is both timely and with a high potential for significant improvement of the current environment. Timely, because the current tools for managing and processing this data are beginning to be overwhelmed. This trend will only worsen as new satellite systems come on line over the next few years, most notably (but not exclusively) EOS. As we have also argued, automation of these tasks shows great potential benefit. Existing research in AI, Operations Research, databases, and distributed systems can be adapted to alleviate the looming data overload, in some cases by freeing humans from the process entirely (e.g., generating browse products on ingest), and in other cases by providing better tools for interactive use (e.g., helping scientists to retrieve and process archived data).

In this paper, we have presented results on the application to image processing of two bodies of work drawn from current research in AI planning: conditional planning and planning with duration and deadlines. These results are promising, but the work is by no means complete. Moving these systems into operational use will require further refinement and development, which we expect to accomplish over the next twelve to eighteen months.

References

[1] Brooks, Rodney A., *A Robust Layered Control System for a Mobile Robot*, A.I. Memo No. 864, MIT Artificial Intelligence Laboratory, 1985.

[2] Campbell, W., Short, N., Jr., Roelofs, L., and Dorfman, E., Using Semantic Data Modeling Techniques to Organize an Object-Oriented Database For Extending the Mass Storage Model, *42nd Congress of the International Astronautical Federation*, 1991.

[3] Chapman, David, *Planning for Conjunctive Goals*, Technical Report AI-TR-802, MIT Artificial Intelligence Laboratory, 1985.

[4] Chapman, David and Agre, Phil, Abstract Reasoning as Emergent from Concrete Activity, Georgeff, Michael P. and Lansky, Amy L., (Eds.), *The 1986 Workshop on Reasoning About Actions and Plans*, Morgan-Kaufman, 1987, 411–424.

[5] Dean, Thomas and McDermott, Drew V., Temporal Data Base Management, *Artificial Intelligence*, **32** (1987) 1–55.

[6] Dozier, J. and Ramapriyan, H.K., Planning for the EOS Data and Information System (EOSDIS), *The Science of Global Environmental Change*, (NATO ASI, 1990).

[7] Firby, R. James, An Investigation in Reactive Planning in Complex Domains, *Proceedings AAAI-87, Seattle, Washington*, AAAI, 1987, 196–201.

[8] Green, J., The New Space and Earth Science Information Systems at NASA's Archive, *Government Information Quarterly*, **7**(2) (1990) 141–7.

[9] McDermott, Drew, Regression Planning, *International Journal of Intelligent Systems*, **6**(4) (1991) 357–416.

[10] Minton, Steven, Bresina, John, and Drummond, Mark, Commitment Strategies in Planning: A Comparative Analysis, *Proceedings IJCAI 12, Sydney, Australia*, IJCAI, 1991.

[11] Peot, Mark A. and Smith, David E., Conditional Nonlinear Planning, *Artificial Intelligence Planning Systems:Proceedings of the First International Conference*, Los Altos, CA, 1992, 189–197, Morgan Kaufmann Publishers, Inc.

[12] Ramapriyan, H.K., The EOS Data and Information System, *American Institute of Aeronautics and Astronautics*, 1990.

[13] Tate, Austin, Generating Project Networks, *Proceedings IJCAI 5, Cambridge, MA*, IJCAI, 1977.

[14] Vere, Steven, Planning in Time: Windows and Durations for Activities and Goals, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5** (1983) 246–267.

N94- 35046

RAVE: Rapid Visualization Environment

D. M. Klumpar
Lockheed Space Sciences Laboratory
3251 Hanover Street
Palo Alto, CA 94304
klump@agena.space.lockheed.com

Kevin Anderson and Evangelos Simoudis
Lockheed AI Center
3251 Hanover Street
Palo Alto, CA 94304
{kevin, simoudis}@aic.lockheed.com

Abstract

Visualization is used in the process of analyzing large, multidimensional data sets. However, the selection and creation of visualizations that are appropriate for the characteristics of a particular data set and the satisfaction of the analyst's goals is difficult. This process consists of three tasks: generate, test, and refine, that are performed iteratively. The performance of these tasks requires the utilization of several types of domain knowledge that data analysts do not often have. Existing visualization systems and frameworks do not adequately support the performance of these tasks. In this paper we present the RApid Visualization Environment (*RAVE*), a knowledge-based system that interfaces with commercial visualization frameworks and assists a data analyst in quickly and easily generating, testing, and refining visualizations. *RAVE* has been used for the visualization of *in situ* measurement data captured by spacecraft.

1. Introduction

Large volumes of multidimensional data are routinely collected in fields that range from space physics to retail marketing. Information is extracted from the collected data through visualization techniques, as well as a variety of analysis methods, e.g., statistical, whose results are comprehended also using visualizations. The creation of useful visualizations is a knowledge intensive task that is usually performed as a **generate and test** process. The analyst must know (1) how to visualize the data set he is analyzing, (2) what visualization package to use to realize the selected visualization(s), and (3) whether the data set will need to be transformed before the selected package can generate the chosen visualization. Very few analysts possess all the necessary types of knowledge. As a result, the generate and test process takes a long time to perform. Existing data visualization systems and frameworks suffer from two limitations. Either they are not easily extensible implying that the generate and test process cannot be performed, or they are hard to use. In this paper we present the RApid Visualization Environment (*RAVE*), a knowledge-based system that interfaces with commercial visualization frameworks and assists a data analyst in quickly and easily generating, testing, and refining visualizations. *RAVE* generates visualizations that satisfy a set of analysis and display goals stated by its user. The system has been used by space scientists for the visualization of *in situ* measurement data captured by spacecraft. It currently interfaces with the PV-Wave and AVS visualization frameworks.

Once the user selects a goal, *RAVE* automatically identifies a set of visualizations that can be used to achieve the goal. The user selects one or more visualizations from this set, and *RAVE* automatically creates and executes the appropriate program to generate each of the selected visualizations. The program is implemented in the language of the visualization frameworks that are interfaced to *RAVE*. The user can compare the effectiveness of the created visualizations with regards to the amount of information that can be extracted and the way the information is presented. In this way, the analyst can quickly and easily select a set of target visualizations which can then be further refined.

2. Data Visualization

Data is collected and analyzed to create models that predict the future behavior of a system, or explain an observed event. For example, a space scientist may try to explain whether the earth's magnetosheath contain solar wind plasma or just noise. Data analysis gives rise to goals that must be achieved. Visualization can be used to achieve these goals. For example, an analyst may create a scatter plot of temperature versus density for protons in an attempt to explain the existence of solar wind in earth's magnetosheath.

Visualization of a data set to achieve a goal implies that the analyst must be able to (1) decide how to visualize a data set, (2) create the decided upon visualization, and (3) assess the created visualization's effectiveness in presenting the information contained in the data set and thus achieving the stated goal. The selection and creation operations can be viewed together as a generate operation. Therefore, visualization in this class of domains can be framed as a **generate and test** process.

Deciding how to visualize the data implies that the analyst understands the benefits of using each visualization, and knows how to map from a space of analysis goals to a space of visualizations. Oftentimes these mappings are one-to-many further complicating the visualization-selection task. Creating a visualization, and later refining it through the addition of features such as color, implies that the analyst must have knowledge about computer graphics and programming. For example, the analyst must know how to write a program to generate a scatter plot, and then how to use color to display the data points whose values are greater than some threshold. Existing visualization languages that are included in frameworks such as AVS, PV-WAVE facilitate the programming task but have steep learning curves. The knowledge included in the *RAVE* system (1) supports the data analyst during the visualization selection operation, and (2) enables the automatic creation of the corresponding programs in the visualization languages of the frameworks to which *RAVE* is interfaced.

3. Application Domain

RAVE has been used for the visualization of space sciences data from *in situ*

measurements of plasmas, fields, and corpuscular radiation. For example a set of instruments for *in situ* measurements may capture the DC magnetic field (a three-component field vector), the DC electric field (also a three component field vector), the flux of particles as a function of their energy of arrival, charge, and mass composition, and the AC electric and magnetic wave field spectra (energy density vs. frequency). We have experimented with a subset of observations from the Hot Plasma Composition Experiment (HPCE) on the AMPTE/CCE spacecraft.

The Charge Composition Explorer (CCE) satellite of the AMPTE program was launched in August, 1984 into a near equatorial orbit (inclination 4.8° with apogee of 8.8 R_e and perigee 1108 km). The CCE was spin-stabilized at 10 rpm with its spin axis pointing approximately toward the sun. The spacecraft carried instrumentation to measure composition and charge state of ions over a very broad energy range, electrons, plasma waves, and the geomagnetic field. The data used here from the HPCE was taken by a set of eight magnetic electron spectrometers that measure the flux of electrons from 50 eV to 25 keV [Shelley et al., 1985]. Each spectrometer is operated at a fixed energy with an energy resolution of 50%. The eight instruments are co-aligned with their fields-of-view perpendicular to the spacecraft spin axis and are operated simultaneously, making measurements in unison every 155 msec. Thus an eight point electron energy spectrum is obtained every 9.5° of satellite rotation. The spectrometers are collimated with a 5° full width conical field-of-view giving an effective full width field-of-view of 5° x 14.5° during each measurement period. As the spacecraft rotates the view directions sweep through a range of pitch angles the amplitude of which depends on the direction of **B** with respect to the spacecraft spin axis. The full pitch angle scan (0°-180°) is achieved when **B** is perpendicular to the spin axis.

The captured measurements may be viewed as a succession of time slices through energy space. Each time slice contains eight measures of the instantaneous electron flux (at the eight energies) and each successive slice is displaced by 9.5° of rotation in the spin plane of the satellite. All captured parameters vary in time and space with rates-of-change that are highly variable as (1) the environment responds to externally applied forces, and (2) the spacecraft moves between different plasma regimes. In addition to the data captured from the HPCE sensors, our data set also contained data that was derived by combining data from multiple sensors. In this way we were able to represent global characteristics of the plasma, e.g., plasma beta, ratio of plasma frequency to gyro frequency, etc.

4. *RAVE*: System Description

RAVE is a knowledge-based system that supports a data analyst in generating, testing, and refining visualizations of a set of relational data. It is layered on top of existing visualization frameworks, e.g., AVS, PV-WAVE, etc., capitalizing on their data representation and rendering strengths, while improving their ease of use. *RAVE* consists of: a graphical user interface, a knowledge base, an inference engine, a facts

database, a visualization program generator, and interfaces to the visualization frameworks to which it is connected. The system's architecture is show in Figure 1. *RAVE*'s graphical user interface is implemented using DevGuide under Open Look, the interfaces to the visualization frameworks are implemented in C, whereas the rest of the system is implemented in Common Lisp using the Common Lisp Object System.

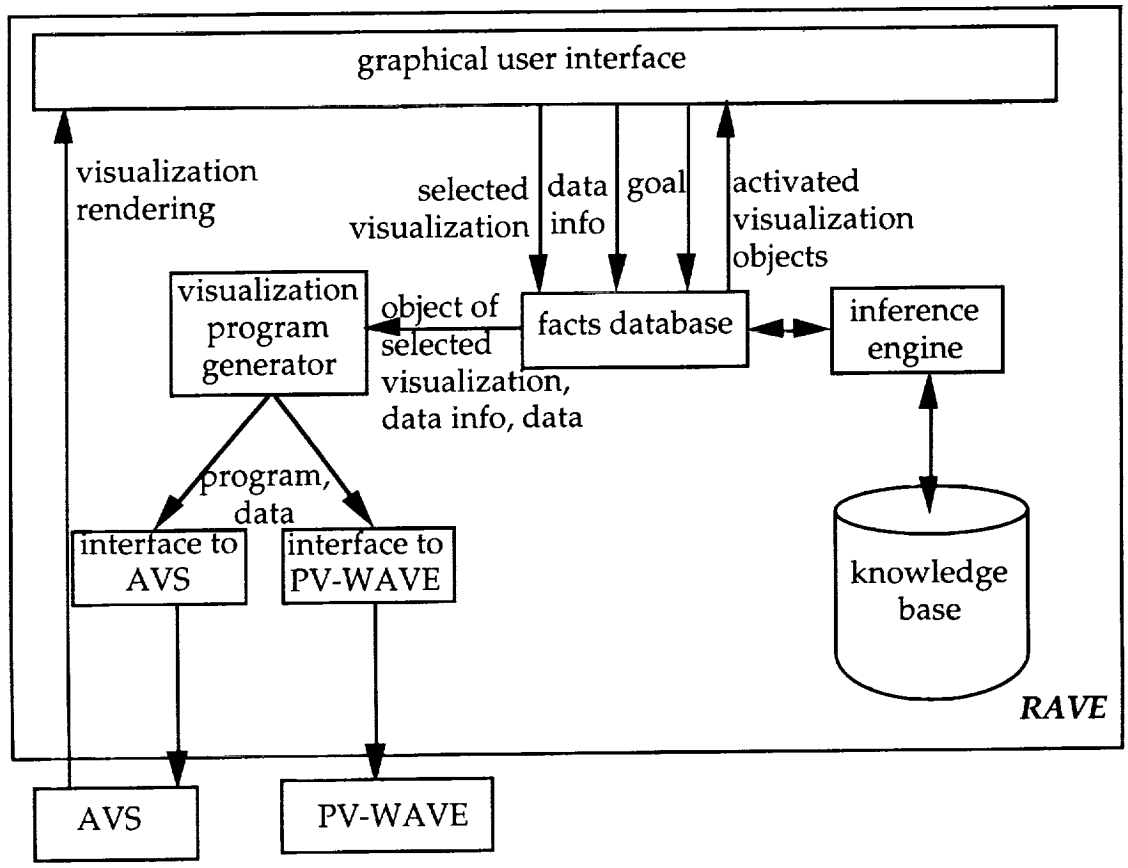


Figure 1: The architecture of the *RAVE* system

RAVE's knowledge base contains (1) a set of visualization objects, and (2) a set of rules that relate visualization goals to visualization objects. Each method for visualizing a data set is represented by a separate object. An object has slots for: (1) the name of the visualization, (2) the goals the corresponding visualization can satisfy, (3) the refinements the visualization can accept, (4) the domain(s) in which it can be used, and (5) the program that implements it in the language of one of the visualization

frameworks with which *RAVE* interacts. For example, the visualization object that corresponds to the two-dimensional scatter plot can satisfy the goal "attribute x is related to attribute y," it can accept zooming and color as refinements, and can be applied in any domain where numeric-valued attributes are compared.

Occasionally, a data set may need to be transformed, e.g., to a different coordinate system, before the visualization can be rendered in a framework. The visualization-generation program included in an object includes the necessary transformations that must performed, and executes them automatically as part of the program-creation process. If a particular visualization can be rendered by more than one of the frameworks with which *RAVE* is interfaced, then a visualization object can include a separate visualization-generation program, along with the necessary transformations, for each of the appropriate frameworks. The user can select the framework(s) where the data will be displayed. We plan to expand the types of knowledge that can be represented in a visualization object by making explicit visualization-selection criteria such as cost of displaying a visualization, type of display device needed for displaying effectively a selected visualization, etc. In this way, not only we will be able to further assist the analyst's selections, but we will also be able to capture the rationale behind each selection. Finally, this knowledge will also be accessible by the rules and will enhance the *RAVE*'s mapping capabilities.

A set of visualization objects that are appropriate for the data of a particular application domain can be organized into a collection. For example, objects that correspond to visualizations used with financial data are organized into a collection. Such collections can be organized hierarchically with the top-level collection (node) containing the set of general visualizations, e.g., scatter plots, line plots, etc.

The antecedents of each rule represent (1) the statement of a goal, and (2) the constraints that need be satisfied before the goal can be achieved. The rule's consequents select the visualizations, i.e., instantiate the visualization objects, that can satisfy the goal. The rules make explicit: (1) what constitutes an appropriate visualization for a data set and a goal, (2) how to develop knowledge that can be shared across several domains, and (3) how to choose a particular visualization. A set of rules can be associated with a collection of visualization objects, e.g., the objects that correspond to visualizations that pertain to analysis of financial data. Two example rules are shown in Figure 2. Variables in these rules are preceded by question marks. Each variable, e.g., ?x, is bound to an object that contains information about each attribute in the data set to be visualized. The information includes: the type of the attribute's values, e.g., integer, real, nominal, etc., the number of distinct values included in the set to be visualized, minimum and maximum values (for numeric-valued attributes), the type of data, e.g., time-series, a pointer to the actual values, etc.

```
(if (and (goal (related-to ?x ?y))
         (equal (value-type ?x) 'numeric)
         (equal (value-type ?y) 'numeric))
 then (activate 2-d-scatter-plot ?x ?y))

(if (and (goal (value-distribution-of ?x))
         (<= (number-of-distinct-values ?x) 5))
 then ((activate 2-d-bar-graph ?x) (activate 2-d-pie-chart ?x) (activate 3-d-pie-chart ?x)))
```

Figure 2: Examples of rules used by *RAVE*

The first rule states that the visualization object corresponding to the two-dimensional scatter plot is enabled for selection by the user if the stated goal seeks to establish whether attribute x is related to attribute y, and the values of both attributes x and y are numeric. The second rule states that the visualization objects corresponding to the two-dimensional bar graph, the two-dimensional pie chart, and the three-dimensional pie chart are enabled for selection by the analyst if the stated goal seeks to identify the distribution of the values of a particular attribute and the number of distinct values this attribute takes in the selected data set is not greater than five.

The user interface initially allows the analyst to: (1) select a data set to be analyzed, (2) specify any special characteristics of the selected data, e.g., time-series data, and (3) choose a visualization goal from a menu of pre-specified goals. The selected data, i.e., attributes and values, is preprocessed so that information such as minimum and maximum values can be established. The data and the resulting metadata are organized into an object that is asserted in *RAVE*'s facts database. The selected goal is asserted as a separate fact in the same database. The inference engine executes the rules whose antecedents match facts in the database. As a result of executing the matching rules, one or more visualizations may be activated. The names of the activated visualizations are displayed to the analyst through *RAVE*'s interface. The analyst can select one or more of the activated visualizations. A portion of *RAVE*'s user interface is shown in Figure 3.

As was stated above, if a selected visualization can be displayed by more than one of the frameworks that are connected to *RAVE*, the analyst must select the frameworks that will be used. The program(s) included with the object that corresponds to each selected visualization is subsequently sent to *RAVE*'s visualization program generator which instantiates it for the specified data set and attributes. The instantiated program is then sent, through the appropriate interface, to the corresponding visualization framework for rendering. Each selected visualization is rendered in a different window. The analyst is given control, through the user interface, of simultaneously displaying all the created visualizations. In this way, the analyst is able to compare the information-extraction and presentation effectiveness of each selected visualization, and either

choose the best one(s), refine one or more of the displayed visualizations, or discard them and modify his initial selections so that new visualization objects may be enabled.

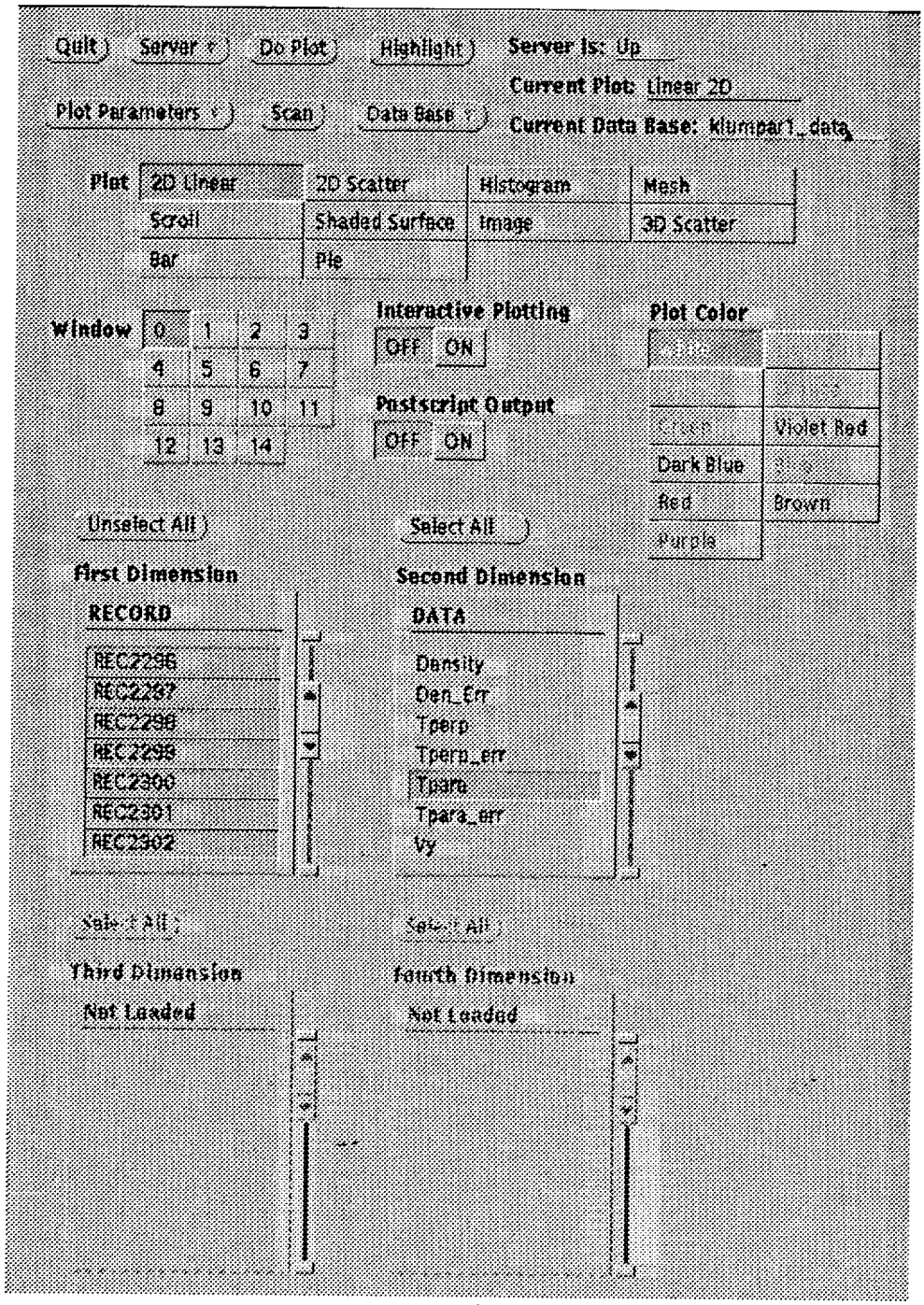


Figure 3: A portion of *RAVE*'s user interface

Should, after seeing the results of a selected visualization, the analyst decide to refine it, he can select one or more of the refinements supported by the particular visualization. The selected refinements are reflected on the instance of the visualization object that is asserted on *RAVE*'s facts database. The updated object is communicated to the program generator which creates an instance of the updated program(s) corresponding to the visualization. This new instance is similarly sent to the appropriate visualization framework for rendering. Figure 4 shows a two-dimensional scatter plot that has been refined by enabling the zooming and coloring capabilities.

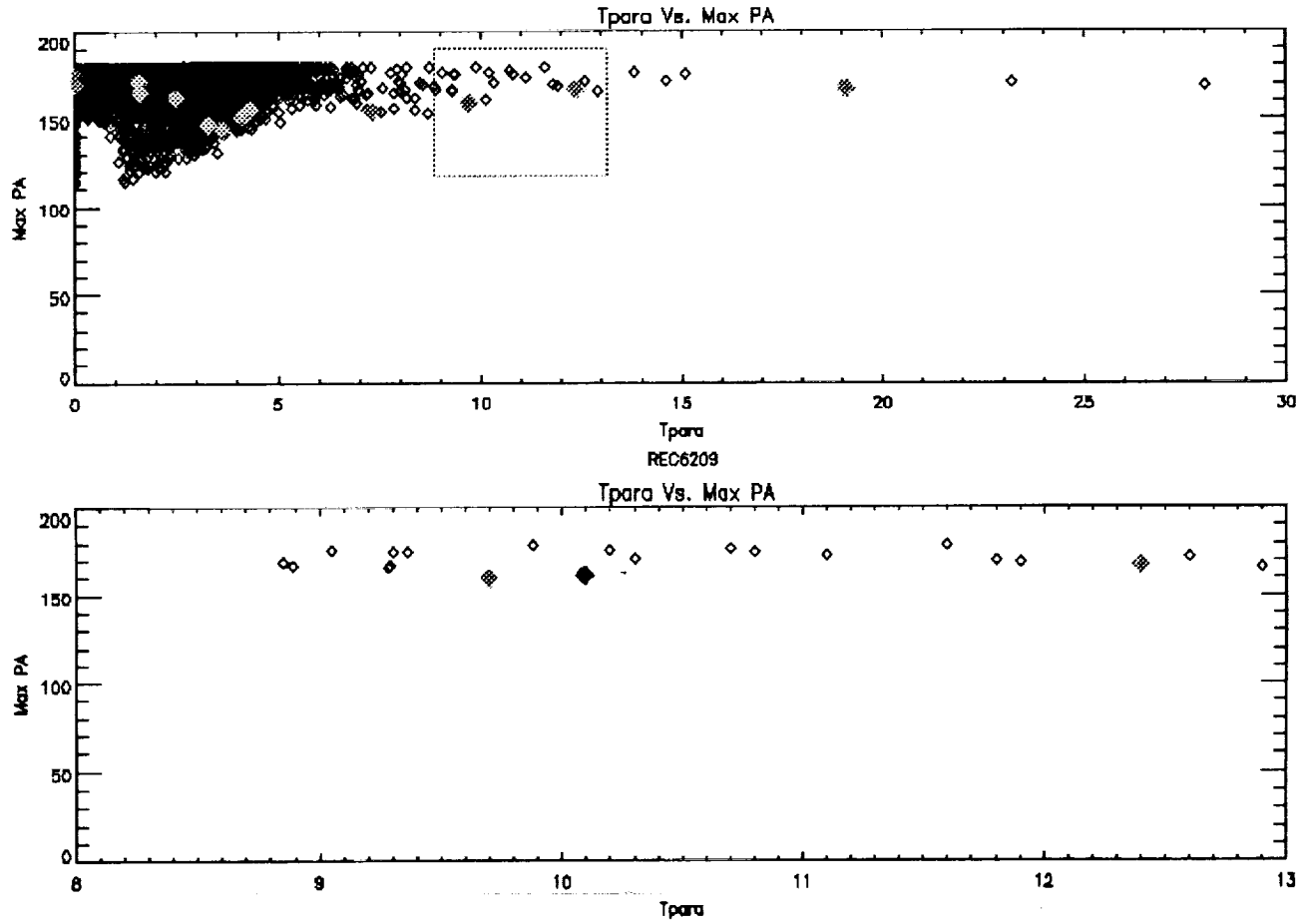


Figure 4: A two-dimensional scatter plot produced by *RAVE*

6. Related Work

We compare *RAVE* to two sets of systems. The first set consists of systems that use artificial intelligence techniques to facilitate the creation of complex user interface. In particular, we examine the Integrated Interfaces system [Arens *et al.* 91], and the APT

system [Mackinlay 91]. The second set consists of two visualization frameworks that have been developed for space sciences data: LinkWinds [Jacobson & Berkin 93], and SAVS [Szuszczewicz *et al.* 92].

Integrated Interfaces is a knowledge-based system whose goal is to alleviate the work of user interface designers and developers. For this reason the system uses knowledge that allows it to automatically select, at run time, the most appropriate way for presenting information. Integrated Interfaces supports output in natural language, text, maps and other graphics, tables, and forms. Its knowledge base represents domain knowledge, and knowledge about user interfaces. In contrast, *RAVE* concentrates on output that can be presented through graphics. Its knowledge base represents deeper and more detailed knowledge both about graphics and the particular domains where the system is applied. Furthermore, through its ability to display several different visualizations that can satisfy a particular goal, *RAVE* provides its user with the ability to perform guided exploration of the visualization space by comparing the effectiveness of several visualizations in extracting the most possible information from the contents of the data set.

The goal of the APT system is to dynamically create effective visualizations of relational information by searching a space of possible designs. In this respect, it is a tool for performing open-ended exploration of the visualization space. It concentrates on bar charts, scatter plots and connected graphs. APT has the ability to decompose a data set into components that can be visualized individually, selecting a visualization appropriate for each component and then composing from the individual pieces the visualization for the overall data set. As was mentioned above, *RAVE* provides less support for a guided rather than an open-ended exploration of the visualization space. It assumes that the user will partition a data set, should the need arise. However, it supports more complex domain-specific and domain-independent visualizations of relational information than the APT system. Finally, *RAVE* is able to interface with existing visualization frameworks making it an easily extensible system able to render complex domain-specific visualizations, in addition to the generic simple ones.

The LinkWinds system provides an environment for rapidly prototyping and executing visualization applications on planetary data. It allows its user to analyze data and creation visualizations through a spreadsheet interface. *RAVE*, in contrast, provides knowledge-based support for selecting appropriate visualizations and automates the creation of each selected visualization template.

The SAVS tool provides data acquisition, manipulation, and visualization capabilities. The system is being applied on solar-terrestrial and planetary data. It is implemented on top of the AVS visualization environment. SAVS does not provide knowledge-based support during the visualization-creation process. In addition, the visualizations it supports are those that can be supported by AVS. In addition to the knowledge-based support it provides, *RAVE* can be interfaced with a variety of visualization languages,

including AVS. As such it can support a wider range of visualizations than SAVS.

7. Conclusions

We continue developing the *RAVE* system expanding its knowledge base with more visualizations that are appropriate for space sciences data, and interfacing it to other visualizations frameworks, such as SGI Explorer, that have desired features. In addition, we are in the process of interfacing *RAVE* to the Recon data mining system [Simoudis *et al* 93] that provides sophisticated data management and analysis capabilities.

The process of generating, testing, and refining visualizations is particularly well-suited for the space sciences domain, as well as for other domains where graphics are used for extracting information from data but where the identification of the appropriate visualizations is not easy. *RAVE* provides knowledge-based selection of visualizations that are appropriate for achieving analysis and data display goals. Once visualizations are selected, *RAVE* can automatically create and execute programs that generate the visualizations. In the process, it automatically transforms the target data set to satisfy constraints imposed by the visualization framework in which the data will be displayed. These capabilities allow the analyst to perform a guided exploration of domain-specific and generic visualization spaces, while concentrating on information extraction.

References

Arens, Y., Miller, L, Sondheimer, N., "Presentation Design Using an Integrated Knowledge Base," Intelligent User Interfaces, Sullivan, J.W., Tyler, S.W. eds., pp. 241-258, ACM Press, 1991.

Jacobson, A. S. and A. L. Berkin, LINKWINDS, A system for Interactive Scientific Data Analysis and Visualization, High Performance Computing 1993, Adrian Tentner ed., The Society for Computer Simulation, 1993.

Mackinlay, J.D., "Search Architectures for the Automatic Design of Graphical Presentations," Intelligent User Interfaces, Sullivan, J.W., Tyler, S.W. eds., pp. 281-292, ACM Press, 1991.

Shelley, E. G., A. Ghielmetti, E. Hertzberg, S. J. Battel, K. Altwegg-Von Burg, and H. Balsiger, The AMPTE CCE Hot Plasma Composition Experiment (HPCE), IEEE Trans on Geoscience and Remote Sensing, GE-23, 241, 1985.

Simoudis, E., Kerber, R., Livezey, B., "Recon: A database mining framework," submitted to the International Journal of Intelligent Information Systems.

Szuszczewicz, E. P., Mankofsky, Alan, and Charles C, Goodrich, SAVS: A Space Analysis and Visualization System, Proceedings of the 2nd NASA AISRP Workshop, 1992.

Planning, Scheduling, and Robotics

On-Board Emergent Scheduling of Autonomous Spacecraft Payload Operations

Craig A. Lindley

CSIRO Division of Information Technology
Locked Bag 17, North Ryde NSW Australia 2113

Abstract

This paper describes a behavioural competency level concerned with emergent scheduling of spacecraft payload operations. The level is part of a multi-level subsumption architecture model for autonomous spacecraft, and functions as an action selection system for processing spacecraft commands that can be considered as "plans-as-communication". Several versions of the selection mechanism are described and their robustness is qualitatively compared.

Keywords: Spacecraft Control, Emergent Scheduling, Behavioural AI.

Introduction

This paper describes an autonomous control architecture for scheduling payload and user service operations of a low Earth orbiting microsatellite, AUSTRALIS-1. The control architecture is based upon a behavioural paradigm of artificial intelligence (see Maes, 1993). Previous work has defined a layered competency model for autonomous orbital spacecraft (Lindley, 1993a), and the detailed design of the level 1 competency for maintaining the spacecraft battery condition (involving power system fault detection, redundant unit switching, charge control, and discharge control; Lindley, 1993b). Planning and Scheduling of Data Acquisition and Transmission is a much higher level competency (level 7), representing very atypical functions for behavioural systems (that have generally addressed low-level robot motion control and guidance functions). Emphasizing the interface between the autonomous system and its human users, the level 7 competency takes into account specifically requested data objects, user requested instrument parameters, and the need to downlink particular data sets at particular times or locations. These needs are catered for while considering the battery charge requirements of Level 1. A *plan* in this system consists primarily of the current set of requests for user operations, generated by users and uplinked to the spacecraft. Hence "a plan" is a communicable list of representations of goals. The term "planning" can be used to refer to the generation of goal lists, or can be used more loosely to refer to the generation of activity that satisfies a set of current goals. A *schedule* is taken to be an association of specific activities with particular times, and hence is a more detailed form of planning in the loose sense. By these definitions, planning and scheduling can be emergent phenomena in that they do not need to involve the generation of action representations. A goal selection algorithm is described that achieves emergent planning and scheduling by choosing a particular represented goal as the basis of action generation with each control cycle of the system. Several versions of the algorithm are described, and their relative merits are qualitatively discussed.

Precedents for Autonomous Planning in Space Systems

Command languages for spacecraft can be viewed as languages for expressing plans that are uploaded to and then obeyed by a spacecraft. Increasing abstraction levels in spacecraft command languages (Pidgeon et al, 1992), or reducing command detail, requires greater spacecraft "intelligence" in the form of the autonomous planning ability needed to elaborate the details of uplinked plans to the appropriate level for the control of spacecraft hardware. For example, in normal operations, the Hipparcos spacecraft is controlled by processed commands that are sent to the onboard computer for distribution to other systems, and for possible time tagging. The ERS-1 spacecraft, which is in a low polar orbit with limited ground access, has a similar command macro system, with four command types providing different functions and levels of authority. The EURECA system, comprising fifteen separate payloads, uses an

onboard Master Schedule that contains a list of time tagged command macros for execution by the onboard data handling system. Further increasing the abstraction level of commands, or further decreasing their level of detail, involves the incorporation of more sophisticated techniques for autonomous planning and scheduling, drawn from research in artificial intelligence (AI).

Representational AI systems, or Knowledge-Based systems (Maes, 1993), use symbolic models of the robot, its operating environment, and the range of tasks and actions that the robot "knows how to perform". System behaviours are typically produced by an inference engine that reasons about the models to produce action plans. Plans and action representations are frequently hierarchically ordered, with high level representations of goals and actions at the top of the hierarchy. Subsequent levels of abstraction decompose goals into increasingly specific subgoals, ending with primitive machine commands that can be executed directly by hardware. Robotic plans are monitored during execution. A robot may attempt to deal with plan failures by replanning from some suitable level of abstraction, with various techniques being employed to deal with constraints of real-time operation. Brooks (1986) has described this traditional, representational approach as the sense/model/plan/act approach. Representational approaches (reviewed in Georgeff, 1987) have made substantial progress in addressing the requirements of many applications, but frequently founder in the inability of system designers to provide sufficient knowledge for the autonomous performance of useful tasks in real environments.

Most space applications of AI to date have concentrated upon the areas of mission planning, sequencing, and control. There have been numerous projects that have addressed the automation of ground-based planning and control of space systems, with many systems successfully prototyped, and a number now in routine operational use (Drabble,1991). These ground-based systems have been developed within the traditional, representational AI paradigm. Autonomous orbital spacecraft, surface exploration robots, and dextrous free-flying robots have particular requirements for producing appropriate behaviour quickly in the face of dynamic and uncertain circumstances. These issues have been addressed using techniques representing a convergence towards some of the techniques involved in behavioural artificial intelligence. Fast response times have been achieved by separating the control architecture into parallel functions, with a combination of deliberative planning and reactive planning being used to achieve goals at strategic and tactical levels, respectively (eg. Rokey and Grenander, 1990, and Erickson et al, 1989). Autonomous orbital spacecraft control models have adopted limited parallel and distributed processing models, but have still tended to rely upon world modelling as the basis of their intelligence (eg. Raslavicius et al, 1989).

Hierarchical approaches to action representation and control have dominated the field of spacecraft automation, and have been incorporated as fundamental structuring principles in proposed reference models for space automation and robotics developed both by NASA and ESA (Elfving and Kirchoff, 1991). The "Theory of Intelligent Machines" proposed by Valavanis and Saridis (1992) associates hierarchy with proposed fundamental analytical measures of intelligence, and the Rensselaer Polytechnic Institute Center for Intelligent Robotic Systems for Space Exploration has constructed a testbed that has the development and implementation of this theory as one of its primary purposes (Watson et al, 1992).

Behavioural theories of artificial intelligence suggest a very different approach to autonomous action generation and control, requiring new reference models for robotic systems (Lindley, 1993a) and new analytical formulations that do not intrinsically depend upon a hierarchy of functional or control flow (Lindley, 1993c). Maes (1993) has described the following characteristics of behavioural systems:

- systems have multiple, integrated, and typically low-level competences
- the system is "open" or "situated" in its environment, having many interactive interfaces with a complex, dynamic, and unpredictable world
- the emphasis is upon autonomy
- systems are designed to produce behaviour, rather than to have knowledge in a

representational sense, thus avoiding the *circumspection problem* of providing sufficiently comprehensive world models for the tasks at hand
- particular functions may emerge from the activity of more primitive behaviours
- there is a strong emphasis upon adaptation

Lindley (1994) has argued that behavioural approaches are justified by philosophical positions that regard represented knowledge as a cultural, discursive, and pragmatic artefact, rather than being the substance of intelligence; in effect, knowledge is reified practice, codified by representation in order to coordinate behaviour. From this viewpoint, *action*, rather than representation, is at the core of intelligence. Behavioural theories constitute a new and vigorous paradigm that has shown superior performance to representational methods in the control of simple robotic functions (Brooks, 1991).

Planning and Behavioural Action Selection

Agre and Chapman (1990) attribute problems with the representation-centred view of "plans-as-programs" to a mistaken notion of what activity is like, and the role plans can play in activity. The plan-as-program view regards activity as a matter of problem-solving and control. The world presents an agent with a series of formally defined problems that require solutions, and a planner produces the solutions. The executive then "implements these solutions by trying to make the world conform to them". Agre and Chapman hold that this view is incorrect, since "Acting in the world is an ongoing process conducted in an evolving web of opportunities to engage in various activities and contingencies that arise in the course of doing so". Hence an agent can be viewed as participating in the flow of events, rather than solving problems.

This view leads to the notion of *plans-as-communication* as part of a general theory of situated activity. Plans-as-communication have a reduced role, with the need for improvisation on the part of the user. A plan is used as a resource among other resources in a process in which an agent engaged in rational, goal-directed activity continually reevaluates what to do. Plan use is a matter of figuring out how to make a plan relevant to a situation at hand. The meaning of the plan is heavily dependent upon the context, and must be constantly reassessed. Plans-as-communication have a fuzzy boundary with other forms of communication, such as lists, schedules, images, and mnemonics. The current context of an agent is a resource that can be used to intepret the plan. Agre and Chapman present the example of a person following a plan which has the form of another person's instructions for how to reach a particular location. Lessons generalised from the example are:

- the list of shared understandings is innumerably long
- all plans depend on shared understandings in this way
- action in the real world is sufficiently difficult to suggest that plans must depend on innumerable shared understandings in order to be expressible
- these points apply regardless of whether the plan's maker and user are the same agent or different agents

Attempts to capture the meaning of plans by representation must therefore encounter the circumspection problem of needing to represent an unspecifiable amount of contextual knowledge. Agre and Chapman suggest that the plans-as-communications view is more plausible than plans-as-programs as an account of human plan creation and use, and may provide a more effective model for artificial autonomous agents, although its tendency to appear like the general problem of automated natural language comprehension makes it impractical for the design of autonomous agents in the short to medium term. However, this is overly pessimistic, since the ability to understand (ie. use) limited forms of plans for particular purposes in a specific domain can be a much simpler competency than the general-purpose natural language skills of human beings.

An agent capable of processing plans-as-communication can be designed using a behavioural goal selection system for action generation without deliberative reasoning. Maes (1990) suggests that the action selection mechanism of an autonomous agent in a complex

dynamic environment should be reactive and fast, favouring actions relevant to the current situation, exploiting opportunities, and adapting to unpredictable and changing situations. The mechanism must favour actions that contribute to any current ongoing goal or plan, and should look ahead, especially to avoid hazardous situations and to handle interacting and conflicting goals. The action selection mechanism must also be capable of operating with minimal, incomplete or incorrect world knowledge, with limited computational and time resources, and be robust (degrading gracefully when components fail).

A goal selection mechanism proposed by Maes (1990), that appears to have these desirable characteristics, views an autonomous agent as a set of modules each having its own specific competence. These modules resemble the operators of a classical deliberative planner. A competence module has a list of preconditions to be fulfilled before the module can become active, lists of expected effects of the module's action in terms of an add list and a delete list, and a level of activation for the module. If all the preconditions of a module are true at a particular time, then the module is executable at that time. Competence modules are linked to form a semantic network, with activating and inhibiting links between modules allowing the activation energy to accumulate in the modules that represent the "best" actions to take in the current situation and given the current set of goals. The pattern of spreading activation among modules, and the input of new activation energy into the network, are determined by the current situation and the current global goals of the agent. Activation iterates through cycles, thereby accumulating until a threshold is exceeded and an action is performed. A decay function ensures that the overall activation level remains constant through continuous iterations.

Initial results with this approach are reported to be very encouraging, with networks exhibiting planning behaviour. Plans are not explicitly represented, but the "intention" of an agent to carry out certain actions is expressed by high activation levels of the corresponding modules. Action selection is non-hierarchical and highly distributed. Global parameters serve as controls for mediating smoothly between different action selection characteristics, and therefore between adaptivity, speed, and reactivity on the one hand and "thoughtfulness" and certainty on the other. The approach is computationally much less expensive that search-based planning. Different paths are evaluated in parallel, and the system does not start from scratch when a particular path fails to produce a solution, nor "replan" at each timestep. The action selection mechanism is capable of processing complex goal interrelationships, and complex, possibly hierarchically organised goal substructures, although particular applications may have very simple, flat goal representations with few interconnections.

AUSTRALIS-1 Spacecraft Operations

The AUSTRALIS-1 spacecraft is currently being designed by an informal consortium of Australian universities. The baseline spacecraft is a 35 cm cube, having an expected mass of less than fifty kilograms. It is intended to operate within an altitude range from five hundred to one thousand kilometres. The spacecraft will carry a near infra-red CCD-based camera system, and telecommunications equipment to support a data store-and-forward communications service. AUSTRALIS-1 has particular requirements for autonomy derived from the need to serve a large number of users over a highly dispersed area, using very cheap and simple ground equipment with minimal centralised control or coordination.

The spacecraft subsystems most directly associated with supporting the level 7 user operational model are shown on Figure 1. The controller initiates image acquisition operations and controls downlinking of images and data units via the CMS (Command Management System) database. Hence, the level 7 control network functions at what will here be called a *transaction* level; that is, it is not concerned with detailed data transfer and timing control between devices, but with control of the transfer of complete data units.

The flow of user data between components does not pass through the level 7 controller, although operation selection and initiation is controlled by this level. The level 7 controller receives inputs from the relevant subsystem components, from the guidance and navigation system, and from the

level 1 competency concerned with battery conditioning, charge control, discharge control, and power control. The AUSTRALIS-1 power control system is described in detail in Lindley (1993b).

AUSTRALIS-1 users will be able to uplink data files (by *Store Data* command), request data broadcast (by *Broadcast Data* command), request image acquisition (by *Acquire Image* command), and request data deletion (by *Delete Data* command). The commands are stored on board the spacecraft, and function as goals within the payload planning and scheduling competency of the autonomous control system. In the data store and forward mode, ground stations can uplink a request for the spacecraft to broadcast data immediately, or broadcast to a distant ground station specified in terms of a latitude and longitude or a time. During image acquisition operations, the spacecraft receives an image acquisition request from a user, schedules and acquires the image, and then treats the image data in the same way that user data packets are treated.

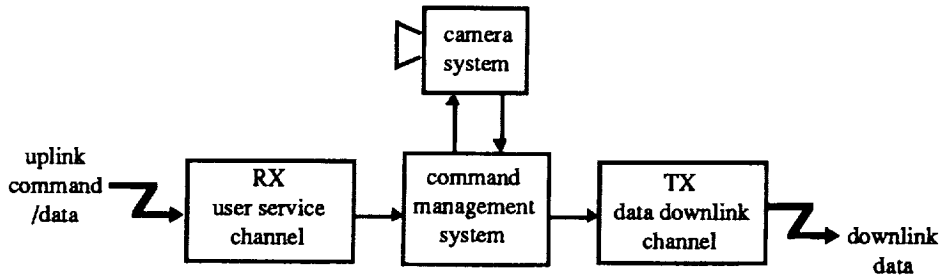


Figure 1. Level 7 subsystems.

Stored data is held in an onboard database within the Command Management System (CMS) for downlinking to specified destinations. Stored data can be deleted upon explicit user request. In all of these transactions the spacecraft will schedule and execute operations without coordination or mediation by a central ground station or command and control network. That is, user stations will interact directly with the spacecraft.

The Command Management System (CMS) consists of a command execution controller, a database management system (dbms), and a database residing in bank switched memory. The CMS command execution controller and other levels of the control system handle detailed protocols and data exchange between devices, allowing the level 7 control competency to be defined at a supervisory control level. The CMS database contains user data packets in addition to user commands, and maintains a file listing all data packets and their parameters or header information. This directory file is treated as a data packet with an identifier of 0. It can be downlinked (with filtering based upon user priority), but cannot be deleted. The directory is updated automatically by the CMS dbms.

A Store Data command has an associated data packet. Upon receipt of a Store Data command, the data packet is stored in a database within the CMS, along with command parameters. A data identifier and descriptor are entered into the directory file within the CMS database. Store Data commands are processed immediately if accepted (ie. within one processing cycle of the control system) to initiate data input to the CMS database from the receiver (channel) for eligible commands. Detailed timing of data storage transactions is not modelled at this level. If the level 7 control cycle time is fast enough, a Store Data command can be validated prior to uplinking any of the associated data to the spacecraft. If the cycle time is too slow for this, uplinked data can be buffered prior to transfer to the CMS database if the command is accepted, or deleted if the command is rejected.

Upon receipt, a Broadcast Data command is stored in the database within the CMS. As specified by the command parameters, broadcasting can be initiated immediately, at a specified time, or in the proximity of a particular target point specified by latitude and longitude. The Broadcast Data command includes parameters specifying the data to be broadcast.

A Delete Data command includes parameters specifying the data to be deleted from the CMS database. Deletion will occur only if the command

carries appropriate authorisation. When data is deleted, the associated data identifier and descriptor are removed from the directory file within the CMS database.

Upon receipt, an Acquire Image command is stored in the CMS database. The command includes all appropriate image acquisition parameters. Execution of an Acquire Image command is triggered when time or positional parameters are satisfied, and subject to resource availability. Once an image has been acquired, it is stored as a data unit in the CMS database, along with its parameters and acquisition details. An image identifier and a descriptor are entered into the directory file within the CMS database.

Level 7 Subsystem Interface Definition

Subsystem Sensors

Each subsystem or component has a range of data outputs, which may include sensor values and computed function values. These outputs are treated as sensors monitored by the level 7 controller. Sensors classified by associated component are as follows:

Command Management System (CMS):

CA	command arrival, 1/0
CREC	command record
BD	broadcasting data via tx, 1/0
SD	storing data from rx, 1/0
CMA	active memory blocks
CM	total memory blocks

Guidance and Navigation System (GNS):

SClat	spacecraft latitude
SClong	spacecraft longitude
SCalt	spacecraft altitude
SCtime	spacecraft system time

CREC records are derived (within the CMS) from information provided by the user. Information associated with all commands upon initial receipt includes: command identifier (CID), user priority (UPRI), command priority (CPRI), and command type (CMD).

Further information within Acquire Image CREC input includes the target latitude and longitude or an image acquisition time, and the image data size. Store Data CREC inputs include the data size and the data itself. Broadcast Data CREC inputs include a data identifier, data size, and the downlink latitude and longitude or a downlink time. A data identifier of 0 indicates the CMS database directory file. Delete Data CREC inputs include a data identifier, and a delete latitude and longitude or a delete time. The level 7 CMS database user data storage format for each data file includes the data identifier, the identifier of the user who supplied the data, the data priority, the data size, and the stored data. A datablock generally includes command parameters from the initial Store Data or Acquire Image command associated with the data record.

Actuators

Control signals (including combined control and data output signals) are:

Command Management System (CMS):

Command id, CID	integer
Control enable, CE	1/0
Command Arrival Acknowledge, CAAK	1/0
Active Memory Blocks, AMB	2 bytes
Operation, CMS_OP	2 bits:
delete data	00
acquire image	01
store data	10
broadcast data	11

Transmitter:

tx_baud	2 bits
(encoding 0, 1200, 4800, or 9600 bps)	

Controller Design

The overall goal of the level 7 competency is to maximise the provision of user services throughout the lifetime of the spacecraft. Hence the goal is to maximise the overall throughput of data in response to user demand. Users and their data requests are prioritised, and higher priority throughput is more important than lower priority throughput. Hence, the goal of this level can be expressed as the maximisation of the value function:

$$Q = \sum_{i=1}^{k} S_i \cdot P_{ui} \cdot P_i / P^2_{max} \quad (1)$$

where k is the number of data packets downlinked over the lifetime of the spacecraft, S_i is the size of the ith data packet downlinked during that time, P_{ui} is the priority of the user who requested data packet i, P_i is the priority requested by the user for data packet i, and P_{max} is the maximum possible user and data priority. P_{max} is an *a priori* constant established by convention. S_i, P_{ui}, and P_i appear to the controller as stochastic variables within respective predefined range limits. The controller has some control over the product $S_i \cdot P_{ui} \cdot P_i$ for the command i executed on any given occasion if there are a number of candidate commands from which to choose that have different values of $S_i \cdot P_{ui} \cdot P_i$. Then, other things being equal, the controller can maximise the value function by choosing to process the command with the maximum product $S_i \cdot P_{ui} \cdot P_i$. It is the goal selection system of the level 7 competency that has the role of maximising Equation (1). Inputs to the goal selection system include command goals representing demands for power and state variables indicating the power available after battery charging (from level 1 of the control system, concerned with power control). The amount of power available constrains the data throughput rate, and also constrains command execution by command type.[1]

There are many ways of defining and implementing a goal selection algorithm, manifesting varying features and degrees of a behavioural approach. Several alternative goal selection systems are described here, compared, and evaluated qualitatively in terms of that aspect of their *robustness* that concerns the tendency of the system to return to proper functioning following disruption due to loss of hardware and/or control logic.

[1]More generally, there is a tradeoff between the throughput and the overall lifespan of the spacecraft. This complicates the optimisation problem, but the complication is avoided here by designing the spacecraft to operate with an average power consumption over a fixed design lifetime of four years.

Design #1: A Procedural Algorithm

The algorithm for a single cycle of goal selection can be written as a conventional procedural algorithm:

1. select eligible commands from the general command list, according to:
   - satisfaction of latitude, longitude, and time constraints
   - Store Data commands are not eligible if a command is currently being stored (ie. SD is asserted)
   - Broadcast Data commands are not eligible if a command is currently being broadcast (ie. BD is asserted)

2. set best goal = NIL

3. set power available = array energy remaining - battery charge energy needed (from Level 1)

4. for each eligible command goal i
   if the command is a *Delete Data* command then
      - execute the command
      - remove the command goal from the controller goal list
   else
      calculate Q_i = UPRI . CPRI . Sd
      if the command is a *Store Data* command then
         if memory required ≤ memory available (that is, Sd ≤ (CM - CMA)) then
            $Cost_i$ = receiver power + memory power
         else
            $Cost_i$ = greater than max power
         end if
      else if the command is an *Acquire Image* command then
         if memory required ≤ memory available then
            $Cost_i$ = camera power + memory power
         else
            $Cost_i$ = greater than max power
         end if

```
    else if the command is a Broadcast Data
    command then
      for each selectable transmission bit
      rate j
            Cost_ij = transmission power for
                  rate j
      end for
  end if
  if Q_i > Q_best goal then
      if command i is a Broadcast Data
      command then
            set j_max = max j for which
              Cost_ij < power available
            if j_max is not zero then
              set tx_baud = bit rate j_max
              set Cost_i  = Cost_ij_max
            else
              Cost_i = greater than max
                  power
            end if
      end if
      if Cost_i < power available then
            set best goal = goal i
      end if
  end if
end for
5. execute best goal
```

This system differs from the goal arbitration and selection network described by Maes (1990) in that, unlike Maes' system, command goals do not have any represented dependencies, substructures of subgoals, or interrelationships; they are treated as independent requests for system resources. Also, the goal list used here is a dynamic structure that may vary in size from 0 to the maximum number of goals that the system memory can hold. The value, Q_i, of each goal i is similar to the activation level used by Meas.

The content of goals (within certain standard formats) and the frequency of goal arrivals are determined by the dynamics of the system environment. The goals have a critical role in determining the value of the autonomous system, since the ultimate purpose of the spacecraft is to satisfy its users on the ground by satisfying their requests communicated to the spacecraft in the form of the goals. However, the goals are partial in the sense that maximising the usefulness of the system over its projected lifetime requires balancing user-generated goals with other goals concerning the health and status of spacecraft subsystems and components. The overall goal structure of the spacecraft control system is partially defined by the levels of the multi-level competency model (see Lindley, 1993a), partially emergent (overall performance is maximised by many competencies acting together but achieving numerous local subgoals), and partially explicit in the form of the command goal list.

Advantages of this procedural algorithm include overall simplicity and computational complexity that is linear in the number of eligible goals. The primary disadvantage is that the procedure has no intrinsic robustness against failure, because it is written according to a model in which a single active process interprets a variable-length list of passive goals.

Design #2: A Declarative Knowledge-Base

The goal selection system is a behavioural approach to goal selection in that it achieves planning and scheduling without world modelling. However, the system can in principle be implemented as a declarative knowledge base, representing a less paradigmatic example of a "behavioural system". The knowledge base can be arbitrarily complex, since there is no limit to the amount of knowledge about the spacecraft and its world that can in principle be modelled. The simplest usable solution is to code the procedural knowledge expressed in the procedural algorithm using a declarative representation language. This will ensure that no knowledge is represented that is not immediately relevant to the solution to the arbitration problem, and avoids the circuspection problem by implementing a behavioural goal selection mechanism that does not depend upon processing a world model. However, such a "knowledge-based" solution is at least as complex as the procedural solution, and will result in little generality - it is very unlikely that any of the knowledge will be reusable for any other purpose.

A declarative solution suffers from the robustness problem of the procedural solution if it depends upon a single inference process. Inference has the further disadvantage of realising the procedural algorithm in an inefficient and "unnatural" way as a search pattern over the declarative model of the algorithm. This inefficiency is avoided by

preunifying or compiling the declarative model into a "flat" rule base that can be executed without search, but at the expense of memory. The declarative solution can then be used to address some aspects of the robustness needed of the control system. In particular, in mapping the controller design onto available physical processors, the natural modularity of the declarative controller definition can be used to adaptively remap control functions across physical processors according to possible competing demands for processors or variable processor availability due to failures or power limitations (Lindley, 1994).

This method can result in a graceful degradation of temporal efficiency in controller execution with decreasing total processing power, and allows the executing controller to be completely rebuilt, if necessary, at any execution cycle (with the loss of some dynamic state information). In general, critically diminishing processor power can be managed systematically by eliminating control functions from the "top down" in a layered control system such as the subsumption architecture, thus eliminating competencies in decreasing order of criticality.

Deficiencies of this approach include the need to maintain a robust and reliable copy of the declarative definition of the control system, and the need for a robust and reliable monitoring and remapping process. The proposed solution is therefore not a general one, since the problems of robustness and reliability are shifted onto those features of the system designed to address the reliability and robustness of the executing controller.

For the proposed strategy to avoid endless regression, it is necessary to reach a point of control design that is *intrinsically* robust and reliable. If intrinsic techniques for achieving robustness and reliability can be defined, parsimony suggests that those techniques should be built into the initial model of the autonomous control system, rather than any level of meta-interpreter. It is the search for intrinsic reliability and robustness that leads to behavioural implentation strategies. As suggested by Lindley (1994), a behavioural control model can be specified declaratively, and formal properties of the syntax can aid design integrity and support verification. However, the use of declarative specifications is neutral in regard to other general characteristics of a design, and adds little to the search for principles of intrinsic robustness.

Design #3: A Subsumption Layer

Taking the basic sequence of operations defined in the procedural algorithm as a sequence of behavioural components, and allowing the messages passed between behavioural components to include lists of complex data items, the procedure can be represented as a behaviour network in the graphical notation used to describe layers of the subsumption architecture, as shown in Figure 2.

In this Figure, list-valued messages are represented by bold arrows. Each box represents a transformation of the data content of the messages passed by input arrows into the box to the data content of messages passed by output arrows. The function and result of the algorithm expressed in this form are the same as in the procedural algorithm. The only substantial change from the procedural algorithm is that instead of iteratively applying the sequence of operations to each command in turn and testing for the best command so far on each iteration, a single sequence of operations is used with each intermediate step in the sequence receiving a list of commands as its input, performing an operation on the list, and producing a new list (and/or single command) as its output. The network is activated once per control cycle, where the inputs (represented by elliptical sensor signal sources) are fixed (sampled) at the start of each cycle and the cycle time is long enough for the outputs (represented by elliptical actuator signal sinks) to stabilise by the end of each cycle.

Advantages of this version of the emergent planning and scheduling competency include those of the procedural algorithm, that is, simplicity and computational complexity that is linear in the number of commands. Additional advantages that might follow from implementation as a *single level* of a subsumption architecture using the mechanisms described by Brooks (1986), but *excluding* the general advantages of behavioural systems, include:

- each block in Figure 2 can be implemented as an augmented finite state machine (AFSM), minimising the complexity of its implementation
- blocks intercommunicate asynchronously, thereby simplifying their interfaces
- each block can be implemented as a separate physical process, thereby improving the robustness of the network against processor failure

The primary disadvantage of this model is that each block is critical to the correct operation of the overall competency, so the failure of a block (or its physical processor) will result in the failure of the competency. Remapping functional components (ie. blocks) to processors can redress this deficiency, but only at the cost of the regression of the robustness requirement (beginning with the remapping process), as in the case of knowledge-based inference processes. The basic subsumption mechanisms as described do not provide intrinsic robustness and fault-tolerance.

Design #4: Distributed Data

A major characteristic contributing to the robustness of a system is that incremental physical element failures merely degrade overall performance. A first step towards this characteristic is to distribute data structures across multiple hardware components. The list-processing functions of the subsumption layer shown on Figure 2 accept lists as input messages

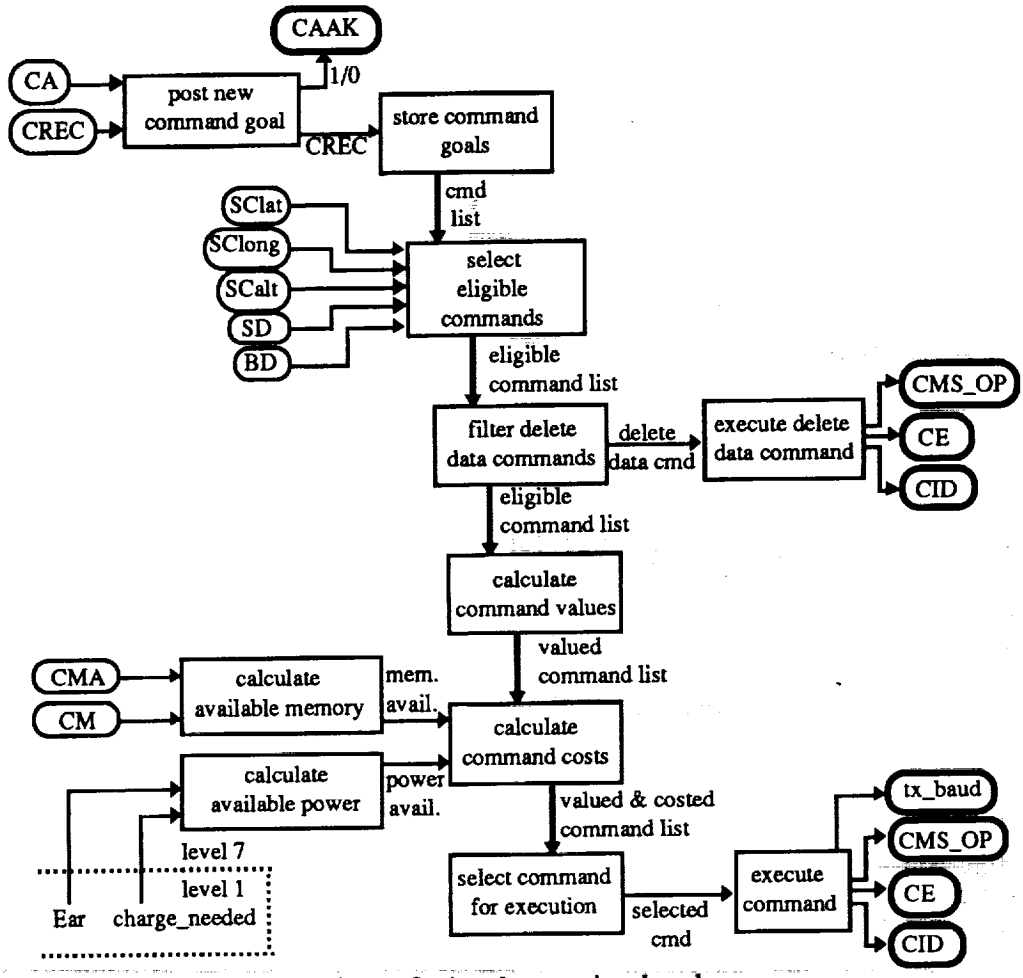


Figure 2. A subsumption level.

and/or generate lists as output messages. A more robust approach would be to carry out the sequence of operations using a separate data structure to represent each command. The transformations carried out by the operations can be represented as a sequence of transformations of the state of each command structure, until the final selection of a command for execution. This approach is modelled on Figure 3.

Posting new goals is assumed to be asynchronous, although only goals that are present at the beginning of a control cycle are processed within that cycle. The procedural sequence is then implemented beginning with the *enable eligible commands* function, with each subsequent function being initiated by the "done" message from the function preceding it in the sequence. The command goals themselves are held in a global memory area that is dynamically mapped onto multiple, modular hardware memory elements by the *post_new_command_goal* component. The wide arrows on Figure 3 indicate multiple independent data paths, and the double-headed paths indicate bidirectional transfer. Actual data transfer for each operation may be parallel or sequential. In the case of parallel transfer, each functional block acts as an SIMD (Single Instruction Multiple Data) processor, speeding up execution of the function. If data transfer is sequential, each block functions iteratively like the blocks of the subsumption layer described in Design #3, but iterating through transformations of shared data structures, rather than processing separate lists.

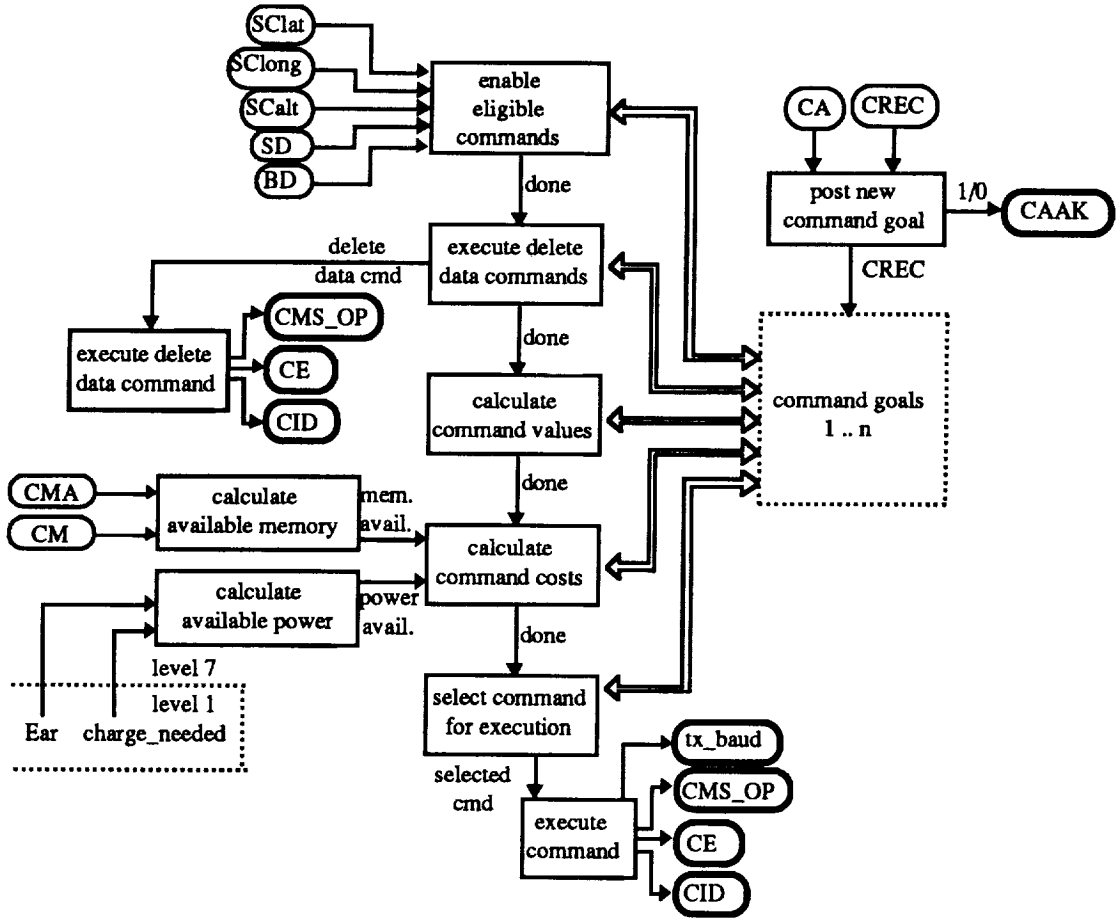


Figure 3. Controller with distributed data.

Each command data structure can be mapped onto a single physical memory unit (or n command

structures can be flexibly mapped onto m physical memory units), so that failures of physical memory units result in a degradation of processing or storage capacity, but do not result in complete loss of functionality. The loss of memory units must be detected by the *post_new_command_goal* component, and that component then assigns new goals to the remaining memory elements. The only loss incurred by the failure of a working memory component is the current set of goals stored by that component. This distributed data solution has superior robustness to the preceding design solutions, but still has no intrinsic robustness against failures of transformational modules.

Design #5: Distributed Data and Distributed Functionality

evaluation functions to be embedded in an iterative algorithm. A simpler solution is to combine the functions into a monolithic goal arbitration module, as shown on Figure 4. Goals can be presented to the network in parallel. While the number of any given goals at any particular time is variable, that number is always in a range from zero to a fixed upper bound (as limited by storage space). The network can be designed for the maximum possible number of goals, with unused inputs assigned to null values and their associated goals having zero probability of selection.

This design will allow distributed data storage and distributed functionality, supporting robustness against hardware loss in both cases by the association of particular data structures and particular connectionist nodes (or sets thereof)

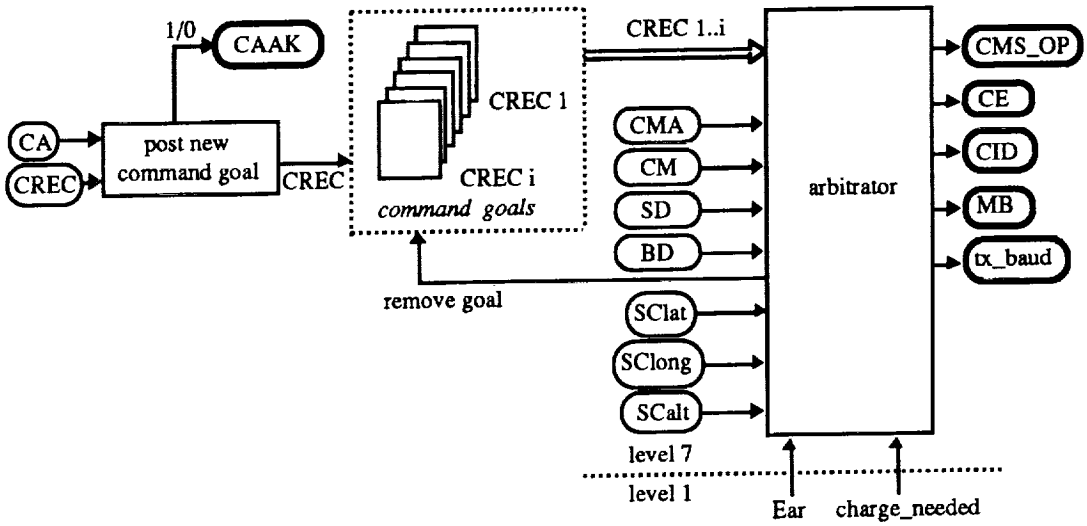


Figure 4. Distributed data with a monolithic arbitrator.

One method of achieving distribution of functionality is by implementing each functional module of the emergent scheduling algorithm as a connectionist network. That is, the transfer function implemented by the module can be implemented as a multi-layer network having an input layer for the inputs to the module, at least one hidden layer, and an output layer. Robustness is achieved because such networks are, in principle, highly tolerant of the loss of intermediate layer nodes. Iterative applications of the basic operations would require connectionist with different physical units. Hardware component failure will degrade performance, but will not immediately halt functionality unless a critical number of components are incapacitated. The detailed degradation characteristics for the goal selection task are currently unknown.

This form of solution may be the most robust considered here. The primary disadvantage of this approach is the need to create the weighted network of the arbitrator. A learning network could use a database of examples to learn from, if

available, or could learn in real time if a suitable reinforcement signal can be defined. The feasibility of this approach is a subject of ongoing research. A more desirable solution would be to derive a robust definition of the network from a formal specification of the transfer function that it is to implement.

Design #6: Distributed Objects/Agents

The final design option considered here is that of integrating data and functionality into discrete computational "objects", each of which represents a particular goal, and each of which can be allocated to a dedicated physical computational component. "Posting" a goal is then a matter of instantiating a goal object, based upon a common class or subclass (according to command type) definition but with particular characteristics defined by data unique to that command instance. Each goal has functionality for calculating its own activation level, based upon sensor inputs, state variables from other competency levels, and its own data values. Sensor inputs and state variables define a virtual environment within which goals compete for execution. The final selection is based upon the overall activation value of each separate goal, and so is a very simple mechanism that can be made robust by redundancy. If a hardware component is lost, the goals that depend upon that component will be lost; all other goals within the system will persist, with their own state and functionality intact.

A serious disadvantage of this approach is the need to robustly maintain a prototypical object or class definition, and a robust instantiation mechanism. This results in a similar regression of the robustness requirement to that seen in the case of a declarative solution (Design #2 above). A strategy for reducing this regressed requirement might be to reduce object functions to a more primitive set of operations, with overall goal selection occurring as an emergent phenomenon of the primitive interactions of goal objects. This is an area of ongoing investigation.

There are many ways in which a fully distributed system might be defined, based upon an object analogy, or upon analogies of processes or agents. Each analogy emphasises different aspects of the behaviour and interaction of goals within the system. Clarifying the variety and respective merits of distributed approaches are important themes of ongoing research.

Conclusion

This paper has described an approach to the autonomous on-board scheduling of spacecraft payload operations, based upon behavioural action selection and upon processing plans as communications. Several variations of the action selection mechanism have been described and evaluated comparatively in terms of robustness against hardware component loss. It has been seen that increasing the distribution of data and functionality can provide greater robustness, validating behavioural approaches to artificial intelligence that seek greater overall functionality and robustness via low level distributed functions that do not rely upon high level representation models. Designs that minimise representation and distribute both data and functionality have the greatest potential robustness, and manifest the most distinctive features of behavioural systems.

The spacecraft model and control designs described in this paper are simplified in a number of ways. Detailed protocols for modelling possibly sequential input of formatted sensor values are not considered. The applicability of on-board image compression has not been considered in detail, although compression could yield substantial benefits in increasing virtual storage capacity and data throughput. It is assumed that FDIR functions are carried out within the level 5 competency (Acquire, Condition, and Downlink Instrument Data); without a detailed model of that level, the integration of operational planning and scheduling with FDIR functions is not clearly defined. This paper has not considered downlinking the contents of the command database to users, acknowledgement of command execution, or elimination of commands that cannot be executed. Finally, the quantisation of input and output variables is not considered, either in the definition of those variables or in the definition of functions that accept or define them, respectively. These simplifications are used to clarify the central issues in developing a behavioural control system for the autonomous scheduling and planning of user services, and do

not invalidate the overall analysis or conclusions obtained.

Current work is addressing the validation of the goal selection mechanisms in a spacecraft simulator, and the development of a more rigorous and quantitative characterisation of the merit of the different goal selection systems, accounting for their overall complexity in addition to their robustness. Ongoing research is extending this work to determine how greater levels of autonomy and robustness can be achieved by incorporating methods for learning, adaptation, and self-organisation into behavioural systems.

References

Agre P. E. and Chaman D. 1990 "What Are Plans For?", in Maes, P. (ed) *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 17-34.

Brooks R. A. 1986 "A Robust Layered Control System For A Mobile Robot", *IEEE Journal of Robotics and Automation*, RA-2, no 1, 14-23.

Brooks R. A. 1991 "Intelligence Without Reason", *12th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, 569 - 595.

Drabble B. 1991 "Spacecraft Command and Control Using Artificial Intelligence Techniques", *Journal of the British Interplanetary Society*, Vol 44, 251-254.

Elfving A. and Kirchoff U. 1991 "Design Methodology for Space Automation and Robotics Systems", *ESA Journal*, Vol 15, 149-164.

Erickson J. D., Phinney D. E., Norsworthy R. S., Zacksenhouse M., Hartness K. T., Pham T. T., Merkel L. F. and Tu E. Y. 1989 "A Prototype Autonomous Agent for Crew and Equipment Retrieval in Space", *Proceedings, Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1052-1058.

Georgeff M. P. 1987 "Planning", *Annual Reviews in Computing Science*, 359-400.

Lindley C. A. 1993a "An Autonomous Satellite Architecture Integrating Deliberative Reasoning and Behavioural Intelligence", *Telematics and Informatics*, 10 (3).

Lindley C. A. 1993b "A Behavioural System for Autonomous Spacecraft Power Distribution and Control", *Artificial Intelligence and Knowledge Based Systems for Space*, 4th Workshop, May 17th - 19th, ESTEC, Noordwijk, The Netherlands.

Lindley C. A. 1993c "A Behavioural Theory of Intelligent Machines as a Framework for the Analysis of Adaptation", *AI-93 Workshop on Evolutionary Computation*, Melbourne, Australia.

Lindley C. A. 1994 "Extending the Behavioural Paradigm for Intelligent Systems", Special Track on Emerging Paradigms for Intelligent Systems, *Twenty-Seventh Annual Hawaii International Conference on System Sciences*, Maui, HI.

Maes P. 1990 "Situated Agents Can Have Goals", in Maes, P. (ed) *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 49-70.

Maes P. 1993 "Behaviour-Based Artificial Intelligence", *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour* , MIT Press.

Pidgeon A., Howard G. and Seaton B. 1992 "Operational Aspects of Spacecraft Autonomy", *Journal of the British Interplanetary Society*, Vol 45, 87-92.

Raslavicius L., Gathmann T. P., and Barry J. M. 1989 "An Artificial Intelligence Framework for Satellite Autonomy", *Proceedings, Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Vol 2, 536-543.

Rokey M. and Grenander S. 1990 "Planning for Space Telerobotics: the Remote Mission Specialist", *IEEE Expert*, 8-15, June.

Valavanis K. P. and Saridis G. N. 1992 *Intelligent Robotic Systems: Theory, Design and Applications*, Kluwer Academic Publishers.

Watson J. F., Lefebvre D. R., Desrochers A. A., Murphy S. H., and Fieldhouse K. R. 1992 "Testbed for Cooperative Robotic Manipulators", in Desrochers A. A. (ed.) *Intelligent Robotic Systems for Space Exploration*, Kluwer Academic Publishers.

A Criterion Autoscheduler for Long Range Planning

Jeffrey L. Sponsler

Space Telescope Science Institute
3700 San Martin Drive, Baltimore, MD 21218 USA
(410) 338-4565 sponsler@stsci.edu

Abstract

A constraint-based scheduling system called SPIKE is used to create long-term schedules for the Hubble Space Telescope. A meta-level scheduler called the Criterion Autoscheduler for Long range planning (CASL) has been created to guide SPIKE's schedule generation according to the agenda of the planning scientists. It is proposed that sufficient flexibility exists in a schedule to allow high level planning heuristics to be applied without adversely affected crucial constraints such as spacecraft efficiency. This hypothesis is supported by test data which is described.

1. Introduction

The scheduling of the Hubble Space Telescope (HST) is complex and involves many software systems. A long range planning system called SPIKE is integral to the process. Recently, SPIKE has been augmented with a new subsystem for the following reasons:

1. The planning scientists (who use SPIKE) were not able to make important changes to the behavior of the scheduling system without requesting that software developers effect code changes.

2. A set of scheduling rules that was provided by the scientists could not easily be encoded in the scheduler. Expert system technology was proposed as a way to give users control over the scheduling process.

The *Criterion Autoscheduler for Long range planning* (CASL) has been implemented to satisfy these requirements. An expert system methodology called *functional knowledge representation* (Lucks, 1992) has been implemented as a generic shell called the multiple criterion network (MCN). MCN uses *criteria* and *knowledge functions* which are defined here:

Criterion: A problem solving heuristic that is associated with some aspect of the domain. CASL employs scheduling criteria such as "Attempt to schedule observations as early as possible."

Knowledge Function: A class of operator that transmutes application data into numeric scores. The mappings are criterion-specific and explicitly stored in the knowledge base.

These concepts are described in detail in later sections. CASL is the focus of this report and experiments with CASL are described and results are presented.

2. Description of the HST

NASA's Hubble Space Telescope (HST) is an orbital observatory that was launched by the Space Shuttle *Discovery* in 1990 and successfully repaired in December 1993 by the *Endeavor* crew. The success of the "First Servicing Mission" should restore HST optics to original specifications and improve its ability to do high quality science.

The Space Telescope Science Institute (STScI) is responsible for managing the ground-based scientific operations of HST. Proposals (i.e., experimental programs) are submitted to the Institute and are processed by a series of software programs. The results are sets of spacecraft commands which (ideally) produce image data that is returned to the STScI for further processing and archiving. For more details about HST, see Hall 1982.

3. Conceptual Description of the HST Scheduling Process

The scheduling problem has been divided into discrete layers of logic. The layers can be briefly described below. Several of these layers will be discussed in detail in a later section. The following list is in order of increasing abstraction from the spacecraft.

- An instrument on the spacecraft is commanded to expose a photosensitive element to light.
- The commands are derived from a micro-scheduled calendar created by astronomers using the Science Planning and Scheduling System (SPSS).
- The Long Range Planning (LRP) system operates on one year time intervals. The time resolution of the LRP is currently one week. The set of observations assigned to one particular week of a completely scheduled LRP are communicated to SPSS.

The long range planning process has the following logical layers:

- At the lowest level is a constraint analysis system, called Micro-SPIKE, that provides information such as "When is the observation's target visible to HST?" and "Since observation B must be after observation A, when is it legal to observe B?"
- The next higher layer is the Constraint Satisfaction Problem (CSP) system which provides a workbench for searching for a feasible set of assignments of observations to time slots. The CSP employs Micro-SPIKE to answer *what if* questions such as the aforementioned.
- At the most abstract layer, CASL employs the CSP system and its utilities to create an LRP that satisfies both the physical constraints of the spacecraft and the practical realities of the planning scientists.

4. The Scheduling Sequence

In this section, the major steps in the scheduling of HST proposals is described. Later sections will discuss the CASL Long Range Planning strategy in detail.

4.1. Proposal Creation

An astronomer currently creates a proposal that takes, as its initial form, a text file containing definitions of *targets* (objects to be observed), *exposures* (the images to be taken), and *special requirements* (constraints on exposures). Syntax checking is done via distributed software tools. The proposal is sent to the STScI where it is submitted to an analysis and correction process called *proposal preparation*. This process creates *scheduling units* (SUs) which are collections of exposures organized by a complex set of rules. The role of SPIKE is to place SUs on a long range plan that spans many months.

The proposal preparation phase includes running portions of the SPIKE software that check important proposal aspects such as violation of HST pointing restrictions (e.g., "Allow no pointing that is within 50 degrees of the sun.") and schedulability (e.g., "The AFTER constraint causes the linked exposures to have no legal scheduling windows"). The principle goal of preparation is to provide planning personnel with a proposal that will schedule without constraint violations "in isolation."

4.2. Long Range Planning

When a large fraction of proposals have been prepared, the Long Range Plan (LRP) is created. The LRP spans approximately a year and consists of week-long time *segments*. The main difference between preparation and planning is that the proposals must, in the context of the LRP, compete for resources such as available *spacecraft time*.

4.2.1. The Constraint Based Scheduling System

The SPIKE system has a subsystem called *micro-spike* that is used to analyze absolute constraints ("Don't point at the moon") and relative constraints ("Schedule SU A after SU B"). This section discusses how it operates.

SPIKE represents scheduling information primarily using the *suitability function*. The suitability function is a means for representing scheduling constraints and preferences (Johnston, 1990). The approach provides a way to represent the concept of "goodness over time." Suitabilities (in this discussion) are in the real numbers and range from zero to one where zero means unsuitable and one means nominally suitable. The encoding of a suitability function is via a *piecewise constant function* (PCF) which is a list of time/value pairs. For example, if one determines that the sun is blocking the view of a target from the first segment of our plan up to, but not including, the fifth segment (when the target becomes visible), a PCF representing this would appear as follows: (1 0 5 1).

For an SU, absolute constraints are computed and represented as suitabilities. The following list enumerates some of these:

Solar Exclusion: Allow no point that is less then 50 degrees from the sun.

Orbital Viewing: Determine first whether there is available viewing time (i.e., that the target is not occulted by the earth) and if so how much. The suitability for a specific time period will be inversely proportional to the number of orbits required for the SU's component observations.

Micro-SPIKE supports relative constraints via a constraint propagation algorithm. Informally, let A and B be SUs. Let abs(A) represent the combined absolute constraint suitabilities of A and after(A, B) represent the constraint that B must be scheduled after A. Micro-SPIKE computes the effects of abs(B) and after(A, B) deriving a new suitability for

B that upon combination with abs(B) produces a new suitability for B. Arbitrary relative constraint networks are supported and so the algorithm iterates until no further changes in suitability (for any SU) is detected. See Miller (1988) and Sponsler (1991) for a more complete description.

In the SPIKE domain scheduling is treated as a *constraint satisfaction problem.* Such problems are known to be NP-complete (Garey, 1979) and so the exhaustive traversal of the entire search space is not computationally tractable if the number of SUs is large.

4.2.2. CSP

Another SPIKE subsystem called the Constraint Satisfaction Problem (CSP) system can be considered an general purpose problem solving engine.

In the context of HST scheduling CSP provides a workbench for searching for feasible LRPs (fully committed with no constraint violations). The SPIKE CSP system performs the following:

- Communicates with the Micro-SPIKE software to obtain data about constraints (e.g., if an SU is planned in a specific time segment, how does this affect related SUs?).

- Supports a framework that records static *preference* values for each SU/segment pair. Preference is (currently) defined to be the integral of the suitability function for a discrete time segment and so collapses a complex description to a single value.

- Supports a mapping from SU/segment pair to a count of constraint violations (called *conflicts*). This count provides information that not only describes where constraints are violated but also how many violations have occurred.

- Provides the capability to *commit* SUs to time segments. A fully committed CSP represents a complete long range plan. The action of commitment causes Micro-SPIKE to be consulted such that other SUs (related by relative constraints) may acquire new conflicts in certain segments).

- Tracks resources for each time segment. When all resources for a segment are consumed, a conflict is logged for all other SUs thereby communicating the undesirability of that segment for further commitments.

Minton (1992) found that heuristic algorithms (including *max preference min conflicts*) could be used to solve CSP problems effectively. In the context of HST, by iteratively searching for SU/segment pairs that maximize suitability and minimize constraint violations, good schedules result.

4.2.3. Meta-Scheduling

The max preference algorithm has been effective in supporting HST scheduling. However, the STScI personnel who employ SPIKE to create LRPs requested that the software provide another (more abstract) layer that would support important constraints provided by the planners themselves. For example, the constraint to schedule SUs as early as possible does not have a physical basis. It is important, however, because the near-term portion of an LRP should be as fully packed, with respect to available spacecraft resources, as possible.

The request for such software by the planners catalyzed the design and implementation of the *Criterion AutoScheduler for Long range planning* (CASL) which is the subject of this report. CASL is described in detail in later sections.

The resource constraint levels for the LRP are set to greater than 100% in order to provide a larger pool of SUs for micro-scheduling. This oversubscription is intended to compensate for the fact that certain SPSS scheduling constraints are not encoded in SPIKE currently. If, therefore, certain commitments are incorrect, the larger pool provides SPSS with alternative SUs.

4.3. Micro-scheduling

The final step in proposal processing, at the STScI, is accomplished by SPSS. The SUs committed to one week time segment in the LRP are communicated to SPSS operators who micro-schedule them very precisely on a *calendar* (a data structure that represents a time line). Upon completion, this calendar is converted into a sequence of spacecraft commands.

5. The Multiple Criterion Network Expert System Engine

An expert system technology called the multiple criterion network system (MCN) has been developed. MCN is based on *functional knowledge representation* which has been applied previously in two other systems (Lucks, 1992 and Lucks, 1990).

Several terms are now defined informally. The MCN *score* is a numeric value in the set of real numbers ranging from zero to one. Conventionally, a zero score is interpreted as poor and a unit score as good.

```
CASL computes a score that describes how well an SU
schedules in a specific time segment.  If the score is
zero, then the fit is poor.
```

The MCN *criterion* is domain-specific and deemed an important attribute of the problem being solved by the expert and so is required information concerning the decision making process.

```
For example, in CASL, scheduling an SU as early as possible
is a criterion.
```

This technology provides system builders with the following capabilities:

i. The ability to store expert knowledge in the form of knowledge functions which are domain-specific and which calculate numeric scores (real numbers in the range from 0 to 1).

ii. The ability to explicitly declare mappings from raw scores to more refined scores. These knowledge mappings clarify the decision making calculations for knowledge engineers and users alike.

iii. The ability to define the way scores are accumulated into a single score via an aggregation function.

These capabilities support trade-offs between competing criterion in an automated decision support system. The components of MCN are described in the following sections.

5.1. The Knowledge Functions

The knowledge function is a program which encodes expert knowledge about the domain. Each function either produces a numeric score or maps one score to another. There are four types of knowledge functions in the MCN model. They are described below.

5.1.1. Measurement Functions

The *measurement function* is the initial source of data in the MCN model. This function is the point where the application information about a specific criterion is accessed directly and converted to a measurement value. This value is not required to be a score and in fact may be quantitative or qualitative.

> CASL's *earliest* criterion measurement function maps (for example) to a measurement value of 0.085 (i.e., the temporal offset of a time segment with respect to the entire schedule). Another criterion, *preference* (i.e., static goodness of fit) might produce a value of 50 (where the maximum is 100).

5.1.2. Intensity Functions

The *intensity function* is defined as a mapping between measurement value and the *intensity value* which is required to be from zero to one. This mapping is a normalization of criteria to a single scale and may be any arbitrary function. The intensity value conventionally is not interpreted as good or poor.

> The earliest criterion measurement of 0.085 maps to an intensity of 0.915 (the mapping is simply the additive inverse). The preference intensity for 50 is 0.50.

5.1.3. Compatibility Functions

The *compatibility function* maps from intensity value to *compatibility value*. This mapping can be any function and must be specified by the knowledge engineer with expert guidance. There are two important attributes to this mapping:

1. This compatibility value conventionally uses the classic 0.0 = poor and 1.0 = good meaning.

2. The mapping also provides a way to adjust the relative power or importance of the criterion. An intensity value may be either tempered or amplified by this mapping to diminish or increase its importance with respect to other criteria.

In CASL, a simple weighting scheme is used for this mapping. The weight is applied as follows. Let w be the weight, m be the maximum intensity value, and v be the intensity value. Computation of compatibility is performed by this formula: m - (w * (m - v)).

> The earliest criterion maps (by application of weight = 0.5) from intensity 0.915 to a compatibility value of 0.941. The preference value maps (with weight = 1.0) from intensity 0.5 to a compatibility value of 0.5.

5.1.4. Aggregation Function

The aggregation function maps from the compatibility value to the aggregation value which is the final score of the network. In the functional knowledge representation technology, an augmented decision tree (and/or graph) is used to compute the score from the individual criteria. The augmentation includes other functions that may be defined by the engineer. These are described in Lucks, 1992. MCN employs a single function (e.g., multiply) to perform the aggregation.

```
Aggregating all criteria for a specific SU/segment pair
yields a value of 0.211.  One can select the segment with
highest score as the winning match where the SU can be
scheduled.
```

5.2. Parallel Observations Matching System

The Space Telescope is capable of executing observations in parallel provided a set of restrictions are satisfied (e.g., the same instrument cannot be used in parallel with itself). Functional knowledge representation has been applied to the problem of matching pre-defined parallel scheduling units with standard SUs (see Lucks, 1992). A large pool of such parallel SUs must compete for scheduled SUs. This system, the Parallel Observations Matching System (POMS), is in operational use at STScI.

6. CASL Architecture

The CASL system employs MCN technology to direct the LRP generation. Specifically, the aggregation decision tree is replaced with a simple combining function (the multiply function, for example). The function of CASL is to create Long Range Plans for HST that:

- do not violate constraints
- maximize the suitability of all commitments
- satisfy the planning goals of the LRP planners

The CASL system requires that prepared proposals be loaded into a CSP Scheduling System which provides the workbench upon which the expert system operates. There are two main phases to the application of CASL. They are described below.

6.1. Prioritization Phase

The prioritization phase of CASL is responsible for analyzing each scheduling unit using MCN techniques to determine the order by which SUs are placed on the schedule. This ordering does not specify the time ordering on the resulting schedule. A subset of the criteria used are described below:

Absolute Time Windows (ABSOLUTE). Characterize an SU based on the amount of time that is available to it based on its absolute constraints.

Relative Timing Links (RELATIVE). Characterize an SU based on the number of relative constraints that are attached to it.

Proposal Completion (COMPLETION). Consider a proposal and its component SUs. The fraction of SUs that have been scheduled by SPSS is used to compute the completion

measurement function. The priority of an SU (with respect to this rule) is proportional to the fractional completion.

The ordering is done based on the aggregate score for each SU. The SU with the highest score (i.e., has the highest priority) will be placed on the plan first.

6.2. Autoscheduling Phase

The next phase of the CASL algorithm is the actual scheduling. An SU is selected from the sorted list and is analyzed with respect to all week-long time segments in the scheduling interval (which can span a year or more).

6.2.1. Time Segment Selection

Some of the criteria used to select a time segment for each SU are described below:

Min-Conflicts (CONFLICTS): This criterion will select for segments that have few or no conflicts.

Max-Preference (PREFERENCE): This criterion will select for segments that have the highest preference value.

Earliest-Segment (EARLIEST): This criterion will tend to place commitments earlier on the schedule if possible.

Minimize-Proposal-Spread (SPREAD): This criterion will tend to keep SUs from the same proposal together on the schedule.

Level-SU-Duration (DURATION): This criterion will tend to place SUs into segments that have less SU Duration resource consumed.

Prior-Commit (PRIOR): This criterion attempts to preserve pre-existing commitments. Periodically, the LRP must be regenerated due to feedback from the micro-schedule, changes in spacecraft status, etc. This criterion attempts to preserve old LRP state by biasing for selection of segments previously selected.

Group-Instruments (GROUP): This criterion attempts to commit SUs with the same science instrument (SI) into the same segment. It determines an SI frequency distribution of the instrument for the SU and this becomes the score. For example, if the SU under study uses the Faint Object Camera (FOC) and the distribution of FOC in the segment is 0.70 then that is the score (since it is fairly good score the segment will be attractive for commitment).

Continuous Viewing Zone (CVZ): The measurement function for this criterion determines whether the SU is a candidate for the CVZ which is a period of time (usually a few days) when the target is not occulted by the earth. Scheduling SUs in the CVZ improves the efficiency of spacecraft operation.

Random-Choice (RANDOM): This criterion should be used only when the autoscheduler is to be executed several times in order to search for relatively better plans. It injects an element of chance into time segment selection thereby allowing schedules to differ. The associated criterion weight specifies the degree to which randomness is inserted.

Table 1. This table provides example data for one SU/segment pair. Only a subset of the criteria is displayed. The aggregate score for this set of criteria is: 0.211.

Criterion	Measurement	Intensity	Weight	Compatibility
CONFLICTS	0.000	1.000	1.000	1.000
PREFERENCE	50.000	0.500	1.000	0.500
SU DUR	0.422	0.578	0.600	0.747
EARLIEST	0.085	0.915	0.500	0.941
SPREAD	8.500	0.200	0.500	0.600

In Table 1, the EARLIEST criterion has a measurement value of 0.085 (i.e., the week under analysis has an offset that is 8.5% of the scheduling interval and therefore very early). Normalization by the intensity function produces an intensity value of 0.915 (which is good). The weight of 0.5 reduces the effect of the criterion producing a compatibility value of 0.941.

6.2.2. Commitment

The result of the time segment selection process is the selection of the week that has the highest aggregate score. The SU is then committed to this week. This action produces the following side effects:

i. Constraint propagation is performed to determine the effect of the commitment on other SUs that are linked (via timing constraints) to the committed SU.

ii. For such a linked SU, conflicts accumulate for any week that has become unsuitable based on constraint propagation.

iii. The committed SU consumes some portion of available resource constraints.

iv. SUs that are yet to be analyzed will be affected by the change in conflict counts and resources.

7. CASL Behavior

In the following sections, CASL experimental data are presented and described. An informal Meta-Scheduling Hypothesis for these experiments is as follows. Let *schedule quality* be defined as average preference for all commitments. Preference takes into account the physical and proposer constraints on the SU.

Sufficient flexibility exists in the placement of scheduling units such that a meta-scheduling agenda can be followed without sacrificing schedule average preference.

The scheduling units, criteria, and weights in these studies have been obtained from the operational database currently in use by HST planners.

7.1. The Cycle 4 Long Range Plan

The Cycle 4 Proposal period extends from January 1994 (following the First Servicing Mission) to June 1995. For this report, 994 scheduling units from 281 proposals were automatically scheduled into 78 week-long time segments using CASL. This dataset has the following attributes:

- It is not randomly generated and therefore cannot be considered a general purpose benchmark dataset. This dataset may not adequately exercise CASL.
- The dataset contains a 4944 SU to SU timing links which make scheduling difficult.
- The pool represents a subset of the complete set of Cycle 4 SUs because the proposals have not been completely prepared.
- The resource ceilings are very high (approximately 200%) compared to the actual available time. This is unrealistic but does not really effect the outcome of the experiments.

The descriptions below include unit tests on several criteria and tests done on the integrated criterion set.

7.2. Criterion Unit Tests

In each of the following tests, the *max preference* and *min conflicts* criteria were in effect and set to a unity weight. The Meta-Scheduling Hypothesis proposes that meta-scheduling criteria (such as earliest) can be applied without violating the preference requirement.

The data format of the following tables is as follows:

Spread: Let n be the number of proposals, last-su_i be the latest commitment date of all SUs in a proposal, and first-su_i be the first commitment date of the same proposal. Spread is defined as: $\frac{\sum_{i=1}^{n} \text{last-su}_i\text{-first-su}_i}{n}$.

Spread SD: This is the standard deviation (in weeks) for proposal spread. This gives an indication of how each proposal varies from the sample mean spread.

Completion: This is the number of SUs committed divided by the total.

Offset: For each proposal a mean offset (segment of commitment/total segments) is determined. This value is the mean of all mean offsets and provides a measure of when the proposal was scheduled relative to the entire schedule. This attribute is directly affected by the *earliest* criterion.

SU Dur Mean: This is the mean amount of available SU Duration resource consumed for all weeks.

SU Dur SD: This is the standard deviation of the SU Dur Mean. A value of 0.0 would indicate perfectly level consumption. This attribute is directly affected by the *level su duration* criterion.

Pref: Let n be the number of SUs and $\pi(SU_i)$ be the percent of maximum preference for the ith SU. Pref is defined as $\frac{\sum_{i=1}^{n} \pi(SUi)}{n}$.

Orbits-Min: This is the number of spacecraft orbits consumed by the schedule minus the theoretical minimum number of orbits. An SU may consume a different number of orbits at different times of the year due to variance in target viewing times. The optimal value for this measure is zero. It should be noted that CASL does not explicitly operate on this attribute and so no attempt is made to minimize changes to it. Preference references this information implicitly along with many other aspects of target viewing. Note that the schedule minimum orbits is equal to 2432 and that an Orbits-Min value of 100 represents 4% difference from the optimal.

The proposal spread criterion was tested with weights of 0.0, 0.1., 0.5, and 1.0. There results are in Table 2.

Table 2. This table contains data obtained from unit tests of the *proposal spread* criterion which tends to keep the SUs from a given proposal together.

Attribute	Wt =0.0	0.1	0.5	1.0
Spread	**14.4**	**11.2**	**8.0**	**5.7**
Spread SD	**18.8**	**16.0**	**13.3**	**10.4**
Completion	0.71	0.71	0.71	0.71
Offset	0.86	0.87	0.87	0.87
SU Dur Mean	0.46	0.45	0.45	0.45
SU Dur SD	0.25	0.22	0.23	0.23
Pref	0.99	0.99	0.98	0.95
Orbits-Min	91	99	104	104

The average Spread was decreased from 14.4 to 5.7 weeks (a 60% change) and the Spread SD decreased by 45% while the Pref value changed by only 4% and the Orbits-Min value change by 14%. These data indicate that this criterion can cause an improvement to a schedule without disrupting the crucial Pref value.

A second criterion, Level SU Duration, was analyzed in the same manner and the results are recorded in Table 3.

Table 3. These data were obtained from unit tests of the *Level SU Duration* criterion.

Attribute	Wt =0.0	0.1	0.5	1.0
Spread	14.4	16.2	16.5	17.2
Spread SD	18.8	18.8	19.0	19.4
Completion	0.71	0.71	0.71	0.71
Offset	0.86	0.89	0.88	0.88
SU Dur Mean	**0.46**	**0.46**	**0.46**	**0.46**
SU Dur SD	**0.25**	**0.11**	**0.09**	**0.08**
Pref	0.99	1.0	0.99	0.98
Orbits-Min	91	102	105	109

The SU Dur SD decreased from 0.25 to 0.08 (a 68% change) while the Pref value changed by only 1% and the Orbits-Min value change by 20%. These data indicate that this criterion can cause measurable changes to a schedule without disrupting the crucial Pref value.

The Earliest Criterion

The Earliest criterion was tested in a context that contained only the preference and conflicts criteria and the results are illustrated in Figure 1.

The behavior of the criterion is apparent in that 100% of the SU Duration resource is consumed in the early segments (e.g., 94.031) and decreases to 0% in the final segments (e.g., 95.065).

The Level SU Duration Criterion

The effect of the Level SU Duration criterion on the LRP can be seen in Figure 2 below.

In Figure 2, the peaks and valleys for SU Duration are tempered by the criterion's effect. The minimum and maximum values for the control LRP are 9% and 100%. The minimum and maximum for the criterion are 33% and 80%. It should be noted that if the preference and conflict criteria were inactivated, the bold line in Figure 2 would be flat.

7.3. Integrated Tests

In the integrated tests, a subset (Preference, Conflicts, Earliest, Spread, and Level SU Duration) of the criteria were activated and given the weights used operationally at STScI. Figure 3 illustrates some of the calculations used for time segment selection for one scheduling unit. Since no conflicted commitments are permitted, this criterion is omitted from further discussion.

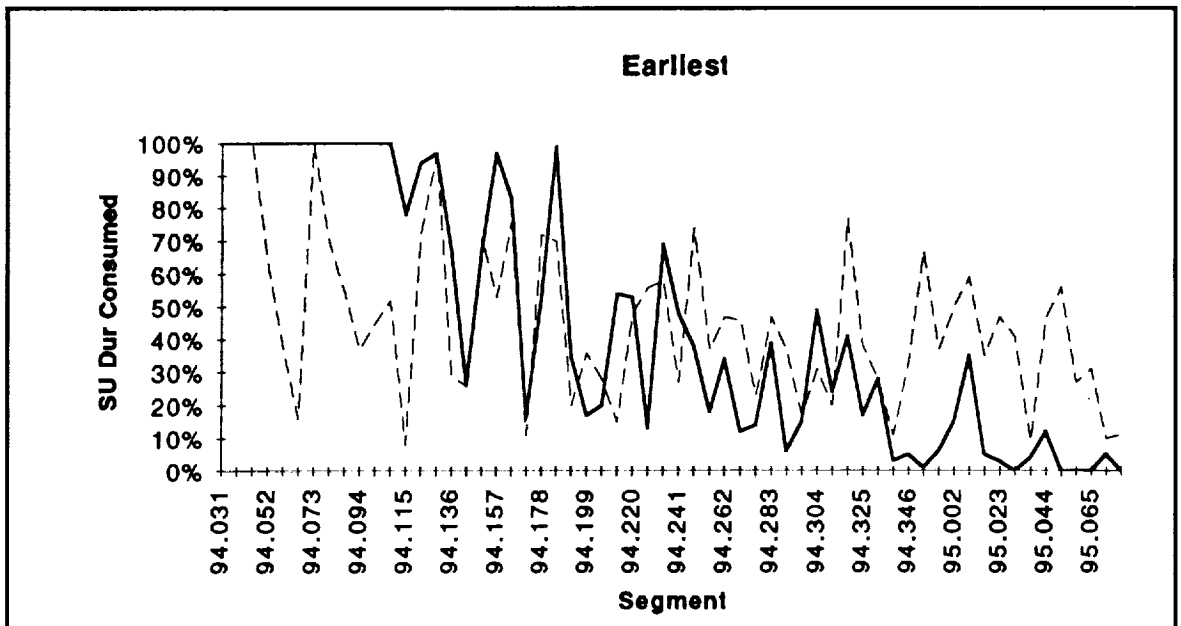


Figure 1. The bold line represents the LRP when the Earliest Criterion is set to weight of 1.0 compared with the control LRP that is scheduled with only Max Preference and Min Conflicts. An earliest week bias exists in the control LRP because the weeks are analyzed chronologically.

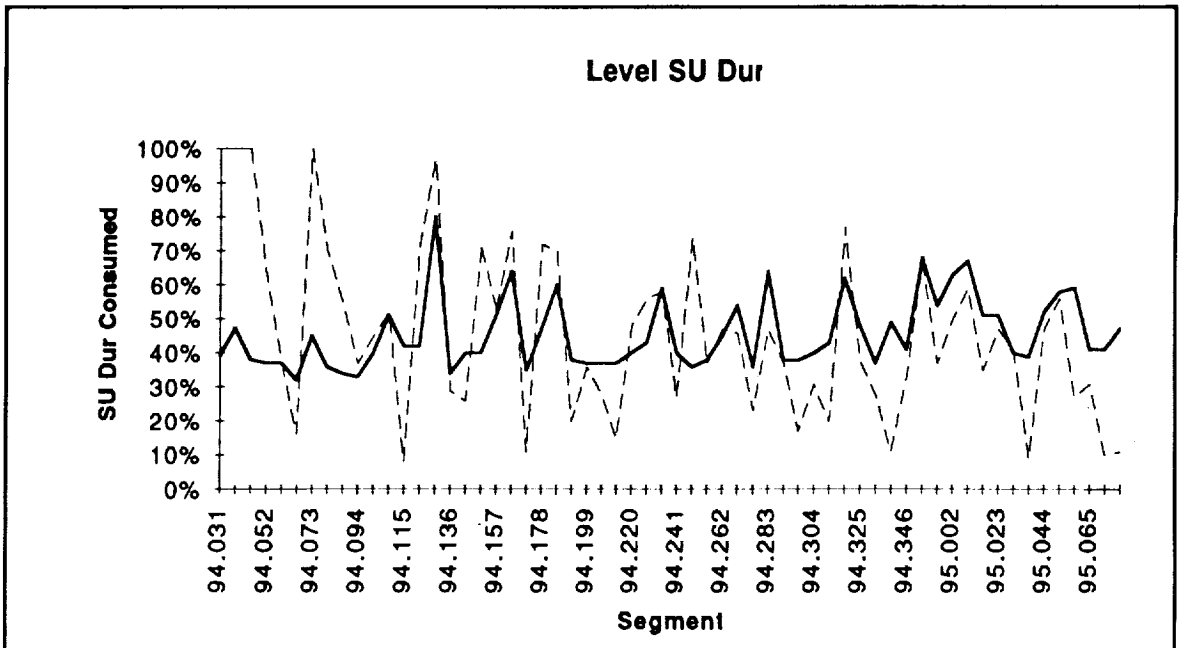


Figure 2. The control LRP is compared with an LRP with the Level SU Duration criterion applied with weight 1.0 (the bold line).

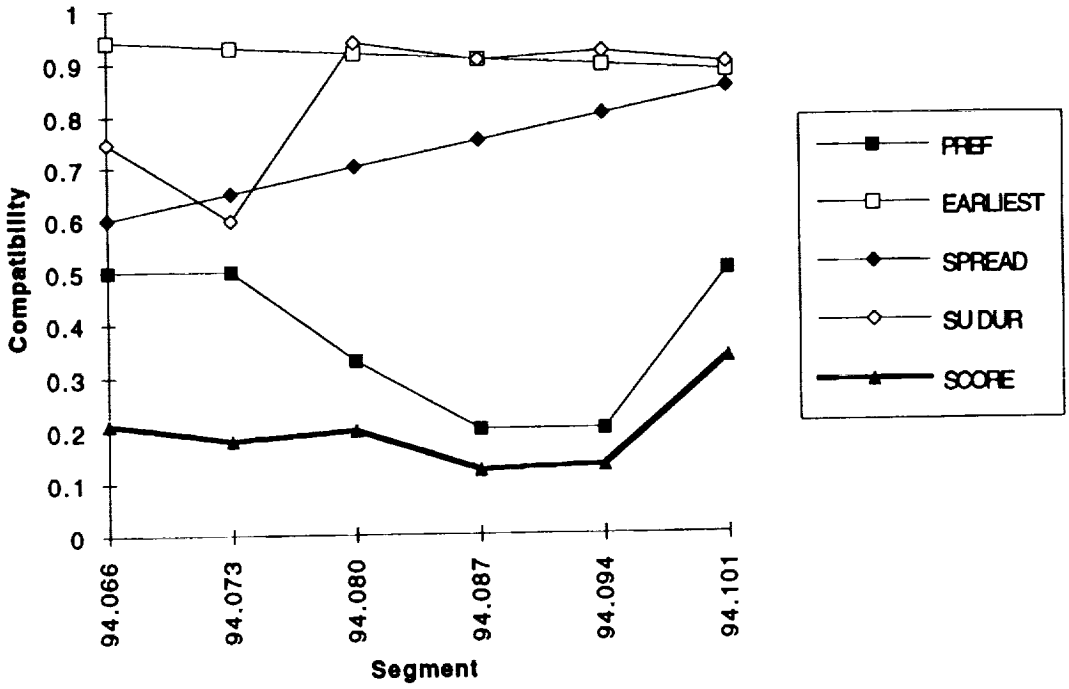


Figure 3. Four criterion compatibility values and the aggregate score (the product) for one scheduling unit are plotted as a function of time segment.

Figure 3 shows how the dominant Preference criterion has its highest value (0.5) in several different weeks. The score reaches its most favorable at 94.101 when the secondary criteria (Earliest, Spread, Level SU Dur) each contributes positively. These criteria can be considered tie-breakers and it is the presence of several weeks that have equally high Preference that allows the meta-scheduling criteria to take effect.

8. Discussion

CASL has demonstrated usefulness in generated long range plans for HST and the tests described in this report indicate a good capability in tailoring a plan according to specific high-level goals. Several important issues are discussed here:

1. Support for the Meta-Scheduling Hypothesis exists in the tests described. A mechanism has been implemented that can improve important scheduling criteria (earliest, su duration, etc.) while producing minimal changes to the average preference of SU placement.

2. CASL currently has a Random Criterion which can introduce non-determinism into the algorithm. Running the algorithm with this criterion activated will generate different results possibly finding better overall solutions. Traub (1994) states that "with a random selection of points, the computation complexity is at most on the order of the reciprocal of the square of the error threshold ($1/\varepsilon^2$)." Testing the usefulness of this criterion is planned.

3. Complex aggregation function is missing. The aggregation of compatibility values by a single function was adequate for CASL. However, a decision tree is a more powerful means of exactly specifying how scores combine.

4. The MCN system is a shell and therefore can be applied to other applications. Currently, there are a number of potential candidates for such application including the intelligent combination of constraint PCFs.

Acknowledgements

The authors would like to thank the following persons for their ideas concerning this work: Mark Giuliano, Mark Johnston, Anthony Krueger, Michael Lucks, Melissa McGrath, Glenn Miller, Mary-Louise Shea.

References

Garey, M., and Johnson, D. (1979). *Computers and Intractability.* New York: W. H. Freeman and Co.

Hall, D., ed. (1982), *The Space Telescope Observatory*, NASA CP2244.

M. Lucks, Detecting Opportunities for Parallel Observations on the Hubble Space Telescope, in *Proceedings of the 1992 Goddard Conference on Space Applications of Artificial Intelligence*, NASA Goddard Space Flight Center, Greenbelt, MD, pp. 22-24. Also in *Telematics and Informatics*, 9, 3-4, November/December 1992.

M. Lucks, and I. Gladwell, "Automated Selection of Mathematical Software," *ACM Transactions on Mathematical Software* 18:1, March 1992, pp. 11-34.

Miller, G., Johnston, M., Vick, S., Sponsler, J., and Lindenmayer, K. (1988). Knowledge Based Tools for Hubble Space Telescope Planning and Scheduling: Constraints and Strategies, in *Proc. 1988 Goddard Conference on Space Applications of Artificial Intelligence*; reprinted in *Telematics and Informatics* 5, p. 197 (1988).

Minton, S., Mark D. Johnston, Andrew B. Philips, Philip Laird (1992). Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence* 58 (1992), pp. 161-205.

Johnston, M. (1990). SPIKE: AI Scheduling for NASA's Hubble Space Telescope. *Proc of the Sixth IEEE Conference on Artificial Intelligence Applications* (March, 1990).

Sponsler, Jeffrey L., Mark D. Johnston, Glenn Miller, Anthony Krueger, Michael Lucks, Mark Giuliano (1991). An AI Scheduling Environment for the Hubble Space Telescope. *Proc AIAA Computing in Aerospace 8*, Baltimore, MD (October, 1991), pp. 14-24.

Traub J. and H. Wozniakowski. Breaking Intractability. *Scientific American* (January, 1994), pp. 102 - 107.

PROPAGATING ORIENTATION CONSTRAINTS FOR THE HUBBLE SPACE TELESCOPE

Ashim Bose
bose@stsci.edu
and
Andy Gerb
gerb@stsci.edu

Space Telescope Science Institute
3700 San Martin Drive
Baltimore, MD 21218.

Abstract

An observing program on the Hubble Space Telescope (HST) is described in terms of exposures that are obtained by one or more of the instruments onboard the HST. Many requested exposures might specify orientation requirements and accompanying ranges. Orientation refers to the amount of roll (in degrees) about the line of sight. The ranges give the permissible tolerance (also in degrees). These requirements may be (i) absolute (in relation to the celestial coordinate system), (ii) relative to the nominal roll angle for HST during that exposure, or (iii) relative (in relation to other exposures in the observing program).

The TRANSformation expert system converts proposals for astronomical observations with HST into detailed observing plans. Part of the conversion process involves grouping exposures into higher level structures based on exposure characteristics. Exposures constrained to be at different orientations cannot be grouped together. Because relative orientation requirements cause implicit constraints, orientation constraints have to be propagated. TRANS must also identify any inconsistencies that may exist so they can be corrected. We have designed and implemented an orientation constraint propagator as part of TRANS. The propagator is based on an informal algebra that facilitates the setting up and propagation of the orientation constraints. The constraint propagator generates constraints between directly related exposures, and propagates derived constraints between exposures that are related indirectly. It provides facilities for path-consistency checking, identification of unsatisfiable constraints, and querying of orientation relationships. The system has been successfully operational as part of TRANS for over seven months. The solution has particular significance to space applications in which satellite/telescope pointing and attitude are constrained and relationships exist between multiple configurations.

1 Introduction

This paper discusses the use of constraint satisfaction technology to solve the problem of propagating constraints that determine the amount of roll of a spacecraft/telescope about the line of sight. The solution is geared specifically to the needs of the Hubble Space Telescope (HST), but is general enough to be useful for other satellite applications. Section 1 provides introductions to constraint propagation, TRANSformation (the expert system within which this problem is solved), and the problem domain. Section 2 describes the problem in detail and the implications of this problem on the rest of TRANS. Section 3 discusses the solution that has been incorporated within TRANS. Section 4 includes a small orientation constraint problem and an enumeration of the important steps in arriving at a solution. Section 5 deals with implementation and experience in running the system.

1.1. Constraint Propagation

A large number of problems in AI and other areas of computer science can be viewed as special cases of the Constraint-Satisfaction Problem (CSP). The basic CSP can be formulated as follows (Kumar 92): We are given a set of variables, a finite and discrete domain for each variable, and a set of constraints. Each constraint is defined over some subset of the original set of variables and limits the combinations of values that the variables in this subset can take. The goal is to find one or more assignments of values to all variables that satisfies all the constraints. We restrict discussion to CSPs in which each constraint is either unary or binary. These are referred to as binary CSPs.[1]

A CSP can be solved using the generate-and-test paradigm. In this paradigm, all combinations of variable instantiations are systematically generated and tested to see if any satisfy all the constraints. Enhancements to this basic paradigm make it more efficient. For instance, in the backtracking paradigm, variables are instantiated sequentially. If an instantiation violates any of the constraints, backtracking is performed to the most recently instantiated variable that still has alternatives available. Although backtracking is strictly better than the generate-and-test-method, its run-time complexity for most non-trivial problems is still exponential. Thrashing, one of the problems with simple back-tracking, can be attributed in many cases to the lack of arc consistency (Mackworth 77). Arc-consistency algorithms have been studied in Mackworth 77, Mackworth and Freuder 85, Mohr and Henderson 86, Han and Lee 88, Chen 91, Bessiere and Cordier 93. In this scheme, constraints between different variables are propagated to derive a simpler problem. In some cases (depending on the problem and the degree of constraint propagation applied), the resulting CSP is so simple that its solution can be found without search. Although any n-variable CSP can always be be solved by achieving n -consistency, this approach is usually even more expensive than simple backtracking. A lower-degree consistency (that is, K-consistency for $K < n$) does not eliminate the need for search except for certain kinds of problems. In Jegou 93, the author uses the concepts of width of hypergraphs and hyper-K-consistency to derive a theorem defining a sufficient condition for consistency of general CSPs.

Several CSP algorithms have been designed for space applications. In order to solve the HST scheduling problem, which is a complex task where between ten thousand and thirty thousand astronomical observations must be scheduled each year subject to a great variety of constraints, Minton and Johnston 90 describe a heuristic repair method called Minimizing-Conflicts, that was inspired by a neural network method described in Adorf and Johnston 90. Miller et al. 88, describe temporal constraints in terms of suitability functions. The suitability function framework has been used successfully as a basis for the SPIKE long-range scheduling system for the HST, as discussed by Johnston and Miller 91. Gerb et al. 92, describe a method based on suitability functions to represent temporal relationships, which is used within the TRANSformation expert system for processing HST proposals.

1.2. TRANSformation - The Big Picture

The TRANSformation expert system (TRANS) converts *Proposals* for astronomical observations with the HST into detailed observing plans. It encodes expert knowledge to solve problems faced in planning and commanding HST observations to enable their processing by the Science Operations Ground System (SOGS). Among these problems are determining an acceptable order of executing observations, hierarchical grouping of observations to enhance efficiency and schedulability, inserting extra observations when necessary, and providing

[1] A CSP with n-ary constraints can be converted to an equivalent binary CSP (Rossi et al. 89).

parameters for commanding HST instruments (Gerb 91). TRANS is currently an operational system and plays a critical role in the HST ground system.

1.3. The Domain - Controlling Spacecraft Roll

An observing program on the HST is described in terms of exposures that are to be obtained by one or more of the instruments that the HST has on board. These exposures have characteristics that are used by TRANS in its decision making. Many requested exposures might specify orientation requirements. Orientation refers to the amount of roll (in degrees) of the spacecraft about the line of sight. Orientation requirements are specified as ranges that incorporate tolerances. These requirements may be (i) absolute (in relation to the celestial coordinate system), (ii) relative to the nominal roll angle for HST during the exposure, or (iii) relative (in relation to other exposures in the observing program). Constraints between exposures arise due to (iii). These have to be propagated and the constraint network made arc-consistent before TRANS makes its orientation-based decisions.

2 The Problem - Orientation Constraints

As already pointed out in the previous section, there are three kinds of observation requirements that an observer may specify for an exposure[2].

Absolute orientation requirements [(i) above] are of the form:
REQUIREMENT: A ORIENT 10 +/ 3D
EXPLANATION: The HST orientation for exposure A is to be within an interval of +7 and +13 degrees in relation to the celestial coordinate system.

Orientation requirements relative to the nominal [(ii) above] are of the form:
REQUIREMENT: X ORIENT 6 +/- 1D FROM NOMINAL
EXPLANATION: The orientation for exposure X should be within an interval of +5 and +7 degrees from the nominal orientation of the HST.

Orientation requirements that are relative [(iii) above] can be further sub-divided into the following categories[3]:
REQUIREMENT: X SAME AS Y
EXPLANATION: An exposure X has to be at the same orientation as Y.

REQUIREMENT: A ORIENT 20 +/- 5D FROM B
EXPLANATION: A's orientation has to be at least +15 and at most +25 degrees from B.

TRANS organizes exposures into higher level groups for efficiency and schedulability. There are several rules that are used to determine if two exposures can be grouped (or merged) together. One of these states that two exposures that have a non-free orientation constraint between them cannot be merged (a free constraint is one with a slack of 360 degrees - it has no constraining effect). Hence, during the TRANS merging phase, TRANS needs a way to find out about the existence of a direct or indirect (derived) constraint between any two exposures that are

---

[2]The inequalities of orientation angles and ranges arising from these requirements are stated in Appendix 1.

[3]The actual syntax for these requirements, as used in the Proposals is slightly different. In reality, the SAME AS requirement can arise out of several different requirements.

being considered for merging. A direct constraint between two exposures exists due to the ORIENT FROM requirement. An indirect constraint between two exposures might come about through constraint propagation. For example, consider the following requirements:
A ORIENT 20 +/- 5D FROM B ...(1)
B ORIENT -10 +/- 2D FROM C ...(2)
This implies that there is an indirect constraint between A and C since they are both directly constrained to B. The indirect constraint between A and C can be derived from (1) and (2) by summing up the median values and ranges of the constraints between A and B (1), and B and C (2):
A ORIENT 10 +/- 7D FROM C ...(3)
There may be more than one constraint between two exposures. For example, in addition to the indirect constraint between A and C (3), there may also exist a direct constraint of the form:
A ORIENT 15 +/- 5D FROM C ...(4)
Now, the real constraint between A and C will have to be derived from (3) and (4). In general, the real constraints between two exposures can only be ascertained after all direct and indirect constraints between them have been considered.

Also, an observer may specify requirements that are inconsistent with each other. Consider the requirements:
A ORIENT 20 +/- 5D FROM B ...(1)
B ORIENT -10 +/- 2D FROM C ...(2)
A ORIENT -20 +/- 5D FROM C ...(5)
The direct constraint between A and C as a result of (5) is inconsistent with the indirect constraint (3) derived from (1) and (2).

Another form of inconsistency is due to cyclic dependencies. Consider:
X ORIENT 10 +/- 2D FROM Y
Y ORIENT 12 +/- 4D FROM X
It is impossible to satisfy both requirements simultaneously. On the other hand, not all cyclic dependencies are inconsistent. For example:
X ORIENT 10 +/- 2D FROM Y
Y ORIENT -12 +/- 4D FROM X
can be satisfied.

Inconsistencies resulting from faulty orientation requirements have to be flagged and reported. This is so that the faulty requirements in the Proposal can be altered and the Proposal TRANSed to produce correct output.

3 The Solution within TRANS

The Orient Constraint Propagation module within TRANS sets up and propagates constraints between related exposures. A basic algorithm to achieve this would be the following:

```
arcs-to-check <- all directly related exposure pairs
while arcs-to-check {
        changes <- nil
        for-each arc in arcs-to-check
                changes <- old changes + forward and backward induced arcs from arc
        arcs-to-check <- changes}
```

The actual propagation takes place inside the for-each loop as constraints between exposures that are indirectly related through a common exposure are set up. For example, if there is an arc between exposures i and j, and there is another arc between j and k, an arc is now set up between

i and k. If an arc already exists between i and k, then the resulting constraint is the intersection of the old and propagated constraints. Note that the absolute and nominal ranges for each exposure, as propagated through the constraint, are noted as soon as the new constraint is set up.

This is ordinarily an $O(n^3)$ operation in the number of related exposures. The number of exposures processed by TRANS can be large (over 100) in some cases. Hence, modifications to the basic algorithm are necessary to ensure that processing time is reasonable.[4]

3.1. Representing Angle Intervals and their Intersections

Angle ranges occur in all three kinds of orientation requirements. They are expressed in Proposals as a median value accompanied with a tolerance (see Appendix 2). A more convenient representation is that of a pair, with the first part being the lower limit and the second part being twice the tolerance (or upper limit - lower limit).[5] So, the angle interval 20 +/- 5D can be expressed as (15, 10). Interval intersections are easy to compute.
If $R_1 = (a_1, b_1)$, and $R_2 = (a_2, b_2)$, then
$R_3 = R_1 \cap R_2 =$
(a_3, b_3) such that

$$a_3 = \max(a_1, a_2) \text{ and } b_3 = \min(a_1+b_1, a_2+b_2) - a_3 \text{ and } b3 \geq 0,$$

null (an empty solution set) otherwise.
So, if $R_1 = (15, 10)$, and $R_2 = (12, 8)$, then $R_1 \cap R_2 = (15, 5)$.

3.2. Representing a Direct Relationship as a Constraint

A direct ORIENT FROM relationship between two exposures is represented as a constraint interval. For example,
A ORIENT 20 +/- 5D FROM B
gives rise to the constraint
$C_{AB} = (15, 25)$.
When A is constrained to be at a certain angle interval from B, B is also constrained to be at a certain interval from A. This gives rise to an inverse constraint C_{BA}. This will be defined in Section 3.4.

3.3. Propagating Angle intervals Through Constraints

If exposure B has an angle interval $R_B = (x_B, y_B)$, and there exists a constraint between exposures A and B, $C_{AB} = (l, r)$, then the angle interval for exposure A, R_A, propagated from R_B through constraint C_{AB},
$R_A = \mathfrak{P}(C_{AB}, R_B) = (l + x_B, r + y_B)$,
where $\mathfrak{P}(C_{AB}, R_B)$ denotes the propagation of the angle interval from B to A through the connecting constraint C_{AB}.

---

[4]Note that setting up and propagating orientation constraints is only a small part of TRANS processing. For the most part however, TRANS runs in O(n).

[5]A more obvious representation would be a pair, where the first member is the lower limit, and the second member is the upper limit. However, in this representation, it is impossible to distinguish a range x+/-0 degrees from x+/-360 degrees.

Note that R_A may already have an angle interval of its own.
Then,
$R_A^{new} = R_A^{old} \cap \Psi(C_{AB}, R_B)$.

3.4. Computing Inverse Relationships

If exposure B has an angle interval $R_B = (x_B, y_B)$, and there exists a constraint between exposures A and B, $C_{AB} = (l, r)$, then the inverse of this constraint C_{BA}, is defined to be such that if $C_{AB} = (l, r)$, then $C_{BA} = C_{AB}^{-} = (-(l+r), r)$.
Let $R_B = (x_B, y_B)$, $C_{AB} = (l, r)$, then
$R_A = \Psi(C_{AB}, R_B) = (x_B + l, y_B + r)$.

Now, let's propagate a range from A to B.
$C_{BA} = C_{AB}^{-} = (-l - r, r)$.
R_B (from range propagated from A) = $R_B^{new} = \Psi(C_{BA}, R_A) = (x_B - r, y_B + 2r)$.
Hence, $R_B^{new} \neq R_B^{old}$.
But,
$R_B^{new} \cap R_B^{old} = (x_B, y_B) \cap (x_B - r, y_B + 2r) = (x_B, y_B)$.
Hence, even though the range propagated over the inverse constraint has more slack, it doesn't really affect the original range.

3.5. Deriving Constraints for Indirect Relationships

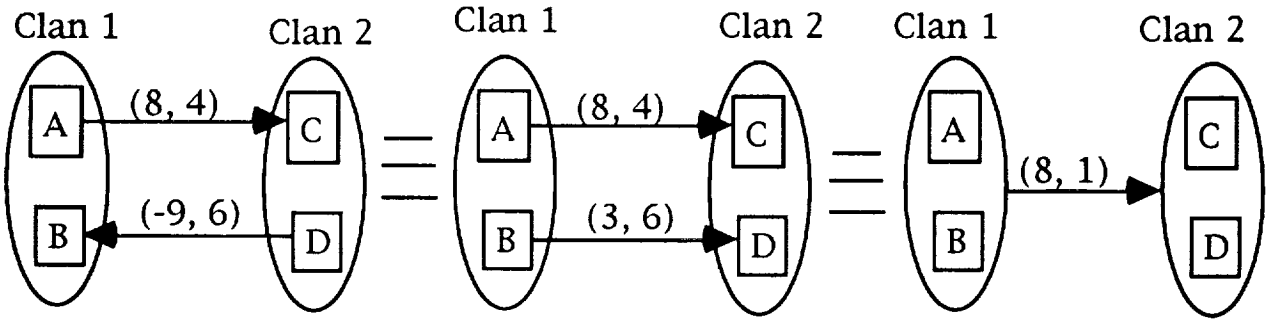
An indirect relationship exists between two exposures A and C when they are both related to a third exposure B. Given two constraints
$C_{AB} = (l_1, r_1)$, and $C_{BC} = (l_2, r_2)$,
$C_{AC} = (l_3, r_3)$ is defined to be such that
$l_3 = l_1 + l_2$, and $r_3 = r_1 + r_2$.
Consider:
A ORIENT 10 +/- 2D FROM B
B ORIENT 12 +/- 2D FROM C.
This gives rise to the constraints
$C_{AB} = (8, 4)$, and $C_{BC} = (10, 4)$. $C_{AC} = (18, 8)$.
Note that as constraints are propagated through a network, the slack (or tolerance - the second part of the constraint pair) can only increase or stay the same. Since the slack is with respect to planar angles, it should be noted that the slack on any constraint cannot exceed 360 degrees. Also, once the slack has reached that limit, the constraint becomes a *free* constraint, i.e. it does not have any constraining effect on the angle intervals of the concerned clan.

3.6. An Informal Algebra for Orientation Constraint Propagation

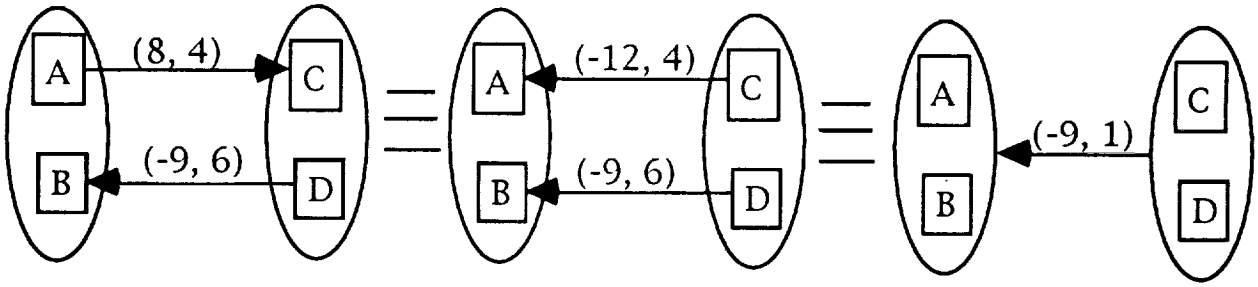
We make modifications to the algorithm mentioned above to make constraint propagation more efficient. These modifications will be described in the following paragraphs.

The SAME AS requirement can be exploited to reduce the value of n not only in the time and space complexity of the algorithm, but also in the hierarchical ordering of exposures. This is because exposures that have the SAME AS requirement are required to be at the same orientation and can be treated as a group called a *clan*. All exposures that belong to a clan are called its *members*. A clan can have one or more members. By definition, there can be no non-free

ORIENT FROM constraint between two members of the same clan. A constraint between two clans is derived from an ORIENT FROM relationship between its members. There may initially be more than one constraint between two clans. Consider the following case:
A SAME AS B
A ORIENT 10 +/- 2D FROM C
C SAME AS D
D ORIENT -12 +/- 3D FROM B
Exposures A and B belong to the same clan (say clan 1), as do C and D (say clan 2). The following constraints exist between clans 1 and 2 due to the ORIENT FROM requirements:
$C_{1A,2C} = (8, 4)$
$C_{2D,1B} = (-9, 6) \Rightarrow C_{1B,2D} = (3, 6)$ (using the inverse relationship formula - Sec. 3.3)
The actual constraints between clans 1 and 2 can be deduced to be:
$C_{12} = C_{1A,2C} \cap C_{1B,2D} = (8, 1)$, and
$C_{21} = C_{12}^{-} = (-9, 1)$.



Row 1: Steps in generating the constraint C_{12}



Row 2: Steps in generating the constraint C_{21}

Figure 1: Setting Up and Simplifying Constraints Between Clans Due to Related Member Exposures (either by traversing Row 1 or Row 2).

In Figure 1, we illustrate how constraints between clans are set up. The constraints can be set up in either direction (by traversing either the first or the second row). Once a constraint between two clans is set up, its inverse yields the constraint in the other direction. Hence there is no need to traverse both rows.

The fact that constraints can be generated between indirectly related clans (via member exposures) helps reduce the propagation time, as the network becomes "stable" faster.

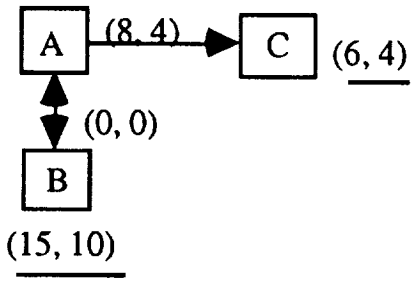
Any interval of the form (x . 360) where x is any real number, is a *free* interval. All free intervals are equivalent. Clans have a default interval of (0 . 360) for both absolute and nominal ranges. [6] The absolute and nominal ranges of a clan can change under the following conditions:

• A member exposure has an absolute or nominal range. If R_c^{old} is the old range for the clan and R_{mem} is the range for the member, the new range for the clan R_c^{new} is:
$R_c^{new} = R_c^{old} \cap R_{mem}$.

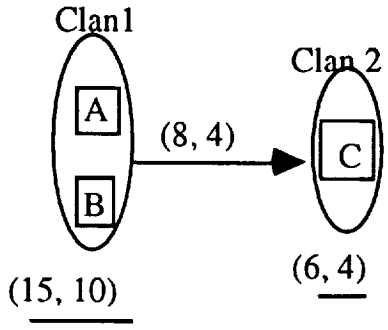
• A range is propagated from a related clan. If R_2 is the range for clan 2 and the constraint between clans 1 and 2 is C_{12}, then,
$R_c^{new} = R_c^{old} \cap \mathfrak{P}(R_2, C_{12})$, where $\mathfrak{P}(R,C)$ is the result of propagating R over C.
If R = (x, y), and C = (l, r), then $\mathfrak{P}(R, C) = (x + l, y + r)$.

Consider:
A SAME AS B
A ORIENT 10 +/- 2D FROM C
B ORIENT 20 +/- 5D
C ORIENT 8 +/- 2D
From the above requirements, we have:
Clan 1 = {A, B}, Cian 2 = {C}, $C_{1A,2C} = (8, 4)$.

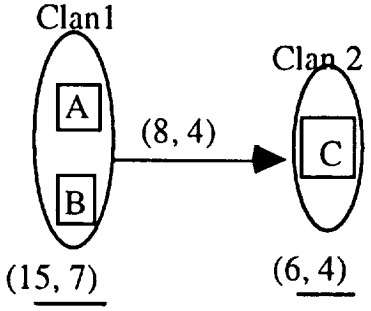
[6] Note that there is a need to keep track of two sets of intervals for each clan - its absolute interval which is obtained directly or indirectly from requirement (i), and its nominal interval which is obtained directly or indirectly from requirement (ii).



The problem represented pictorially. Absolute ranges are underlined.



Before Propagation



After Propagation

Figure 2: Propagation of an angle interval Between Two Clans that are Related Through Member Exposures.

Figure 2 illustrates the propagation of an angle interval from Clan 2 to Clan 1 through the constraint C_{12}.

$R_1^{initial} = R_A \cap R_B = (0, 360) \cap (15, 10) = (15, 10)$. $R_2 = (6, 4)$.

$R_1^{new} = R_1^{old} \cap \Re(C_{12}, R_2) = (15, 10) \cap (14, 8) = (15, 7)$.

A constraint network is stable when the absolute and nominal angle intervals of all clans are consistent with the constraints connecting them. This happens only when the constraints between clans cease to change.

3.7. Interfacing with the Rest of TRANS

As has already been pointed out, the results of propagating orientation constraints are used within TRANS as one of many inputs to decide on hierarchical exposure groupings. These decisions are made through *merging rules*. The merging rule related to orientations of exposures states that two exposures can be merged only when their absolute and nominal angle intervals are identical and they can be at the same orientation.[7] These requirements will hold only when one of the following conditions is met:

- Both exposures belong to the same clan.
- The exposures belong to different clans that have identical absolute and nominal angle intervals, and either there is no constraint, or a free constraint, or a constraint that includes zero in its interval, between them.

Defective orientation requirements in the Proposal result in null (infeasible) angle intervals for one or more clans and/or null constraints between two or more clans. Warning messsages are generated when these are detected, other orientation constraints continue to be propagated, and TRANS uses the results of these propagations to make its merging decisions. In this manner, TRANS products that are not dependent on the faulty orientation requirements can still be used, but the warnings alert the TRANS user to correct the faulty requirements and TRANS the Proposal again, to obtain fully correct TRANS products.

4 A Small Example

We will now illustrate some of the important aspects of orient constraint propagation with the help of a small example. Consider the following orientation requirements:

A ORIENT 20 +/- 5D
B SAME AS A, C
B ORIENT 10 +/- 2D FROM D
C ORIENT 9 +/- 2D FROM F
E ORIENT -10 +/- 2D FROM A
E SAME AS F

The problem is represented pictorially in Figure 3a. Clan 1 = {A, B, C}, Clan 2 = {D}, Clan 3 {E, F}. $R_1 = (15, 10)$, $R_2 = (0, 360)$, $R_3 = (0, 360)$. $C_{1B,3D} = (8, 4)$, $C_{1C,2F} = (3, 4)$, $C_{2E,1A} = (-12, 4)$.

[7] Organizing exposures into groups enhances viewing efficiency since the orientation of the spacecraft is the same for all exposures in the group. However, merging exposures that could possibly be at different orientations, into the same group might cause scheduling problems downstream due to the inflexibility that is introduced. Also, the existence of a non-free constraint between two exposures precludes them from being in the same group.

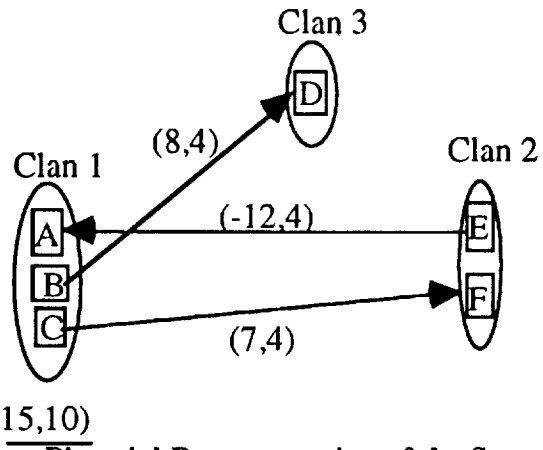


Figure 3a: Pictorial Representation of the Sample Problem

Consider the pair Clan 1, Clan 2.

$C_{21} = C_{2E,1A} \cap C_{1C,2F}^{\sim} = (-12, 4) \cap (-11, 4) = (-11, 3)$

$C_{12} = C_{21}^{\sim} = (8, 3)$

$R_2 = \Phi(C_{21}, R_1) = (4, 13)$

Consider the pair Clan 1, Clan 3

$C_{13} = (8, 4)$

$C_{31} = (-12, 4)$

$R_3^{new} = R_3^{old} \cap \Phi(C_{31}, R_1) = (-4, 17) \cap (3, 14) = (3, 10)$

Now, we propagate constraints.

Deducing the constraint between the pair (Clan 3, Clan 2) from the constraints between the pairs (Clan 3, Clan 1) and (Clan 1, Clan 2),

Clan 3 <=> Clan 1 <=> Clan 2

$C_{32} = (C_{31} + C_{12}) = (-4, 7)$ Change

$C_{23} = (-3, 7)$

$R_3^{new} = R_3^{old} \cap \Phi(C_{32}, R_2) = (3, 10) \cap (0, 20) = (3, 10)$

Clan 2 <=> Clan 3 <=> Clan 1

$C_{21new} = C_{21old} \cap (C_{23} + C_{31}) = (-11, 3) \cap (-15, 11) = (-11, 3)$ - No Change

Clan 3 <=> Clan1 <=> Clan2

$C_{32} = (C_{31} + C_{12}) = (-4, 7)$ No Change

So, the final solution is:

$R_1 = (15, 10), R_2 = (4, 13), R_3 = (3, 10)$

5 Implementation and Experience

The Orient Constraint Propagator, like the rest of TRANS is implemented in LISP, using an object-oriented paradigm (CLOS). Clans are objects, absolute and nominal ranges and constraints are dotted pairs of real numbers. Pairs of related clans, and the constraints between

them, are stored in a hash-table. In order to have a uniform representation for angle intervals, all angles are positive and modulo 360.

The implementation has met all expectations related to run-time efficiency and correctness since it was installed as a TRANS module over seven months ago. It was mentioned in Section 3.5 that the slack in the intervals of constraints can grow to 360 degrees, thus causing propagated constraints to become free. Since in reality, the slack in the intervals of directly related exposures is of the order of ten degrees, this can only happen when there is a long ($\approx$ 36) "chain" of related exposures with no interactions with other constraints (note that when there is more than one constraint between two clans, we take their intersection, which has a "constraining effect" on the slack of the resulting constraint). Since the number of exposures in an ORIENT FROM chain is seldom more than five, we have not encountered problems with free constraints.

6 Conclusions

HST observations, expressed as exposures may be related to each other via relationships regulating the roll of the telescope. In this paper, we have formulated these relationships as a CSP, and described a method to "solve" it using an informal algebra. Solving the problem entails determining the ranges of roll for each exposure so that all roll relationships are satisfied. The solution has been implemented using object-oriented technology as a sub-system of the TRANSformation expert system. It has been operational for over seven months and has met all expectations.

References

Adorf, H. M., and Johnston, M. D. 1990. A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems. Proceedings of the International Joint Conference on Neural Networks, San Diego, CA.

Chen, Y. 1991. Improving Han and Lee's Path Consistency Algorithm. In Proceedings of the Third International Conference on Tools for AI, 346-350. Washington, D.C.: IEEE Computer Society.

Gerb, A. 1991. TRANSformation Reborn: A New Generation Expert System for Planning HST Operations. *Telematics and Informatics* v8n4:283-295.

Gerb, A., Henry, R., and Johnston, M. 1992. Applying Constraint Propagation Technology to an Expert System Planning Hubble Space Telescope Observations. In Proceedings of the IEEE International Conference on Developing and Managing Intelligent System Projects, 131-138. Washington, D.C.

Han, C.C., and Lee, C. H. 1988. Comments on Mohr and Henderson's Path-Consistency Algorithm. *Artificial Intelligence* 36: 125-130.

Johnston, M., and Miller, G. 1991. Long Range Science Scheduling for the Hubble Space Telescope. Proceedings of the 1991 Goddard Conference on Space Applications of Artificial Intelligence, Greenbelt, MD.

Kumar, V. 1992. Algorithms for Constraint-Satisfaction Problems: A Survey. AI Magazine v13(Sp 92):32-44.

Mackworth, A. K. 1977a. Consistency in Networks of Relations. *Artificial Intelligence* 8(1): 99-118.

Mackworth, A. K., and Freuder, E. 1985. The Complexity of Some Polynomial Network-Consistency Algorithms for Constraint-Satisfaction Problems. *Artificial Intelligence* 25:65-74.

Miller, G., Johnston, M., Vick, S., Sponsler, J., and Lindenmayer, K. 1988. Knowledge-based Tools for Hubble Speace Telescope Planning and Scheduling, Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence, Greenbelt, MD.

Minton, S., Johnston, M. D., Phillips, A., and Laird, P. 1990. Solving Large-Scale Constraint-Satisfaction and Scheduling Problems Using a Heuristic Repair Method. Proceedings of the Eigth National Conference on Artificial Intelligence, 1990:17-24. Menlo Park, Calif.

Mohr, R., and Henderson, T.C. 1986. Arc and Path Consistency Revisited. *Artificial Intelligence* 28:225-233.

Rossi, F.; Petrie, C.; and Dhar, V. 1989. On the Equivalence of Constraint-Satisfaction Problems, Technical Report ACT-AI-222-89, MCC Corp., Austin, Texas.

APPENDIX

(1) ORIENTATION RELATIONSHIPS BETWEEN EXPOSURES:

In order to determine the merging of exposures based on their orientation requirements, the orientation relationship between pairs of exposures needs to be computed. In the following discussion, we will call the orientation of exposure A Oa. Orientation relationships between exposures come about in six ways:

a. Default - Normally two exposures may be 0d +/- 180d from each other (i.e.they can be at any two orientations.

b. ORIENT FROM - If two exposures A and B are related by A ORIENT <angle> +/- <range> FROM B, the relationship between Oa and Ob should be limited as follows:

Ob + <angle> - <range> <= Oa <= Ob + <angle> + <range>

Oa - <angle> - <range> <= Ob <= Oa - <angle> + <range>

c. SAME ORIENT/SAME POS - If two exposures A and B have SAME POS/ORIENT for A as B. The relationship between Oa and Oc should be limited as follows:

Oa = Ob

d. Specified orientation - If exposure A has ORIENT <angle1> +/- <range1> and exposure B has ORIENT <angle2> +/- <range2>, or exposure A has ORIENT <angle1> +/- <range1> FROM NOMINAL and exposure B has ORIENT <angle2> +/- <range2> FROM NOMINAL, then the relationship between A and B should be limited as follows:

Ob - <range2> - <angle2> + <angle1> - <range1> <= Oa <= Ob + <range2> + <angle1> + <range1> - <angle2>

Oa - <range1> - <angle1> + <angle2> - <range2> <= Ob <= Oa + <range1> + <angle2> + <range2> - <angle1>

e. Merging - If exposure A is merged into the same scheduling unit (a higher level structure) as exposure B, then the relationship between A and B should be limited as follows:

Oa = Ob

f. Propagation of constraints - If exposure B is limited by the orientation of A by Ob = Oa + <angle1> +/- <range1> and exposure C is limited by the orientation of B by Oc = Ob + <angle2> +/- <range2>, then the relationship between Oa and Oc should be limited as follows:

Oa + <angle1> + <angle2> - <range1> - <range2> <= Oc <= Oa + <angle1> + <angle2> + <range1> + <range2>

Oc - <angle1> - <angle2> - <range1> - <range2> <= Oa <= Oc - <angle1> - <angle2> + <range1> + <range2>

If more than one of the above apply to two exposures A and B, the relationship between Oa and Ob should be the intersection of the multiple rules. If this process leads to pairs of exposures A and B where there is no legal angle or range between Oa and Ob, TRANS should generate an error.

(2) ORIENTATION RANGES

Exposures have two sets of limits where they can be scheduled, a nominal range and an absolute range. These ranges are set as follows:

a. An exposure with an ORIENT <angle> +/- <range> has an absolute range extending from <angle> - <range> to <angle> + <range>.

b. An exposure with an ORIENT <angle> +/- <range> FROM NOMINAL has a nominal range extending from <angle> - <range> to <angle> + <range>.

c. An exposure without an ORIENT <angle> +/- <range> has an absolute range from zero to 360.

d. An exposure without an ORIENT <angle> +/- <range> FROM NOMINAL has a nominal range from zero to 360.

e. If an exposure B is limited by the orientation of A by Ob = Oa + <angle1> +/- <range1>, where A has an an absolute range from <low-angle> to <high-angle> then B has an absolute range from <low-angle> + <angle1> - <range1> to <high-angle> + <angle1> + <range1>.

If more than one of the above rules apply to an exposure A, the ranges of A should be the intersection of ranges specified by the multiple rules. If the resulting interval is empty for any given exposure, TRANS should generate an error.

N94-35050

A Linguistic Geometry for Space Applications

Boris Stilman
Department of Computer Science & Engineering, University of Colorado at Denver,
Campus Box 109, Denver, CO 80217-3364, USA.
Tel. (303) 556-3614, Fax: (303) 556-8369, Email: bstilman@gothic.denver.colorado.edu

Abstract: *We develop a formal theory, the so-called Linguistic Geometry, in order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, and apply them to different systems. This research relies on the formalization of search heuristics of high-skilled human experts, which allow to decompose complex system into the hierarchy of subsystems, and thus solve intractable problems reducing the search. The hierarchy of subsystems is represented as a hierarchy of formal attribute languages. This paper includes a formal survey of the Linguistic Geometry, and new example of a solution of optimization problem for the space robotic vehicles. This example includes actual generation of the hierarchy of languages, some details of trajectory generation and demonstrates the drastic reduction of search in comparison with conventional search algorithms.*

There are many real-world problems where human expert skills in reasoning about complex systems are incomparably higher than the level of modern computing systems. At the same time there are even more areas where advances are required but human problem-solving skills can not be directly applied. For example, there are problems of planning and automatic control of autonomous agents such as space vehicles, stations and robots with cooperative and opposing interests functioning in a complex, hazardous environment. Reasoning about such complex systems should be done automatically, in a timely manner, and often in a real time. Moreover, there are no highly-skilled human experts in these fields ready to substitute for robots (on a virtual model) or transfer their knowledge to them. There is no grand-master in robot control, although, of course, the knowledge of existing experts in this field should not be neglected – it is even more valuable. It is very important to study human expert reasoning about similar complex systems in the areas where the results are successful, in order to discover the keys to success, and then apply and adopt these keys to the new, as yet, unsolved problems. The question then is what language tools do we have for the adequate representation of human expert skills? An application of such language to the area of successful results achieved by the human expert should yield a *formal, domain independent knowledge* ready to be transferred to different areas. Neither natural nor programming languages satisfy our goal. The first are informal and ambiguous, while the second are usually detailed, lower-level tools. Actually, we have to learn how we can formally represent, generate, and investigate a *mathematical model* based on the *abstract images* extracted from the expert vision of the problem.

There have been many attempts to find the optimal (suboptimal) operation for real-world complex systems. One of the basic ideas is to decrease the dimension of the real-world system following the approach of a *human expert in a certain field*, by breaking the system into smaller subsystems. These ideas have been implemented for many problems with varying degrees of success [1, 2, 15]. Implementations based on the formal theories of linear and nonlinear planning meet hard efficiency problems [4, 12, 17, 22, 25]. An efficient planner requires an intensive use of heuristic knowledge. On the other hand, a pure heuristic implementation is unique. There is no general constructive approach to such implementations. Each new problem must be carefully studied and previous experience usually can not be applied. Basically, we can not answer the question: what are the formal properties of human heuristics which drove us to a successful hierarchy of subsystems for a given problem and how can we apply the same ideas in a different problem domain?

In the 1960's a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky [5], Ginsburg [10], and others. This development provided an interesting opportunity for dissemination of this approach to different areas. In particular, there came an idea of analogous linguistic

representation of images. This idea was successfully developed into syntactic methods of pattern recognition by Fu [8], Narasimhan [16], and Pavlidis [18], and picture description languages by Shaw [23], Feder [6], Rosenfeld [20].

Searching for the adequate mathematical tools formalizing human heuristics of dynamic hierarchy, we have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems [27]. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. The origin of such languages can be traced back to the research on programmed attribute grammars by Knuth [11], Rozenkrantz [21], Volchenkov [36].

A mathematical environment (a "glue") for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson [17], Fikes [7], Sacerdoti [22], McCarthy, Hayes [13, 14], and others based on first order predicate calculus.

To show the power of the linguistic approach it is important that the chosen model of the heuristic hierarchical system be sufficiently complex, poorly formalized, and have successful applications in different areas. Such a model was developed by Botvinnik, Stilman, and others, and successfully applied to scheduling, planning, and computer chess [2].

In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex control systems, we develop a formal theory, the so-called *Linguistic Geometry* [28-35]. This research includes the development of syntactic tools for *knowledge representation* and *reasoning* about large-scale hierarchical complex systems. It relies on the formalization of *search heuristics*, which allow one to decompose complex system into a hierarchy of subsystems, and thus solve intractable problems, reducing the search. These *hierarchical images* were extracted from the expert vision of the problem. The hierarchy of subsystems is represented as a *hierarchy of formal attribute languages* [28, 33].

1 Complex System

A **Complex System** is the following eight-tuple:

$$< \mathbf{X}, \mathbf{P}, \mathbf{R_p}, \{\mathbf{ON}\}, \mathbf{v}, \mathbf{S_i}, \mathbf{S_t}, \mathbf{TR}>,$$

where $\mathbf{X}=\{x_i\}$ is a finite set of *points*; $\mathbf{P}=\{p_i\}$ is a finite set of *elements*; **P** is a union of two non-intersecting subsets P_1 and P_2; $\mathbf{R_p(x, y)}$ is a set of binary relations of *reachability* in X (x and y are from X, p from P); **ON**(p)=x, where ON is a partial function of *placement* from P into X; **v** is a function on P with positive integer values; it describes the *values* of elements. The Complex System searches the state space, which should have initial and target states; $\mathbf{S_i}$ and $\mathbf{S_t}$ are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, which matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from S_i or S_t is described by a certain set of WFF of the form $\{ON(p_j)=x_k\}$; **TR** is a set of operators, TRANSITION(p, x, y), of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

Remove list: ON(p)=x, ON(q)=y;
Add list: ON(p)=y;
Applicability list: $(ON(p)=x)\wedge R_p(x, y)$,

where p belongs to P_1 and q belongs to P_2 or vice versa. The transitions are carried out in turn with participation of elements p from P_1 and P_2 respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets P_1 and P_2. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x, y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from point x to point y. The element q from point y must be withdrawn (eliminated) if p and q belong to the different subsets P_1 and P_2.

The problem of the optimal operation of the System is considered as a search for the optimal

sequence of transitions leading from one of the initial states of S_i to a target state S of S_t.

It is easy to show formally that robotic system can be considered as the Complex System (see below). Many different technical and human society systems (including military battlefield systems, systems of economic competition, positional games) which can be represented as twin-sets of movable units (of two or more opposite sides) and their locations, thus, can be considered as Complex Systems.

With such a problem statement for the search of the optimal sequence of transitions leading to the target state, we could use formal methods like those in the problem-solving system STRIPS [7], nonlinear planner NOAH [22], or in subsequent planning systems. However, the search would have to be made in a space of a huge dimension (for nontrivial examples). Thus, in practice no solution would be obtained.

We devote ourselves to the search for an approximate solution of a reformulated problem.

2 Measurement of distances

To create and study a hierarchy of dynamic subsystems we have to investigate geometrical properties of the Complex System.

A ***map of the set*** **X** relative to the point x and element p for the Complex System is the mapping: $\mathbf{MAP_{x,p}}$: X —> $\mathbf{Z_+}$, (where x is from X, p is from P), which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets of X $\{M^k_{x,p}\}$ (Fig.1):

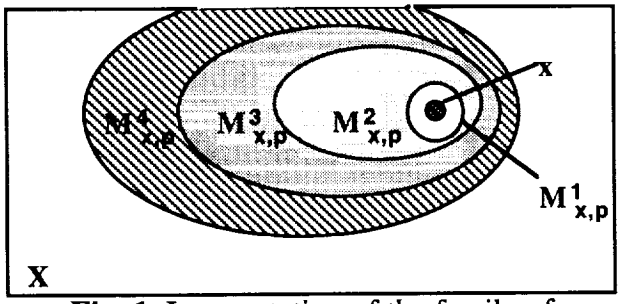


Fig. 1. Interpretation of the family of reachability areas

k=1: $M^k_{x,p}$ is a set of points *m reachable in one step* from x: $R_p(x,m)$=T;

k>1: $M^k_{x,p}$ is a set of points *reachable in k steps and not reachable in k-1 steps,* i.e., points *m* reachable from points of $M^{k-1}_{x,p}$ and not included in any $M^i_{x,p}$ with numbers *i* less than *k*.

Let $MAP_{x,p}(y)=k$, for y from $M^k_{x,p}$ (*number of steps from x to y*). In the remainder points let $MAP_{x,p}(y)=2n$, if y≠x (n is the number of points in X); $MAP_{x,p}(y)=0$, if y=x.

It is easy to verify that the map of the set X for the specified element p from P defines an *asymmetric distance function* on X:

1. $MAP_{x,p}(y) > 0$ for x≠y; $MAP_{x,p}(x)=0$;
2. $MAP_{x,p}(y)+MAP_{y,p}(z) \geq MAP_{x,p}(z)$.

If R_p is a symmetric relation,

3. $MAP_{x,p}(y)=MAP_{y,p}(x)$.

In this case each of the elements p from P specifies on X its *own* ***metric***. Various examples of measurement of distances for robotic vehicles are considered later.

3 Language of Trajectories

This language is a formal description of the set of lowest-level subsystems, the set of different paths between points of the Complex System. An element might follow a path to achieve the goal "connected with the ending point" of this path.

A ***trajectory*** for an element p of P with the beginning at x of X and the end at the y of X (x ≠ y) with a length *l* is a following string of symbols with parameters, points of X:

$$t_0=a(x)a(x_1)\ldots a(x_l),$$

where x_l = y, each successive point x_{i+1} is reachable from the previous point x_i, i.e., $R_p(x_i, x_{i+1})$ holds for i = 0, 1,..., *l*–1; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of trajectories in which p, x, y, and *l* coincide. $\mathcal{P}(t_0)=\{x, x_1, \ldots, x_l\}$ is the set of parameter values of the trajectory t_0.

A ***shortest trajectory*** t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x, end y and element p.

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the shortest trajectories. A general grammar (Table I) and its application to

generating the shortest trajectory for a robotic vehicle will be presented later.

Reasoning informally, an analogy can be set up: the shortest trajectory is analogous with a straight line segment connecting two points in a plane. An analogy to a k-element segmented line connecting these points is called an ***admissible trajectory of degree k***, i.e., the trajectory which can be divided into k shortest trajectories. The admissible trajectories of degree 2 play a special role in many problems. As a rule, elements of the System should move along the shortest paths. In case of an obstacle, the element should move around this obstacle by tracing an intermediate point aside and going to and from this point to the end along the shortest trajectories. Thus, in this case, an element should move along an admissible trajectory of degree 2.

A ***Language of Trajectories*** $L_t^H(S)$ for the Complex System in a state S is the set of all the shortest and admissible (degree 2) trajectories of the length less than H. Different properties of this language and generating grammars were investigated in [32].

4 Languages of Trajectory Networks

After defining the Language of Trajectories, we have new tools for the breakdown of our System into subsystems. According to the ideas presented in [2], these subsystems should be various types of trajectory networks, i.e., the sets of interconnected trajectories with one singled out trajectory called the *main trajectory*. An example of such network is shown in Fig. 2. The basic idea behind these networks is as follows. Element p_0 should move along the main trajectory $a(1)a(2)a(3)a(4)a(5)$ to reach the ending point 5 and remove the target q_4 (an opposite element). Naturally, the opposite elements should try to disturb those motions by controlling the intermediate points of the main trajectory. They should come closer to these points (to the point 4 in Fig. 2) and remove element p_0 after its arrival (at point 4). For this purpose, elements q_3 or q_2 should move along the trajectories $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of element p_0 at point 4. Similarly, element p_1 of the same side as p_0 might try to disturb the motion of q_2 by controlling point 9 along the trajectory $a(13)a(9)$. It makes sense for the opposite side to include the trajectory $a(11)a(12)a(9)$ of element q_1 to prevent this control.

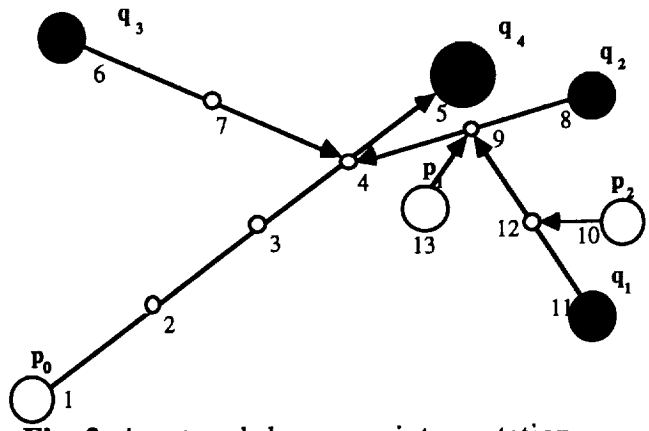


Fig. 2. A network language interpretation.

Similar networks are used for the breakdown of complex systems in different areas. Let us consider a linguistic formalization of such networks. The Language of Trajectories describes "one-dimensional" objects by joining symbols into a string employing reachability relation $R_p(x, y)$. To describe networks, i.e., "multi-dimensional" objects made up of trajectories, we use the relation of *trajectory connection.*

A ***trajectory connection*** of the trajectories t_1 and t_2 is the relation $C(t_1,t_2)$. It holds, if the ending link of the trajectory t_1 coincides with an intermediate link of the trajectory t_2; more precisely t_1 is connected with t_2, if among the parameter values $\mathcal{P}(t_2)=\{y,y_1,\ldots,y_l\}$ of trajectory t_2 there is a value $y_i = x_k$, where $t_1=a(x_0)a(x_1)\ldots a(x_k)$. If t_1 belongs to a set of trajectories with the common end-point, then the entire set is said to be connected with the trajectory t_2.

For example, in Fig. 2 the trajectories $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$ are connected with the main trajectory $a(1)a(2)a(3)a(4)a(5)$ through point 4. Trajectories $a(13)a(9)$ and $a(11)a(12)a(9)$ are connected with $a(8)a(9)a(4)$.

To formalize the trajectory networks we define and use routine operations on the set of trajectories: $C_A^k(t_1,t_2)$, a ***k-th degree of connection***, and $C_A^+(t_1,t_2)$, a ***transitive closure***.

Trajectory $a(11)a(12)a(9)$ in Fig. 2 is connected degree 2 with trajectory $a(1)a(2)a(3)a(4)a(5)$, i.e., $C^2(a(11)a(12)a(9), a(1)a(2)a(3)a(4)a(5))$ holds. Trajectory $a(10)a(12)$ in Fig. 2 is in transitive closure to the trajectory $a(1)a(2)a(3)a(4)a(5)$ because $C^3(a(10)a(12), a(1)a(2)a(3)a(4)a(5))$ holds by means of the chain of trajectories $a(11)a(12)a(9)$ and $a(8)a(9)a(4)$.

A ***trajectory network*** **W** relative to trajectory t_0 is a finite set of trajectories $t_0,t_1,\ldots,t_k$ from the language $L_t^H(S)$ that possesses the following property: for every trajectory t_i from W (i = 1, 2,...,k) the relation $C_W^+(t_i,t_0)$ holds, i.e., each trajectory of the network W is connected with the trajectory t_0 that was singled out by a subset of interconnected trajectories of this network. If the relation $C_W^m(t_i, t_0)$ holds, trajectory t_i is called the ***m negation trajectory***.

Obviously, the trajectories in Fig. 2 form a trajectory network relative to the main trajectory $a(1)a(2)a(3)a(4)a(5)$. We are now ready to define network languages.

A ***family of trajectory network languages*** $\mathbf{L_C(S)}$ in a state S of the Complex System is the family of languages that contains strings of the form

$$t(t_1, param)t(t_2, param)\ldots t(t_m, param),$$

where *param* in parentheses substitute for the other parameters of a particular language. All the symbols of the string $t_1, t_2,\ldots, t_m$ correspond to trajectories that form a trajectory network W relative to t_1.

Different members of this family correspond to different types of trajectory network languages, which describe particular subsystems for solving search problems. One of such languages is the language that describes specific networks called Zones. They play the main role in the model considered here [2, 26, 33, 34]. A formal definition of this language is essentially constructive and requires showing explicitly a method for generating this language, i.e., a certain formal grammar, which is presented in the [33, 34]. In order to make our points transparent, here, we define the Language of Zones informally.

A ***Language of Zones*** is a trajectory network language with strings of the form

$$Z=t(p_0,t_0,\tau_0)\ t(p_1,t_1,\tau_1)\ldots t(p_k,t_k,\tau_k),$$

where $t_0,t_1,\ldots,t_k$ are the trajectories of elements $p_0,p_2,\ldots,p_k$ respectively; $\tau_0,\tau_1,\ldots,\tau_k$ are positive integer numbers (or 0) which "denote the time allotted for the motion along the trajectories" in a correspondence to the mutual goal of this Zone: to remove the target element – for one side, and to protect it – for the opposite side. Trajectory $t(p_0,t_0,\tau_0)$ is called the *main trajectory* of the Zone. The element q standing on the ending point of the main trajectory is called the *target*. The elements p_0 and q belong to the opposite sides.

To make it clearer let us show the Zone corresponding to the trajectory network in Fig. 2.

$$Z=t(p_0,a(1)a(2)a(3)a(4)a(5), 4)$$
$$t(q_3,a(6)a(7)a(4), 3)$$
$$t(q_2, a(8)a(9)a(4), 3)t(p_1, a(13)a(9), 1)$$
$$t(q_1, a(11)a(12)a(9), 2)\ t(p_2, a(10)a(12), 1)$$

Assume that the goal of the white side is to remove target q_4, while the goal of the black side is to protect it. According to these goals element p_0 starts the motion to the target, while blacks start in its turn to move their elements q_2 or q_3 to intercept element p_0. Actually, only those black trajectories are to be included into the Zone where the motion of the element makes sense, i. e., the *length of the trajectory is less than the amount of time (third parameter* τ*) allocated to it.* For example, the motion along the trajectories $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$ makes sense, because they are of length 2 and time allocated equals 3: each of the elements has 3 time intervals to reach point 4 to intercept element p_0 assuming one would go along the main trajectory without move omission. According to definition of Zone the trajectories of white elements (except p_0) could only be of the length 1, e.g., $a(13)a(9)$ or $a(10)a(12)$. As far as element p_1 can intercept motion of the element q_2 at the point 9, blacks include into the Zone the trajectory $a(11)a(12)a(9)$ of the element q_1, which has enough time for motion to prevent this interception. The total amount of time allocated to the whole bunch of black trajectories connected (directly or indirectly) with the given point of main trajectory is determined by the number of that point. For example, for the point 4 it equals 3 time intervals.

A language $L_Z(S)$ generated by the certain grammar $\mathbf{G_Z}$ [33, 34] in a state S of a Complex System is called the ***Language of Zones***.

Network languages allow us to describe the "statics", i.e., the states of the System. We proceed with the description of the "dynamics" of the System, i.e., the transitions from one state to another. The transitions describe the change of the descriptions of states as the change of sets of WFF. After each transition a new hierarchy of languages should be generated. Of course, it is an inefficient procedure. To improve an efficiency of applications in a process of the search it is important to describe the change of the hierarchy of languages. A study of this change should help us in modifying the hierarchy instead of regenerating it in each state. The change may be described as a hierarchy of mappings – translations of languages. Each language should be transformed by the specific mapping called a *translation*. Translations of Languages of Trajectories and Zones are considered in [34].

5 Complex System of Space Robotic Vehicles

The robotic model can be represented as a Complex System naturally (Fig. 3). A set of **X** represents the operational district which could be the area of combat operation broken into smaller cubic areas, "points", e.g., in the form of the big cube of 8 x 8 x 8, n = 512. It could be a space operation, where X represents the set of different orbits, or an air force battlefield, etc. **P** is the set of robots or autonomous vehicles. It is broken into two subsets P_1 and P_2 with opposing interests; $\mathbf{R_p(x,y)}$ represent moving capabilities of different robots for different problem domains: robot p can move from point x to point y if $R_p(x, y)$ holds. Some of the robots can crawl, the other can jump or ride, sail and fly, or even move from one orbit to another. Some of them move fast and can reach point y (from x) in "one step", i.e., $R_p(x, y)$ holds, others can do that in *k* steps only, and many of them can not reach this point at all. **ON(p)=x**, if robot p is at the point x; **v(p)** is the value of robot p. This value might be determined by the technical parameters of the robot. It might include the immediate value of this robot for the given combat operation; $\mathbf{S_i}$ is an arbitrary initial state of operation for analysis, or the starting state; $\mathbf{S_t}$ is the set of target states. These might be the states where robots of each side reached specified points. On the other hand S_t can specify states where opposing robots of the highest value are destroyed. The set of WFF $\{ON(p_j) = x_k\}$ corresponds to the list of robots with their coordinates in each state. **TRANSITION(p, x, y)** represents the move of the robot p from the location x to location y; if a robot of the opposing side stands on y, a removal occurs, i.e., robot on y is destroyed and removed.

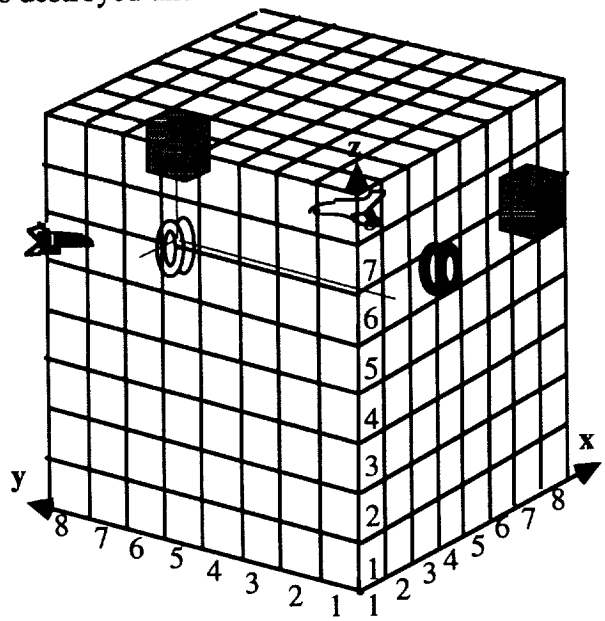


Fig. 3. A problem for autonomous space robotic vehicles.

Space robotic vehicles with different moving capabilities are shown in Fig. 3. The operational district X is the cubic table of 8 x 8 x 8. Robot W-INTERCEPTOR (White Interceptor) located at 118 (x=1, y=1, z=8), can move to any next location, i.e., 117, 217, 218, 228, 227, 128, 127. The other robotic vehicle B-STATION (double-ring shape in Fig. 3) from 416 can move only straight ahead towards the goal area 816 (shaded in Fig. 3), one square at a time, e.g., from 416 to 516, from 516 to 616, etc. Robot B-INTERCEPTOR (Black Interceptor) located at 186, can move to any next square similarly to robot W-INTERCEPTOR. Robotic vehicle W-STATION located at 266 is analogous with the robotic B-STATION; it can move only straight ahead to the goal area 268 (shaded in Fig. 3). Thus, robot W-INTERCEPTOR on 118 can reach any of the points $y \in \{117, 217, 218, 228, 227, 128, 127\}$ in one step, i.e., RW-INTERCEPTOR(118, y) holds, while W-STATION can reach only 267 in one step.

Assume that robots W-INTERCEPTOR and W-STATION belong to one side, while B-INTERCEPTOR and B-STATION belong to the opposite side: W-INTERCEPTOR $\in P_1$, W-STATION $\in P_1$, B-INTERCEPTOR $\in P_2$, B-STATION $\in P_2$. Also assume that both goal areas, 816 and 268, are the safe areas for B-STATION and W-STATION, respectively, if station reached the area and stayed there for more than one time interval. Each of the STATIONs has powerful weapons capable to destroy opposing INTERCEPTORs at the next diagonal locations ahead of the course. For example W-STATION from 266 can destroy opposing INTERCEPTORs at 157, 257, 357, 367, 377, 277, 177, 167. Each of the INTERCEPTORs is capable to destroy an opposing STATION approaching its location from any direction, but it also capable to protect its friendly STATION approaching its prospective location. In the latter case the joint protective power of the combined weapons of the friendly STATION and INTERCEPTOR (from any next to the STATION area) can protect the STATION from interception. For example, W-INTERCEPTOR located at 156 can protect W-STATION on 266 and 267.

The battlefield considered can be broken into two local operations. The first operation is as follows: robot B-STATION should reach the strategic point 816 safely and stay there for at list one time interval, while W-INTERCEPTOR will try to intercept this motion. The second operation is similar: robot W-STATION should reach point 268, while B-INTERCEPTOR will try to intercept this motion. After reaching the designated strategic area the (attacking) side is considered as a winner of the local operation and the global battle. The only chance for the opposing side to revenge itself is to reach its own strategic area within the next time interval and this way end the battle in a draw. The conditions considered above give us S_t, the description of target states of the Complex System. The description of the initial state S_i is obvious and follows from Fig. 3.

Assume also that due to the shortage of resources (which is typical in real combat operation) or some other reasons, each side can not participate in both operations simultaneously. It means that during the current time interval, in case of White turn, either W-STATION or W-INTERCEPTOR can move. Analogous condition holds for Black. Of course, it does not mean that if one side began participating in one of the operations it must complete it. Any time on its turn each side can switch from one operation to another, e.g., transferring resources (fuel, weapons, human resources, etc.), and later switch back.

It seems that local operations are independent, because they are located far from each other. Moreover, the operation of B-STATION from 418 looks like unconditionally winning operation, and, consequently, the global battle can be easily won by the Black side. Is there a strategy for the White side to make a draw?

Of course, this question can be answered by the direct search employing, for example, minimax algorithm with alpha-beta cut-offs. Experiments with the computer chess programs showed that for the similar 2-D problem (in chess terms – the R.Reti endgame) the search tree includes about a million moves (transitions). Of course, in the 3-D case the search would require billions of moves. It is very interesting to observe the drastic reduction of search employing the Linguistic Geometry tools. In order to demonstrate generation of the Hierarchy of Languages for this problem, below we consider generation of the Language of Trajectories for the robotic system on example of generation of the shortest trajectory from point 336 to point 816 for the robot W-INTERCEPTOR (Fig. 4, see also Fig. 16). (Point 336 is the location of W-INTERCEPTOR in one of the states of the System in the process of the search.)

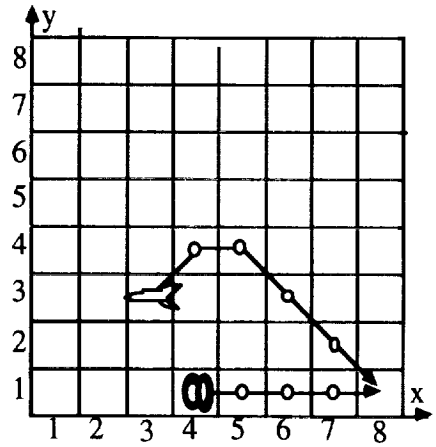


Fig. 4. Interpretation of Zone for the Robotic System (projection to xy-plane).

Table I. A grammar of shortest trajectories $G_t^{(1)}$

L	Q	Kernel, π_k	F_T	F_F
1	Q_1	$S(x,y,l) \rightarrow A(x, y, l)$	two	ø
2_i	Q_2	$A(x,y,l) \rightarrow a(x)A(next_i(x,l),y,f(l))$	two	3
3	Q_3	$A(x, y, l) \rightarrow a(y)$	ø	ø

$V_T = \{a\}$ is the alphabet of terminal symbols,

$V_N = \{S, A\}$ is the alphabet of nonterminal symbols,

$V_{PR} = Truth \cup Pred \cup Con \cup Var \cup Func \cup \{$symbols of logical operations$\}$ is the alphabet of the first order predicate calculus PR,

$Truth = \{T, F\}$

$Pred = \{Q_1, Q_2, Q_3\}$ are predicate symbols:

$Q_1(x, y, l) = (MAP_{x,p}(y)=l)$ $(0<l<n)$

$Q_2(l) = (l \geq 1)$

$Q_3 = T$

$Var = \{x, y, l\}$ are variables;

$Con = \{x_0, y_0, l_0, p\}$ are constants;

$Func = Fcon$ are functional symbols;

$Fcon = \{f, next_1, ..., next_n\}$ (n=|X|, number of points in X),

$f(l)=l-1$, $D(f)=\mathbf{Z}_+ \setminus \{0\}$

($next_i$ is defined lower)

$E = \mathbf{Z}_+ \cup X \cup P$ is the subject domain;

$Parm$: $S \rightarrow Var$, $A \rightarrow Var$, $a \rightarrow \{x\}$, is such a mapping that matches each symbol of the alphabet $V_T \cup V_N$ a set of formal parameters;

$L = \{1,3\} \cup$ two, two$=\{2_1, 2_2, ..., 2_n\}$ is a finite set called the set of labels; labels of different productions are different;

Q_i are the WFF of the predicate calculus PR, the conditions of applicability of productions;

F_T is a subset of L of labels of the productions permitted on the next step derivation if Q=T; it is called a permissible set;

F_F is analogous to F_T but these productions are permitted in case of Q=F.

At the beginning of derivation: $x=x_0$, $y=y_0$, $l=l_0$, $x_0 \in X$, $y_0 \in X$, $l_0 \in \mathbf{Z}_+$, $p \in P$.

$next_i$ is defined as follows:

$D(next_i) = X \times \mathbf{Z}_+ \times X^2 \times \mathbf{Z}_+ \times P$

(This is the domain of function *next*.)

$SUM = \{v \mid v \in X, MAP_{x0,p}(v) + MAP_{y0,p}(v) = l_0\}$

$ST_k(x) = \{v \mid v$ from X, $MAP_{x,p}(v)=k\}$,

$MOVE_l(x)$ is an intersection of the following sets: $ST_1(x)$, $ST_{l_0-l+1}(x_0)$ and SUM.

If $MOVE_l(x) = \{m_1, m_2, ..., m_r\} \neq \emptyset$

then

$next_i(x, l) = m_i$ for $i \leq r$;

$next_i(x, l) = m_r$ for $r < i \leq n$,

otherwise

$next_i(x,l) = x$.

Consider the Grammar of shortest trajectories $G_t^{(1)}$(Table I). This is a controlled grammar [32]. Such grammars operate as follows. The initial permissible set of productions consists of the production with label 1. It should be applied first. Let us describe the application of a production in such grammar. Suppose that we attempt to apply production with label l to rewrite a symbol A. We choose the leftmost entry of symbol A in the current string and compute the value of predicate Q, the condition of applicability of the production. If the current string *does not* contain A or $Q=F$, then the application of the production is ended, and the next production is chosen from the failure section F_F ; F_F becomes the current permissible set. If the current string *does* contain the symbol A and $Q=T$, A is replaced by the string in the right side of the production; we carry out the computation of the values of all formulas either standing separately (section π_n) or corresponding to the parameters of the symbols (π_k), and the parameters assume new values thus computed. Then, application of the production is ended, and the next production is chosen from the success section F_T, which is now the current permissible set. If the applicable section is empty, the derivation halts.

The controlled grammar shown in Table I can be used for generation of shortest trajectories for robots with arbitrary moving capabilities. Values of $MAP_{336,W\text{-}INTERCEPTOR}$ are shown in Fig.5. Thus, the distance from 336 to 816 for W-INTERCEPTOR is equal to 5. To be transparent we will show generation of trajectories located completely within the plane xy6 only. Thus, for this generation we will use 2-D coordinates.

Applying the grammar $G_t^{(1)}$ we have (symbol $l\Rightarrow$ means application of the production with the label l):

$$S(33, 81, 5) \overset{1}{\Rightarrow} A(33, 81, 5)$$

$$2_1 \Rightarrow a(33)A(next_1(33, 5), 81, 5)$$

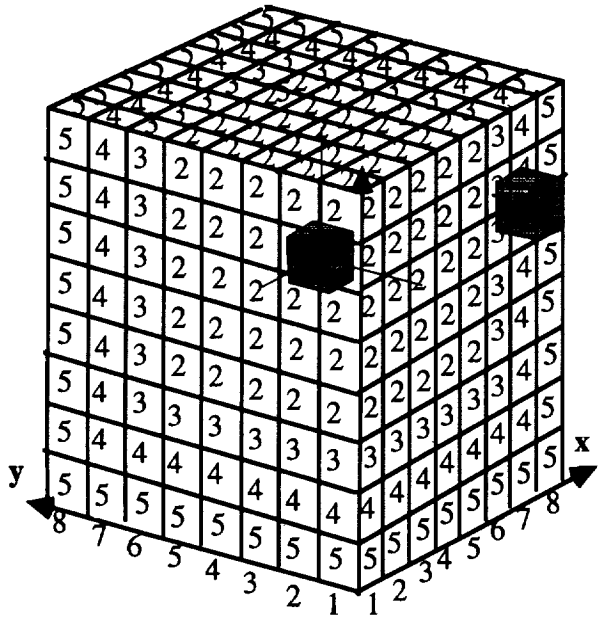


Fig. 5. MAP_{336}, INTERCEPTOR

Thus we have to compute MOVE (see definition of the function $next_i$ from the grammar $G_t(1)$). First we have to determine the set of SUM, that is, we need to know values of

MAP_{33},W-INTERCEPTOR and
MAP_{81},W-INTERCEPTOR
(shown in Fig. 6) on X.

5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	5
3	3	3	3	3	3	4	5
2	2	2	2	2	3	4	5
2	1	1	1	2	3	4	5
2	1	0	1	2	3	4	5
2	1	1	1	2	3	4	5
2	2	2	2	2	3	4	5

7	7	7	7	7	7	7	7
7	6	6	6	6	6	6	6
7	6	5	5	5	5	5	5
7	6	5	4	4	4	4	4
7	6	5	4	3	3	3	3
7	6	5	4	3	2	2	2
7	6	5	4	3	2	1	1
7	6	5	4	3	2	1	0

Fig.6.MAP_{33},W-INTERCEPTOR (left) and MAP_{81},W-INTERCEPTOR (right)

Adding these tables as matrices we compute

$$SUM = \{v \mid v \in X, MAP_{33}, \text{W-INTERCEPTOR}(v) + MAP_{81}, \text{W-INTERCEPTOR}(v) = 5\} \text{ (Fig. 7).}$$

For the general 3-D case we should add 3-D matrices like those shown in Fig. 5.

The next step is the computation of $ST_1(33)$= {v |v from X,MAP_{33},W-INTERCEPTOR(v)=1} which is shown in Fig. 8. In order to complete computation of the set $MOVE_5(33)$ we have to determine the following intersection:
$ST_1(33)$, $ST_{5-5+1}(33)=ST_1(33)$ and SUM.

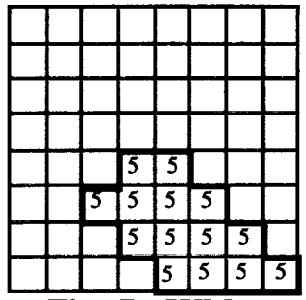


Fig. 7. SUM.

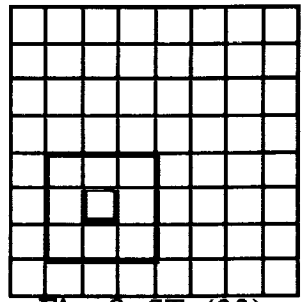


Fig. 8. $ST_1(33)$.

Consequently, $MOVE_5(33)=\{44, 43, 42\}$; and $next_1(33, 5)=44$, $next_2(33, 5)=43$, $next_3(33, 5)=42$. Since the number of different values of *next* is equal to 3 (here r=3, see definition of the function *next*, Table I) we could branch at this step, apply productions 2_1, 2_2 and 2_3 simultaneously, and continue both derivations independently. This could be accomplished in a parallel computing environment. Let us proceed with the first derivation.

$$a(33)A(44, 81,4)\ 2_1 => a(33)a(44)$$
$$A(next_1(44, 4), 81, 3)$$

We have to compute $next_1(44, 4)$ and, as on the preceding step, have to determine $MOVE_4(44)$. To do this we have to compute

$$ST_1(44)=\{v \mid v \in X, MAP_{44}, \text{W-INTERCEPTOR}(v)=1\}, \text{(Fig. 9)}$$

$$ST_{5-4+1}(33) = ST_2(33) = \{v \mid v \in X, MAP_{33}, \text{W-INTERCEPTOR}(v)=2\}, \text{(Fig. 10).}$$

The set of SUM is the same on all steps of the derivation. Hence, $MOVE_4(44)$ is the intersection of the sets shown in Fig. 7, 9, 10; $MOVE_4(44)=\{54, 53, 52\}$; and $next_1(44, 4) = 54$; $next_2(44, 4) = 53$, $next_3(44, 4) = 52$. Thus, the number of different values of the function *next* is equal to 3 (r=3), so the number of continuations of derivation should be multiplied by 3.

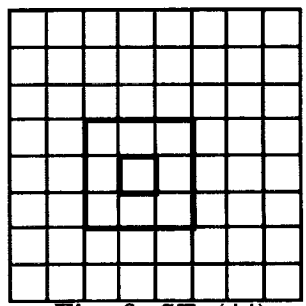


Fig. 9. $ST_1(44)$

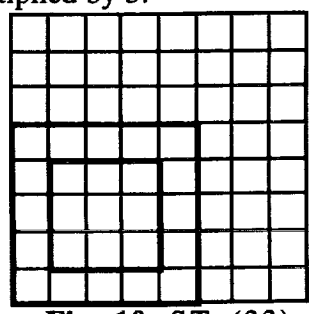


Fig. 10. $ST_2(33)$.

Let us proceed with the first one: $a(33)a(44)A(54, 81, 3)\ ^{2}1 =>$ ... Eventually, we will generate one of the shortest trajectories for the robot W-INTERCEPTOR from 33 to 81:

$a(33)a(44)a(54)a(63)a(72)a(81)$.

Similar generating techniques are used to generate higher level subsystems, the networks of paths, i.e., the Language of Zones. For example,, incomplete Zones shown in Fig. 4 is as follows (in 2-D coordinates):

t(B-STATION,t_B,5)t(W-INTERCEPTOR,t_F,5),

where $t_B = a(41)a(51)a(61)a(71)a(81)$,

$t_F = a(33)a(44)a(54)a(63)a(72)a(81)$.

The details of generation of different Zones are considered in [33, 34].

6 Search Generation for Space Robotic System

Consider how the hierarchy of languages works for the optimal control of the space robotic system introduced above (Fig. 3). We generate the search of the Language of Translations representing it as a conventional search tree (Fig. 12) and comment on its generation. In fact, this tree is close to the search tree of the relative 2-D problem [35]. Moreover, it is close to the search tree of the R.Reti endgame generated by program PIONEER in 1977 and presented at the World Computer Chess Championship (joint event with IFIP Congress 77, Toronto, Canada). Later it was published in different journals and books, in particular in [2].

First, the Language of Zones in the start state is generated. The targets for attack are determined within the limited number of steps which is called a horizon. In general, the value of the horizon is unknown. As a rule, this value can be determined from the experience of solving specific classes of problems employing Linguistic Geometry tools. In absence of such experience, first we have to consider the value of 1 as a horizon, and solve the problem within this value. If we still have resources available, i.e., computer time, memory, etc., we can increase the horizon by one. After each increase we have to regenerate the entire model. This increase means a new level of "vigilance" of the model, and, consequently, new greater need for resources.

In our case it is easy to show that within the horizons of 1, 2, 3, 4 all the models are "blind" and corresponding searches do not give a "reasonable" solution. But, again, after application of each of the consecutive values of the horizon we will have a *solution* which can be considered as an approximate solution within the available resources. Thus, let the horizon H of the language $L_Z(S)$ is equal to 5, i.e., the length of main trajectories of all Zones must not exceed 5 steps. All the Zones generated in the start state are shown in Fig. 11.

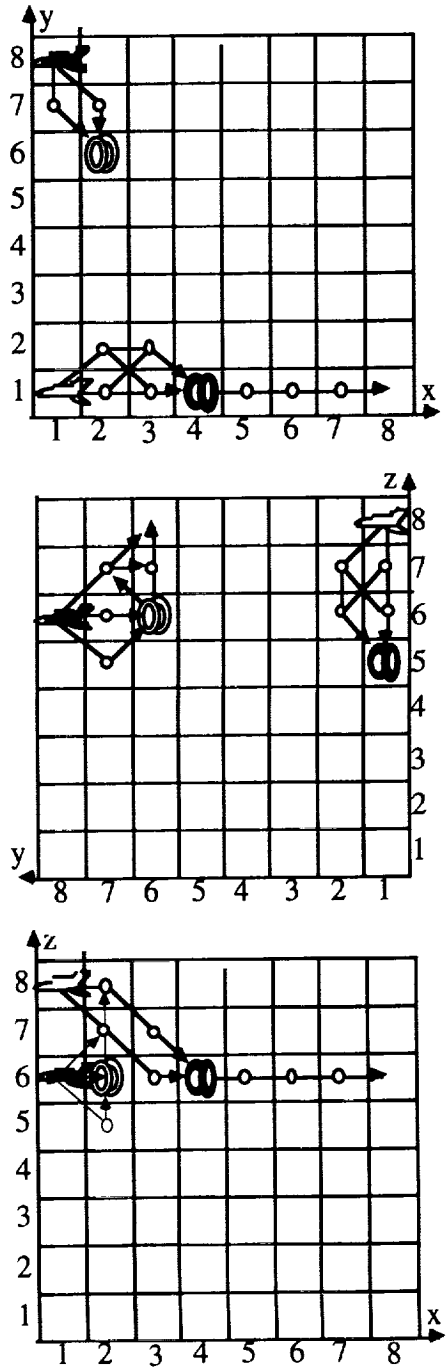


Fig. 11. Interpretation of Zones in the initial state of the space robotic system (3 projections).

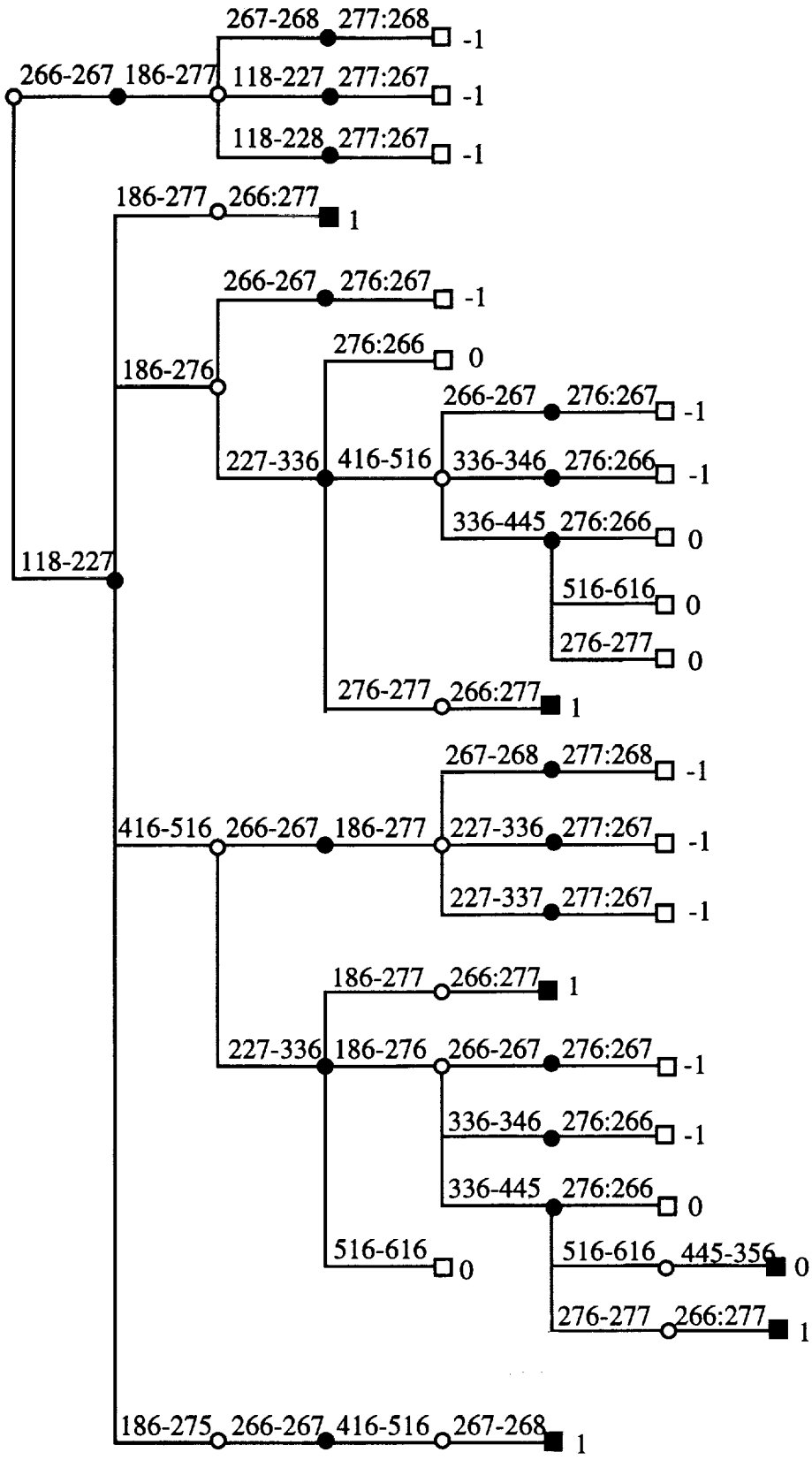


Fig. 12. Search tree for the optimization problem for space robotic vehicles.

Zones for INTERCEPTORSs as attacking elements are shown in the top diagram, while Zones for STATIONs – in the bottom one. For example, one of the Zones for W-STATION, Z_{WS} is as follows:

Z_{WS}=*t*(W-STATION, *a*(266)*a*(267)*a*(268), 3)
t(B-INTERCEPTOR, *a*(186)*a*(277)*a*(268), 3)
t(B-INTERCEPTOR,*a*(186)*a*(276)*a*(267), 2)
t(W-STATION, *a*(266)*a*(277), 1)

The other trajectories of B-INTERCEPTOR, e.g., the second trajectory, *a*(186)*a*(177)*a*(268), leading to the point 268 is included into different Zone; for each Zone only one trajectory from each bundle of trajectories with the same beginning and end is taken.

Generation begins with the move 1. 266-267 in the "white" Zone with the target of the highest value and the shortest main trajectory. The order of consideration of Zones and particular trajectories is determined by the grammar of translations. The computation of move-ordering constraints is the most sophisticated procedure in this grammar. It takes into account different parameters of Zones, trajectories, and the so-called chains of trajectories.

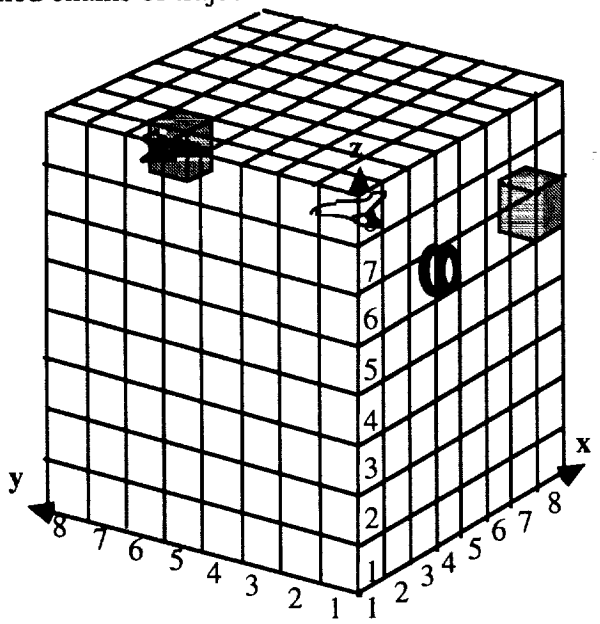


Fig. 13. The state where control Zone from 118 to 268 was detected.

Next move, 1. ... 186-277, is in the same Zone along the first negation trajectory. The interception continues: 2. 267-268 277:268 (Fig. 13). Symbol ":" means the removal of element. Here the grammar terminates this branch with the value of -1 (as a win of the Black side). This value is given by the special procedure of "generalized square rules" built into the grammar.

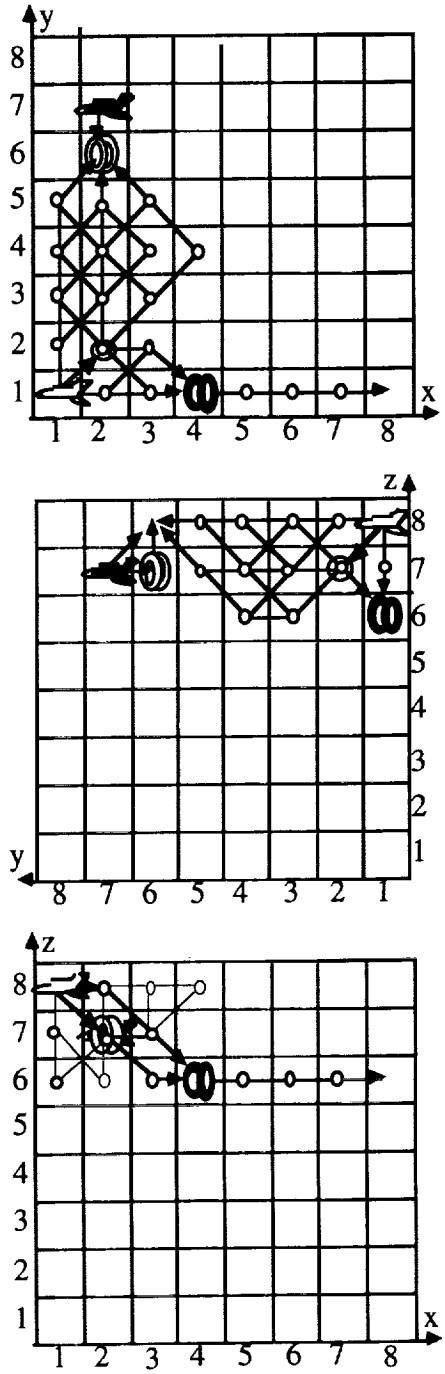


Fig. 14. Interpretation of Zones in the state where control Zone from 118 to 268 was included first (3 projections).

Then, the grammar initiates the backtracking climb. Each backtracking move is followed by the inspection procedure, the analysis of the subtree generated in the process of the earlier search. After climb up to the move 1. ... 186-277, the

tree to be analyzed consists of one branch (of two plies): 2. 267-268 277-268. The inspection procedure determined that the current minimax value (-1) can be "improved" by the improvement of the exchange in the area 268 (in favor of the White side). This can be achieved by participation of W-INTERCEPTOR from 118, i.e., by generation and inclusion of the new so-called "control" Zone with the main trajectory from 118 to 268. The set of different Zones from 118 to 268 (the bundle of Zones) is shown in Fig. 14. The move-ordering procedure picks the subset of Zones with main trajectories passing 227. These trajectories partly coincide with the main trajectory of another Zone attacking the opposing B-STATION on 516. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously.

The generation continues: 2. 118-227 277-267. Again, the procedure of "square rules" cuts the branch, evaluates it as a win of the black side, and the grammar initiates the climb. Move 2. 118-227 is changed for 2. 118-228. Analogously to the previous case, the inspection procedure determined that the current minimax value (-1) can be improved by the improvement of the exchange on 267. Again, this can be achieved by the inclusion of Zone from 118 to 267. Of course, the best "time-gaining" move in this Zone is 2. 118-227, but it was already included (as move in the Zone from 118 to 268). The other untested move in the Zone from 118 to 267 is 2. 118-228. Obviously the grammar does not have knowledge that trajectories to 267 and 268 are "almost" the same.

After the next cut and climb, the inspection procedure does not find new Zones to improve the current minimax value, and the climb continues up to the start state. The analysis of the subtree shows that inclusion of Zone from 118 to 268 in the start state can be useful: the minimax value can be improved. Similarly, the most promising "time-gaining" move is 1. 118-227. The Black side responded 1. ... 186-277 along the first negation trajectories $a(186)a(277)a(267)$ and $a(186)a(277)a(268)$ shown in Fig. 12 (better see yz-projection). Obviously, 2. 266:277, and the branch is terminated. The grammar initiates the climb and move 1. ... 186-277 is changed for 1. ... 186-276 along the trajectory $a(186)a(276)a(266)$. Note, that grammar "knows" that in this state trajectory $a(186)a(276)a(266)$ is active, i.e., B-INTERCEPTOR has enough time for interception. The following moves are in the same Zone of W-STATION: 2. 266-267 276:267. This state is shown in Fig. 15. The "square rule procedure" cuts this branch and evaluates it as a win of the Black side.

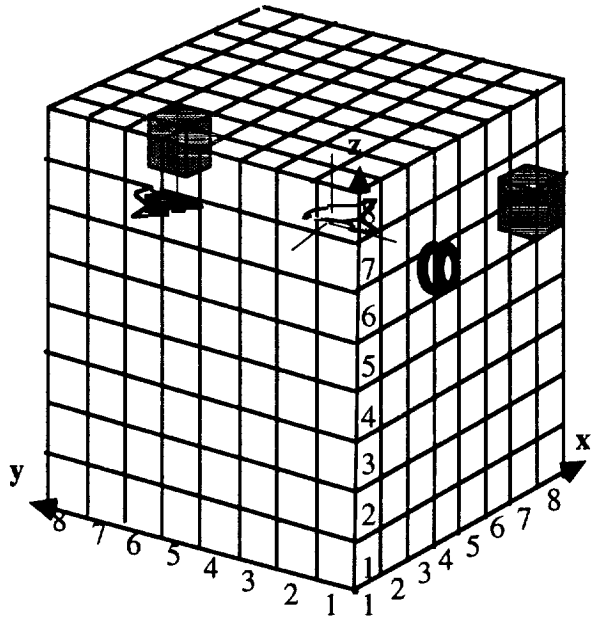


Fig. 15. The state where control Zone from 227 to 267 was detected.

New climb up to the move 2. ... 186-276 and execution of the inspection procedure result in the inclusion of the new control Zone from 227 to 267 in order to improve the exchange in the area 267. The set of Zones with different main trajectories from 227 to 267 is shown in Fig. 16. Besides that, the trajectories from 227 to 516, 616, 716, and 816 are shown in the same Fig. 16. These are "potential" first negation trajectories. It means that beginning with the second symbol $a(336)$, $a(337)$, $a(338)$, or $a(326)$, $a(327)$, $a(328)$, or $a(316)$, $a(317)$, $a(318)$, these trajectories become first negation trajectories in the Zone of B-STATION h5. Speaking informally, from the areas listed above W-INTERCEPTOR can intercept B-STATION (in case of white move). The main trajectories of control Zones passing one of three points, 336, 337, or 338, partly coincide with the potential first negation trajectories. The motion along such trajectories allows to "gain the time", i.e., to approach two goals simultaneously. The move-ordering procedure picks the subset of Zones with the main trajectories passing 336. Thus, 2. 227-336.

This way proceeding with the search we will

generate the tree that consists of 56 moves. Obviously, this is a drastic reduction in comparison with a billion-move trees generated by conventional search procedures.

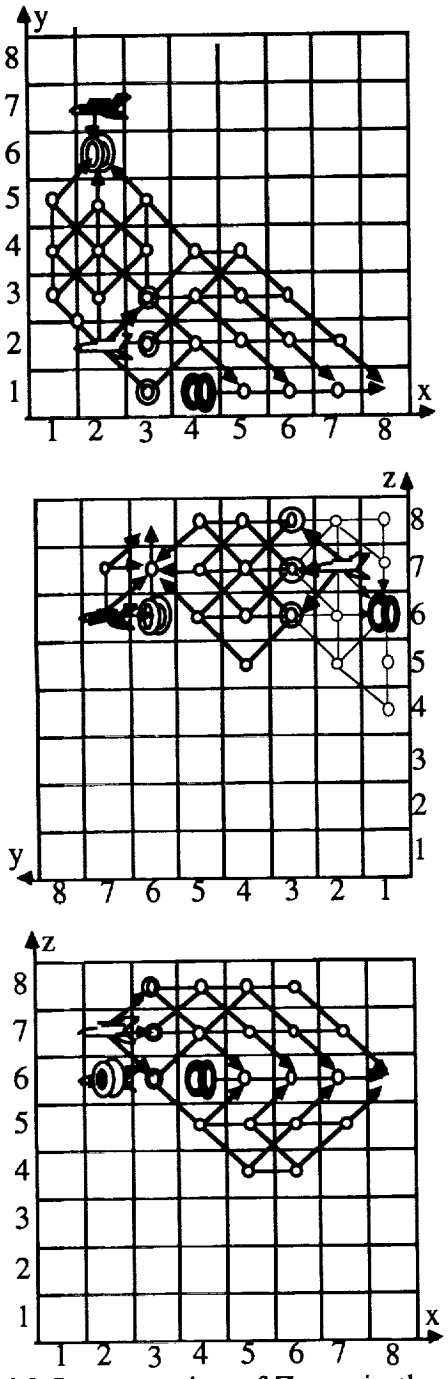


Fig. 16. Interpretation of Zones in the state where control Zone from 227 to 267 was included first (3 projections).

7 Discussion

The approach to understanding of dynamic hierarchical systems considered here will encompass the discovery of geometrical properties of subsystems and details of interactions between the elements within subsystems and between different subsystems. We will understand the details of influence of this complex hierarchical structure on the reduction of the search for suboptimal operation. Most importantly, it should allow a better understanding of the evaluation and control of the solution quality.

This contribution to the formalization and generalization of human search heuristics should allow for the expansion of advanced human heuristic methods discovered in different complex systems to other real-world systems where existing methods are not sufficient. The research will lead to the development of *efficient applications* to autonomous navigation in hazardous environment, robot control, combat operations planning as well as applications in different nonmilitary areas. The development of applications will be accomplished by the design of separate programs, and, later on, by the program implementation of the general hierarchy of formal grammars and applying it to a given problem.

References

1. Albus, J. (1991) "Outline for a Theory of Intelligence", *IEEE Trans. on Systems, Man and Cybernetics*, 3:473-509.
2. Botvinnik, M.M. (1984) *Computers in Chess: Solving Inexact Search Problems.* Springer Series in Symbolic Computation, Springer-Verlag, New York.
3. Botvinnik, M., Petriyev, E., Reznitskiy, A., et al. (1983) "Application of New Method for Solving Search Problems For Power Equipment Maintenance Scheduling", *Economics and Mathematical Methods*, 6:1030-1041 (in Russian).
4. Chapman, D. (1987) "Planning for conjunctive goals", *Artificial Intelligence* 32(3).
5. Chomsky, N. (1963) "Formal Properties of Grammars", in *Handbook of Mathematical Psychology*, eds. R.Luce, R.Bush, E. Galanter., vol. 2, John Wiley & Sons, New York, pp. 323-418.
6. Feder, J. (1971) "Plex languages", *Information Sciences*, 3: 225–241.
7. Fikes, R.E. and Nilsson, N.J. (1971) "STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving", *Artificial Intelligence* 2: 189–208.
8. Fu, K.S. (1982) *Syntactic Pattern*

Recognition and Applications, Prentice Hall, Englewood Cliffs.
9. Garey, M.R. and D.S.Johnson D.S. (1991) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco.
10. Ginsburg, S. (1966) *The Mathematical Theory of Context-Free Languages*, McGraw Hill, New York.
11. Knuth, D.E. (1968) "Semantics of Context-Free Languages", *Mathematical Systems Theory* 2:127–146.
12. McAllester, D. and Rosenblitt, D. (1991) Systematic Non-Linear Planning, *Proc. of AAAI-91*, pp. 634-639.
13. McCarthy, J. (1980) "Circumscription–A Form of Non-Monotonic Reasoning", *Artificial Intelligence* 13:27-39.
14. McCarthy, J. and Hayes, P.J. (1969) "Some Philosophical Problems from the Standpoint of Artificial Intelligence", *Machine Intelligence* 4:463–502.
15. Mesarovich, M.D., Macko, D., and Takahara Y. (1970) *Theory of Hierarchical Multilevel Systems.*, Academic Press, New York.
16. Narasimhan, R.N. (1966) "Syntax–Directed Interpretation of Classes of Pictures", *Communications of the ACM* 9: 166–173.
17. Nilsson, N.J. (1980) *Principles of Artificial Intelligence*, Tioga Publ., Palo Alto, CA.
18. Pavlidis, T. (1977) *Structural Pattern Recognition*, Springer-Verlag, New York.
19. Reznitskiy, A.I. and Stilman, B. (1983) "Use of Method PIONEER in Automating the Planning of Maintenance of Power-Generating Equipment," *Automatics and Remote Control*, 11: 147-153, (in Russian).
20. Rosenfeld, A. (1979) *Picture Languages, Formal Models for Picture Recognition*, Academic Press.
21. Rozenkrantz, D.J. (1969) "Programmed Grammars and Classes of Formal Languages," *J. of the ACM*, 1:107–131.
22. Sacerdoti, E.D. (1975) "The Nonlinear Nature of Plans," *Proc. Int. Joint Conference on Artificial Intelligence.*
23. Shaw, A.C. (1969) "A Formal Picture Description Scheme as a Basis for Picture Processing System," *Information and Control* 19: 9-52.
24. Simon, H.A. (1980) *The Sciences of the Artificial*, 2-nd ed., The MIT Press, Cambridge, MA.
25. Stefik, M. (1981) "Planning and meta-planning (MOLGEN: Part 2),"*Artificial Intelligence*, 2:141-169.
26. Stilman, B. (1977) "The Computer Learns", in Levy, D.,*1976 US Computer Chess Championship*, Computer Science Press, Woodland Hills, CA, 83-90.
27. Stilman, B. (1985)"Hierarchy of Formal Grammars for Solving Search Problems," in *Artificial Intelligence. Results and Prospects, Proceedings of the International Workshop*, Moscow, 63–72, (in Russian).
28. Stilman, B. (1992) "A Linguistic Geometry of Complex Systems, *Abstr. Second Int. Symposium on Artificial Intelligence and Mathematics*," Ft. Lauderdale, FL. The full paper was submitted to *Annals of Math. & Artificial Intelligence.*
29. Stilman, B. (1992) "A Syntactic Structure for Complex Systems", *Proc. Second Golden West Int. Conf. on Intelligent Systems*, Reno, NE, June, 269-274.
30. Stilman, B. (1992) "A Geometry of Hierarchical Systems: Generating Techniques", *Proc. Ninth Israeli Conference on Artificial Intelligence and Computer Vision*, pp. 95-109, Dec., Tel Aviv, Israel.
31. Stilman, B. (1992) "A Syntactic Approach to Geometric Reasoning about Complex Systems," *Proc. Fifth Int. Symp. on Artificial Intelligence*, Cancun, Mexico, Dec., 115-124.
32. Stilman, B. (1993) "A Linguistic Approach to Geometric Reasoning," *Int. J. Computers and Mathematics with Applications*, 26(7), 29-57.
33. Stilman, B. (1993) "Network Languages for Complex Systems," *Int. J. Computers and Mathematics with Applications*, 26(8), 51-79.
34. Stilman, B. (1994) "Translations of Network Languages," *Int. J. Computers and Mathematics with Applications*, 27(2), 65-98, (to appear).
35. Stilman, B.(1994) "A Formal Model for Heuristic Search," *Proceedings of the 22nd Annual ACM Computer Science Conf.*, Phoenix, AZ, (to appear).
36. Volchenkov, N.G. (1979) "The Interpreter of Context-Free Controlled Parameter Programmed Grammars," in L.T. Kuzin, Eds., *Cybernetics Problems. Intellectual Data Banks*, The USSR Academy of Sci., Moscow, pp. 147–157, (in Russian).

N94- 35051

Multiresolutional Schemata for Unsupervised Learning of Autonomous Robots for 3D Space Operation

Alberto Lacaze, Michael Meystel, Alex Meystel
Simulation and Modeling Group
IMPAQT Center, Drexel University, Philadelphia, PA
schema@impaqt.drexel.edu

ABSTRACT

This paper describes a novel approach to the development of a learning control system for autonomous space robot (ASR) which presents the ASR as a "baby" -- that is, a system with no a priori knowledge of the world in which it operates, but with behavior acquisition techniques that allows it to build this knowledge from the experiences of actions within a particular environment (we will call it an Astro-baby). The learning techniques are rooted in the recursive algorithm for inductive generation of nested schemata molded from processes of early cognitive development in humans. The algorithm extracts data from the environment and by means of correlation and abduction, it creates schemata that are used for control. This system is robust enough to deal with a constantly changing environment because such changes provoke the creation of new schemata by generalizing from experiences, while still maintaining minimal computational complexity, thanks to the system's multiresolutional nature.

Experimenting with ASR is especially interesting because the rules of input control do not coincide with human intuitions. Actually, we want to see that the simulated device can learn the unexpected schemata from its own experience. Although the traditional approach to autonomous navigation involves off-line path planning with a known world map (such as the potential fields algorithm), in most of the real tasks the environment is not well known because of ever-changing conditions of the assignment absence of gravity, and sophisticated, hard to predict obstacles like components of the space stations, etc. Astro-baby gathers data from its sensors and then by using a schema-discovery system it extracts concepts, forms schemata and creates a quantitative/conceptual semantic network.

When the Astro-baby is first dropped into the space it does not have any experiences and its sensors and actuators are sets that do not have any distinction among its elements. Then, by trial and error, the ASR learns the function of its actuators and sensors; and how to activate them to achieve a the goal given by its creator, or the sub-goals that it finds. In our simulation the initial goal is to minimize the distance to a beacon.

The learning techniques are rooted in a nested hierarchical algorithm molded from processes of early cognitive development in humans. The algorithm extracts data from the environment and by means of correlation, it creates schemata (rules) that are used for control. This system is robust enough to deal with a constantly changing environment because such changes provoke the creation of new schemata using generalization, while still maintaining minimal computational complexity, thanks to the system's multiresolutional nature.

The results of simulation are positive. Astro-baby displays the ability to learn a number of maneuvers.

I. INTRODUCTION

Although the traditional approach to autonomous navigation involves off-line path planning with a known world map (such as the potential fields algorithm shown in [1]), in most of the tasks assigned to autonomous robots, the environment is not well known because of ever-changing conditions of the space, complicated conditions of visibility, and diversified obstacles like trusses, other automated machines, unpredictable objects from other planets. Thus, a system robust enough to cope with changes by means of learning rules about the situation is needed. Motion planning and control for autonomous ground vehicles can be approached based upon substantial human experience of dealing with a diversity of ground vehicles. We believe that 3-D dynamic motion in space requires control rules which are not easily available and are not a part of the intuition of a human designer. Therefore, our intention is to allow the ASR to collect its own rules based upon a system of unsupervised (teacher-independent) conceptual learning.

We have developed a system for early cognition that is capable of extracting concepts from the environment and using them for planning and controlling the ASR. Astro-baby gathers data from its sensors and then by using a rule-discovery system and a concept formatting system it extracts and stores the concepts and schemata to create a quantitative/conceptual semantic network as a system of knowledge representation. The natural growth of the rule-base can be compared with the "subsumption" architecture. However, the subsumption concept does not emphasize the early learning, and is usually designed from prior experience of operation.

Our approach focuses on self-developing knowledge base which starts with a minimal amount of knowledge, which we call "bootstrap-knowledge". The bootstrap knowledge does not include any implicit or explicit information about the world or the robot. It has a minimal set of learning rules which the Astro-baby uses to create a world model, decision-making rules, rules of motion, and rules of perception.

The main idea of our approach is knowledge-base generation by applying generalization recursively to obtain the schemata, or rules of behavior at different levels of resolution from the stored information of experiences properly labeled and organized. During the life of the

unmanned vehicle, these rules are constantly reviewed and updated based on new sensor information and deductions which the Astro-baby makes on the basis of algorithms which are hard-coded in the system ("bootstrap knowledge")

In the beginning, Astro-baby does not have any rule of operation and its sensors and actuators are sets that do not have any distinction among its elements. Then, by trial and error, the space robot learns the function of its actuators and sensors; and how to activate them to achieve a certain goal given by its creator or learned sub-goals. In our simulation the initial goal is to minimize the distance to a beacon (with sensors measuring angle and distance to the beacon with some error) which could be a sunken ship, a lost diver, etc., but because of its learning capabilities, the system's applications could be very broad. Given the goal (expressed as a cost functional) the Astro-baby learns concepts like direction, passageway, or obstacle. If these actuation rules were not to apply in a different environment, it would extract a new set of rules.

The world in our simulation consists of a fully dynamic 3-D environment. We have attempted to incorporate as many variables from the real world as possible, so as to fully test the robustness of the learning algorithm. The environment is constantly changed and no map is given. Astro-baby is a very adaptable system that can both create rules of planning and control and deal with situations that were not envisioned by its creators.

II. LEARNING

Standard Approach

The Artificial Intelligence community has made attempts to write "intelligent" programs, or programs which learn from mistakes, for many decades. Some of the early work is Newell, Shaw, and Simon's General Problem Solver (1956), and Samuel's checkers playing program (1959). Most of these learning systems were built to solve very specific problems of learning. In our Astro-baby, although we take into consideration as many variables from the environment as possible in our simulation, we do not give this knowledge to the learning system. In our research we decided to develop a system which arrives at this knowledge on its own. This cannot be done unless the system is given some initial knowledge [2, 3]. One of the attempts we have made is to find what this minimum initial knowledge should be.

Differences in our approach with other existing approaches are classified below.

A. Drawbacks of Subsumption Architecture

The subsumption architecture is also a multiresolutional one, as is ours. However, in a subsumption architecture, the set of rules of control is predetermined by the designer of the system. This means that the designer must be aware of all possible situations that the asr will encounter. This precludes the assumption of an open environment and that the system will be able to store all the rules for that open environment and that the designer of the system has all these rules to begin with. This makes applying existing approaches to subsumption for astro-robots impossible. We do not include any heuristic schemata in our system. Instead, we include rules (called "bootstrap knowledge") which help the system to acquire, by itself, through learning, the rules that are given a priori in a subsumption architecture.

B. Multi-Agent versus Centralized Decision-Making

In a multiagent system, the decision-making is decentralized. Thus, it has a set of entities which have their own goals and an arbiter who is in charge of switching or deciding the weight or power of each agent depending on the urgency of the situation. For example, [4] uses a subsumption-based, multiagent approach, generating potential fields of attraction and repulsion in various areas of the map. Some examples of preprogrammed agents are "Follow Object", "Forward Attraction", "Open Space Attraction", "Wall Following". In this approach the environment must be entirely known because of the necessity to determine placement of the potential fields. Moreover, the behavior that the robot should take in front of these potential fields must also be known in order to preprogram these agents.

A centralized control system, in the opinion of its critics, creates a bottleneck by forcing each separate unit of the control system, regardless of the type or resolution of its task, to query one decision maker for instructions. Indeed, this happens if the centralized system is not based upon proper (multiresolutional) task decomposition. The latter not only eliminates the bottleneck but actually reduces the complexity dramatically. In [5] it is proven that a hierarchical system largely reduces the complexity of the computations involved in search.

C. Flat Schemata and Multiresolutional Schemata

An example of learning using centralized decision-making and flat schemata is shown in [6]. When we have a centralized decision-making control system working in a complex environment, the amount of rules that must be dealt with is so large that working in a flat-level is impossible. When we work with centralized learning systems, we must use a multiresolutional configuration to avoid complexity.

The approach of our paper is based upon M. Arbib's theory of motor schema [7] applied to a multiresolutional structure. We believe that high-resolution schemata generalize in such a way as to create a low-resolution level of schemata. This procedure of generalization is recursive

in nature, and is inherent in the learning loop. The reality of computation requires it for complexity reduction.

The Multiresolutional Schemata Approach

A. Theory of Multiresolutional Schemata

There exists a multiplicity of definitions for the idea of "schema" which takes into consideration different aspects of this powerful concept. The concept has existed for centuries, and has recently been applied in the area of neurobiology by [6-8] and others. Schema is a construct which represents an entity related to the areas of perception, knowledge organization, and control.

As far as problems of motion control are concerned, we believe that "schema" should be defined as follows:

Schema is an implication

"situation→action" [9] formulated as an *entity* for a particular i^{th} level of resolution of the world representation. More formally, this statement can be represented as a notation (from [10])

$$\Sigma_i = \{[s_i(t) \rightarrow a_i(t)], \rho_i\}, \quad (1)$$

where Σ_i is the "schema",

s_i is the "situation" determined only by a set of the "entity discovered in the set of sensor information at a resolution ρ_i" so that $s_i = \{\pi_i, \kappa_i, \gamma_i\}$.

π_i is the percept: "a set of information delivered from the sensors",

κ_i is the context: "a set of information delivered from the sensors at time t-ρ_i",

γ_i - is the final goal at a level: "an entity defined by the assignment at a lower resolution level ρ_{i-1}"

a_i - is the action: "an entity defined upon a set of dynamic changes in a position and orientation at a resolution ρ_i"; action is a string of subgoals $\gamma_{sk}(k=1,2,...,m; y_{sm} = \gamma_i)$ to be reached before the final goal is achieved, in other words $a_i \rightarrow (\gamma_{s1i}, \gamma_{s2i}, ..., \gamma_{sm})$,

ρ_i - is a vector which contains the minimum distinguishable discrete of a spatial dimension or time in the i^{th} level.

The storage of schema is done based upon a concept called semantic network, exemplified in Figure 1.

B. Learning in Multiresolutional Schemata

(1) Bootstrap knowledge

Bootstrap is a minimal set of algorithms which allow us to manipulate a multiresolutional representation of our schemata, which include generalization and task decomposition. The minimal set also includes the rule: "IF <no rule for this situation> THEN <give random signal to actuators>". Other than this, only a "goal" percept and a corresponding cost function are given. This capability and associated learning-related functions are examined in detail below.

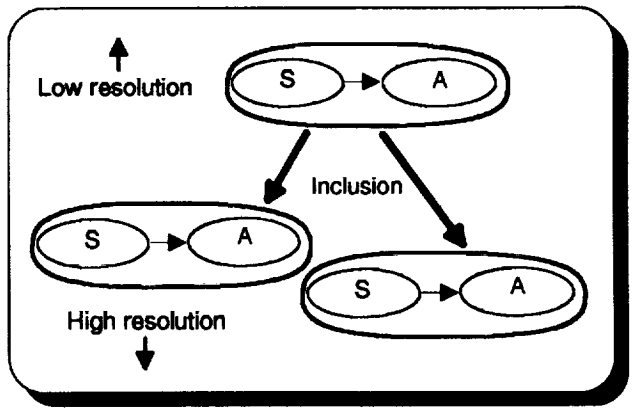


Figure 1. Multiresolutional Schema Representation

(2) Multiresolutional representation

A perfect example of a multiresolutional organization is any linguistic unit. Words form sentences. The sentences form paragraphs, paragraphs form sections, sections form chapters, chapters form articles, and all these articles make books, which also form libraries. Without its multiresolutional hierarchical organization, any book would be a gigantic word. This word would carry all the meaning of all the articles written here. This would create problems not only from an implementation standpoint but also from the point of view of searching through, storing, and communicating. The sentences and paragraphs do not need to be referenced frequently so we do not label them. On the other hand, subsections, sections, and articles carry a label and each of them has different broadness, granularity, or resolution. The title of the book summarizes the content of the book, and is of lower resolution than each of the titles of the articles; the titles of the articles refer to topics that are more specific than the book title. So, we can say that this structure is also nested in the sense that the title of the book includes information about its contents, and so on.

A distinctive property of a multiresolutional organization is the property of "nesting": sets of a particular resolution level are "nested" in a single unit of the lower resolution level. As a result, any multiresolutional representation is a multiple representation of a system at different scales: each level of resolution can represent the same entity with different degree of detail.

Now lets analyze why this multiresolutional nested organization is ever-present:

(a) Search time

Every time we store data -- and in our system we need to do it very often -- this data needs to be retrieved. The Baby Robot stores different percepts and different contexts for further use. These percepts need to be compared to the current percept. Thus, we have to search through the stored percepts. In general, we are interested in performing an

NP-complete procedures without paying for this by any increase in complexity.

It was demonstrated [5] that it is possible to do by repeating the same search several times at different resolution levels: starting with the lowest level (coarse granularity) and performing the search in a large envelope, and ending with a very high resolution space (fine granularity) however, in a very narrow envelope of search. Unlike the search processes (which propagate top-down) the processes of concept generation propagate bottom-up: fine granularity events and entities merge into lower resolution events and entities until the hierarchical tree of percepts and concepts can be assembled. If we store these percepts in a nested multiresolutional manner, our search time will be greatly reduced; it was proven by [5] that searching a nested multiresolutional structure reduces search time.

(b) Creation of schemata

Experiences are stored in a form opposite to the form in which the schema is presented (1). Experiences $E_{i,k}$ formulated at the i-th level of resolution for the k-th moment of time are interpreted as our memories about actions $a_i(t_{k-1})$ we performed in response to a particular situation $s_i(t_{k-1})$ and what was the result $s_i(t_k)$ of these actions

$$E_{i,k} = \{[s_i(t_{k-1}), a_i(t_{k-1}), \rho_i \rightarrow s_i(t_k)]\ g_{i,k}, \rho_i\}, \qquad (2)$$

where -is the value of the increment of "goodness" achieved during the interval of time $\Delta t = t_k - t_{k-1}$.

Experiences are grouped by their goodness in a class of "good experiences". Within this class, a set of subclasses can be created "good experiences at particular situations $\{s_n, n=1,2,...,N\}$. A generalized statement of experience is declared typical for a particular situation $\mathcal{G}[E_{i,n}]$ where $\mathcal{G}$ is an operator of generalization. In this paper we will use only the least sophisticated operator of generalization: weighted averaging assuming all weights equal to 1.

Generalized inverted experiences can be considered the basis for transforming them into hypotheses of the future schemata. After a while a set of schemata emerges as a result of inverting classes of similar experiences based upon the value of goodness delivered by a particular action in a particular situation. When creating schemata it is possible, even necessary, to create them to apply a recommended action to an entire class of situations, not just to its members. For example, the Astro-baby might create a rule such as "IF <obstacle visible> THEN <avoid it>". This will include every obstacle that it could sense, and it would not require to create specific rules for every kind of obstacle, every velocity and every direction. This is only possible in a system where lower resolution concepts include higher resolution ones as the components which are required to accomplish the lower resolution task.

(c) Task decomposition

If we create and store schemata in a nested multi-resolutional manner, then actions of a lower resolution level can be decomposed into sub-tasks that are goals for higher levels of resolution. This is done by the virtue of string generation for the higher resolution level in the following manner:

$$a_i(t_{k-1}) \Rightarrow a_{i+1}(t_{i+1,1}), a_{i+1}(t_{i+1,2}), \ldots, a_{i+1}(t_{i+1,m}) \qquad (3)$$

For example, in the previous given rule, "IF <obstacle visible> THEN <avoid it>", the action "<avoid it>" can be decomposed into "turn right", "orient up", and "slowly accelerate" (Astro-baby creates decomposition which vary with the type of situation). Each one of these actions can again be subdivided until we have a direct command to our actuators.

(3) Reasoning and Decision Making

We use only the most fundamental tools of reasoning which are critical for development continuous ("never-ending") processes of learning. Thus, all reasoning is based upon three major operations: a) determining whether a particular entity and/or event are related to a particular class, or not ("issuing the acknowledgment of inclusion"); b) finding an appropriate member of a particular class ("instantiating the class"); and c) forming a new class by determining a group of entities and/or events similar in some respect ("generalization").

The operator of generalization $\mathcal{G}$ is the key operator in a multiscale system . It is evoked and utilized to drastically reduce the required amount of computations by allowing to use the "typical" class representative instead of using different particular elements of the class.

For example, generalization is the kernel of the operation that takes schemata in one level of resolution, groups them in order of goodness (given by the cost function), and by means of correlating them it finds features that are in common in the good schemata. These features create schemata of lower resolution that create a new lower level of resolution.

Generalization is recursive in the sense that the highest level of resolution creates a level that is of lower resolution and this lower level creates other level given that sufficient instantiations of this schemata were collected to create clusters form the correlation. In the simulation part of this paper examples of how this generalization works in Astro-baby are given.

III. ASTRO-BABY

One of the possible realizations for Astro-baby is shown in Figure 2. It is possible to demonstrate that this simple configuration is able to provide for all necessary motions. In this paper, we won't concentrate on the subtleties of control for the configuration in Figure 2; we use abstracted translational and rotational vectors of control which should be obtained for any configuration.

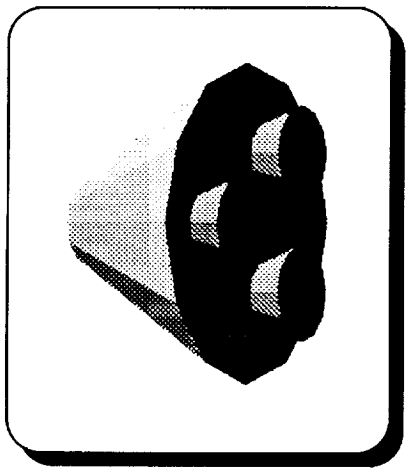


Figure 2. A configuration of Astro-baby

Early learning processes are studied here as applied to the systems which can be represented in a form of six-box-diagram (see Figure 3).

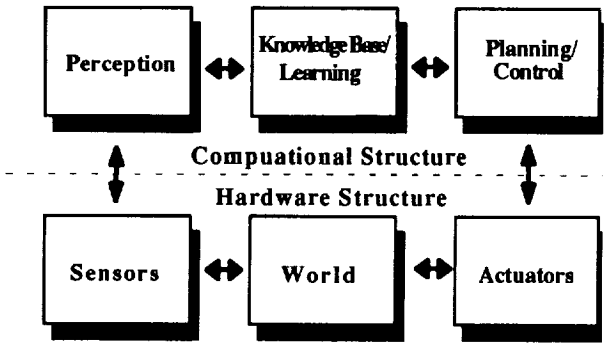


Figure 2. Six-Box-Diagram

The diagram is divided into the Computational Structure and the Hardware Structure which are mapped into another. The Hardware Structure is composed into three blocks: sensors, world, and actuators which are simulated by our program in order to be able to test BR. The other three boxes constitute the structure of intelligence and include: Perception, Knowledge Base and Planning/Control, are the basic components of BR or any control system for that matter. Perception receives the signals coming from the sensors, quantifies them, encodes them into a language suitable for storage and manipulation, and organizes them.

The Knowledge Bas module receives the encoded sensor information (percepts), puts it into correspondence with the rest of previously stored knowledge and finds relationships (rules) between the actions performed and the concepts perceived. Finally, the Planning/Control ("decision making") module uses all available information and the decision making mechanism to find the command sequence for the actuators.

It was demonstrated that the systems which can be represented by six-box-diagrams has to be equipped by at least two modalities of sensing and have at least two degrees of freedom in their actuation.

In a multiresolutional system, the six-box-diagram is becoming multiresolutional too. Thus it forms a structure of loops which can be called a multiresolutional nested structure (see [11]). In this structure, each lower resolution loop includes generalized activities of the adjacent higher resolution loop. In this paper we will consider a single loop but the results of reasoning can always be expanded to other loops.

The setting for sensing part is easily understood from Figure 3.

Sensors

The following sensors are given to Astro-baby:

A. Distance, Angle to Goal

In our simulation, we simulate real-world sensors by introducing errors. The angles to goal are expressed as Euler angles between the local axes of the ASR and the imaginary vector pointing towards the goal. Some error is introduced in distance to goal.

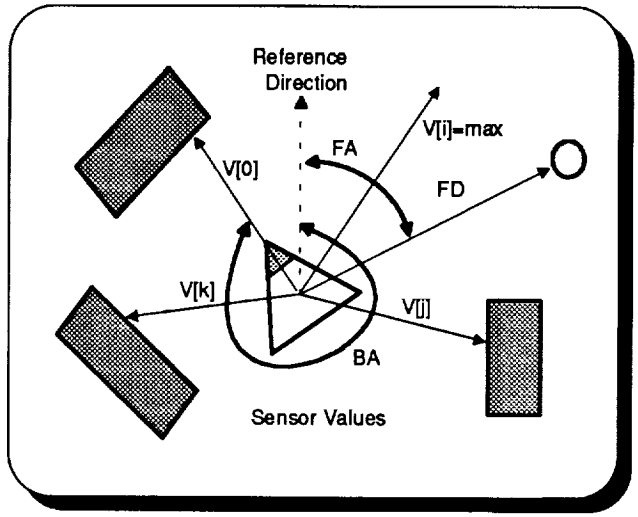


Figure 3. Sensing the position and orientation

B. Dynamic Avoidance Regions (DAR)

In [11] a DAR system for ground autonomous vehicles was introduced as a technique of substantially reducing the amount of information to be dealt with by fuzzifying the sensor. DARs are regions that grow bigger and fuzzier the further away they are from the astro-baby. This is a multiresolutional sensor where each one of these zones is a boolean sensor for Astro-baby. [4] describes a non-multiresolutional DAR. In contrast, our sensor will allow Astro-baby to create obstacle avoidance schemata of different resolution. In [12] an implementation example is

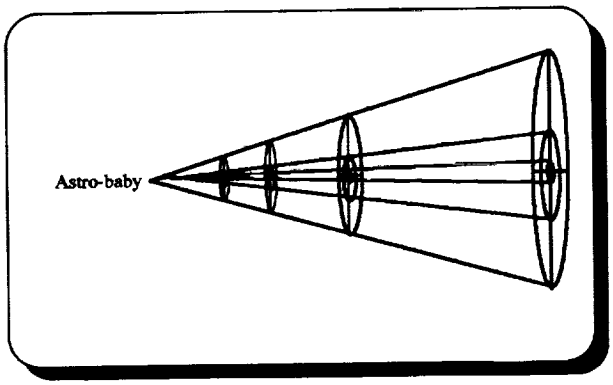


Figure 4. The DAR Sensors

given using sonar with only one DAR. By overlapping these fixed beam sonars we could create multiple DARs as shown in Figure 4.

C. Proximity sensors

A set of proximity sensors is included surrounding the body of the Astro-baby covering higher resolution proximity zones that are beyond the sensitivity of DARs. Errors and maximum reach are introduced in the simulation to make these sensors closer to real world sensors.

Actuators

Astro-baby has a source of translational and a source of rotational motion which it controls with three forces: F_x, F_y, F_z in local coordinates.

The Structure of Learning

Figure 5 describes one level of resolution in Astro-baby (the only one at the beginning of the learning). However, one can proceed with several levels of resolution by using the same picture; at the next resolution level one should use the same loop. The system is divided into two parts:

(a) Simulation of the hardware is composed of S (sensors), W (world), A (actuators). Actuators produce changes in the world, and the sensors sense the world. Our simulation includes dynamics. The existence of dynamics makes learning motion difficult, especially in 3-D.

(b) Astro-baby is composed of the Percept Knowledge Base (KB), Context KB, Schema KB and the learning loop. Astro-baby is unaware of the information stored in the hardware simulation, the only communication between the two boxes is done via sensors and actuators. An explanation of how this structure works is done as follows.

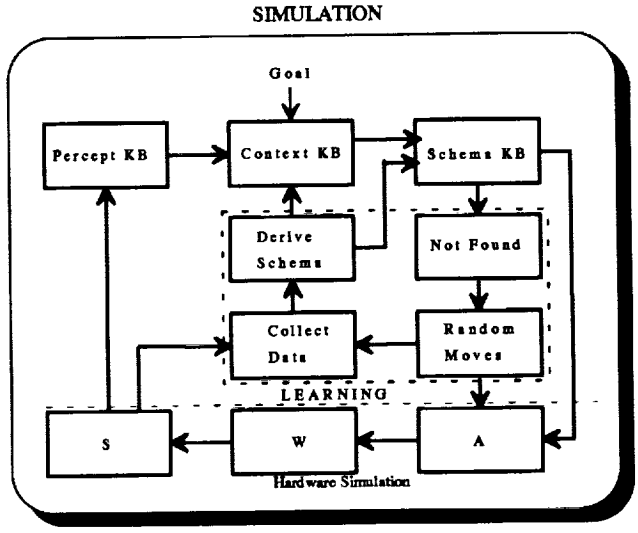


Figure 6. The structure of learning

When the Astro-baby is started, there is a first set of sensor values that come from the sensors, since our Percept KB is empty save for the goal percept. These sensor values are tagged and then stored in the Percept KB. So, since there is no previous percept, there is no context and therefore there is no schema for this percept. Thus, the Astro-baby must execute its first random movement. The random commands generator is a part of bootstrap knowledge (see Figure 6).

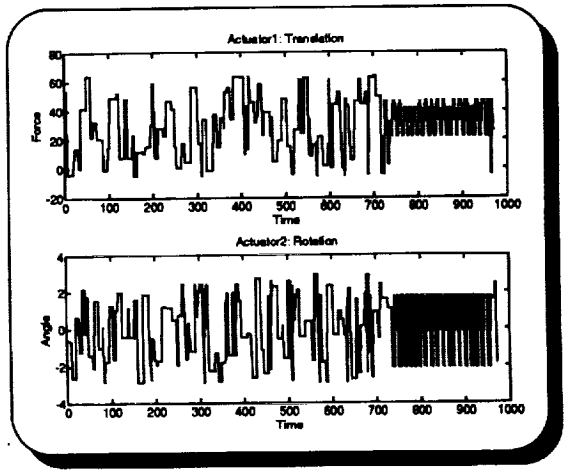


Figure 6. Random commands

These random commands generate random motion which is shown in Figure 7. As a result we have a change in the environment and a change in the goodness (change in the distance to goal divided by the step size). So, when the next percept is coming, it has the previous percept, change in goodness, and a context, but it still does not have a rule. But we have the following expression: previous

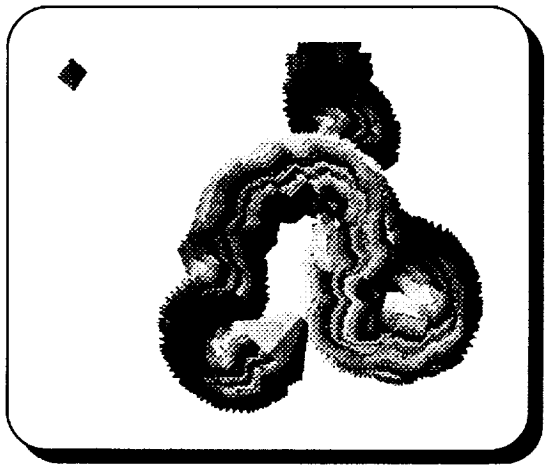


Figure 7. Random movements

percept, action, percept(now) and change in goodness for this action. Thus, we can create, via abduction, hypotheses (schemata that do not have enough statistical data collected either to become full Schemata or to be rejected, we call "Baby Schemata"). The following is a "Baby Schema": IF ((percept(n-1)) && (percept(n)) && (delta(goodness) is desired)) THEN action. The desirable goodness is given by the user as a threshold.

This Baby Schema will probably apply only in very selected situations and as a matter of fact they might give a different goodness in the same situation (because of dynamics) and its goodness could be very low (i.e. going away from the goal). But after we go through this process several times, we have a set of "Baby" schemata that cover some situations. If we have two or more Baby Schemata for the same situation, then the schema with better goodness is applied.

We can see in Figure 8 that this baby schemata causes Astro-baby to "spiral" towards the goal, The use of the baby schemata by Astro-baby improves its operation and at the same time it helps to collect more data of "good" baby schema. Then the generalization process starts working. Baby Schemata are ordered by goodness, and a correlation "engine" tries to find similarities among the baby schemata. First it tries to see if some of the values have been kept constant (within a fuzzy region), then it checks if the bad baby schemata also have this quality, If not it decides that this is a good characteristic in this class. In the case of the Astro-baby the Euler angle between the nose of the sub and the goal are

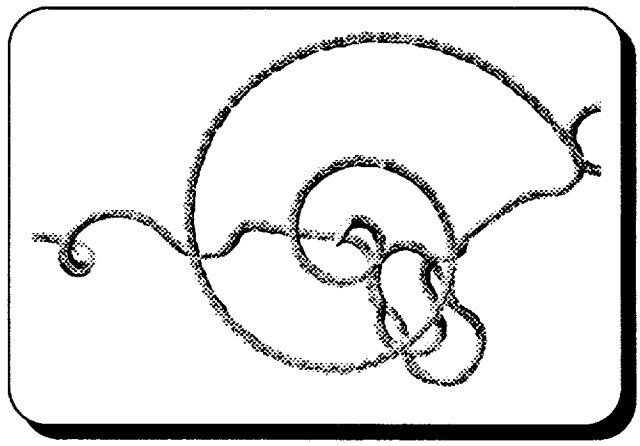


Figure 8. Testing Baby Schemata

very small in all the good schemata, so it creates a new low resolution schema that could be understood as:

$$\text{if <empty> and } goal_1 \text{ are required then} \quad (4)$$
$$\text{make } sensor_n = 0$$

Where $sensor_n$ is the Euler angle between the nose of the sub and the goal, and goal is minimize delta distance to goal/ delta step. The reason that it puts an <empty> in the Percept and Context parts of the situation is that it could not find any relationships between them in the good situations. When it will encounter obstacles, this part of the schema will not be empty.

Other relationships that we check if they where within a fuzzy boundary are the following: addition, subtraction, multiplication and division of two sensors and deltas of individual sensors (giving Astro-baby the ability to derive). These other relationships could also create schemata if they were good characteristics.

At this point Astro-baby has two levels of resolution, thus our goal (minimize delta distance to goal/ delta step) is passed to the lower resolution level. And this lower resolution level passes to the higher resolution level "make $sensor_n = 0$". The higher resolution level does not know how to do this, so it starts again to give random commands collecting them in baby schemata. But these Baby Schemata are judged with the new cost function. After a few trials Astro-baby creates some schemata that perform a Bang-Bang control on the Astro-baby (see Figure 9) trying to point at all times the nose of the sub towards the goal. In the traces of the tail of Astro-baby can be seen clear marks of this kind of control. The oscillations are big because it

does not have enough data to apply the exact amount of control needed, thus is overshooting.

Once some schemata are formed, new schemata are created that are very similar to the ones already created. These new schemata are used to quantitatively improve the previous ones. For example when schema (2) is formed, $sensor_n$ is not exactly zero but a small number; this number is refined every time the same schema is encountered. Thus the overshoot that we see in Figure 6 will become smaller and smaller.

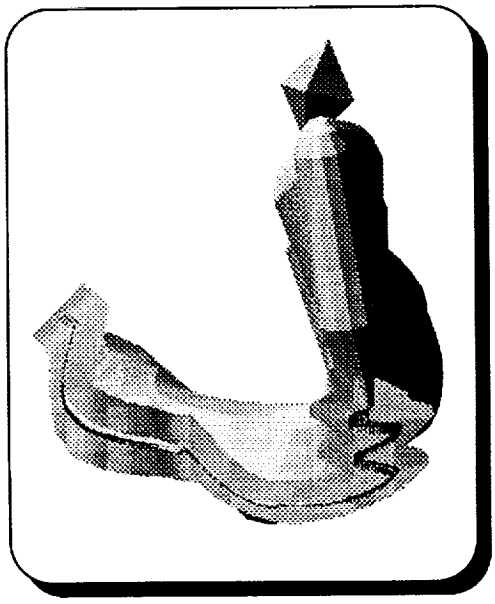


Figure 9. Bang-bang Control

The process of learning does not stop here; new levels of lower resolution appear when obstacles (of any kind: currents, low visibility, etc) are included in its world. The more the variety of circumstance which Astro-baby encounters, the more complex its own control system becomes and the richer its world representation becomes.

LEARNING CURVES

The following experiment was performed:

a) all knowledge was deleted from the database (except bootstrap)
b) the vehicle was set in a random position in the screen.
c) the goal was set in a random position.
d) when vehicle achieves the goal then go to b)

The learning curves where built by calculating the Euclidean distance between the vehicle and the goal in the initial position and dividing it by the number of steps (time) used to achieve it. The second graph shows the number of schemata versus time.

Case 1

Figure 10a and 10b show the case where no initial random moves where assigned before allowing generalization of schemata. It is possible to see that the learning curve is not very stable, although the performance of the submarine is improves, in some trials it has to perform several new random movements to be able to generalize rules that it does not have. It is also possible to see that the number of schemata levels up, the reason for this is that since the simulation is a closed environment, The set of rules that it found it is sufficient for its operation.

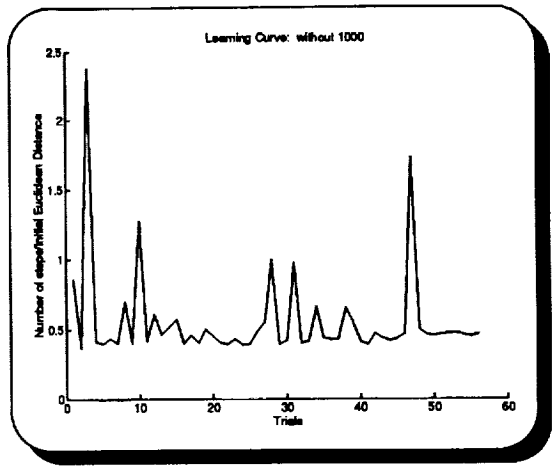


Figure 10a: Learning Curve

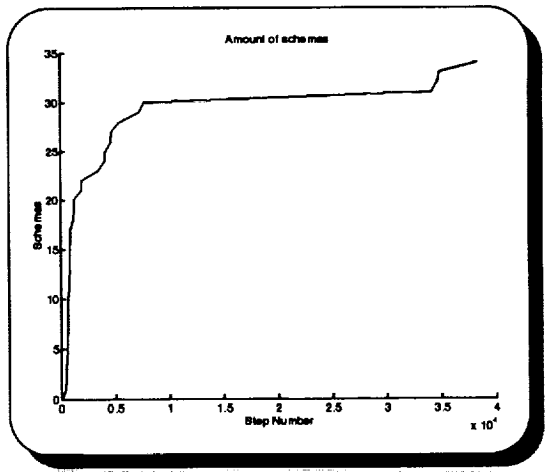


Figure 10b: Number of Schemata

Case 2

Figure 11a and 11b show the case where 1000 random moves where assigned before allowing any generalization of schemata. It is possible to see that the learning curve is a lot more consistent. It is also shown by this curve that the number of schemata found increases faster than in the previous case.

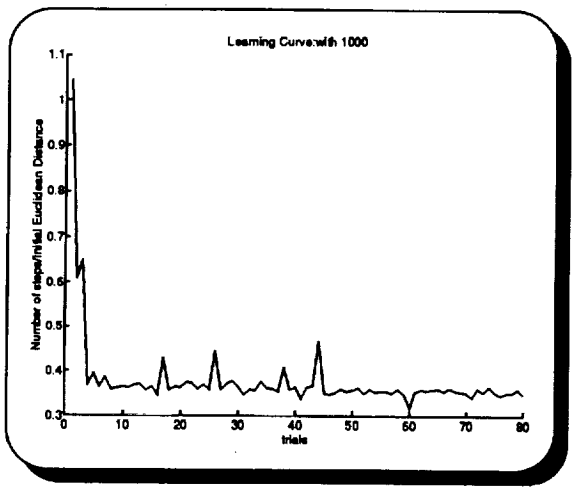


Figure 11a: Learning Curve

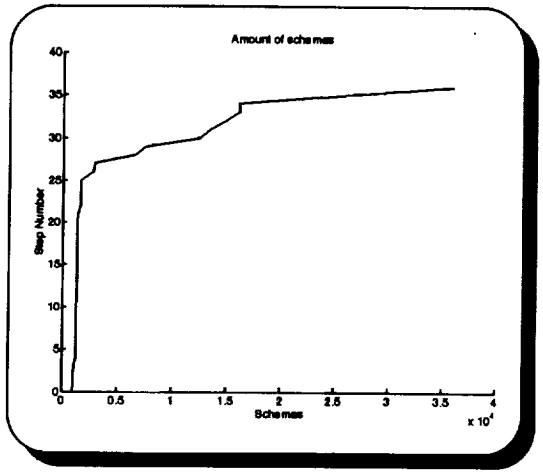


Figure 11b: Number of Schemata

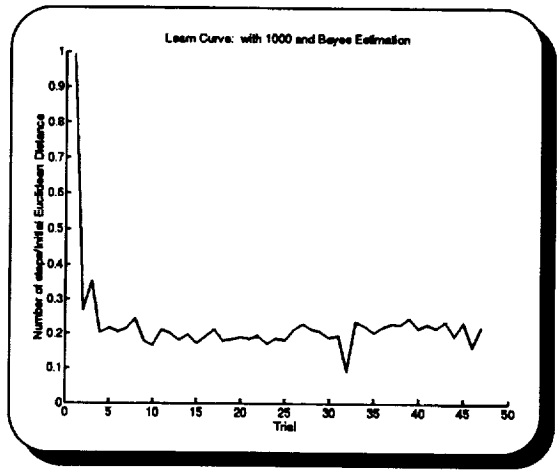


Figure 12a: Learning Curve

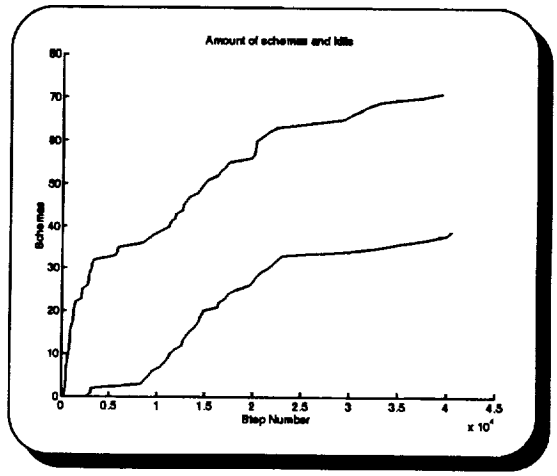


Figure 12b: Number of Schemata and Kills

Case 3

Figure 12a and 12b show also a case where 1000 random moves where assigned before allowing any generalization of schemata and once a Bayes estimator is used to rank the performance of the found schemata. If the Bayes estimator is low then the rule is eliminated. It is possible to see that the learning curve sinks lower than in the previous 2 cases. This is interpreted as the vehicle improving its performance, thus, achieving more distance per number of steps. Figure 12b shows two curves: the one on the top represents the total number of schemata that where created and the one on the bottom shows the total number of schemata that where eliminated using a Bayes estimator. It is possible to see that the number of schemata remains constant.

MAIN RESULT

The early learning process explored in this paper has demonstrated the following sequence of stages:

1. After the random sequence is completed, the learning structure determines that the way to achieve the goal is to minimize (null) the Euler angles. Now this result is considered to be the new goal of operation.

2. As the new goal is pursued, the system learns that it can be achieved by bang-bang control (or variable structure control). The system assign bang-bang control objectives and they become a new goal.

3. The results presented in positions 1 and 2 can be considered a formation of the low and high resolution levels. If the control objectives of the bang-bang control are considered to be a new goal, the next (the highest) resolution level is formed where the system learns how to provide the oscillation free motion.

4. These stages of the learning process constitute a multiscale system of dealing with experiences and creating a rule based controller. Our conjecture is that the process of

the ("never-ending") learning will continue in the multiresolutional fashion demonstrated above.

5. The bootstrap-knowledge set confirmed to be conducive of the ("never-ending") learning process.

CONCLUSIONS

1. A structure of learning mechanisms for an ASR has been described, based upon a theory of multi- resolutional schemata.
2. A system of simulation has been constructed which allows for testing the process of early learning.
3. The following observations have been made:

Astro-baby is a very adaptable learning system. Adaptable in two senses:

a) when it is installed in a robot it can deal with different kinds of situations and incorporate the knowledge extracted from the environment in its knowledge bases as percepts, contexts and schemata; and

b) in the sense that could be applied different platforms almost without modification and given the "proper" sensors and actuators for the goal assigned it will learn schemata about its own operation and its interaction with the environment.

Astro-baby discovers Bang-Bang Control and applies it efficiently to perform the assigned task.

GLOSSARY

An attempt is made to formally and concisely define terminology used frequently throughout this paper, so as to minimize interdisciplinary misunderstandings.

Resolution - The granularity at which a particular situation is viewed based upon the size of a minimum distinguishable unit of space

Multiresolutional System - A system which views the world at multiple levels of granularity

Multiresolutional Hierarchy - A graph-like structure used to demonstrate the organization of data in a multiresolutional system

Learning - The process of acquiring knowledge about the world and developing behavior patterns to deal with accomplishing a specified task within the framework of the acquired knowledge

Bootstrap Knowledge - An initial set of information or knowledge, including, more specifically, techniques required for learning

Goal - A desired outcome of events

Intelligence - The ability to efficiently process and organize knowledge acquired through learning

Intelligent System - A system exhibiting the properties of intelligence and using them for control

Task Decomposition - A process whereby a given Goal is subdivided into sub-Goals which are achievable at a particular level of Resolution in a given temporal discrete. Often used by Intelligent Systems to reduce computational complexity

Percept - A set of sensor values acquired at a particular level of resolution at a particular moment of time

Context - Various information about the world at a particular time. Context may include Percept information, as well as data from other sources

Action - A set of activation of actuators in a body, to perform a task, usually set by a Goal

Situation - A grouping of information about the world and the task at hand, consisting of a Percept, a Context, and a Goal

Schema (*pl. schemata*) - A logical operation relating a Percept, Context, and Goal with an Action. Can be expressed as follows: *IF* (Percept & Context & Goal) *THEN* Action

REFERENCES

[1] Warren, Charles W., "A technique for autonomous underwater vehicle route planning", IEEE J. Oceanic Eng., v.15 n.3, July 1990 pp. 199-204

[2] Meystel, A., "Baby-Robot: on the analysis of cognitive controllers for robotics", Proc. IEEE Int'l Conf. on Systems, Man, Cybernetics, Tuscon Arizona, 1985

[3] Eilbert, J., Meystel, A., Venetsky, L., Zietz, S., "Baby Robot: Learning to control goal-oriented behavior", Proc. IEEE Workshop on Intelligent Control, 1985

[4] Mataric, M.J., "Integration of representation into goal-driven behavior-based robots", IEEE Trans. on Robotics and Automation v.8 n.3, pp. 304-312

[5] Maximov, Y., Meystel, A., "Optimum Design of Multiresolutional Hierarchical Control Systems," Proceedings of the 1992 IEEE International Symposium on Intelligent Control, pp. 514-520 , Glasgow, Scotland, U.K.

[6] Dresher, G.L., Made-Up Minds, MIT Press, 1991

[7] Arbib,M.A."Modelling neural mechanisms of visuomotor coordination in frog and toad", Competition and Cooperation in Neural Nets, Volume 45 of Lecture Notes in Biomathematics, Chapter 21, Proceedings, Kyoto, Springer-Verlag 1982

[8] Lyons,D., "RS: A formal model of distributed computation for sensory-based robot control", COINS Tech. Report 86-43, UMass Amherst, 1986

[9] Arbib, M.A., The metaphorical brain 2: Neural networks and beyond, "Schema Theory" (Chapter 5), John Wiley, 1989

[10] Meystel, A., "Multiresolutional schemata for motion control", Drexel Univ. IMPAQT Center Technical Report, 1993

[11] Meystel, A., "Knowledge-based nested hierarchical control", Advances in Automation and Robotics, pp. 63-151, JAI Press 1990

[12] Floyd, C.A., Kanayama, Y., Magrino, C., "Underwater obstacle recognition using a low-resolution sonar", Seventh Int'l Symp. on Unmanned Untethered Submersible Technology, September 1991

N94- 35052

A Genetic Technique for Planning a Control Sequence to Navigate the State Space with a Quasi-Minimum-Cost Output Trajectory for a Non-Linear Multi-Dimensional System

C. Hein,
Martin Marietta,
Moorestown NJ 08057

A. Meystel,
Drexel University,
Philadelphia, PA 19104

Abstract:

There are many multi-stage optimization problems that are not easily solved through any known direct method when the stages are coupled. For instance, we have investigated the problem of planning a vehicle's control sequence to negotiate obstacles and reach a goal in minimum time. The vehicle has a known mass, and the controlling forces have finite limits. We have developed a technique that finds admissible control trajectories which tend to minimize the vehicle's transit time through the obstacle field. The immediate application is that of a space robot which must rapidly traverse around 2-or-3 dimensional structures via application of a rotating thruster or non-rotating on-off thrusters. An air bearing floor test-bed for such vehicles is located at the Marshal Space Flight Center in Huntsville Alabama.

However, it appears that the developed method is applicable to a general set of optimization problems in which the cost function and the multi-dimensional multi-state system can be any non-linear functions, which are continuous in the operating regions.

Other applications include the planning of optimal navigation pathways through a traversability graph; The planning of control input for under-water maneuvering vehicles which have complex control state-space relationships; The planning of control sequences for milling and manufacturing robots; The planning of control and trajectories for automated delivery vehicles; And the optimization of control for automotive racing vehicles and athletic training in slalom sports.

Introduction:

Many optimization techniques have been developed such as: Bolza, Mayer, Lagrange, Green, Gradient Methods, Dynamic Programming, and Optimum Spacing of Corrective Thrusts[2]. Other methods that are based upon searching tend to be less efficient, exhausting much time searching the entire space at a given resolution level.

Techniques that are based on direct methods tend to be limited to specific types of cost functions and systems. In such methods, the system must be expressible as a single function. For example, a discrete maximum principle method (from Pontryagin), is based upon Calculus of Variations. The problem is formulated as finding the control function, u[k], which minimizes J=Sum of F[x,u,k] with constraint x(k+1) = f[x,u,k]. Where x are the system states, f is the system transition function, and F is the cost function. The solution is found by first, defining the costate vector, p(k), and the Hamiltonian, H[x,u,p(k+1),k], as:

$$H = F[x,u,k] - \langle p(k+1, f[x,u,k] \rangle$$

Let p and x take variations, lambda. Then form 2*n first order difference equations / canonical state equations. Solve:

$$\frac{\text{partial } dH}{\text{partial } dU} = 0,$$

for optimum (minimum) relative to the boundary conditions. Then apply the condition of transversality to find u.

Background:

The purpose of our algorithm is to determine and plan an ideal path. Since it is a plan, unpredictable perturbation forces, and sensor, actuator errors cannot be considered. It is assumed that a final design must incorporate additional methods such as closed-loop feedback, periodic re-evaluation, or variable structure control methods in conjunction with this path-planning method. Planning is necessary since, none of the feedback methods could be used without a basic path plan such as that developed by this technique. In the absence of general rules for directly deriving an optimal control sequence, given arbitrary constraints such as vehicle dynamics, control limitations, and an arbitrary obstacle environment, a basic strategy is to search the control space for a control sequence which avoids obstacles while minimizing the time to reach the goal. We call such a control space search an "input-space" search.

An alternate method would be to search the "output-" space for an optimal trajectory, and then to find a corresponding admissible control sequence that produces it. However, there are many trajectories which cannot be achieved through any admissible control sequence due to limitations in vehicle dynamics. Namely, these limitations are: vehicle mass and the appliable acceleration, force, or power. The set of all trajectories includes those which are produced from admissible control, plus those which cannot be.

Likewise, the input-space corresponds to all trajectories that can be traversed, plus those which cannot be due to obstacles or other constraints. Consequently, it is not obvious which of the two candidate search spaces is more efficient to search. The decision then rests upon ease of implementation or computation.

Searching the input space requires posing various permutations of admissible control sequences and testing their outcomes for conformance to the output constraints, which in our case would be obstacle collisions. The output trajectory due to a control input sequence is easily computable as long as we have a system equation. Since the output trajectory is a function of the control input (and not vice-versa), this is a one-to-one mapping. In other words, there is exactly one output trajectory corresponding to a given control input sequence.

Searching the output-space requires posing various valid output trajectories, and working backwards for each one to derive an admissible control input sequence that would produce it, if any exist. The inverse to the system equation is required to compute the input sequence from the output trajectory. Therefore, since it appears more direct and tends to be more well-formed (since inversion is avoided), we have chosen to search in the control input space.

In general, the control space for this type of problem consists of a vector function, C(t), which is continuous in time. At any time instance, t, C(t) represents the control force vector on the vehicle. That is, it represents a force magnitude and direction (F,theta). Both quantities may change abruptly at any time. The magnitude is usually limited to some admissible range, Fmax, so it is continuous from 0 to Fmax, while the direction may vary continuously from 0 to 360-degrees.

The continuous quantities (F,theta) can be quantized into discrete quanta. As the quanta grow infinitesimally small, the quantization approaches the continuous function. Similarly, the functions can be discretized in time, so as to be approximated by discrete time series. Again, as the sample period grows infinitesimally small, the discrete approximation approaches the continuous function in time. Therefore, through quantization and discretization, the time/value continuous system can be approximated with arbitrary precision.

Discretization in time, and quantization of

values of the control-input, allows a systematic search to be conducted over the possible sequences of control-input values. The control string, C(t), is analogous to a gene. We search for improved solutions by inserting random mutations into the control gene, and then evaluating the outcome as a whole. Mutations with favorable outcomes are retained, and further mutations are applied and evaluated recursively. This technique therefore belongs to the class of genetic algorithms.

As we attempt to approximate the continuous system with smaller force and time quanta (higher resolution), the number of terms, -and the values they may take-, grows infinite. Therefore, the search must be conducted at reduced resolution.

The best solution from a reduced resolution search will generally not be equivalent to the best solution of a search conducted at a higher, or infinite, resolution. Since, the higher resolution search space includes the lower resolution search space, if their best solutions differ, then the higher resolution solution must be better. Therefore, to approach a more optimal solution, we wish to conduct a search at the highest practical resolution.

It is desired to obtain the best solution that can be found in a given practical amount of computation time. A high resolution search consumes much computation time searching the entire space at high resolution. The maximum resolution is thus limited by the available compute time.

The required compute time is related to the number of permutations that must be examined in the search space. The number of potential permutations is:

$$(M * D)^N$$

Where M is the number of F-force magnitude quanta, D is the number of theta-direction quanta, and N is number of time samples or potential switching-points from the origin to the goal. From this, we can see that the computation time is exponentially related to the resolution. For instance, a doubling in force (magnitude or direction) resolution implies a 2^N increase. A doubling in time resolution implies a squared increase in compute time. Therefore, minimizing resolution is computationally expedient.

Note that we use the term "potential" permutations above. The actual number tends to be less-than the potential number since many potential permutations are obviated by spatial boundaries and obstacles in a depth-first search. Therefore, this is an upper-bound.

There is a time-space duality between the resolution of the switching times, and the resolution of the force magnitude/direction quantities. For instance, if the force resolution is halved while the time resolution is doubled, equivalent control trajectories can still be maintained by time averaging the now more rapid, but less precise force-switching quantizes.

In 2-D space, the minimum force resolution, without loosing any degrees of freedom in movement directions, is one magnitude quantity and four direction quantities, or two magnitude quantities and two direction quantities. In either case, the M*D product equals four (4). In the former case the magnitude would correspond to the maximum admissible force, and the directions could be 0, 90, 180, and 360-degrees. In the later case the magnitudes would correspond to (+) and (-) the maximum admissible force, and the directions could be 0 and 90-degrees. (Clearly this is just two ways of saying the same thing.)

We now describe an iterative refinement based method to achieve a solution in less time than would otherwise be required by performing a single search at the equivalent resolution level to produce a solution of similar quality. The strategy is to perform a course resolution search over all the control input space to determine the most promising

region. Next a higher resolution search is performed only within this region to find a more optimal sub-region. Then continually higher resolution searches are recursively performed within the sub-region. The search complexity is roughly equal at each level.

For computational simplicity, the minimum force resolution of four (4) force magnitude/directions is used throughout the process, with increasing switch-point time resolutions. Time-resolution may later be traded for increased force-resolution, as a final integration step, once sufficient resolution has been obtained.

Construct Initial Coarse-Resolution Path:

The initial number of switch-points for the initial coarse resolution search depends upon the complexity of the feature space, the available computation time, and the search rate of the computer. Since a direct method for determining the initial number of switch-points is unknown, a variety of techniques can be used such as, experience, trial and error, and heuristics which take into account features of the obstacle space. For instance, fewer switch-points may be placed in areas which are far from obstacles and therefore require fewer course corrections.

The number of initial switch-points can impact the quality of the final solution, since the coarse resolution search determines the region of focused attention. Therefore, care should be taken to investigate the larger space before delving into a fine resolution search of a selected sub-region.

In general, the initial search should use the largest number of points possible to complete the search computation in the allotted time. This is estimated by measuring the rate at which switch-point nodes are processed by the computer. The maximum number of nodes to be processed is given by the $(M*D)^N = 4^N$ formula. It is recommended that at least half of the remaining computation time be devoted to the remaining resolution levels. For instance, a Sun Sparc-10 processes about 80,000-nodes/second. If the total allowable computation time is 60-seconds, then allowing 30-seconds for the initial search would imply:

$$N = Log_4(T*R) = Log_4(30*80{,}000) = 10\text{-nodes}$$

Actually, due to obstacles reducing the number of available paths, the search can be completed in much less time, which allows more time for refinement. Consequently, this estimate for N forms a conservative upper bound.

In general, there is no way of knowing what the distribution of the N switch points in time should be, nor of whether a solution can be found using N switch points. Therefore, it is useful to have an estimated upper bound on the solution cost. Then, the points can be distributed along this cost. This distribution and cost bound can be obtained through experience, trial and error, or a heuristic method.

For our navigation application, we determined the initial switch point set-size and distribution by the applying the following 2-step process:

1. Perform a Dijkstra search on the obstacle vertices, where cost is taken to be the Euclidean distance between vertices. This finds the minimum distance path from the origin to the goal, and it ignores the vehicle dynamics.
2. For each segment in the path from (1), compute the optimal switching times to move the vehicle along each segment ensuring the vehicle velocity is zero at each segment's end-points. This decouples the stages of the problem into a series of simple linear re-positioning sub-problems.

The result of this process is an initial control string, C(t), and a fixed upper bound cost.

The resulting initial pathway may not coincide with the optimal dynamic path, but it is usually a good one that is close. If it is close enough, then the refinement process can find the optimal path.

In the general case, if an intuitive heuristic approach is unknown for a given state-space, but a reasonable upper-bound cost can be estimated, then an initial control string can often be found by concatenating and evaluating N variable switching points that are distributed uniformly along the cost axis. If no solution can be found within the estimated upper bound cost with N-points, than either a higher resolution search (>N) or a higher upper-bound cost is required. If it is the former, an increase in N would exceed the allowable computation time, thus the problem would be partitioned into sub-problems which would be solved similarly but separately. The separate solutions would be combined to form the initial control string which would then be refined as described below.

Genetic Iterative Refinement or Increasing the time-resolution:

Once an initial trajectory is found, attempts are made to refine the switching points by inserting new variable switching points between the existing switching points. The switching point string, C(T), is considered to be a control-string or list of force-directions with time-deltas between them. A new switch-point B can be inserted between two elements A and C by relating their delta times as follows:

$$Tc - Ta = (Tc - Tb) + (Tb - Ta).$$

The time relationships of the remaining elements in the string remain unchanged.

The force-directions and time-deltas of the existing switch-points are held constant, while a search is performed over a newly inserted variable point, by simulating the trajectory determined by the new string. The trial is made four times, once with the new point set in each of the four switch directions. If a new trajectory reaches the goal in less time than the previous best trajectory, then the new trajectory becomes the best trajectory. At least one of the force-directions will equal the previous best trajectory, since one force-direction matches that of the previous switching-point in the string, and therefore represents no force-change.

Inserting the new switch nodes at random distances between the existing nodes seems to yield the best results. This is apparently due to the greater variation available through random insertion points.

To control the size of the trajectory change introduced by a varying-point, it is useful to insert a second switching-point, between the new varying point and the next switching-point, that resumes the force application to its value prior to the new varying point. In this way, the size of the trajectory change is controlled by the length of time the new force is applied. Inserting the second switch-point at a random distance between the new varying point and the next switch-point seems to be advantageous, especially if the distribution is weighed heavier close to the varying point.

Note that as the density of points increases in time, the time resolution is effectively increased, and the length of new force applications becomes shorter, which creates smaller trajectory variations. Therefore the search space automatically becomes more restricted as the time resolution increases. Consequently, the search complexity is maintained at a constant level by reducing the search space as the resolution increases.

It is usually beneficial to insert many new varying points into the control string at once, and then to test all their permutations together, since often multiple simultaneous trajectory changes are very beneficial to minimize the path yet avoid obstacles. Once again, the maximum number that can be inserted at any one time is N due to computational limitations.

References:

[1] Anderson, B., Moore, J., "Linear Optimal Control", Prentice Hall.

[2] Leitmann, G., "Optimization Techniques with Applications to Aerospace Systems", Academic Press.

[3] Brayton, R., Spence, R., "Sensitivity and Optimization Simultaneous changes in M-components", Elsevier Scientific Pub.

[4] Kuo, B. "Digital Control Systems", Holt, Rienhart, and Winston.

[5] Slottine, Li, W., "Applied Non-linear Control", Prentice Hall, Inglewood Cliffs, NJ, 1991.

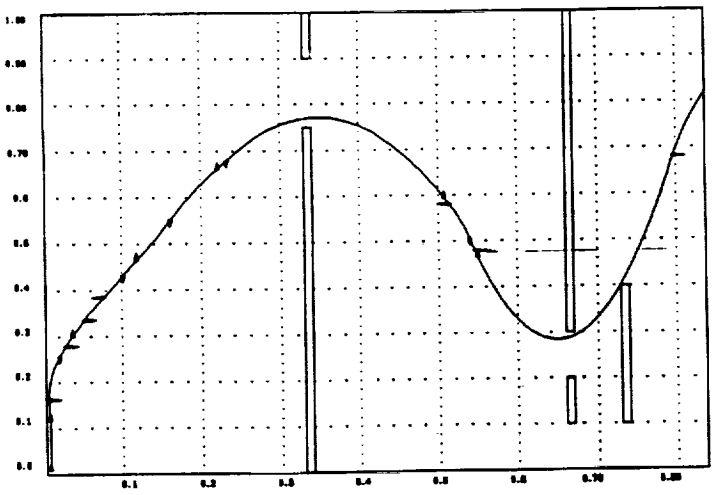


Figure 8 - Trajectory for obstacle field B in x,y.

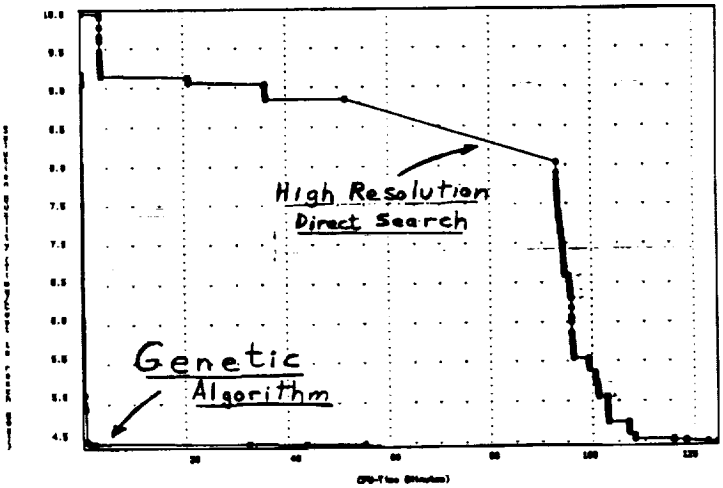


Figure 9 - Convergence rate comparison.

N94-35053

Modelling Heterogeneous Processor Scheduling for Real Time Systems

J. F. Leathrum, R. R. Mielke, and J. W. Stoughton
Old Dominion University

Abstract

A new model is presented to describe data-flow algorithms implemented in a multiprocessing system. Called the resource/data flow graph (RDFG), the model explicitly represents cyclo-static processor schedules as circuits of processor arcs which reflect the order that processors execute graph nodes. The model also allows the guarantee of meeting hard real-time deadlines. When unfolded, the model identifies statically the processor schedule. The model therefore is useful for determining the throughput and latency of systems with heterogeneous processors. The applicability of the model is demonstrated using a space surveillance algorithm.

Introduction

Improvement in throughput and latency in hard real-time algorithms increasingly is realized through the use of parallel constructs. However, it is known that the characterization of performance in parallel systems is particularly difficult, and is compounded when a heterogeneous processing environment is introduced. The processing time of a particular piece of code is dependent on the processor type scheduled to execute it. This may complicate the analysis of throughput and latency. Current strategies focus on the use of data-flow graphs to describe algorithm play, and then an external processor scheduling scheme is imposed for graph execution. A method for the development of processor scheduling within the data-flow model is presented. It provides a deterministic method of predicting the effects of various schedules, both in homogeneous and heterogeneous processor environments.

Scheduling tasks on parallel computers can be divided into two categories, static scheduling and dynamic scheduling. Static scheduling methods [1,2,3,4,5,6] allocate tasks to processors during compile time. The time required to schedule is incurred only once, independent of the number of times the application is executed. Dynamic scheduling methods [7,8] allocate tasks to processors at run time, taking advantage of current knowledge of the state of the system. Dynamic scheduling methods generally provide better resource utilization, but at the penalty of real-time overhead to complete the scheduling. Dynamic scheduling also may incur a degradation in expected performance resulting from slight variations in task time, even when the task time decreases. Heterogeneous systems generally employ dynamic schedulers [7,8].

The applications under consideration in this paper have hard real-time deadlines. Once an input arrives, the corresponding output must be generated by some maximum time deadline. Inputs must be accepted at some maximum rate while still meeting performance deadlines. Consequently, completing tasks as early as possible is not as important as avoiding a late finish.

The paper starts by reviewing the data-flow graph model implemented on a homogeneous processor system [9]. Methods of analyzing performance are described, and are followed by a discussion of how the graph plays in steady state in order to establish the

requirements for schedules.

The paper then introduces a new model, based on the data-flow paradigm, which allows a schedule to be represented in the model. The schedules used are cyclo-static [1]. These schedules are characterized by processors being scheduled to nodes in a cycle so that each processor periodically revisits the nodes in its cycle. Once the schedule is included in the model, the resulting graph can be implemented on dynamic scheduling systems with full guarantee of achieving hard deadlines, even in the presence of time varying tasks.

The model is then extended to heterogeneous processor systems. This extension allows the same deterministic analysis of system performance. In this manner different schedules can be compared for their ability to meet deadlines. Finally the system is applied to an example of a space surveillance algorithm [9] to demonstrate the analytical ability of the model.

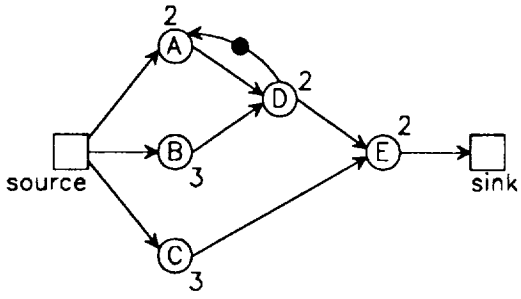


Figure 1. Data-flow graph (DFG)

Data-flow Model

The ensuing discussion is based on a data-flow model which employs a data-flow graph (DFG) [10]. Nodes in the graph represent tasks or modules of code. Each module has a maximum execution time, but times may vary to smaller values. Arcs in the graph represent data dependencies between the code modules which must be satisfied prior to a module executing. Tokens on an arc represent the communication of data from one module to another. The graph plays under the rule of earliest fire - as soon as a node is enabled by tokens on all incoming arcs, it fires. Figure 1 shows an example DFG with node times indicated next to each node.

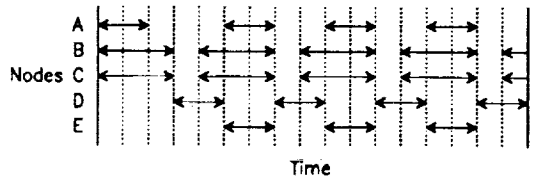


Figure 2. Graph play.

Execution of the graph is initiated by source nodes injecting data into the graph. The graph operates with repeated inputs which are periodic in nature. Data is consumed by sink nodes to signify completion of a computation. The rate of information flow through the graph is managed by injection control at the source. The injection rate is set to guarantee graph performance. The expected graph play if each node requires its maximum time is given in Figure 2 for the DFG in Figure 1.

The goal of the graph is to guarantee hard real-time deadlines. These guarantees are made based on steady state graph play. Steady state is defined by operation with maximum node times under repeated inputs. Any deviation in node times should result in no worse than steady state completion times for nodes.

This section defines the system performance and the associated steady state graph play. Then the requirements a cyclo-static

schedule must meet to satisfy steady state play are defined. This establishes the data-flow model on which current work is based.

Performance

Performance is based on the assumption that there are sufficient processor resources to execute the graph in steady state. Performance is determined by steady state graph play and is characterized by the throughput and latency [1,9]. There is no guarantee that the resource requirement is a minimum bound, but it is sufficient to play the graph.

Throughput is bounded by the length of the critical circuit where the length of a circuit is the time per token in the circuit. The critical circuit length (CL_{cr}) is defined by

$$CL_{cr} = \max\left(\frac{NT_i}{IT_i}\right) \quad \text{for } i=1...C \quad (1)$$

for a graph with C circuits, where NT_i is the total node time in circuit i and IT_i is the number of tokens in circuit i. Throughput is then bound by

$$TP \le \frac{1}{CL_{cr}} \quad (2)$$

The maximum rate that data can be injected into the graph is defined in terms of the time between inputs (*TBI*) which is equivalent to CL_{cr}.

Latency is bounded by the length of the critical path, or the longest path from source to sink. Latency (L) is then defined as

$$L \ge \max(PL_i) \quad \text{for } i=1..P \quad (3)$$

for P paths from source to sink, where PL_i is the length of path i.

With sufficient resources, throughput can be maximized, and latency minimized, for a given graph. The number of resources (R) is bounded to meet performance requirements and can be found as the number required to meet steady state requirements as defined later.

Steady State Graph Play

The purpose of the proposed model is to predict performance of DFGs operating in steady state with repeated inputs and where the assigned node times are worst case values. Steady state graph play is completely characterized by the node firings which occur in one time period of length TBI [9]. This is demonstrated by a Gantt chart termed the Total Graph Play (TGP) diagram.

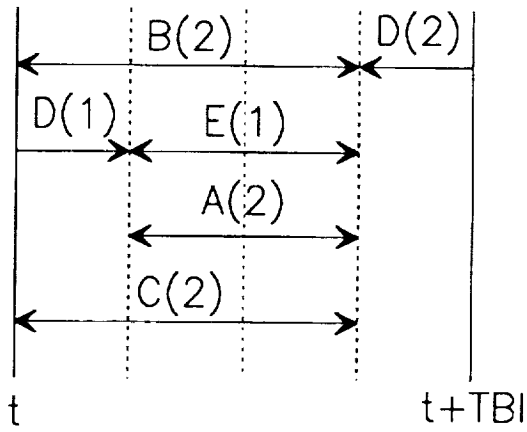


Figure 3. Total Graph Play.

The TGP diagram is constructed by drawing a Gantt chart of length TBI, with time ranging from t to $t+TBI$. The nodes are inserted into the TGP diagram using their worst case values. The nodes in the critical path are placed in the TGP diagram first. Node firing times are wrapped around from time $t+TBI$ to time t as needed to indicate a change in data packet number. Then the remaining graph nodes are placed in the

TGP based on their precedence relationship with existing nodes. Each node in the TGP diagram is associated with an iteration number which identifies the input data packet that produced a firing of the node. The iteration number is decremented by one during a wrap around the TGP diagram. This is a result of the times t and $t+TBI$ being equivalent in steady state, but for different input data packets. Thus all node firings in a TGP diagram are not necessarily associated with the same input. The TGP diagram for the graph in Figure 1 is shown in Figure 3.

Steady State Scheduling Requirements

The goal of steady state scheduling is to provide a schedule which will guarantee graph play with performance no worse than steady state play. This requires that processors be available to graph nodes no later than worst case steady state graph play dictates. It is not the purpose of this paper to determine a schedule, but rather to model and analyze schedules. However, some important criteria which schedules must meet are presented.

The first criterion for a schedule concerns the use of processor idle time. Different schedules cause different distributions of processor idle time. However, all schedules must meet the constraints established by the TGP diagram and data-flow graph precedence relations. Cyclo-static schedules can be represented by circuits of nodes over which processors traverse during the play of the graph. There may be one or more circuits for a given graph and schedule, and each processor may reside on only one circuit. As a processor traverses from one node to another on this circuit, it accumulates a non-negative idle time while waiting for the next node firing. Each transition appears in the TGP diagram once since each node fires exactly once in the interval TBI. Thus, the total time processors spend in graph play during the interval of TGP diagram is the sum of the total computing effort (TCE), which is the sum of all node times, and the idle time (T_{idle}). This must equal the total computing time available in the TGP diagram which is given by the number of available processors (R) times TBI resulting in the requirement

$$R \cdot TBI = TCE + T_{idle}. \qquad (4)$$

This equation represents a necessary requirement for a schedule, where the schedule determines the value for T_{idle}.

An example schedule for the TGP diagram in Figure 3 consists of two processor circuits, one containing nodes A, D, E, C and the other containing only node B. The idle times that processors spend between the node pairs are: node A to node D, 0; node D to node E, 0; node E to node C, 1; node C to node A, 2; and node B to node B, 1. Thus there are 4 processors, TBI is 4, TCE is 12, and T_{idle} is 4. This results in Equation 4 taking the form 4*4 = 12+4, which meets the requirements.

Graph Markings

The graph markings (token-arc pairings) are necessary for the following work. First a set of steady state markings are found. Then the initial graph markings are determined.

For steady state markings, the approach is to find a set of markings which satisfy the TGP diagram. The original graph is also needed, including the tokens necessary to meet its initial conditions. A timing point is then established in the TGP diagram (time t). For convenience, t is taken as that point in the TGP diagram when the source fires. Then the set of nodes with activity at time t^+ are examined. Two conditions arise: a node fires at time t, or a node was already

executing at time t and continues at time t^+. A final condition is handled at time $(t+TBI)^-$ where if a node completes before time $t+TBI$ but the ensuing node on a outgoing arc does not fire before time $t+TBI$, then a token must be placed on the outgoing arc. The conditions are handled by the following procedure:

A) All nodes firing at time t must have all tokens available to fire. Therefore, tokens are placed on all incoming edges for these nodes. If a token already existed on an edge from the original graph, an additional token should not be added.
B) All nodes currently executing are considered as going short and completed at time t^-. When a node finishes, it deposits tokens on all of its outgoing edges. Therefore appropriate tokens are placed to indicate the completion of each node in this category.
C) For all nodes in the graph, check all nodes associated with outgoing arcs. If these nodes do not fire before $t+TBI$ in the TGP diagram, then place a token on the arc between the two nodes.

When performed on the DFG in Figure 1, two new tokens are added: between nodes D and E, and between nodes C and E.

Given the markings for the graph in steady state, a set of initial markings are determined to allow the graph to initiate play and achieve steady state. The purpose for finding the initial markings is to remove some tokens from the graph to simplify analysis. These markings are found by advancing the steady state markings as far as they will go without injecting new inputs. This is a marking coinciding with the graph being played in steady state and then having inputs stopped and allowing the graph to come to rest. This all happens at some time less than zero. Then at time 0, new inputs are injected initiating graph play. The transient from allowing the graph to execute from steady state with no inputs and the transient from initiating graph play compliment each other allowing the graph to return to steady state. The resulting initial markings for the graph in Figure 1 match the initial markings provided.

Processor Schedule Model

A model is presented which explicitly incorporates processor scheduling in the data-flow context. The model is intended for use with hard real-time systems where meeting deadlines is the primary concern. The model is developed by utilizing the features of cyclo-static scheduling [1] in the data-flow graph such that the graph will play by the rules of the schedule in the presence of time varying nodes and dynamic schedulers.

The model is first presented for a homogeneous processor system and then later extended to heterogeneous systems. The homogeneous system demonstrates the new model which includes resource scheduling in the data-flow graph. It will also provide the mechanism for development of the necessary tools for heterogeneous systems.

Resource/Data-Flow Graph (RDFG)

A new data-flow graph model, called the resource/data flow graph (RDFG), is introduced which explicitly shows the relationship between resources and nodes, . In this paper, the resources are considered to be processors, although the model is extendable to other resources such as communication channels. The RDFG introduces new arcs in the DFG to describe how processors migrate through the graph nodes in a cyclo-static schedule. Processors

are represented as tokens in the graph, and thus are treated similarly to data in the evaluation of the graph.

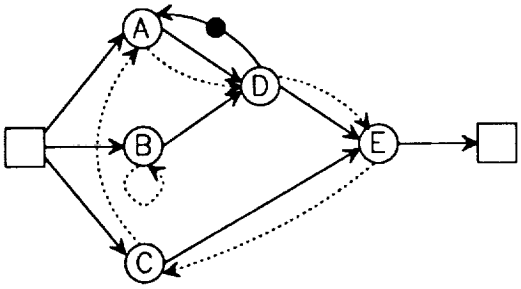


Figure 4. Resource/Data Flow Graph (RDFG).

The RDFG is developed from the DFG and a given cyclo-static schedule. A cyclo-static schedule may be represented as a set of circuits, called processor circuits, describing the order in which processors execute nodes. Processor circuits are formed by adding arcs, called processor arcs, to the original DFG so that a token representing a processor, when placed in this circuit, visits in order all nodes assigned to the processor. Each circuit may have multiple processors, but each processor can reside on only one circuit. The resulting circuits are disjoint and encompass all nodes. An example RDFG for the cyclo-static schedule developed earlier and for the graph in Figure 1 is illustrated in Figure 4.

Given a RDFG, finding a set of steady state markings is the next step in the analysis. This is done by the same process as for the DFG. These markings introduce the necessary processor tokens representing processors onto each processor circuit. A fully static schedule is represented as an RDFG where all processor circuits have only one token. Figure 5 is a possible set of steady state markings imposed on Figure 4. The initial graph markings are shown in Figure 6.

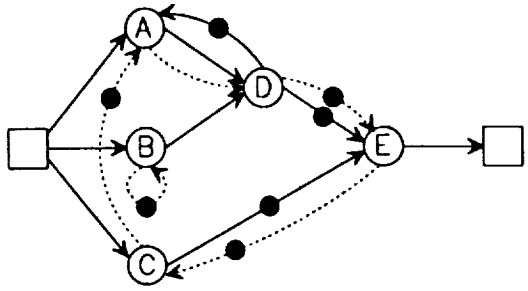


Figure 5. Steady state graph markings.

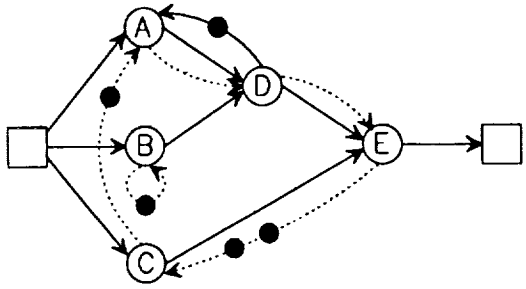


Figure 6. Initial graph markings.

Graph Reduction

The addition of processor arcs to the DFG may create redundant arcs in the RDFG in terms of necessary precedence relationships. It is helpful to delete redundant arcs to reduce the number of paths and circuits which must be considered during analysis and to better allow observance of graph activities.

The determination of which arcs are redundant is based on node precedence. An arc from node *a* to node *b* is redundant (and thus can be removed) if an alternative path exists from *a* to *b*. The alternate path may contain several arcs (and thus have stronger precedence relationships) or may be a single arc. If a single arc, the two arcs are equivalent, and thus either arc may be removed. The following rule may be used to eliminate one of the arcs. If the two arcs are of the same type (data or

control/processor), then randomly select one for elimination. If they are different, then eliminate the control arc since the data arc has more meaning in data-flow, maintaining the original graph in the RDFG, while the processor arc is a means only to scheduling. Figure 7 was reduced in preparation for analysis.

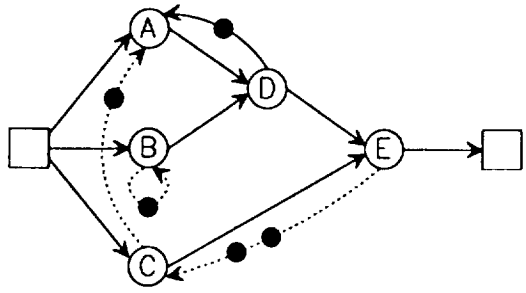


Figure 7. Reduced RDFG.

Unfolded Graph

A third graph is introduced to allow improved observation of individual node and processor behavior which is particularly useful in the heterogeneous system discussion. This graph is an unfolded graph such that all processor-node combinations are presented. The object is to create an equivalent graph having only one processor token per processor circuit. This means that each node in the unfolded graph is executed by only one processor which allows the examination of the relationship of a node to a specific processor. The foundation of the unfolded graph is found in [2] where the purpose was to transform a data-flow graph to allow fully static scheduling. The difference between their work and the current work is the RDFG already has a schedule imposed which may reduce the necessary unfoldings.

The basic method of developing the unfolded graph is to replicate the graph k times, where k is the greatest common multiple of the number of processor tokens on each processor circuit. Arcs with initial tokens are moved to reflect the data dependencies between iterations. The source of the arc remains intact, but the destination is moved from the node in iteration i to the corresponding node in iteration $(i+n) \bmod k$ for n initial tokens on the arc. The modulus function results in iteration $k-1$ cycling back to iteration 0 for repeated inputs. Figure 8 is a k-unfolded RDFG $(k=3)$ for the graph shown in Figure 7.

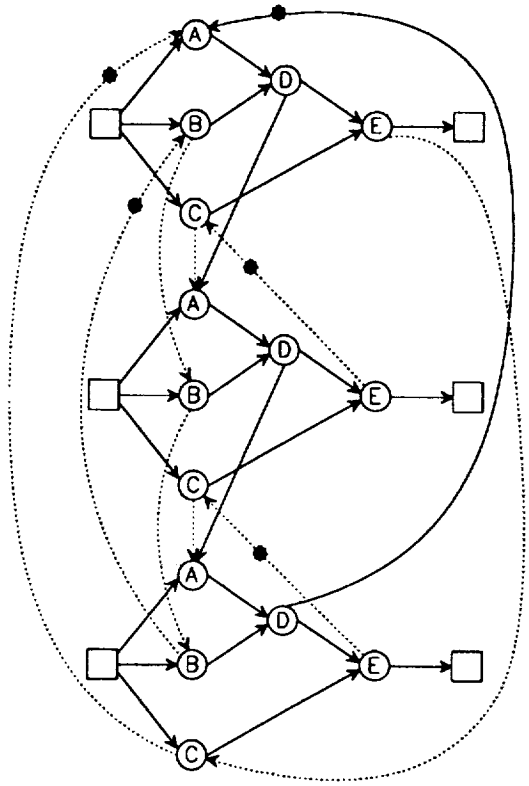


Figure 8. Unfolded RDFG (k = 3).

Hard Deadline Guarantee

It is a well known phenomenon that small reductions in the execution time of a node may adversely affect performance, thus

preventing the system from meeting deadlines. The reason performance may degrade is that the graph may enter resource saturation where there are more enabled nodes than processors and some work must be delayed beyond the worst case start time. Much work has been done to address this problem [11,12].

Static scheduling inherently avoids resource saturation and thus guarantees worst case performance. The new model extends this property to dynamically scheduled systems. The RDFG is a graph that plays on a dynamic system just as the equivalent DFG would play on a static system. However, the RDFG has more restrictive precedent constraints as a result of the processor arcs. The processor arcs guarantee that a processor is available for a node to fire without dictating which processor is used. The static schedule simply makes a direct association between processors and tokens on processor edges which is not necessary in dynamic scheduling.

Heterogeneous Processor Systems

The purpose of this section is to demonstrate the utility of the unfolded RDFG in analyzing heterogeneous systems. The construction of the unfolded RDFG depends on the specification of a processor schedule. The scheduling approach used for this paper is to assume a worst case processor schedule. It is known that this precludes the representation of certain valid heterogeneous schedules, but developing heterogeneous schedules directly for the RDFG is left for future work.

The method used addresses the standard practice often used for heterogeneous systems where a node is scheduled exclusively on one class of processors. In the RDFG, such a schedule appears as multiple processor circuits, where each such circuit is assigned a single class of processor. For the schedule used in the previous section in Figure 7, node B would execute on one class of processor and the other nodes would execute on a second class of processor. This is a special case of the systems under consideration.

This section first demonstrates the benefits of using the unfolded RDFG. Then, the analysis of the graph to determine available throughput and latency improvement for a given schedule is presented. The effect of different processor assignments to the RDFG is shown followed by a discussion of heuristic approaches for improving performance for a given schedule operating in a heterogeneous environment.

Unfolded RDFG

The unfolded RDFG is particularly useful for evaluating heterogeneous systems. The reason for this is the fully static nature of the RDFG which allows each node to be mapped to a single processor. Thus, the effect of a processor on the node processing time can be characterized and included in the analysis.

For example, suppose that two of the four processors in the unfolded RDFG in Figure 8 are capable of executing nodes one time unit faster than the time indicated in each node. Then the node times in the processor circuits can be adjusted to reflect the capabilities of the specific processor assigned to the circuit. Figure 9 illustrates an unfolded RDFG for a given processor assignment.

Performance Characteristics

Once the unfolded RDFG is found for a given schedule and processor assignment, the system performance can be computed. The methods of finding throughput and latency in

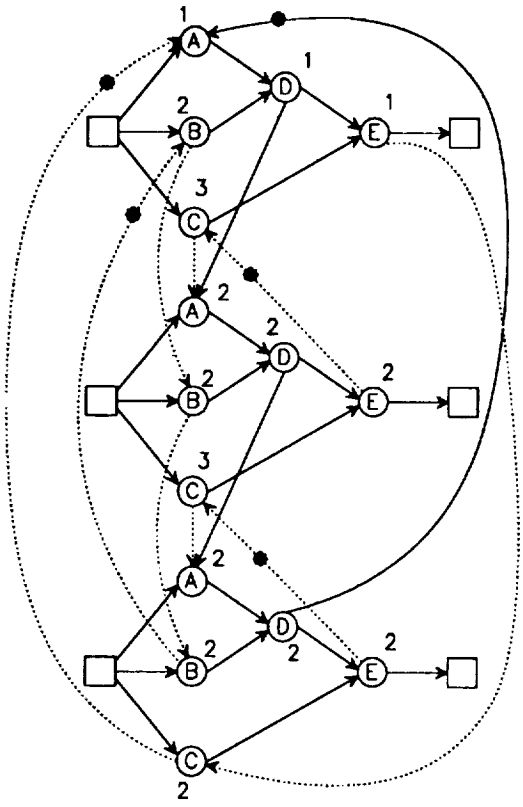


Figure 9. Unfolded RDFG with initial tokens and processor assignment.

the unfolded graph are similar to those used in the original DFG. Performance is still limited by the critical path and critical circuit for the given graph.

Throughput: The throughput for the heterogeneous system is found by determining the critical circuit in the unfolded RDFG. However, this critical circuit defines the throughput for k inputs given a k-unfolded RDFG. Thus the lower bound on throughput (TP_{LB}) is

$$TP_{LB} = \frac{k}{CL_{cr}(unfolded\ RDFG)} \quad (5)$$

though the graph as defined may not be able to play with periodic inputs at this throughput.

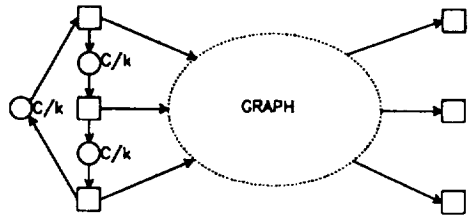


Figure 10. Injection control of an unfolded RDFG.

To reestablish periodic inputs, injection control is imposed on the k sources. The sources are placed in a circuit of length C with k nodes of time C/k between each source firing, as shown in Figure 10. In this manner, each source fires periodically every C time units. Provided $C/k \geq TP_{LB}$, C/k will dictate the actual throughput. This is possible since the addition of the circuit at the set of sources creates no other circuits. Injection control allows a throughput to be defined with periodic inputs by imposing a critical circuit about the sources.

The critical circuit for the graph in Figure 9 has a length of 10. Therefore, each source can fire every 10 time units, and the 3 sources can be made periodic by firing them in sequence separated by $\geq 10/3$ time units.

Latency: Prior to controlling the sources, latencies can be computed from each source to its corresponding sink (L_i for iteration i). The lower bound latency for the given schedule and assignment is

$$L_{LB} = \max(L_i) \quad for\ i=1...k. \quad (6)$$

Imposing periodic inputs may have adverse effects on latency, possibly even increasing latency above L_{WC}. This occurs because some sources may fire before the graph is ready to accept them. Thus, tokens will wait at some point in the critical path,, increasing the total time spent on the critical path.

To compute the periodic latency (L_p), the aperiodic and periodic firing sequences for sources must be compared. The issue is whether a periodic source fires earlier or later than its corresponding aperiodic source. Firing later is not an issue since the source then lags behind the graph, and thus will not add any time to the latency. Firing early means the graph will have to catch up adding the amount the source fired early to the latency (EF where $EF = 0$ if the source fires late). The resulting latency is

$$L_p = \max(L_i+EF_i) \quad for\ i=1...k \quad (7)$$

for the graph. Note that the latency can be improved up to L_{LB} by increasing the throughput which reduces EF_i. Therefore, a trade off between throughput and latency is available for improving performance.

For the graph in Figure 9, the three iterations have latencies of 4, 6, and 6 respectively prior to injection control at the inputs. If the inputs are fired periodically every 10/3 time units, then iteration 0 fires on time, iteration 1 fires 1 1/3 time units after the earliest it could, and iteration 2 fires 1/3 time units early. Therefore, the latencies remain the same under periodic inputs.

Processor Assignment

An alternative to decreasing throughput to preserve latency is to consider other processor assignments. A one to one assignment of processors to processor arcs is made for a given schedule. Different assignments produce different performance characteristics. Therefore, the performance obtained by various assignments should be considered.

The different assignments change performance in several ways. Assignments potentially affect latency on different iterations, the critical circuit of the unfolded RDFG, and the idle time spent on the critical path on different iterations. The combination of these three factors may result in many different performance points.

The various assignments should be compared to determine the appropriate choice. Not all markings need to be considered since many potential markings can be proven equivalent. The appropriate assignment will then be based on the desired goal of the implementation, either to improve throughput or latency.

Schedule Potential

An attempt is made to provide insight to the ability for a given schedule to have improved performance over other potential schedules for a given graph. A schedule ideally should allow nodes to have lower execution times without providing further graph constraints beyond those present in the original DFG. Different schedules change node times in heterogeneous systems, and evaluating how different schedules improve performance requires knowledge of the peformance goals for a particular heterogeneous system. But the ability of a schedule to avoid imposing further graph constraints can be characterized.

Ideally, processor arcs in the unfolded RDFG should have little influence on throughput improvement. If the destination node of a processor arc is allowed to fire

early by its incoming data arcs, the processor arc may delay the node until a processor is available. In this manner the processor arc imposes further constraints upon the graph. This restricts how much faster the graph runs over and above the data-flow requirements. Thus schedules which result in more idle time on processor arcs are desirable.

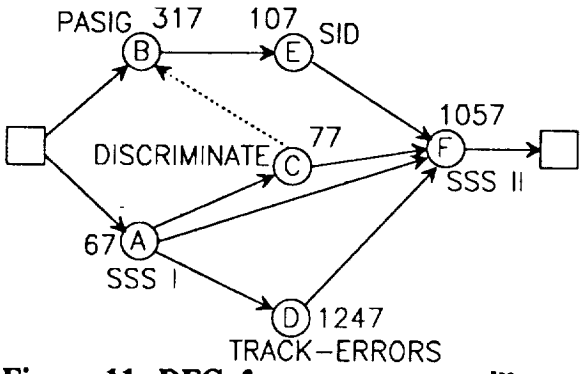


Figure 11. DFG for a space surveillance algorithm.

A Space Application

An example algorithm is presented as a test case for the model. The algorithm is a space surveillance algorithm [9]. Figure 11 illustrates the DFG for the algorithm. At each iteration, the algorithm accepts as inputs the current position coordinates of multiple targets located in the instrument's field of view. The algorithm identifies each target and plots the target's trajectory in three dimensional space. Node labels describe the algorithm operations, and the relative time required for each operation is shown beside the node. Note that a control edge has been added to the graph. The extra control edge reduces the number of required processors from 4 to 3 on a homogeneous processor system.

The performance measures for the DFG are TBI = 1247 and latency = 2371 with 3 processors. Suppose that one of the processors used was capable of executing nodes at 75% of the specified time. The result of scheduling this processor is presented.

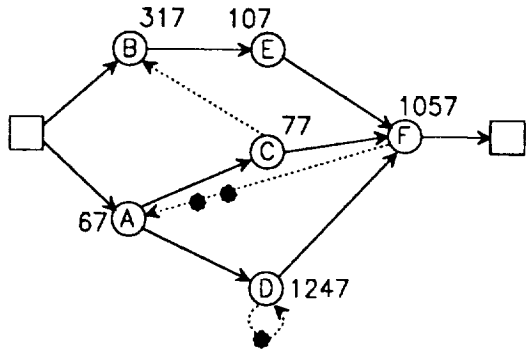


Figure 12. RDFG for a space surveillance algorithm.

A possible cyclo-static schedule for the space surveillance algorithm is for one processor to execute node D and the other processors to cycle through the other nodes in a circuit consisting of nodes A-C-B-E-F in the specified order. The resulting RDFG is shown in Figure 12 and the unfolded RDFG in Figure 13. When the faster processor is applied to the D node, each D node will execute in 936 time units. This results in a new TBI of 1057 and a new latency of 2060. The faster processor cannot lower TBI further since the slow processors handling node F now dictate the throughput.

Conclusion

The RDFG is a graph model which allows schedule development with data-flow constructs on parallel computing systems. The RDFG also guarantees that hard real-time deadlines are met. The graph will meet deadlines even when executed on a system with a dynamic scheduling scheme such as a queue.

An extended RDFG, termed the unfoled RDFG, allows addressing heterogeneous

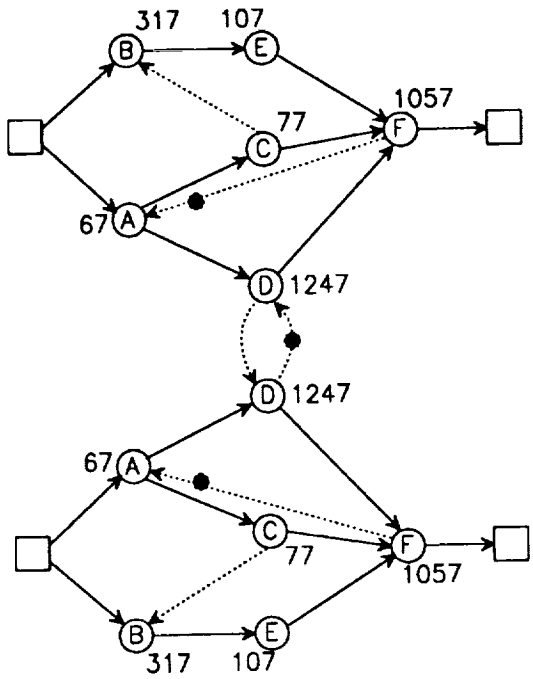


Figure 13. Unfolded RDFG for a space surveillance algorithm.

processor systems. It is a fully static form of the RDFG where each node executes on one processor. Then each node time can be adjusted to describe the execution on a specific processor for a given processor assignment. The graph is then analyzed to find the available performance in terms of throughput and latency.

The unfolded RDFG was demonstrated on an example problem for heterogeneous processors. The space surveillance algorithm is an example of a control system with repeated inputs for which the model is well suited. The unfolded RDFG provided the ability to analyze the performance in terms of throughput and latency.

References

[1] D. A. Schwartz, T. P. Barnwell, III, "Cyclo-static multiprocessor scheduling for the optimal implementation of shift invariant flow graphs," *Proc. ICASSP-85*, Tampa, FL, Mar. 1985.

[2] K. K. Parhi, D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Transactions on Computers*, v. 40, n. 2, pp. 178-195, Feb. 1991.

[3] B. Shirazi, M. Wang, G. Pathak, "Analysis and evaluation of heuristic methods for static task scheduling," *Journal of Parallel and Distributed Computing*, v. 10, pp. 222-232, 1990.

[4] S. M. Heemstra de Groot, S. H. Gerez, O. E. Herrmann, "Range-chart-guided iterative data-flow graph scheduling," *IEEE Transactions on Circuits and Systems-I*, v. 39, n. 5, pp. 351-364, May 1992.

[5] M. Marrakchi, "Optimal parallel scheduling for the 2-steps graph with constant task cost," *Parallel Computing*, North-Holland, v. 18, pp. 169-176, 1992.

[6] S. Ha, E. A. Lee, "Compile-time scheduling and assignment of data-flow program graphs with data-dependent iteration," *IEEE Transactions on Computers*, v. C-40, pp. 1225-1238, Nov. 1991.

[7] G. C. Sih, E. A. Lee, "A Compile-Time Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," *IEEE Transactions on Parallel and Distributed Systems*, v. 4, No. 2, pp. 175-, February 1993.

[8] Y. Hu, Z. Xie, X. Lu, "Approaches to Decentralized Control of Job Scheduling for Homogeneous and Heterogeneous Parallel Computing Systems," *Future Generation Computer Systems*, North-Holland, v. 6, pp. 91-96, 1990.

[9] S. Som, R. R. Mielke, J. W.

Stoughton, "Prediction of performance and processor requirements in real-time data flow architectures," *IEEE Transactions on Parallel and Distributed Systems*, v. 4, n. 11, Nov. 1993.

[10] K. M. Kavi, B. P. Buckles, "A formal definition of data flow graph models," *IEEE Transactions on Computers*, v. C-35, pp. 940-948, Nov. 1986.

[11] G. K. Manacher, "Production and stabilization of real-time task schedules," *Journal of the Association for Computing Machinery*, v. 14, n. 3, pp. 439-465, July 1967.

[12] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, v. 17, n. 2, pp. 416-429, March 1969.

C++ Planning and Resource Reasoning (PARR) Shell

James McIntyre, Alan Tuchman, David McLean and Ronald Littlefield
AlliedSignal Technical Services Corporation
Goddard Corporate Park
7515 Mission Drive
Seabrook, MD 20706

ABSTRACT

This paper describes a generic, C++ version of the Planning and Resource Reasoning (PARR) shell which has been developed to supersede the C-based versions of PARR that are currently used to support AI planning and scheduling applications in flight operations centers at Goddard Space Flight Center. This new object-oriented version of PARR can be more easily customized to build a variety of planning and scheduling applications, and C++ PARR applications can be more easily ported to different environments. Generic classes of PARR objects for resources, activities, constraints, strategies, and paradigms are described along with two types of PARR interfaces.

Keywords: AI, Planning, Scheduling, Shell

INTRODUCTION

The Artificial Intelligence (AI) software group at AlliedSignal Technical Services Corporation has been developing expert system software for Goddard Space Flight Center since 1985. These systems use expert system technology to automatically create conflict-free schedules. Many expert systems use commercial programs known as expert system shells written in either Lisp or Prolog. These systems often suffer a performance penalty when they are used on personal computers or engineering workstations. Because we developed our programs using conventional programming languages on readily available computer hardware, they are able to rapidly schedule hundreds of events.

We have adapted an evolutionary prototyping approach to software development. The prototyping aspect of this approach dictates that we gather initial requirements, create or evolve a prototype, evaluate the prototype to update the requirements, and repeat the process until the prototype satisfies the software users. The evolutionary aspect dictates that our prototypes are not developed to be thrown away, but instead are based on generic, reusable software tools. The prototypes are updated, not rebuilt, with each prototyping cycle. As we build a system, we develop new software tools. When the tools appear to be useful outside the project where they were first implemented, we add them to our tool library. In many cases, we are able to base a new system entirely on tools that exist in our library.

BACKGROUND

We delivered the Earth Radiation Budget Satellite (ERBS)-Tracking and Data Relay Satellite System (TDRSS) contact planning system to the ERBS flight operations team in 1987. It was the first AI expert system application at Goddard that supported flight operations. It is used to build schedules of requests to TDRSS for communications contacts to send commands to ERBS and to download data.

The ERBS-TDRSS system created its schedules using the first version of the Planning And Resource Reasoning (PARR) tool (McLean *et al.*, [1]). PARR is an expert system shell designed expressly for scheduling applications. Its knowledge base is organized into activity classes. Each activity class represents something that can be scheduled; in the ERBS-TDRSS system, the activity classes represent the scheduling of communication contacts between ERBS and a Tracking and Data Relay Satellite (TDRS) via a specific antenna. Each activity class contains both strategies which suggest ways that the activity can be scheduled and constraints which specify limitations on when the activity can be scheduled.

PARR works as an intelligent tactical planning tool to put specific activities on a timeline by following the strategies and checking the constraints found in its knowledge base. Unlike traditional scheduling systems that try to anticipate all possible scheduling conflicts before scheduling an activity, PARR uses conflict resolution strategies to reschedule activities that do not meet their constraints (McLean, *et. al.*, [2]). PARR's knowledge base forms a strategic plan, a list of broad strategies used to schedule activities. PARR uses this strategic plan to create a tactical plan, a list of specific activities with specific times and durations.

PARR is unusual in its use of a combination of conflict avoidance and conflict resolution. While many scheduling systems backtrack and change their partial schedules extensively, PARR mimics the way people create schedules. We have found that people obtain insight into a scheduling problem and quickly develop rules like "that requires most of the resources, so schedule it first" or "if we have trouble fitting this into the schedule, we can reduce its duration by a few minutes and make it fit." PARR captures this kind of strategy in its knowledge base along with the more traditional conflict avoidance rules like "this can only be scheduled when that resource is available." By dynamically applying both types of rules, PARR limits the amount of search that is required to build a timeline and is able to produce a conflict-free schedule much faster than other systems.

After developing the ERBS-TDRSS system, we realized we had the appropriate tools in our library to quickly build more complex scheduling systems (McLean, *et. al.*, [3]). In 1991, we delivered the Explorer Platform Planning System (EPPS) to the flight operations team for the Extreme Ultraviolet Explorer (EUVE) spacecraft (McLean, *et. al.*, [4]). EPPS shares much of its implementation with the ERBS-TDRSS system, including a revised version of PARR. Like the ERBS system, EPPS generates schedules of TDRSS requests. In the development of EPPS, we found that our simple way of expressing constraints was insufficient. The ERBS system simply checked whether a conflicting resource or activity was scheduled during the period under consideration. EPPS needed to schedule activities that affected power and tape consumption, and the system needed constraints that could track these resources. We realized that new scheduling problems would need new resource models to support new kinds of constraints. Rather than creating new code to support each new type of resource, we created more general resource models that could be used to model many different kinds of resources. EPPS was the first system to use resource models

The next significant PARR development occurred as we developed the Hubble Space Telescope (HST) Servicing Mission Planning and Replanning Tool (SM/PART) which we delivered in 1992 to help the HST flight operations personnel build integrated timelines and command plans to control the activities of the HST, the space shuttle and the space shuttle crew for the HST servicing mission in December 1993 (Bogovich, *et. al.*, [5]). SM/PART uses PARR in a slightly different way. In our previous systems, each activity class in the knowledge base is scheduled repeatedly over time. In SM/PART, each activity class is scheduled only once during the schedule, but the activity class definitions and resource availability may change, causing the need for replanning. This changed our understanding of the relationship between tactical and strategic planning. Users may not always be able to completely define all of their activities before scheduling and may need to tweak the knowledge base during the tactical planning process.

In addition to these three systems, we also developed several prototype scheduling systems that gave us insight into other useful capabilities. We prototyped systems that merged schedules that were created by other scheduling systems and built a new version of PARR that takes multiple schedules as inputs and merges them into a single conflict-free schedule. Another prototype scheduled observations of stars and represented each potential target star as a different activity class. To create an efficient schedule, we created a system that found the shortest path between the targets stars using a Hopfield net (Yen, *et. al.*, [6]). These prototype systems pointed up the need for a more general system to select the

activity classes to be scheduled, the order in which they are considered for scheduling, and which of their strategies are used.

A NEW PARR

After using PARR to develop three major systems and several prototypes, we began to find limitations with our implementation of PARR. Each new project required new strategies, new constraints, and new resource models to support the new constraints. It was more and more difficult to make changes needed to add the new strategies, constraints, and resource models without affecting other parts of the program. PARR also needed ways to add broader kinds of strategies to encompass more than one activity class.

We also gained insight into PARR's interfaces. PARR has two primary interfaces: an interactive interface and an interface using files. Each new system or prototype required a different interactive interface. Some of the differences were dictated by the fact we implemented each system on different hardware platforms and operating systems; others were driven by the user's needs. We saw we would continue to need to adapt the interactive interface to meet the users' needs. The file-based interface also continued to change. New strategies, resource models, and constraints required changes in the formats of the knowledge base files. The prototypes that explored the relationships between the activity classes also needed changes in the file formats. We saw that if the file structures were extensible, old versions of the files would still be usable after new features were added.

We decided that the best approach to these problems was to reimplement PARR in C++. C++ is an object-oriented language based on C, the language used to implement all previous versions of PARR. The objects defined in C++ contain both data and methods, or functions that operate on the data. Each object has a well-defined interface, so interactions between objects are easy to understand and an object can be changed with little impact on other objects. Objects can inherit data, methods, and interfaces from other objects, so new objects can reuse existing objects. Objects also have the ability to provide their own implementations of methods they inherit, giving them the ability to use the same interface to accomplish different tasks.

We found the functionality provided by C++ to be a good match for our needs. Strategies can be represented by an abstract class that describes a set of general methods that are appropriate for any strategy. Each of the specific strategies can be represented by a class derived from the abstract strategy class. Each derived class can implement the general methods in a way that is appropriate for that specific strategy. New strategies can be added by deriving additional classes without modifying the abstract strategy class. These new derived classes can be integrated into the system without modifying the rest of the program. Constraints and resource models can be represented the same way. The more general strategies that handle interactions between activity classes can also benefit from the same representation. C++ and object-oriented techniques also offer solutions to our interface concerns. Different interactive interfaces for different users can share much of their implementation code by using a class library made up of classes which implement the scheduling algorithms.

IMPLEMENTING PARR IN C++

The Solar and Heliospheric Observatory (SOHO) Experimenters' Operations Facility (EOF) Core System (ECS) requires a scheduling system to find and resolve conflicts between the schedules for each of the satellite's eleven instruments. The scheduling system in the EOF needed the type of schedule merging capability provided by one of the prototype systems. This presented the opportunity to redesign and reimplement PARR using object-oriented methods.

When designing or implementing a program using object-oriented methods, the primary organizing construct is the class. Each class describes a type of object. By definition, an object is anything that can be thought about, a concept. In most object-oriented systems, the classes are abstractions of concrete, real-life objects, like spacecraft, stars, or people. Because PARR is a generic scheduling system that can schedule many kinds of things,

PARR's classes represent abstractions of scheduling objects, such as activities, strategies and constraints.

Resource Classes

PARR refers to any data it does not directly schedule as a resource. Resources include both data that PARR cannot change and data that changes as a result of the schedule PARR is creating. In the satellite scheduling domain, spacecraft daylight is an example of a resource PARR cannot change, and available power is an example of data that is changed by the activities PARR schedules.

PARR has different resource models it uses to track different types of resources. Each resource model is represented by a different resource class. PARR implements an abstract class, ResourceClass, from which each of the classes representing a type of resource is derived. ResourceClass has an interface that is appropriate for any type of resource model. Its interface has a method that accesses the resource's value at any given time. It also has a method that confirms that a constraint can be satisfied and one that updates the data to reflect the application of the constraint.

Subscribable is the resource class used to represent data that PARR cannot change. A class called SubscribableResourceClass is derived from ResourceClass to represent these resources. It implements the access method so that it returns the number "one" when the resource is available and the number "zero" when it is not. The method that confirms a constraint checks the value against the constraint. The update method simply modifies bookkeeping data because the data itself cannot be changed.

Limited capacity is one of the resource classes that can be used to model data that is modified as PARR creates a schedule. It is used to model resources that have a maximum amount that can be used at one time, like water supplied through a pipe from a public utility. It is implemented with a class called LimitedCapacityResourceClass. Each instance of this class contains data stating its maximum level of consumption. It updates the access method from its parent class, ResourceClass, to provide the amount of capacity that remains at any specific time. The constraint checking method verifies the constraint against this value and the update method modifies it to reflect the application of the constraint.

Consumable is the third resource class that PARR provides. It is used to model resources that can be consumed and replenished. The gasoline in a car is a good example of a consumable resource. The class that is derived from ResourceClass to implement this model is called ConsumableResourceClass. Instances of this class contain both a maximum capacity for the resource and a base level that is used when no other data is available to initialize the resource availability. The class implements the access method that gives the amount of resource available at any time. The constraints that are used with this resource class are different in that they can also specify the replenishment of the resource. The constraint validation method checks that the constraints that specify consumption do not consume more of the resource than is available. The update method modifies the data to reflect the amount of consumption or replenishment the constraint indicates.

Activity Class

PARR also needs to represent the types of activities that it is to schedule. They are defined in activity classes. Each activity class represents one type of activity that can be scheduled. Activity classes are represented by a class called ActivityClass. ActivityClass acts primarily as a container for data describing its class and for keeping track of when activities of its class have been scheduled. It has methods for input and output, methods that can tell when activities of its class have been scheduled, and methods to update this information when PARR adds or removes specific activities from the schedule. Instances of ActivityClass contain some simple data including the activity class name, priority, and how activities of this class can be shifted. An activity class can be derived from another activity class in the same way a C++ class can be derived from another C++ class. ActivityClass has a reference to the activity class it is derived from, if any, and it has data access methods that automatically obtain any

data that the ActivityClass inherits from its parent. Most importantly, ActivityClass contains a list of constraints and a set of strategies.

Constraints

Constraints represent PARR's conflict avoidance rules. A constraint can state how an activity must be scheduled in relationship to other activities or resources. For instance, a constraint might say that an activity can only be scheduled during spacecraft daylight, or only when no other activity is scheduled. If the resource represents data that the schedule indirectly modifies, the constraint will say how that data is modified. For example, if the resource is water from a pipe, the constraint will state how much water the activity consumes. PARR implements an abstract class, Constraint, to establish an interface for all of its constraints. Constraint establishes methods to validate the constraint for a given activity, and to update the resource data for an activity when the activity is actually scheduled. There is a different subclass of constraint for each resource class, and one that represents the constraints between different activity classes.

SubscribableConstraint is the class that represents constraints between activity classes and resources represented by a subscribable resource class. Each instance of SubscribableConstraint contains a reference to the SubscribableResourceClass the activity is constrained by and the desired state of the constraint. For example, if the constraint stated that an activity could only be scheduled during spacecraft daylight, the SubscribableConstraint would contain a reference to the SubscribableResourceClass representing spacecraft daylight and the value "one", indicating that the resource should be present. Its methods simply call the corresponding methods in the resource class to which it refers.

LimitedCapacityConstraint is the class that represents constraints between activity classes and resources represented by a limited capacity resource class. The instances of LimitedCapacityConstraint each contain a reference to the specific instance of LimitedCapacityResourceClass that represents the desired resource and the amount of the resource that the activity consumes. For example, if the constraint says that an activity consumes 25 gallons of water, the resource class reference would be set to the instance of LimitedCapacityResourceClass that represents water and the amount would be set to 25. As with the SubscribableConstraint, LimitedCapacityConstraint implements its methods by calling the methods in the resource class to which it refers.

Constraints between activity classes and resources represented by a consumable resource class are handled by the class ConsumableConstraint. Its instances contain a reference to the instance of ConsumableResourceClass that models the resource under consideration and the amount of the resource under consideration. Because activities can either supply or consume resources of this class, the instances also contain an indicator telling whether this is a constraint where the resource is being consumed, or whether the constraint is indicating that the activity supplies more of the resource being modeled. For example, if the constraint was being used for an activity that records data to the data tape recorder on a satellite and it requires 25 feet of tape, the resource class reference would be set to the instance representing tape, the amount would be set to 25, and the indicator would be set to "consume." If the constraint was being used for an activity that downlinks data from that same tape recorder and the activity reads back 200 feet of tape, the resource class reference would remain the same, the amount would be set to 200, and the indicator would be set to "supply." As in the other constraints, the methods in ConsumableConstraint are implemented by calling corresponding methods in the ConsumableResourceClass to which it refers.

All of the constraints which represent relationships between resources and activity classes follow a similar pattern. Because the class ActivityClass keeps track of information on when an activity has been scheduled in a way that parallels the way resource classes keep track of resource consumption, ActivityClassConstraint can be implemented the same way that the other classes derived from Constraint are implemented.

ActivityClass constraints represent constraints between two different activity classes. The first of the two activity classes is implied by the location of the constraint within an activity class. Each instance of ActivityClassConstraint contains a reference to the second ActivityClass and an indicator that represents whether the constraining activity class must be present or absent. In other words, if the constraint stated that an activity could only be scheduled when a certain other activity was not scheduled, the constraint would contain a reference to the ActivityClass representing the constraining activity and its indicator would be set to "avoid," indicating that the constraining activity must not be on the schedule during the same period. Its methods call the methods provided by ActivityClass indicating when an activity has been scheduled.

Strategies

Strategies represent PARR's conflict resolution rules. Strategies are used to place activities on the schedule, and to move activities when the constraint checking process discovers conflicts. For instance, a strategy might suggest scheduling an activity when a certain resource becomes available, or putting an activity after an activity it conflicts with. Although many strategies may be appropriate for rescheduling, only certain strategies make sense when initially adding an activity to a schedule. PARR implements an abstract class, Strategy, to provide an interface for classes that implement strategies. Strategy defines methods that take an activity and a list of conflicts and trys to reschedule the activity to avoid the conflicts.

Because only a limited number of strategies can be used to schedule an activity initially, and they require a different interface when used in this way, PARR provides a second abstract class, InitialStrategy, that is derived from Strategy. InitialStrategy provides methods in addition to those defined by Strategy that can be used when initially scheduling an activity. It defines a method that schedules an activity given no other input than the current schedule. This method returns a list of conflicts if the activity cannot be scheduled, and an indicator that tells if the activity should be scheduled again. InitialStrategy also implements the rescheduling method. It reschedules by calling the scheduling method, ignoring the list of conflicts passed into the method. The three different initial strategies that are provided are the start strategy, the stop strategy and the at strategy.

The start strategy tries to schedule an activity when a resource becomes available. For instance, if having a star in view of a satellite is a resource, a start strategy may indicate that an activity is to be scheduled when the resource becomes available, indicating the activity should be scheduled when the star is in view of the satellite. PARR implements a class, StartStrategy, that is used to represent a start strategy in an activity class. Each instance of StartStrategy contains a reference to the ResourceClass that corresponds to the appropriate class. It also has an offset duration that can be set to indicate that the activity should be scheduled the specified amount of time before or after the start of resource availability.

The start strategy is considered a repeatable strategy; the activity will be scheduled every time the resource becomes available. The instances contain a data item that gives the user the ability to indicate a count that makes the strategy skip a number of instances of resource availability before scheduling again. For instance, if the count was set to 3 in the previous example, the system would try to schedule the activity every third time the star came into view. StartStrategy implements the method for scheduling by searching the current schedule for activities of this type. When it finds the last activity of this type, it searches for when the key resource becomes available. If the count is set, it skips the appropriate number of times the resource becomes available. It then attempts to schedule the activity and checks its constraints. If there are conflicts, it returns a list of them. It also returns an indicator that the strategy should be retried so that the scheduling of this resource continues.

The end strategy is quite similar to the start strategy, except that is attempts to schedule at the end of a resource's availability. It is implemented via the class StopStrategy. StopStrategy's implementation parallels

StartStrategy's. Each instance of StopStrategy contains a reference to the ResourceClass corresponding to the key resource, an offset, and a skip count. Its implementation of the schedule strategy is the same as in StartStrategy, except that it looks for the end of a resource's availability.

The at strategy is the simplest of the initial scheduling strategies. It simply schedules an activity at a given time. The class AtStrategy is used to represent this strategy in an activity class. The instances of AtStrategy store the start time that the user specifies. It implements the scheduling method to put the activity at the given time and check for conflicts. The method returns a list of any conflicts it finds and an indicator that the strategy is not to be applied again.

The initial strategies can also be used as alternative strategies to reschedule activities that could not be initially added to the schedule because of conflicts. In addition, PARR also provides several strategies that can only be used as alternative strategies. These alternative strategies include: the before strategy, the after strategy, the next strategy, the prior strategy, the duration strategy, the bump strategy, the activity strategy, the delete strategy and the shift strategy.

The before and after strategies try to resolve a conflict by scheduling an activity either before or after the conflicts. The before strategy is represented by the class BeforeStrategy and the after strategy by the class AfterStrategy. BeforeStrategy implements the rescheduling method to search through the constraints to find the time before all of the conflicts and attempts to schedule the activity at that time. AfterStrategy implements the method using a similar algorithm to find a time after all the conflicts.

The next and prior strategies look for the next or previous time that a given resource becomes available and tries to reschedule the activity at that time. The next strategy is represented by the class NextStrategy and the prior strategy is represented by the class PriorStrategy. Instances of each of these classes contain a reference to the ResourceClass representing the key resource. NextStrategy implements the rescheduling method to start at the time the activity was to be scheduled, search for the next time the key resource becomes available, and attempt to schedule the activity at that time. PriorStrategy implements the method using the same algorithm but searching for the last time the key resource was available.

The duration strategy tries to make an activity fit into the schedule by reducing its duration. The duration strategy is represented by the class DurationStrategy. Each instance of DurationStrategy contains a duration by which the activity can be reduced. It implements the reschedule method by reducing the duration of the activity and adding the activity to the schedule.

The bump strategy is closely related to the duration strategy. Rather than reducing the duration of the activity, it moves its starting time. The bump strategy is implemented by the class BumpStrategy. The instances of BumpStrategy each contain an amount of time by which the start time of the activity can be moved. It implements the reschedule method by changing the start time of the activity by adding the offset and putting the activity into the schedule.

The activity strategy resolves a conflict by scheduling an activity of a different class. The activity strategy is represented by the class ActivityStrategy. Each instance of ActivityClass contains a reference to the ActivityClass that represents the alternative type of activity that should be scheduled. ActivityStrategy implements the reschedule method by calling the schedule method in the InitialStrategy in the alternative ActivityClass. If it encounters conflicts, it calls the reschedule method of each of the alternative strategies in turn until the conflict is resolved.

The delete strategy deletes the activities that are causing the conflict. Before deleting an activity, PARR checks that the activity causing the conflict is of a lower priority than the activity containing the delete strategy. The delete strategy is represented by the class DeleteStrategy. It implements the reschedule method by finding the activities that cause all of the conflicts. After checking that all of them

are of a lower priority than the activity being scheduled, it deletes them. Then it tries again to add the original activity to the schedule.

The shift strategy moves the activities that are causing the conflict. Before moving an activity, PARR checks that the activity causing the conflict is a shiftable activity and is of a lower priority than the activity containing the shift strategy. The shift strategy is represented by the class ShiftStrategy. It implements the reschedule method using an algorithm similar to the one used by DeleteStrategy. It starts by finding the activities that cause all of the conflicts. After checking that all of them are shiftable and of a lower priority than the activity being scheduled, it moves them either before or after where the original activity was to be scheduled. Then it tries again to add the original activity to the schedule.

Paradigms

The strategies described above each describe ways to schedule individual classes of activities. Although the strategies can delete or move activities that cause conflicts with an activity being scheduled or schedule an activity of another class if the desired activity creates conflicts, they cannot indicate which activity classes are to be scheduled or in what order the activity classes are to be scheduled. They also cannot get activities from alternative sources, such as schedules created outside of PARR. PARR handles these types of problems by using paradigms. Paradigms work at a higher level than regular strategies. PARR provides an abstract class called Paradigm that provides an interface for paradigms. The interface is similar to the Strategy interface; it defines a method that takes the action that the paradigm describes. It calls this method "schedule".

Paradigms are a recent addition to PARR, so only a few of them have been implemented. PARR currently provides the activity paradigm, the merge paradigm, and the delete paradigm.

The activity paradigm is used to select an activity class to be added to the schedule. In pervious versions of PARR, this was the only type of paradigm. This paradigm has been extended to accommodate the kind of tactical planning used in SM/PART. The activity paradigm lets the user override any of the data in the activity class, including the initial strategy, and add new constraints and alternative strategies to the activity class. This gives the user the ability to create slight variations on an activity without creating an entirely new class that may only be used once.

The activity paradigm is represented by the class ActivityParadigm. Instances of this class contain a reference to an instance of ActivityClass that represents the type of activity to be scheduled. The instance also contains most of the data contained in an instance of ActivityClass: the priority, the information on how the activity can be shifted, the constraints, and the strategies. This data is used only if it has been provided. It implements the schedule method by calling the schedule method in the InitialStrategy provided either locally or in the ActivityClass. If it encounters conflicts, it calls the reschedule method of each of the alternative strategies from the ActivityClass in turn until the conflict is resolved. If none of them resolve the conflict, it calls the reschedule method of each of the alternative strategies provided locally in turn until one of them resolves the conflict.

The merge paradigm is used to merge schedules created outside of PARR into a single conflict-free schedule. It expects that this externally created schedule is in a file in a simple, pre-defined format. The merge paradigm is represented by the class MergeParadigm. Each instance of this class stores the name of the file that contains the schedule to be merged. It implements the schedule strategy by first reading an activity from the schedule file. It then finds the ActivityClass that represents the activity and uses the start time and duration from the external schedule file to add the activity to the schedule. It uses the constraints from the ActivityClass to validate that the activity causes no conflicts. If it does cause conflicts, it calls the reschedule method of each of the alternative strategies from the ActivityClass in turn until the conflicts are resolved. It then repeats the process for each activity in the external schedule file.

The delete paradigm is used when the schedule is modified interactively. When the

user directs PARR to delete a specific activity from the schedule, an instance of the delete paradigm is used to represent this action. It may seem that the paradigm that was used to schedule the activity could be removed, but most of the paradigms can add many activities to the schedule. The delete paradigm is represented in the system by the class DeleteParadigm. Each instance of this class contains a reference to the ActivityClass that corresponds to the activity to be deleted and the start time and duration of the activity. This information is enough to uniquely identify a specific activity. DeleteParadigm implements the schedule method (although the name schedule is somewhat of a misnomer in this case) by checking that the removal of the activity will not cause any conflicts, and then deleting it from the schedule.

Files

The set of classes used to implement PARR is completed with classes used to organize the other classes into logical sets that can be saved to or read from files. These classes are derived from an abstract class called File. Having a single abstract class representing the main files simplifies writing the part of the user interface that handles saving and loading information from disk. The File class provides an interface defining methods to transfer the information its derived classes contain between memory and disk. The three classes derived from File are ResourceBase, KnowledgeBase and Plan.

A resource base serves as a repository for resource classes. PARR represents this concept with the class ResourceBase. Each instance or ResourceBase holds a list of instances of the class ResourceClass. It provides methods for retrieving the resource classes by name, adding new resource classes, and deleting existing ones.

A knowledge base is a set of information that can be used to reason about a given topic. In PARR, the knowledge bases are used to store the activity classes. PARR provides a class, KnowledgeBase, that contains instances of ActivityClass to represent the knowledge needed to schedule each type of activity. Instances of KnowledgeBase also contain references to one or more instances of ResourceBase. These resource bases provide the resource classes to which the constraints refer. KnowledgeBase implements methods for retrieving the activity classes by name, adding new activity classes and deleting existing ones. It also provides methods to add or remove resource bases and to access the resource classes contained in its resource bases.

The class that is used to combine all of the information in PARR in order to create the final schedule is called Plan. Plan represents a planning problem in its entirety. Each instance of plan contains the starting date and time for the schedule and its duration. It has references to one or more instances of KnowledgeBase that it uses to provide the knowledge needed to schedule activities. Most importantly, it has references to instances of the class Paradigm. These instances represent the tactical plan and the information needed to select the proper activity classes in the correct ways to create the schedule. It has methods to access and update the data it contains, including methods to access the resource classes and activity classes it can access indirectly. Most importantly, it has a schedule method that calls the schedule method of each of its paradigms in turn to create the final schedule.

PARR'S INTERFACES

As stated earlier, PARR has two primary types of interfaces: a file-based interface and an interactive interface. The current version of PARR uses different approaches to make each of these interfaces easier to modify.

The format for PARR's input files has been changed in the current version. The files that represent resource bases, knowledge bases, and plans now allow the free use of spaces, line breaks and comments between any words in the file. The file formats use keywords to help clarify the meaning of each data item. This helps both the user, who is given more information on what the data in the files represents, and the program, which can determine if any pieces of data are missing. Any new types of data that are added to the file formats will be optional, so older files will not need to be changed to accommodate the new format.

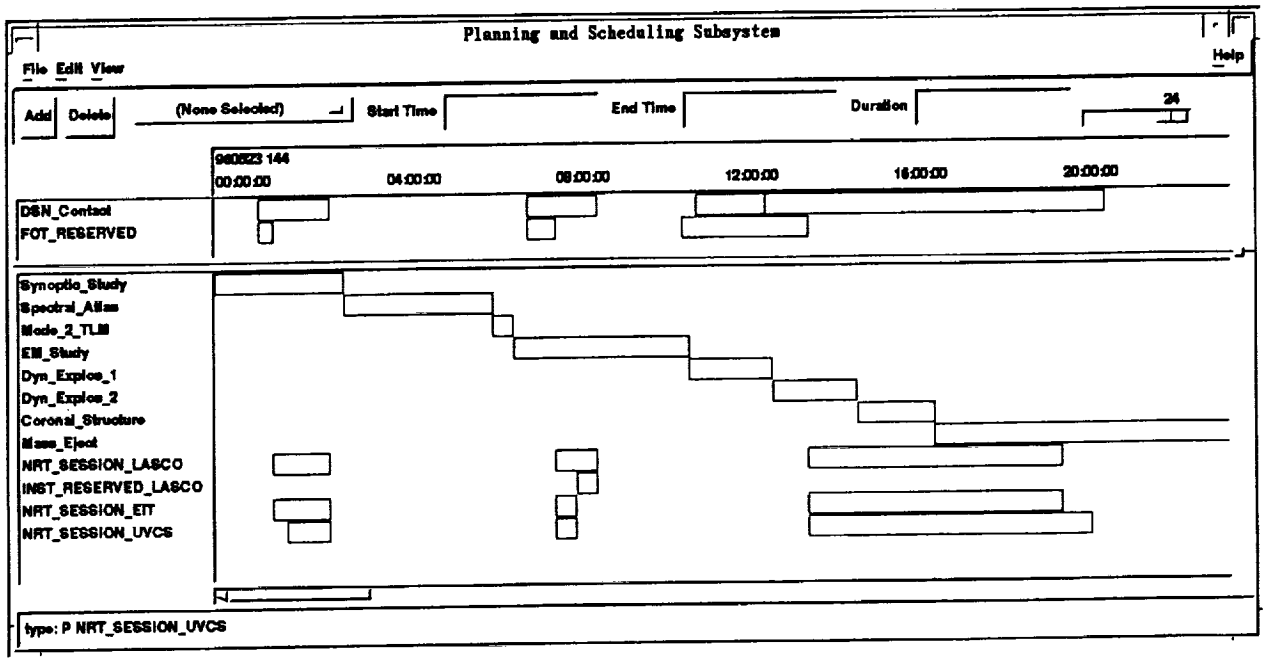


Figure 1. Motif-Style User Interface Display for the PARR Timeline Editor

Each class defined in PARR is responsible for handling its own file data. Every class has methods that can retrieve data from a file, create an instance based on that data, and store an instance to a file. The abstract classes are designed to retrieve data for any class that is derived from them. When derived classes are written, the abstract classes can handle the new data that may appear in a file without modifying their methods.

Because different scheduling systems require different types of interactive interfaces, the classes that implement PARR's scheduling algorithms do not have any interactive interface built into them. They can call outside functions when instances are created, modified or destroyed. They use a mechanism similar to the one used in the X Windows toolkit for accepting functions that are called when their instances change state.

We have developed an interactive user interface to C++ PARR for the SOHO EOF project. It uses the OSF/Motif libraries and a set of X Windows toolkit widgets that we developed in-house. Its primary window is the timeline editor shown in Figure 1. This display lets the user view a graphical timeline that represents the schedule and interactively change it by adding or deleting activities. It also provides a window for editing the activity classes that make up the knowledge base. This window is shown in Figure 2. It displays an activity class textually, but lets the user modify it using the mouse for most of the necessary input.

CONCLUSIONS

The object oriented nature of the new C++ version of PARR allows it to be relatively easily customized to build new planning and scheduling applications. For each new PARR application, the classes of generic objects for resource classes, constraints, and strategies can be supplemented with application-specific types.

In addition, C++ PARR applications can be more easily ported to different environments because the object oriented user interface code has been more completely separated from the algorithmic code.

Finally, the new C++ version of PARR provides paradigm constructs. One of the current paradigms PARR provides lets the user

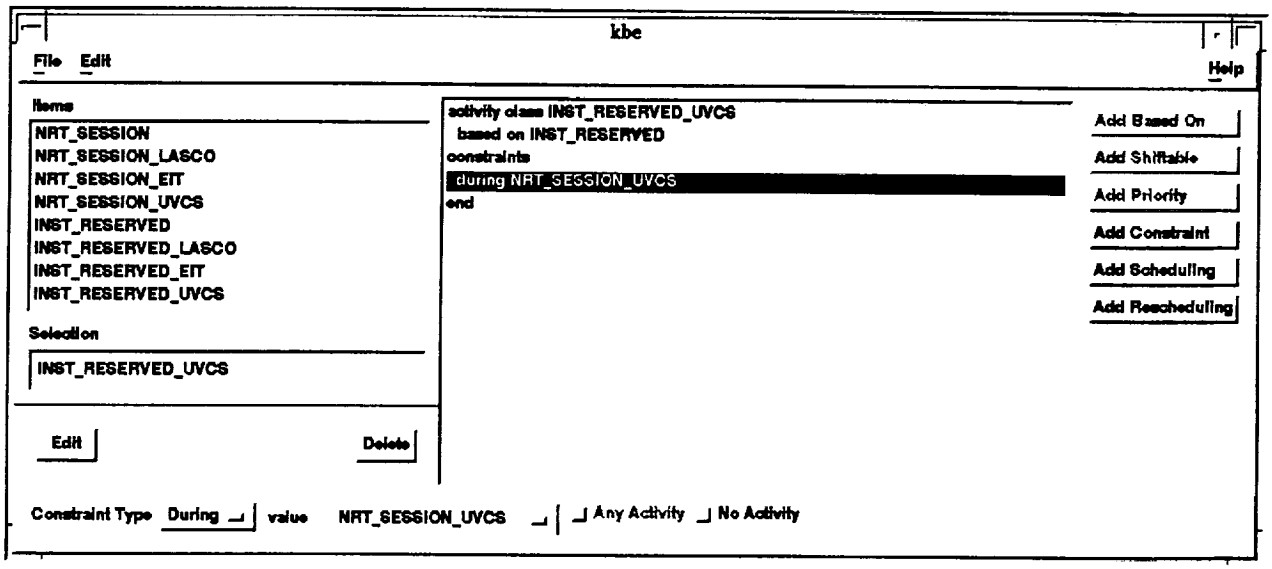


Figure 2. Motif-Style User Interface Display for the PARR Knowledge Base Editor

create variations on the activity classes when creating a tactical plan. Another paradigm provides for the input of schedules created by other systems. The paradigm constructs provide capabilities for extending PARR to handle additional kinds of scheduling problems more easily.

ACKNOWLEDGMENTS

The authors would like to thank Goddard Space Flight Center Code 514, AlliedSignal Technical Services Corporation, and the SOHO Project for their support of this task. This work was supported by NASA Contract NAS5-27772.

REFERENCES

1. McLean, D., Littlefield, R., and Beyer, D., "An Expert System for Scheduling Requests for Communications Links between TDRSS and ERBS," *Proceeding of the 1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics,* Goddard Space Flight Center, Greenbelt, MD, May 13-14, 1987.

2. McLean, D., Page, B., Tuchman, A., Kispert, A., Yen, W., and Potter, W., "Emphasizing Conflict Resolution versus Conflict Avoidance during Schedule Generation," *Expert System With Applications* , Vol. 5, Pergamon Press, 1992.

3. McLean, D. and Yen, W., "PST & PARR: Plan Specification Tools and a Planning and Resource Reasoning Shell for use in Satellite Mission Planning," *Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence,* Greenbelt, MD, 1989.

4. McLean, D., Page, B., and Potter, W., "The Explorer Platform Planning System: An Application of a Resource Reasoning Planning Shell," *Proceedings of the First International Symposium on Ground Data Systems for Spacecraft Control,* Darmstadt, Germany, 1990.

5. Bogovich, L., Johnson, J., Tuchman, A., McLean, D., Page, B., Kispert, A., Burkhardt, C., and Littlefield, R., "Using AI/Expert System Technology to Automate Planning and Replanning for the HST Servicing Missions," *Proceedings of the 1993 Goddard Conference on Space Applications of Artificial Intelligence,* pp. 3-10, Goddard Space Flight Center, Greenbelt, MD, May 10-13, 1993.

6. Yen, W., and McLean, D. "Combining Heuristics for Optimizing a Neural Net Solution to the Traveling Salesman Problem," *Proceedings of the First International Joint Conference on Neural Networks,* San Diego, CA, June, 1990.

N94-35055

Accurate Estimation of Object Location in an Image Sequence Using Helicopter Flight Data[1]

Yuan-Liang Tang and **Rangachar Kasturi**[2]

Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
Phone: (814) 863-4254
Email: *kasturi@cse.psu.edu*

Abstract

In autonomous navigation, it is essential to obtain a three-dimensional (3D) description of the static environment in which the vehicle is traveling. For rotorcrafts conducting low-altitude flight, this description is particularly useful for obstacle detection and avoidance. In this paper, we address the problem of 3D position estimation for static objects from a monocular sequence of images captured from a low-altitude flying helicopter. Since the environment is static, it is well known that the optical flow in the image will produce a radiating pattern from the focus of expansion. We propose a motion analysis system which utilizes the epipolar constraint to accurately estimate 3D positions of scene objects in a real world image sequence taken from a low-altitude flying helicopter. Results show that this approach gives good estimates of object positions near the rotorcraft's intended flightpath.

1 Introduction

To relieve the heavy workload imposed upon the pilots, there is a need for automatic obstacle detection systems onboard rotorcrafts. The success of the system depends upon the ability to accurately estimate object positions near the rotorcraft's flightpath. Several approaches for obstacle detection and range estimation have been investigated at NASA Ames Research Center [Bhanu89, Cheng91, Roberts91, Smith92, Sridhar89]. In this paper, we propose an approach for object position estimation using known camera location and motion parameters.

For a rotorcraft with inertial-guidance systems, the information about camera state is continuously available as the rotorcraft moves. This information can thus be used to facilitate the processes of motion estimation and scene reconstruction. For example, the location of the focus of expansion (FOE) in the image plane can be readily determined. In addition, we assume the image acquisition rate is high enough that an image feature will not move by more than a few pixels in the next image frame. Such closely sampled images will minimize the correspondence problem between successive images. The forward moving camera situation is the worst case in depth estimation because the optical flow in the image is small compared to other motion cases. We overcome this

[1]This research is supported by a grant from NASA Langley Research Center, Hampton, Virginia (NAG-1-1371).
[2]Address all correspondence to Professor Kasturi.

problem by integrating information over a long sequence of images. As image frames are accumulated, the base line between the current frame and the first frame increases, which gives better motion estimates. Baker and Bolles [Baker89, Bolles87] used the *Epipolar Plane Image (EPI) Analysis* for motion analysis. In their approach, camera moving path is known and linear. Therefore, each image frame can be decomposed into a set of epipolar lines. An epipolar plane image can thus be created by collecting corresponding epipolar lines in each image frame. Furthermore, when the viewing direction is orthogonal to the direction of motion, the apparent motion track of a feature on the EPI is a straight line and the motion analysis becomes merely a line fitting process. For forward linear camera motion, however, the feature tracks will be hyperbolas and curve fitting becomes necessary. Sawhney et. al. [Sawhney93] have reported that curve fitting is much more difficult and noisy, making this approach less robust. Matthies et. al. [Matthies89] built a framework which gives depth estimates for every pixel in the image. Kalman filtering is used to incrementally refine the estimates. In their experiments also, the side-viewing camera is assumed and the camera motion is only translational in the vertical direction. Under such conditions, feature tracks will follow the vertical image scan lines and feature matching becomes simpler. In our situation, the problem of general camera motion is dealt with. We handle this by breaking the camera motion path into piece-wise linear segments. Through this process, the camera path determined by two consecutive camera positions is approximated as a straight line. Epipolar planes can thus be set up for each pair of images and motion analysis is recursively performed on each pair of image frames.

Our algorithms were tested on the helicopter images provided by NASA Ames Research Center [Smith90]. There are two sequences of images, namely the *line* and the *arc* sequences. Each sequence consists of 90 image frames with size 512x512 pixels and each frame contains a header information which records the helicopter body and camera positions and orientations, body and camera motion parameters, camera parameters, etc. Time stamps are projected directly on the image frames. Fig. 1(a) and 1(b) show the first and the last frames of the *line* and the *arc* sequences, respectively. For the *line* sequence, the helicopter's flightpath is approximately a straight line and there are five trucks in the scene during the whole sequence. For the *arc* sequence, the helicopter is making turning flight and truck 1 is not visible in all frames. The trucks are labeled in terms of their range (X) value; truck 1 is the nearest and truck 5 is the farthest. Ground truths for the positions of the trucks are also given.

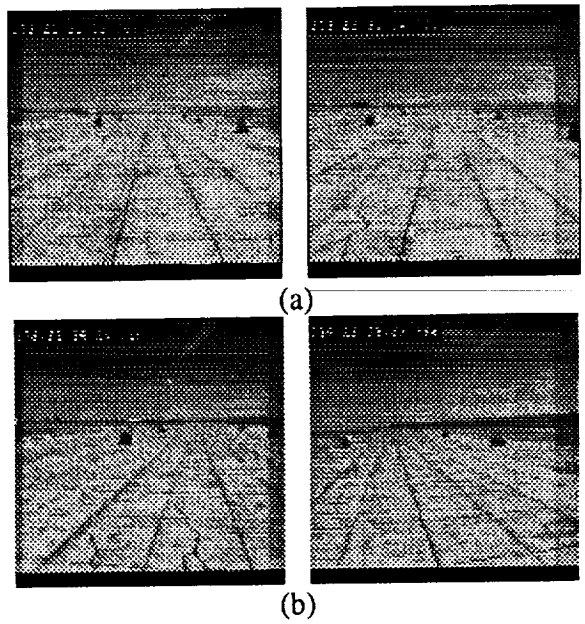


Fig. 1 The first (left) and the last (right) frames of the *line* sequence (a), and the *arc* sequence (b).

The remaining of this paper is organized as follows. In Section 2, we describe how to construct the epipolar lines. Section 3 discusses the feature extraction and tracking processes. In Section 4, we present the three-dimensional position estimation by tracking outputs. Experimental results and discussions are also given. Section 5 gives the conclusion.

2 Constructing the Epipolar Lines

The epipolar constraint gives a strong tool in confining the apparent motion directions of image features. In fact, the constrained directions are determined by epipolar planes as shown in Fig. 2(a), where all the image frames share a common set of epipolar planes. Since we are dealing with general 3D camera motion, the moving path of the camera is not a straight line and its orientation is not constant during motion. Fig. 2(b) illustrates such a case, where the camera's path is an arc. Even if the camera is fixed on the helicopter, its orientation is still changing because the orientation of the helicopter body is changing during nonlinear motion. The location of the FOE also changes significantly. In this case, there is no common set of epipolar planes for all the image frames. We solve this problem by using the piece-wise linear approximation for the camera path. Between two consecutive image frames, the camera path is approximated as linear and hence a pencil of epipolar planes can be created in the 3D space which all intersect at this segment of camera path. For each pair of image frames, we first compute the camera path parameters from the input camera state data (positions and orientations). We then define a pencil of Q epipolar planes, P_i, $i=0, .., Q-1$, which all intersect at the camera path. Q determines the resolution of the 3D space and hence the number of features to be detected in the image. In our experiments, Q is set to 100. The angle between two adjacent epipolar planes P_i and P_{i+1} is equal to π/Q. The result of this process is the construction of a pencil of epipolar planes equally spaced in terms of angular orientations and they all intersect at the camera path. After creating the epipolar planes, epipolar lines on each image plane can thus be determined by intersecting the image plane and the epipolar planes. The process of creating the epipolar lines is recursively performed on each pair of image frames in the sequence. Fig. 3 shows the epipolar lines superimposed on the edge detected images. The intersection of all the epipolar lines shows the FOE. Even for the *line* sequence in which the FOE is expected to remain at the same pixel location in the image, it changes by about 25 pixels both horizontally and vertically. For the *arc* sequence, the FOE location varies by about 70 pixels horizontally and 30 pixels vertically.

3 Feature Detection and Tracking

The purpose of constructing epipolar lines are two folds: to detect features and to facilitate feature tracking. Features in an image are defined to be the intersecting points between the epipolar lines and the edge pixels detected by the Canny's edge detector [Canny86]. The features are extracted from the first image by tracing the pencil of epipolar lines, l_i, $i=0, .., Q-1$. The result will be a number of Q sets of feature points and feature detection and tracking on the following frames are completely independent among these sets. To obtain good localization of detected features, the edges in the image should be nearly perpendicular to the epipolar lines.

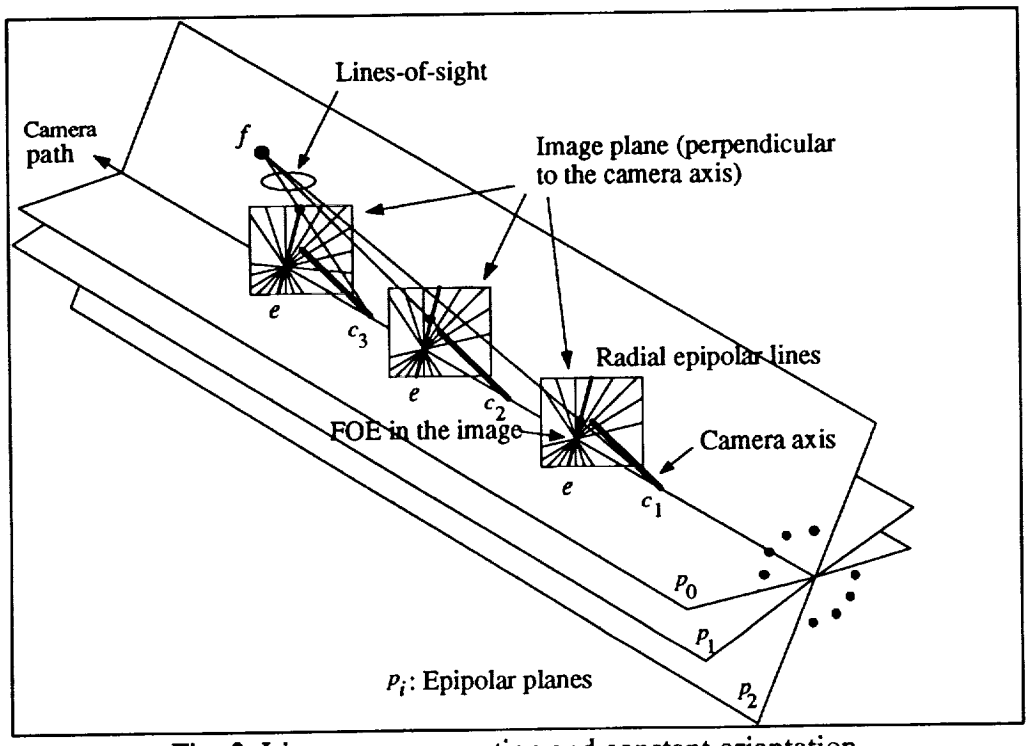


Fig. 2 Linear camera motion and constant orientation.

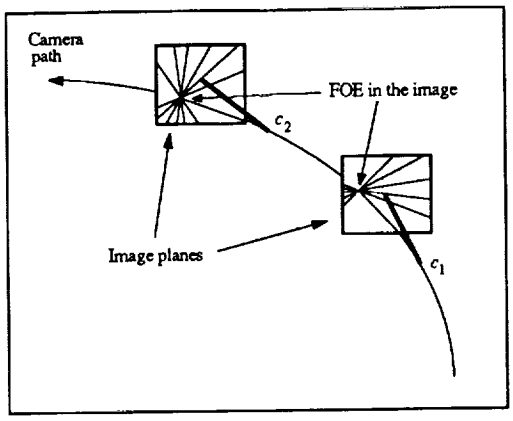


Fig. 3 Non-linear camera motion and varying orientation.

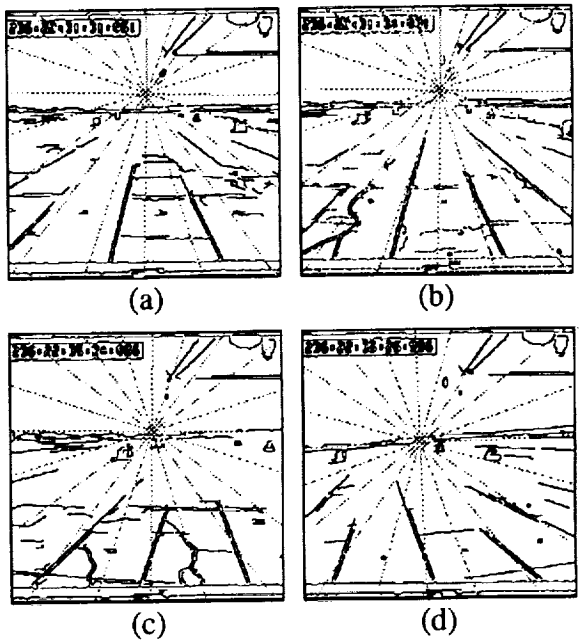


Fig. 3 The epipolar lines superimposed on the edge detected images (Every tenth line is shown). (a), (b): the first and the last frame of the *line* sequence, respectively. (c), (d): those of the *arc* sequence.

Since we are dealing with general camera motion, the camera orientations for two frames are likely to be different. Traditional feature matchers try to search within the *neighborhood*, i.e. the search window, of the feature to be matched (the source feature). With the information of camera state, we argue that this blindly positioning of the search window is inappropriate because the term *neighborhood* is incorrectly defined. The following statement is one of the implicit assumptions in defining the *neighborhood* to be a window centered around the source feature: "Since the image acquisition rate is high enough such that the camera will not move for a long distance between two consecutive frames, the source feature will not move more than a few pixels in the next image plane." We find this statement sustains only if the camera orientation keeps approximately constant during the camera motion. As is well known, even a small amount of camera rotation can actually create large image feature motion in the image plane. Hence, the camera rotation has much more influence on image feature motion compared to camera translation, especially when the distance from the camera to a world feature is very large compared to the distance the camera travels between two consecutive frames. This can be illustrated with the 2D world shown in Fig. 4, where the camera moves from c_1 to c_2 with orientation changes. The projections of a far away world feature onto the two frames are p_1 and p_2, respectively, and p_1' specifies the same image location as p_1 in the second frame. Due to camera rotation, p_2 is not, however, at the neighborhood of p_1'. Under such condition, the search window should be positioned around p_2 instead of p_1'. This is exactly our situation where the velocity of the camera is about 1 foot/frame and the major features of interest in the image are hundreds of feet away. Also, for the *arc* sequence, the instantaneous orientation of the camera is continuously changing since the flightpath is not linear, as described in Section 2. Experiments showed that if we perform tracking on the *arc* sequence by searching the neighborhood of the source feature, the correct corresponding feature may be completely out of the search window.

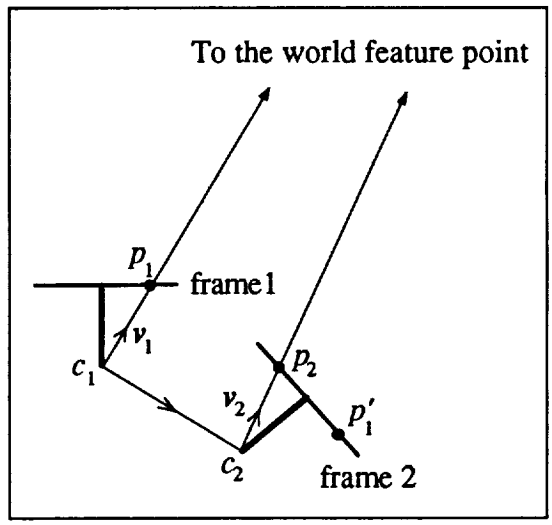


Fig. 4 Camera motion in a 2D world.

From the argument above, we redefine *neighborhood* as follows. We observe that the only thing guaranteed by high image acquisition rate is that the vector of the line-of-sight to a far away world feature will not change drastically between two consecutive frames, i.e. the vectors v_1 and v_2 in Fig. 4 should be approximately the same. Hence, the *neighborhood* of feature p_1 is redefined as "the region surrounding the intersection between the image plane of frame 2 and the vector through c_2 and parallel to v_1." And the feature tracker will search within this *neighborhood* for a match for feature p_1. This is the main reason of the success of our feature tracker. We implement the feature tracker as follows. For each pair of image frames, the locations of the FOE on each image plane are first computed. Let subscripts

1 and 2 denote the time instance of the first and the second frame, respectively. For each image feature at location p_1 in the first frame I_1, we first compute v_1 which is the 3D vector from the camera center c_1 to p_1. And then, the hypothetical location p_2 is obtained by intersecting I_2 with the vector v_1 passing through camera center c_2. Incorporating the epipolar constraint, instead of searching within a region centered around p_2, the feature tracker follows the direction of the epipolar line, which is determined by the FOE and p_2. In our experiment, we use seven pixels as the 1D window size. Feature detection and tracking are then performed within this window on the second image. Note that, in order to reduce the amount of computation we perform the feature detection and tracking based on the edge pixels only. For more robust feature tracking, the intensity distribution around a feature should be considered.

Within a small 1D *neighborhood*, we have a number of features (source features) in the first image and a number of features (target features) in the second image to be matched. Depending on the matching results, a feature will be labeled as **matched**, **new**, **no match**, and **multiple matches**:

1. **Matched**: There is only one target feature in the neighborhood. If there are several source features which are competing for the target (i.e. the occlusion case), the target will be matched to the source feature which has a stable position estimate. The position estimate of a feature is considered *stable* if its estimated position remains approximately the same through several frames (see Section 4). If more than one source feature have a stable position estimate, local slope of their tracks is compared. Fig. 5(a) shows such a situation. Source features A and B can both match to target C. The track with steeper slope (B) should correspond to the feature which is more distant from the camera than A. Since only the near feature can occlude the farther one, target C will be matched to feature A. For the matched source feature, its image (2D) position will be updated and its 3D position estimated (described in Section 4). Feature B will be labeled **no match** and handled as described later.
2. **New**: When a new feature appears in the current image, it has no source feature to match. A feature node will be created and inserted in the database.
3. **No match**: This may be due to the failure of feature detector, occlusion, or feature moving out of the image. For the last case, which can be easily detected, the source feature is simply removed. For the other two cases, if the source feature already has a stable 3D position estimate, the feature tracker will make a hypothesis about its image position based on its 3D position estimate. No estimation will be performed on these features except for updating their 2D position using the hypothesis. For other features, since we have no reliable information about their position, they remain in their current state awaiting possible matches in the future. A maximum number of consecutive no matches is defined to remove those features being occluded or missed by the detector for a long time.
4. **Multiple matches**: In this case, more than one target feature appears within the neighborhood. This may result from feature disocclusion or due to the noise from the feature detector. The goal here is to choose the best match. Cox [Cox93] reviewed some of the approaches including *nearest-neighbor* [Crowley88], *Mahalanobis* distance [Therrien89], *track-splitting filter* [Smith75], *joint-likelihood*

[Morefield77], etc. In our problem, since the epipolar constraint already gives an effective means to improve the tracking process, simple techniques are used to resolve this confusion and at the same time reduce the algorithm complexity. This idea is also supported by three observations. First, if the feature already has a stable 3D position estimate, the best match can be easily found by projecting its position estimate onto the current image. Second, according to the epipolar constraint, actually only *one* direction (away from FOE) is possible for the feature motion under noise-free circumstances. Hence, the feature motion conforming this constraint should be favored. And finally, the size of the search window is small, giving only a small number of multiple matches. Hence, the matching problem is simplified. With these observations, we use the following three priority criteria for choosing a feature to be the best match: (1) the one which satisfies the position estimate; (2) the one which is in the direction away from the FOE and is nearest; and (3) the one which is in the direction towards the FOE and is nearest. The main reason to include (3) as a legal match is to compensate the feature detection noise. These three criteria will also determine the weights in the position estimation (see Section 4). Fig. 5(b) gives a demonstration of how complicated the multiple match can be. Source feature *A* in frame 2 is searching for a best match among the target features in frame 3. Correct track is *AEFG*, but feature *E* (the blank circle) is missed by the detector. There exist also disocclusions (squares and triangles) and noise (star). Several scenarios and consequences are possible in our tracking process: (1) The tracker chooses feature *B* (the triangle) as the best match, feature *A* is already stable, and feature *F* is detected and within *B*'s search window in frame 4. Since *B* does not satisfy the estimated location of *A*, it will be lightly weighted in the position estimation and has little contribution to the estimate. However, according to the estimated position, the tracker will pick the correct match, feature *F*, as the best match in frame 4. (2) The tracker chooses feature *B* as the best match, feature *A* is stable, and feature *F* is not in *B*'s search window or is missed again. The tracker will follow the track *ABC*, which is wrong for feature *A*. The tracker, however, may still make correction on its tracking if it is possible to match feature *G* to feature *C* in frame 5. Otherwise it will follow the path of the triangles and the position estimates will gradually become unstable. Such features can be recognized by noting that their position estimates are still unstable after lengthy tracking. We then reset their estimates and start a new estimate similar to that for newly appeared features. (3) The tracker chooses *B* as the best match and feature *A* has no stable estimates. The tracking will either follow the circles or the triangles depending on which one is nearer. This does not matter too much since no reliable information has been accumulated and these errors will be lightly weighted in the position estimation. (4) The tracker chooses the noise (star) as the best match. This will be similar to what has been discussed, i.e. the tracker may correct its tracking if it is possible to pick feature *F* in frame 4. Otherwise, the estimate will be reset if track *BCD* is followed.

Feature matching is difficult and may contribute to most of the error in image analysis. From the analysis above, we can see that the epipolar constraint helps to simplify and to improve the matching quality. The matcher may be further improved by in-

corporating more clues such as the intensity distribution, edge strength, edge orientation, etc.

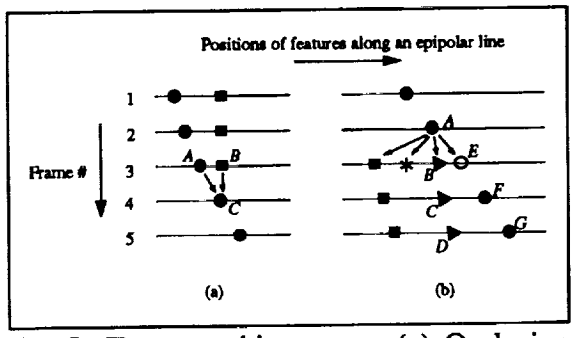


Fig. 5 Two matching cases: (a) Occlusion; (b) Disocclusion, missed features, and noise.

4 Position Estimation and Experimental Results

We use *incremental weighted least squares* for estimating the 3D position of the features being tracked. The main reason for that is its simplicity. Kalman filtering is another choice as described in [Matthies89]. However, we would like to demonstrate that using our feature tracker, a simple estimation method can still achieve accurate estimates. As we know, the line-of-sight determined by the camera center and the image feature position will also pass through the 3D position of the world feature. As time elapses, we can obtain for a certain feature a set of lines-of-sight which, theoretically, will intersect at its true 3D position. However, due to finite pixel size and inaccuracies in the known values of camera parameters and feature's image plane estimate, these do not exactly intersect at a point. This becomes a *point fitting* process. Let (a_k, b_k, c_k) be the normalized direction vector of the line-of-sight at time instance k and (X_{ck}, Y_{ck}, Z_{ck}) be the position of the camera center. The distance d_k between the 3D feature position (X, Y, Z) and the line-of-sight can be computed as:

$$d_k = \left\{[c_k Y_k - b_k Z_k]^2 + [a_k Z_k - c_k X_k]^2 + [b_k X_k - a_k Y_k]^2\right\}^{1/2},$$

where

$$(X_k, Y_k, Z_k) = (X - X_{ck}, Y - Y_{ck}, Z - Z_{ck}).$$

The objective function J can be defined as the sum of the weighted squares of the distances, i.e.,

$$J = \sum_{k=1}^{t} w_k d_k^2,$$

where w_k is the weight and t is the current time. J is to be minimized to obtain a least squares estimate. The weight will be determined by the result of matching. It is defined as:

$$w_k = \begin{cases} w_p, & \text{if sastifying the position estimation, else} \\ w_e, & \text{if satisfying the epipolar constraint, else} \\ w_n, & \text{if nearest,} \end{cases}$$

where $1 \geq w_p \geq w_e \geq w_n \geq 0$. The position estimate is considered stable at time t if

$$\frac{1}{r} \sum_{j=t-r}^{t-1} \left\{(\hat{X}_j - \hat{X}_t)^2 + (\hat{Y}_j - \hat{Y}_t)^2 + (\hat{Z}_j - \hat{Z}_t)^2\right\} < T,$$

where $(\hat{X}_j, \hat{Y}_j, \hat{Z}_j)$ is the position estimate at time j, T is a threshold and r is the number of frames considered.

For the *line* sequence, the tracking for the five trucks are all successful. Fig. 6 shows the position estimates and the relative estimate errors of truck 1 (right most) as a function of number of frames processed. The relative error is defined as the difference between the estimated value and the ground truth divided by the Euclidean distance between the camera center and the ground truth. We can see from the figures that the estimation converges after 10 frames and the relative errors of the estimates for the range, cross range, and height after convergence are within 5%, 1%, and 1%, respectively. As the frame acquisition rate is 30 frames/sec, the estimate converges in less than one second. The reason for this fast convergence is

that truck 1 is the nearest and the average number of pixels moved between two consecutive frames is about 0.7. The fast moving image features provide better optical flow information for motion estimation. The more accurate are the cross range (Y) and the height (Z) estimates since there is little lateral or vertical motion. While the least accurate is the range (X) estimate since the helicopter is moving forward. Due to this fact, in all our experiments the cross range and the height estimates are always more accurate than the range estimates and their relative errors are about 1%. Table I summarizes only the range estimates for all the five trucks in the *line* sequence. It is also shown that the number of frames needed for range estimate to converge within 10% error increases as the distance to the truck increases. Truck 5 needs more than 90 frames to obtain a more accurate estimate because it is farthest and the average number of pixels of image feature movement is only 0.1 pixel.

Experiments are also performed on the *arc* sequence and Table II summarizes the estimates. The tracking of truck 4 is not quite successful. The reasons being that during the initial several frames the detected edge is approximately parallel to one of the epipolar lines, making the tracking more difficult. However, its position estimate converges to within 25% errors after tracking through 90 frames. The estimate would converge to the correct value provided that more image frames are accumulated. Comparing Tables I and II, we find that the range estimates obtained from processing the *arc* sequence tend to be better than those from the *line* sequence. This verifies our previous statement that the forward moving camera case presents more difficult problem in range estimation. Since the helicopter is making a turning flight in the *arc* sequence, features move faster in the image than those in the *line* sequence, and hence provide larger optical flow for motion estimation.

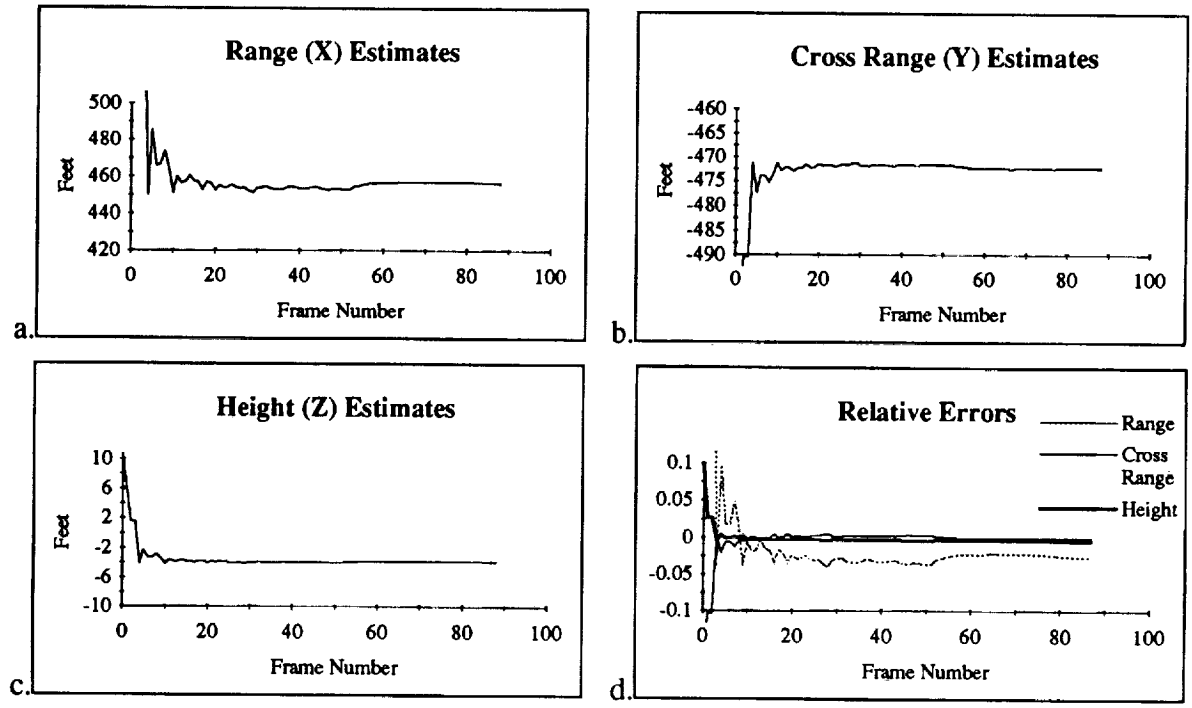


Fig. 6 Position estimates (a), (b), (c) of truck 1 and relative errors (d) as a function of the number of frames processed. The ground truth is (X, Y, Z)=(461.5, -472.3, -3.0) with accuracy ±2 feet.

Table I Position estimates of various trucks after processing 90 *line* frames.

	Initial and final range (unit: feet)	Absolute errors (unit: feet)	Relative errors (unit: %)	# frames for 10% range estimate error
Truck 1 (right most)	(272.7, 176.1)	(5.1, 0.1, 3.8)	(2.7, 0.03, 2.0)	5
Truck 2 (left most)	(365.5, 268.9)	(5.1, 0.6, 2.0)	(1.8, 0.2, 0.7)	20
Truck 3 (between center and right most)	(503.0, 406.4)	(12.2, 0.5, 1.2)	(2.8, 0.1, 0.3)	30
Truck 4 (between center and left most)	(615.9, 519.3)	(37.3, 6.0, 0.3)	(7.0, 1.1, 0.1)	70
Truck 5 (center)	(771.8, 675.2)	(83.7, 0.4, 0.5)	(12.4, 0.1, 0.1)	> 90

Table II Position estimates of various trucks after processing 90 *arc* frames. (Truck 1 is not visible in all frames).

	Initial and final range (unit: feet)	Absolute errors (unit: feet)	Relative errors (unit: %)	# frames for 10% range estimate error
Truck 2 (left most)	(224.9, 107.5)	(2.6, 1.3, 0.6)	(1.7, 0.8, 0.4)	20
Truck 3 (right most)	(384.6, 267.2)	(4.1, 1.9, 1.7)	(1.5, 0.7, 0.6)	20
Truck 4 (second from left)	(475.3, 357.9)	(90.0, 19.2, 0.6)	(24.5, 5.2, 0.2)	> 90
Truck 5 (second from right)	(631.2, 513.8)	(79.6, 3.7, 1.3)	(14.9, 0.7, 0.2)	> 90

5 Conclusion

In this paper, we proposed a motion estimation system which is capable of accurately estimating the 3D object positions in the scene. We used the piece-wise linear approximation for the flightpath to facilitate the construction of the epipolar planes and lines. This is very suitable for rotorcrafts having on-board inertial navigation systems, where the camera states are always available. The piece-wise linear approximation turns out to be a good method to solve the difficult motion problem resulting from general 3D camera motion. We also built an efficient feature tracker which makes use of the epipolar constraint to achieve good feature matching results. Weighted incremental least squares estimator then performs the position estimation. The final relative error of the range estimate for the object approximately 176 feet away (truck 1 in the *line* sequence) is less than 3%, which corresponds to only 5 feet absolute error. In addition, the range estimation for truck 1 takes less than 10 frames to converge within 10% error. Since the speed of the helicopter is 35 feet/sec in the *line* sequence, this gives the pilot about 8 seconds to avoid the obstacle. For the estimation on other trucks, the pilot actually has ample time to make decision about the flightpath. Currently, our algorithms run at the rate of two frames per second on a DECstation 5000/240 for tracking several hundreds of features on 512×512 image sequences (excluding edge detection). However, optimization for throughput was not a consideration in this work.

Reference

[Baker89] Baker, H.H. and R.C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface," *Int. J. Computer Vision*, Vol. 3, pp. 33–49, 1989.

[Bhanu89] Bhanu, B., B. Roberts, and J.C. Ming, "Inertial Navigation Sensor Integrated Motion Analysis," *Proc. DARPA Image Understanding Workshop*, pp. 747–763, 1989.

[Bolles87] Bolles, R.C, H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *Int. J. Computer Vision*, Vol. 1, pp. 7–55, 1987.

[Canny86] Canny, J., "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679–698, 1986.

[Cheng91] Cheng, V.H.L. and B. Sridhar, "Considerations for Automated Nap-of-the-Earth Rotorcraft Flight," *Journal of the American Helicopter Society*, Vol. 36, No. 2, pp. 61–69, 1991.

[Cox93] Cox, I.J., "A Review of Statistical Data Association Techniques for Motion Correspondence," *Int. J. Computer Vision*, Vol. 10, No. 1, pp. 53–66, 1993.

[Crowley88] Crowley, J.L., P. Stelmaszyk, and C. Discours, "Measuring Image Flow by Track Edge-lines," *Proc. Int. Conf. Computer Vision*, pp. 658–664, 1988.

[Matthies89] Matthies, L., T. Kanade, and R. Szeliski, "Kalman Filter-based Algorithms for Estimating Depth from Image Sequences," *Int. J. Computer Vision*, Vol. 3, No. 3, pp. 209–236, 1989.

[Morefield77] Morefield, C.L., "Application of 0-1 Integer Programming to Multitarget Tracking Problems," *IEEE Trans. Automatic Control*, AC-22(6), 1977.

[Roberts91] Roberts, B., B. Sridhar, and B. Bhanu, "Inertial Navigation Sensor Integrated Motion Analysis for Obstacle Detection," *Digital Avionics Systems Conference*, pp. 131–136, 1991.

[Sawhney93] Sawhney, H.S., J. Oleinsis, and A.R. Hanson, "Image Description and 3-D Reconstruction from Image Trajectories of Rotational Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 9, pp. 885–898, 1993.

[Smith75] Smith, P. and G. Buechler, "A Branching Algorithm for Discriminating and Tracking Multiple Objects," *IEEE Trans. Automatic Control*, AC-20: pp. 101–104, 1975.

[Smith90] Smith, P.N., "NASA Image Data Base User's Guide," NASA Ames Research Center, Moffett Field, CA., Version 1.0, 1990.

[Smith92] Smith, P.N., B. Sridhar, and B. Hussien, "Vision-Based Range Estimation Using Helicopter Flight Data," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 202–208, 1992.

[Sridhar89] Sridhar, B., V.H.L. Cheng, and A.V. Phatak, "Kalman Filter Based Range Estimation for Autonomous Navigation Using Imaging Sensors," *Proc. 11th IFAC Symposium on Automatic Control in Aerospace*, 1989.

[Therrien89] Therrien, C.W., *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*, Wiley, New York, 1989.

Image Processing and Data Classification

Automated Analysis of Complex Data

Robert St. Amant and Paul R. Cohen
Computer Science Department
University of Massachusetts
Amherst, MA 01003
stamant@cs.umass.edu, cohen@cs.umass.edu

1. Introduction

Igor is a knowledge-based system designed for exploratory statistical analysis of complex systems and environments. Igor has two related goals: to help automate the search for interesting patterns in data sets, and to help develop models that capture significant relationships in the data. Igor supports and complements the efforts of a human analyst in examining large or complex data sets [11].

Igor explores a data set with the statistical operations of exploratory data analysis (EDA). Examples of EDA operations are univariate power transforms, bivariate relationship partitioning to distinguish separate effects, and analysis of patterns in residuals. Through these operations Igor incrementally constructs partial descriptions or models of data. When interesting patterns are observed in a model or its residuals in the data, Igor opportunistically selects appropriate operations to explore the new phenomena. As the analysis proceeds, a more complete picture of the data set gradually emerges.

We must consider many different tradeoffs in designing a system for statistical reasoning: operator generality versus power, language expressiveness versus efficiency, opportunism versus control. Of these the key tradeoff is between opportunism and control. EDA operations are powerful: they can be applied in many situations in different combinations, producing results whose proper interpretation depends on context. Effective analysis depends on taking advantage of new results to guide search in appropriate directions [7].

In terms of representation, we must decide how primitive statistical operators should be combined into higher level ones, and how these operators should interact with each other. If EDA operations are implemented as procedures that call other statistical procedures, as in a programming language, we have strict control over the results we produce, but little flexibility in responding to unexpected findings and applying appropriate context-specific techniques. If EDA operations are implemented as rules that fire whenever "interesting" intermediate results appear, on the other hand, we have the necessary opportunism, but find it difficult to capture strategic aspects of analysis.

We can approach this problem with the techniques of opportunistic planning. We have developed in Igor a planning language that balances control and opportunism. By explicit representation of the goals of exploratory analysis, we can take advantage of strategic knowledge about data analysis to structure and reduce search. Monitoring structures opportunistically detect intermediate and end results with interesting characteristics. The

This research is supported by the Advanced Research Projects Agency and Rome Laboratory under contracts F30602-91-C-0076 and F30602-93-C-0100.

plan and goal representation lets Igor select appropriate context-dependent sequences of operations during the analysis.

Igor is intended for robust assistance in domains with an enormous search space of hypotheses. Conventional packages offer statistical tests and an environment in which the user may either program scripts or run through an analysis by hand. Igor allows more flexible interaction. An analyst may apply a variety of heterogeneous strategies to produce and confirm results. Some processes may be scripted to proceed without interaction, while others may be defined to incorporate human guidance.

Our work has dealt mainly with analyzing the behavior of artificial intelligence programs. Because an AI system may react in complex ways to the influences of its environment, the reasons for its behavior may not be obvious from execution traces. This is a challenging domain for Igor. We have also looked into automating the analysis of Landsat wildlife habitat data, but we have gone no farther than designing a few basic procedures for data preparation [8].

In the next section we discuss elementary strategies for exploring data. In Section 3 we outline the planning representation, the relevance of planning to statistical analysis and how the representation supports the process. Section 4 describes an example of Igor in practice. Section 5 concludes with a discussion of the benefits of the general approach and plans for future work.

2. Strategies for Exploring Data

Exploratory studies are the informal prelude to experiments, in which questions and procedures are refined. Exploration is like working in a test kitchen: before one writes down the final version of a recipe, one first tries out possible alternative procedures and evaluates the results. Exploratory results influence confirmatory studies in a cycle of successively more refined exploration and confirmation [2, 5, 6, 12].

We apply two general strategies in exploring data: one generates simplifying descriptions of data, the other extends and refines surface descriptions of data. We simplify data by constructing partial descriptions and models that capture particular characteristics of the data. The descriptions range from simple summary statistics, such as means and medians, to complex domain models. We make descriptions more effective by looking beyond surface descriptions at what is left unexplained. For example, a regression line may be a good description of the general trend in a relationship, but an analysis of residuals can give an entirely different picture at a lower level of detail. Exploratory strategies generate increasingly detailed, complementary descriptions of data.

EDA strategies often takes advantage of intermediate results that suggest further areas of exploration. We can best illustrate the process with a brief example, adapted from Tukey [12]. Consider Figure 1, which plots the population of the U.S. between 1800 and 1950.

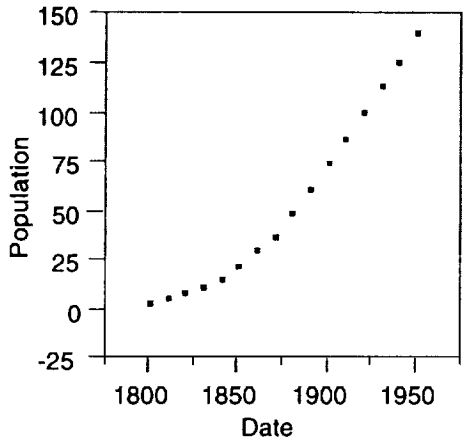


Figure 1: Population vs Date

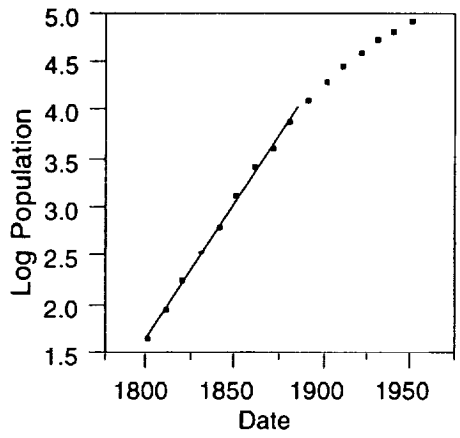


Figure 2: Log Population vs Date

We first notice that population in earlier decades increases at a slower rate than in later years. We can fit a small-degree polynomial such as a quadratic to the data, or, equivalently, transform each point by taking its square root. This kind of transformation serves to straighten the data points so that further structure may be observed. What we find, however, is that the points are not collinear after the transformation. A pattern in the residuals tells us that there is further structure to be elucidated. We thus return to the original data scale to try a different approach.

While the earlier data points seem to curve upward, the later points, from about 1900, seem to have constant slope. We thus partition the relationship at this point, and analyze each partition separately. It turns out that higher order curves fit the later partition little better than a straight line. From the residuals of this fit we may be able to extract further information--for example, that population took a dip in the decade of 1940. The linear fit and residuals, which we interpret as descriptions of trend and local detail, together form a satisfactory model of the later partition. For the earlier partition we first apply a log transform to straighten the data points, as in Figure 2, and then proceed as with the later partition.

Taken together, we now have a reasonably detailed model of population during these years. It shows that population increased exponentially between 1800 and 1900, but then slowed to a linear rate. Local effects can be seen in the residuals of the trend descriptions. Our construction of this composite model was guided by observation of an inflection point in the data, and the realization that the first, simpler model was not sufficient to capture significant characteristics of the data.

Given a basic set of statistical tools, humans can easily follow similar reasoning. The task is considerably more difficult for a computer. Our description has neglected much of the complexity of the decisions involved. Many intermediate results arise, but only a few of them turn out to be interesting. In general there are always choices to be made about which techniques best suit the situation and which results are most interesting to pursue.

When we explore data effectively we exploit a tension between opportunism and control. Opportunism lets us explore new interesting results that arise unpredictably. Control determines which results are promising, how they might best be evaluated, and when particular paths of reasoning might be abandoned. Without a proper balance between opportunism and control, an automated system may be unable to make the simplest discoveries, or may face an explosive search space.

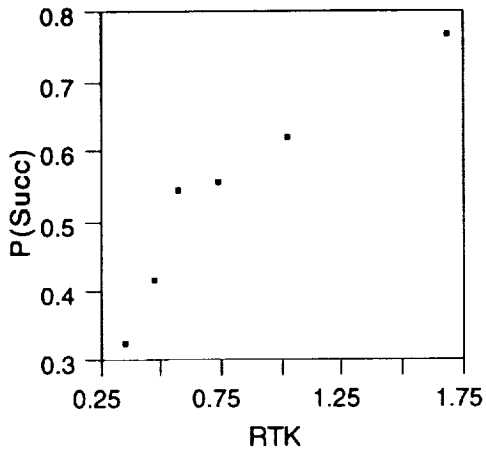


Figure 5: RTK-pS relationship

The result of the discrete-vs-mean-transformation is a transformed relationship. A monitor at the explore-relationships level detects the creation of the RTK-pS relationship and adds it to the context structure for the explore-relationships goal. It can then be explored in turn.

One of the exploration goals generated for RTK-pS is the sequence-fit goal, which is matched by the linear-fit script. The linear-fit script fits a line to the relationship, generating both a simple regression equation and a sequence of residuals. The residuals sequence is detected by a monitor and examined for patterns. The simplest pattern-detecting script finds that the residuals of RTK-pS are bitonic, increasing then decreasing. A monitor makes this available in the context of the exploration goal of RTK-pS.

Now the activation of scripts to match the sequence-fit goal is controlled by a focusing heuristic. If the linear-fit script produces a good enough fit (i.e., with enough variance accounted for,) and there is no structure in the residuals, then the goal is satisfied. Here, however, because of the residuals, the focusing heuristic must continue the search. There are two further possibilities available to the focusing heuristic. The first is to produce a transform for RTK-pS and retry the linear-fit script. The second is to partition the relationship at or near the inflection point in the residuals, and generate a fit for each partition separately during exploration.

Both of these approaches are plausible. Because we know that the mean of Success can never rise above 1.0, we could fit a sublinear function to the relationship, with asymptote at 1.0, as shown in Figure 6. This gives a better fit than a straight line, but still not a perfect one. Alternatively we recall that the initial setting of RTK is 1.0, the setting at which the system was designed and tested. We observe that the discontinuity in RTK-pS occurs near this value. Thus we may be seeing not a continuous relationship between the variables but two separate modes of operation, one at low thinking speed and the other at high thinking speed. Within each mode the RTK-pS relationship is linear. This interpretation is given in Figure 7.

Neither of these competing interpretations clearly rules out the other. If we believe that the design of Phoenix provides smooth degradation in performance as thinking speed decreases, we might prefer the first interpretation. The second interpretation is also plausible, except that degradation is not smooth after some threshold value of RTK is

reached. The additional evidence that this threshold occurs near the RTK setting of 1.0, possibly indicating that Phoenix plans rely implicitly on this setting, inclines us to credit the second interpretation.

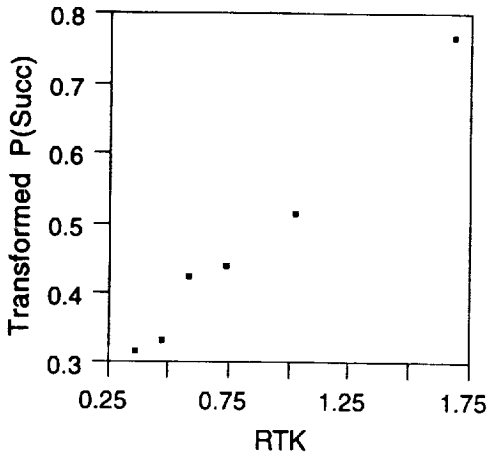


Figure 6: Straightened RTK-pS

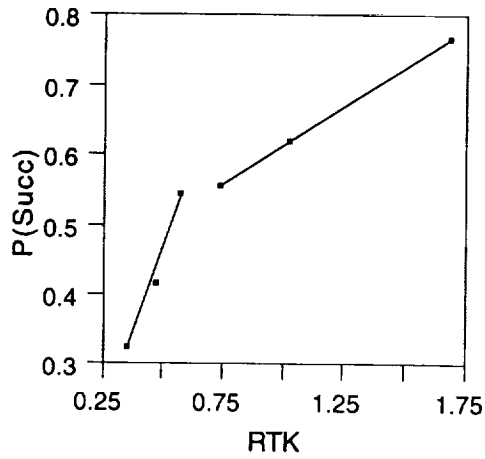


Figure 7: Partitioned model of RTK-pS

The focusing heuristic can clearly take advantage of such domain-dependent knowledge. In this case, however, we have taken a simpler approach. Applying a transformation to RTK-pS is relatively inexpensive, in terms of increase in the size of the search space. Partitioning the relationship, on the other hand, is a great deal more expensive. In general when partitioning Igor must consider the partitions separately, and in some cases recombine the individual results. Thus the focusing heuristic runs the sequence-fit scripts in sequence, rather than in parallel, and will not proceed to the later ones if the initial inexpensive ones succeed.

Finally we present an overview of Igor's processing. Figure 8 traces significant points in plan and goal instantiation during the exploration process.

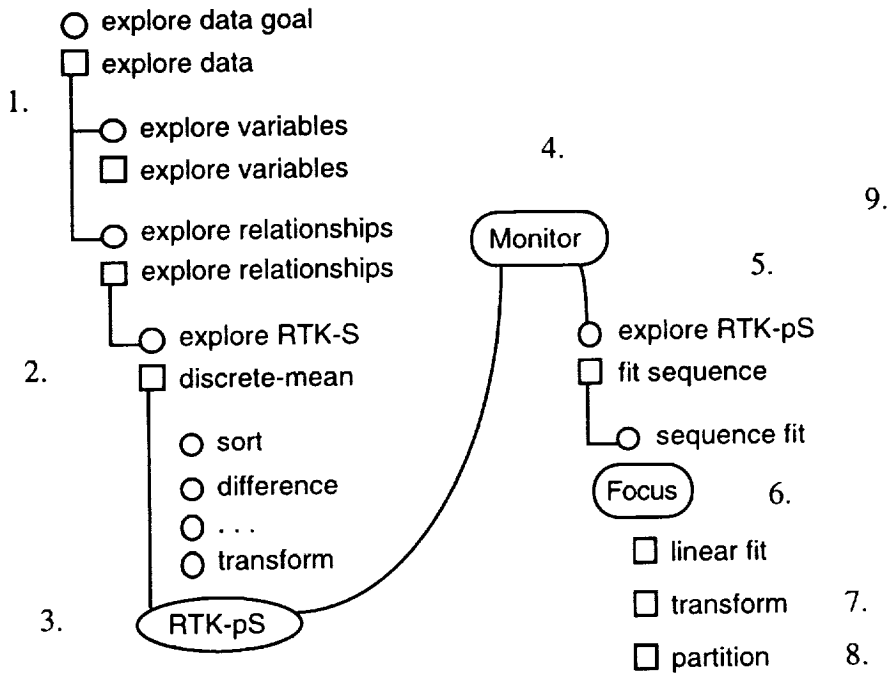


Figure 8: Exploration trace

1. Exploration goals are established for variables and relationships.
2. Relationship RTK-S is explored, triggering the discrete-vs-mean-transformation.
3. Relationship RTK-pS is generated.
4. A monitor detects RTK-pS and establishes an exploration goal for the relationship.
5. A sequence-fit goal is established for RTK-pS.
6. A focusing heuristic determines that a linear fit is appropriate.
7. Residual patterns cause a nonlinear fit to be tried.
8. Inadequate nonlinear fit causes a partitioning of the relationship.
9. Further exploration on the partitions occurs.

5. Conclusions

We have examined some of the issues involved in automating exploratory data analysis, in particular the tradeoff between control and opportunism. We have proposed an opportunistic planning solution for this tradeoff, and have implemented a prototype, Igor, to test the approach. Our experience in developing Igor has been surprisingly smooth. In contrast to earlier versions that relied on rule representation, it has been straightforward to increment Igor's knowledge base without causing the search space to explode. The planning representation appears to be both general and powerful, with high level strategic knowledge provided by goals and plans, and hooks for domain-specific knowledge provided by monitors and focusing heuristics.

Our future plans include incorporating detailed domain-specific knowledge into Igor, which will let us explore the interaction between general strategic knowledge and domain-specific control knowledge [8, 9, 10]. A domain we find especially interesting is experimental results describing the behavior of AI systems such as planners and schedulers.

We also wish to examine the relationship between exploratory data analysis and automated causal modeling. Some of the results produced by EDA appear to be useful cues to humans in developing causal models. We may be able to exploit such results in an automated system as well. Further, a causal model may be able to give search guidance to Igor, for example in distinguishing between proximal and distal causes of an observed effect [3, 4].

Our final concern is with evaluation. How sensitive is Igor to its parameter settings? How much information does Igor require to draw conclusions about a particular effect? Are the chains of statistical reasoning Igor produces coherent to humans? We will explore these and other questions in future work.

Acknowledgements

This research is supported by the Advanced Research Projects Agency and Rome Laboratory under contracts F30602-91-C-0076 and F30602-93-C-0100. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

[1] Carver, N., & Lesser, V., 1993. A Planner for the Control of Problem-Solving Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, vol 23, no. 6.

[2] Cohen, P.R., 1993. *Empirical Methods for Artificial Intelligence*. MIT Press. Forthcoming.

[3] Cohen, P.R., Carlson, A., Ballesteros, L., St. Amant, R., 1993. Automating Path Analysis for Building Causal Models from Data. *Proceedings of the Ninth International Conference on Machine Learning*. Morgan Kaufmann.

[4] Cohen, P.R & Hart, D.M., 1993. Path analysis models of an autonomous agent in a complex environment. *Proceedings of The Fourth International Workshop on AI and Statistics*. Ft. Lauderdale, FL.

[5] Gale, W.A., 1986. REX Review. In Gale, W.A. (Ed.), *Artificial Intelligence and Statistics*. Addison-Wesley.

[6] Hand, D.J., 1986. Patterns in Statistical Strategy. In Gale, W.A. (Ed.), *Artificial Intelligence and Statistics*. Addison-Wesley.

[7] Kulkarni, D., & Simon, H.A., 1990. Experimentation in Machine Discovery. In Schrager, J., & Langley, P., (Eds.), *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann.

[8] Lansky, A.L, & Philpot, A.G., 1993. AI-Based Planning for Data Analysis Tasks. CAIA, IEEE Conference on AI Applications, Orlando, Florida.

[9] Langley, P., Bradshaw, G., & Simon, H.A., 1983. Rediscovering Chemistry with the BACON System. In R.S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning An Artificial Intelligence Approach*. Morgan Kaufmann.

[10] Schrager, J., & Langley, P., 1990. Computational Approaches to Scientific Discovery. In Schrager, J., & Langley, P., (Eds.), *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann.

[11] Silvey, P., 1993. IGOR: The MAD Scientist's Assistant or Building Models from Data. COINS Technical Report 93-??. University of Massachusetts, Amherst.

[12] Tukey, J.W., 1977. *Exploratory Data Analysis*. Addison-Wesley.

An Adaptive Technique To Maximize Lossless Image Data Compression of Satellite Images.

Robert J. Stewart [1]; Y. M. Fleming Lure [1]; C. S. Joe Liou

Abstract

Data compression will play an increasingly important role in the storage and transmission of image data within the NASA science programs as the Earth Observing System comes into operation. It is important that the science data be preserved at the fidelity the instrument and satellite communication systems were designed to produce. Lossless compression must therefore be applied, at least, to archive the processed instrument data. In this paper we present an analysis of the performance of lossless compression techniques and develop an adaptive approach which applied image remapping, feature-based image segmentation to determine regions of similar entropy, and high-order arithmetic coding to obtain significant improvements over the use of conventional compression techniques alone. Image remapping is used to transform the original image into a lower entropy state. Several techniques were tested on satellite images including differential pulse code modulation, bi-linear interpolation, and block-based linear predictive coding. The results of these experiments are discussed and trade-offs between computation requirements and entropy reductions are used to identify the optimum approach for a variety of satellite images. Further entropy reduction can be achieved by segmenting the image based on local entropy properties then applying a coding technique which maximizes compression for the region. Experimental results are presented showing the effect of different coding techniques for regions of different entropy. A rule-base is developed through which the technique giving the best compression is selected. The paper concludes that maximum compression can be achieved cost effectively and at acceptable performance rates with a combination of techniques which are selected based on image contextual information.

[1]The author is with Caelum Research Corp., Silver Spring, Md 20901, Phone: (301) 593-1748.

1 Introduction

With a steadily growing use of imaging technology in almost all computerized scientific fields the need for better image compression for purposes of minimizing transmission time and storage space/cost is ever present. Lossless image compression which permits faithful reconstruction of the original image is important in maintaining accurate image archives of digitized documents, in remote sensing image storage and retrieval, and in medical imaging where loss of fidelity due lossy compression can compromise radiological diagnosis. The techniques for lossless compression basically attempt to re-code image data such that redundant elements are coded with the least number of bits possible by using the frequency of occurrence of elements to determine the number of bits to use to code the image [1] [2]. The amount of compression obtained is related to the degree of redundancy present in the image. To obtain higher compression over the basic approach, preprocessing techniques which attempt to decorrelate image pixel values [3] and source modeling techniques which attempt to use the context of local pixel values [4] have been tried. One- and two-dimensional discrete pulse code modulation (DPCM), [5] bi-linear interpolation [6], and hierarchical interpolation [7] techniques are typical of the decorrelation approaches that have been applied to improve image compression. Prior to coding an image, a statistical model of the image is needed which can be developed in a separate pass over the image or adaptively as the image pixels are being coded [8]. Zero-order models consider each pixel to be independent of its neighbors. Higher-order models collect statistics on sequences of adjacent pixels [9]. Higher-order statistical models provide better compression where images are smooth and regular and zero order models are more effective where the images have a large high frequency content [10]. Investigators have shown that combinations of these techniques when applied to particular classes of images provide improved compression, but no one combination is suitable for all images, particularly satellite images where the image texture can change dramatically over small spacial distances, from smooth desert regions to rough snow-capped mountains. Based on work done on lossless image compression for medical images by the authors [11] and others [12], it is believed that satellite imagery would best be compressed with multiple compression techniques which are adaptively selected based on properties of local image regions. Image regions which contain a large degree of prominent fine texture will not compress well because little data redundancy and inter-pixel correlation will exist. For such regions, the simplest least computationally expensive compression technique should be applied. For regions with smooth textures or regular patterns, image decorrelation and high-order modeling will produce the highest compression. For regions with characteristics between these two extremes, the most compression will be achieved with some combination of decorrelation and statistical modeling with the order of the statistics selected based on some measure of inter-pixel pattern repetition. The problem then becomes the identification of image features which can be used to select the decorrelation, modeling, and encoding techniques to give maximum compression at minimum "cost": Cost being a measure of the computational requirements for a given approach against the degree of improvement in compression achieved by using it. If such a feature set can be found, then the image to be compressed could be divided into arbitrarily small regions, the features calculated for each region, the regions classified according to the best compression techniques to apply, and then similarly classified regions compressed appropriately (Figure 1). A pipelined process of image analysis, feature ex-

traction, classification, decorrelation followed by modeling and coding is shown in Figure 1. Parallelization is possible due to the region processing approach used. Feature extraction and region classification are performed in parallel on each image region. The classifier determines the correct compression approaches to used based on a decision tree approach using the calculated features. Image regions that will use a decorrelation preprocessor will be passed through that path, other regions will be processed by the modeler and coder appropriate to the type of region. The modeler and coder are serialized processes in this architecture because adaptive modeling will be used, that is, the statistical model is built as the image pixels are being compressed. All image regions that are similarly classified will be processed through the same modeler/coder path so that a unique statistical model is generated for each type of region. The modeler and coder will process all pixels in a region then proceed to the next region, and so on. This procedure preserves any two-dimensional correlation of pixel values and should maximize compression. To be able to decompress the image file, it is necessary that the compressed file output by this architecture contains a header which records a classifier identifier for each region. The image can be reconstructed by reading the region classifier identifier and decompressing the region through the reverse of the compression process.

The application of a decorrelation step prior to modeling and coding was also considered after proving the suitability of this approach without it. Decorrelating the image reduces pixel value variance [10] and therefore improves compression. Two decorrelation methods were used, DPCM and bi-linear interpolation. Both are relatively simple to implement with DPCM using one or more previous pixel values to predict the current pixel value, and the lattice points in a 2-dimensional kernel of pixels being used to predict the other kernel pixel values. In the case of bi-linear interpolation, the output is two data streams, a set of prediction errors, and the value of the lattice points for each non-overlapping 2-dimensional kernel in the image. The computational cost of the bi-linear interpolation process is considerably more than the DPCM approach, but the improvement in compression can be considerable for certain types of images.

2 Image Analysis and Feature Extraction

The key to the success of this adaptive scheme for compression lies in the selection of features which can usefully classify image regions into the optimum compression approach and thereby minimize the compressed file size. Their exist no theoretical foundation for determining the feature set that would provide optimum compression selection. An empirical approach was take to determine the best feature set: Satellite images were selected for compression which contained a large variety of textures. The images were divided into symmetrical regions and each region was compressed using several decorrelation and lossless compression techniques. Several features were calculated for each region. Using cluster analysis techniques to identify unique reions in feature space, a binary decision tree classifier was developed using features from a large variety regions. The features selected for the initial training of the classifier were determined by evaluation of the compression process and from previous work on medical image compression [11], [13]. There are several well known techniques for estimating the amount and degree of texture in an image, including co-occurrence matrices [14], sum and

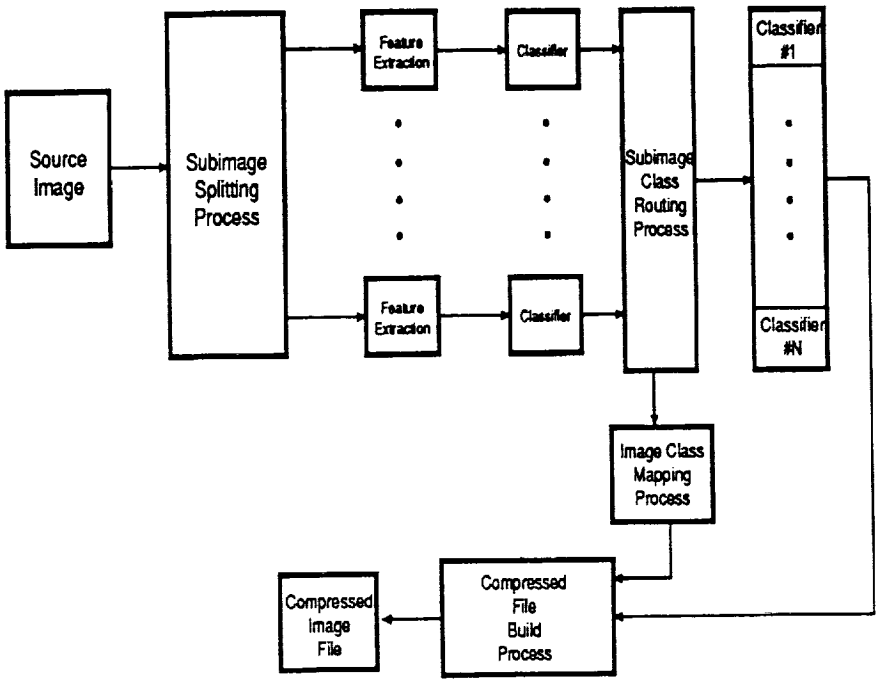


Figure 1: Architecture of the image compression approach.

difference matrices [15], and local grey-scale dependance matrices [16]. Due to the memory requirements of these approaches (at least the same amount as the original image), we elected to use an feature set less resource demanding. The compressibility of an image is related to the range and distribution of intensities in the image, the individual pixel entropy, and the degree of variability in local pixel intensity values. Based on this knowledge, the initial feature included:

Average pixel intensity - The mean over the region of the pixel values, selected so that regions similar brightness would be group together:

$$\bar{x} = \frac{\sum_{i,j} p(i,j)}{N}$$

where, i, j are the rows and columns, respectively, in the n x n subimage, N = n + n, and p(i,j) the intensity of pixel i,j.

Pixel Intensity Variance - The variability in brightness over the region:

$$P_v = \frac{\sum_{i,j}(p(i,j) - \bar{x})^2}{N}$$

First-Order Entropy - The amount of pixel value redundancy present in the region:

$$H = \sum_{i,j}(-P_{i,j} log_2 P_{i,j})$$

Average Run-length – The average value of the run-length sections generated by thresholding the subimage with the mean gray-level:

$$R_a = \frac{\sum_{l,j} R_{l,j}}{M}$$

where, l,j are the rows and columns in an m x n subimage runlength matrix, M = m + n, and R is the distance of the run length.

Run-length Variance – The variance in run lengths obtained by thresholding the subimage with the mean gray-level:

$$R_v = \frac{\sum_{l,j} (R_{l,j} - R_a)^2}{M}$$

The images analyzed in this process were all of sea-ice taken from spectral band 3 from AVHRR satellite data. As shown in Figure 2, sea-ice images (128x128x16-bit) in this spectral band provide a wide variety of textures from smooth to very rough. The compressibility of regions within this large image varies considerably: with the left, middle and right images in the Figure compressing to 35%, 39%, and 28%, respectively, when arithmetic coding is used with first-order modeling. As would be expected, the less textured images compress better.

3 Decision Tree Classifier

An unsupervised classifier method was used to form clusters in feature space, and cluster analysis [17] used to allocate class regions based on minimizing intra-class second-order moments. In the training process, subimages feature vectors containing the above features were calculated for a training set. As each subimage was processed, its feature vector was calculated, it's nearest neighbor in feature space located, the new centroid of the region calculated, and the moments of the vector in that class calculated. Class region boundaries were recalculated when the second moments for the region started to diverge. A binary decision tree was selected for classification using the Kolmogorov-Smirnov test [18] to determine the threshold value for each feature at each node in the decision tree. This results in the selection of a feature at each node which has maximum separation from the nearest neighbor in feature space. Figure 4 shows a representation of a binary decision tree based on the features identified in the previous section.

The classifier output is a class identifier for the subimage processed. This is then used to create a map of the subimages within the image. The map is used as header information, prepended to the compressed image file, and used by a decompression process to recover the original image.

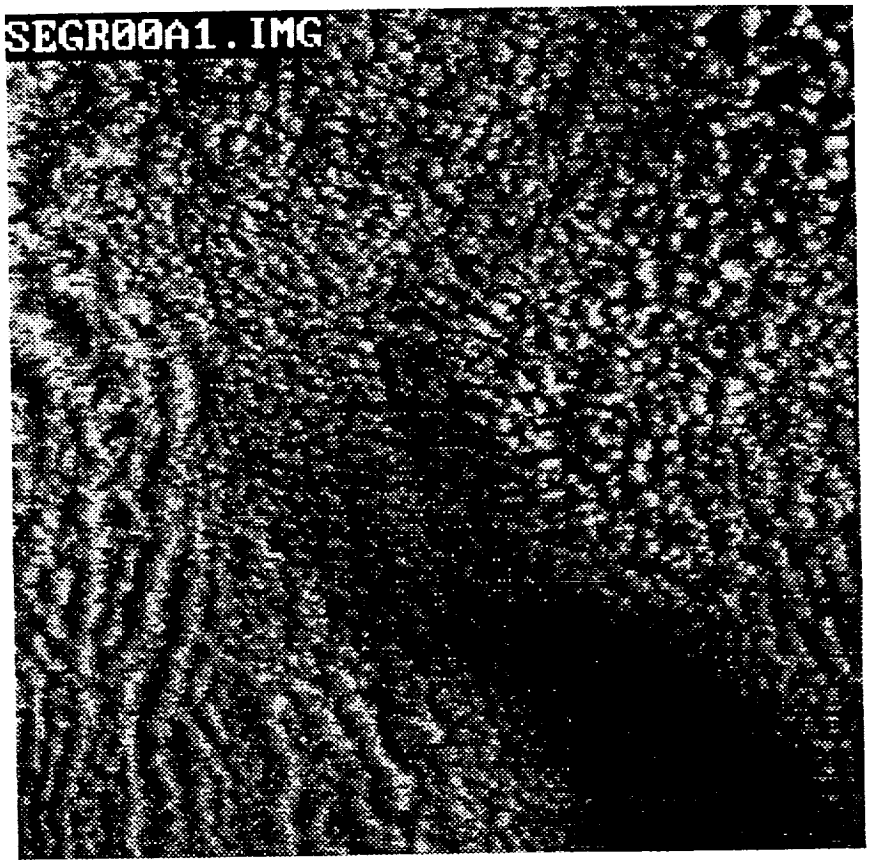


Figure 2: A 512x512x16-bit sea-ice image showing typical texture variety

Figure 3: Sea-ice images showing light and dark evenly textured regions, and a region with high edge content

4 Compression Results

We selected several compression methods based on their ability to compress medical images. These were : Lempel-Ziv dictionary coding with a 15-bit code [19] (LZ), Huffman coding with adaptive modeling (HC) arithmetic coding with static modeling (Ar) arithmetic coding with zero-order modeling (Ar-0) arithmetic coding with adaptive first-order modeling (Ar-1) arithmetic coding with adaptive second-order modeling (Ar-2)

We ran sixteen 512x512x16-bit sea ice images through each of these processes to obtain a baseline of performance for each compression method. Table 1 shows the results of this baseline for three types of texture: Smooth regions with very little or gradual changes in textures; moderately textured regions similar to the middle image in Figure 3; and, regions including a high density of gray-scale variations. Next images were divided into 64x64x16-bit subimages and the compression methods run on each subimage. The compression results were examined to determine the best compression method for each of the subimages. The best compression technique was always one from LZ, Ar- 0, Ar-1, or Ar-2 with a bilinear interpolation decorrelation preprocessing step. The number of classes (k) to be considered was then taken as four. The feature set mentioned above was extracted for each image and cluster analysis performed for the subimages from the entire image set. An automatic classifier was built using the previously mentioned procedure and a binary decision tree was generated. The 16 512x512 images were then run through the adaptive process shown in Figure 1 and the compression values shown in Table 2 obtained.

5 Conclusions

The adaptive application of a variety of compression techniques on satellite images as opposed to applying one technique appears to give an improvement in lossless compression in order of 15we have presented to select the lossless compression technique is to calculate features of subimages with the image and use then feature values in a binary decision tree to select the best compression technique for that subimage. While these results by no means

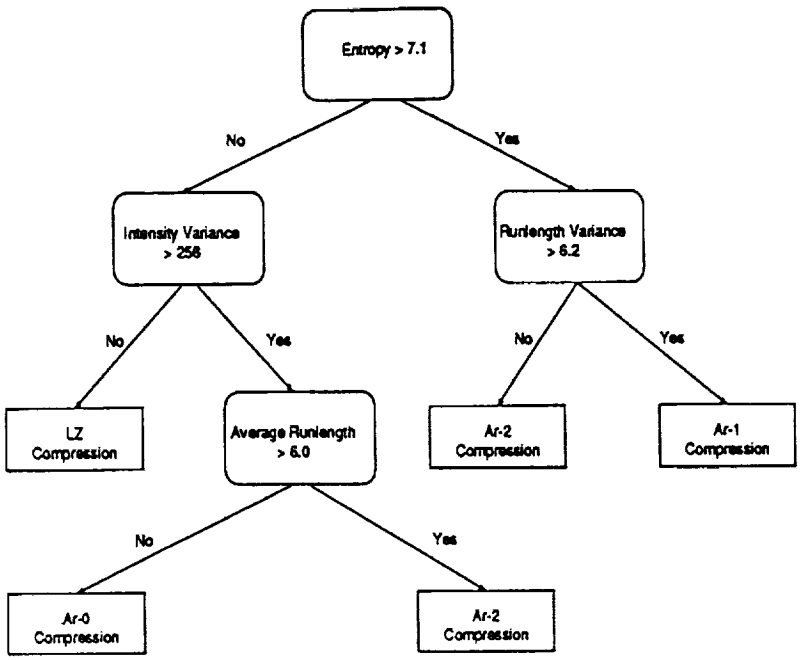


Figure 4: Binary decision tree to select the optimal compression method based on selected feature values

Compression Method	Smooth Image	Moderate Image	Granular Image
Huffman	46	34	26
AR	46	35	27
LZ	54	46	35
Ar-0	51	47	44
AR-1	68	55	49
AR-2	68	51	43

Table 1: The averaged values for the compression methods tested with compression expressed as a percentage of source/compressed file size

Smooth Image	Moderate Image	Granular Image
71	62	56

Table 2: Adaptive compression applied to the same images showing better compression than the best of Table 1

imply that this approach will universally provide better compression for all satellite images, they do indicate that further work in expanding the test database is merited. Expansion of the compression approaches to include a decorrelation preprocessing step, as described previously should be undertaken. The feature set used to select the compression method was adequate, but attention needs to be paid to using the most cost/effective features from a processing point-of-view. It is believed that this compression approach using low-cost multiprocessing architectures and large secondary caches on each processor, will provide compression performance suitable of use in satellite imagery systems such as TRMM and EOS.

6 References

1 D. A. Huffman, " A Method for the construction of minimum-redundancy codes," Proc. IRE, vol. 40, pp. 1098-1101, Sept. 1952.

2 I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," Commun. ACM, vol. 30, pp. 520-540, June 1987.

3 G. R. Kuduvalli and R. M. Rangayyan, "Performance analysis of reversible image compression techniques for high-resolution digital teleradiology," IEEE Trans. Med. Imaging, vol. 11, no. 3, September 1992.

4 T. V. Ramabadran and Keshi Chen, "The use of contextual information in the reversible compression of medical images," IEEE Trans. Med. Imaging, vol. 11, no. 2., June 1992.

5 . R. C. Gonzales, and P. Wintz, Digital Image Processing. Reading, MA.: Addison-Wesley, 1977, pp. 276-281.

6 M. Rabbani and P. W. Jones, "Image compression techniques for medical diagnostic imaging systems," Jnl. of Digital Imaging, vol. 2, no. 2, pp 65-78, May 1991.

7 . P. Roos, M. A. Viergever, M. C. A. Van Duke, and J. H. Peters, "Reversible intraframe compression of medical images," IEEE Trans. Med. Imaging, vol. 7, pp. 328-336, Dec. 1988.

8 J. Rissanen and G. G. Langdon, Jr., "Universal modeling and coding," IEEE Trans. Inform. Theory, vol. IT-27, pp. 12-23, Jan. 1981.

9 J. G. Cleary and I. H. Witten, "Data compression using adaptive coding and partial string matching," IEEE Trans. Comm., COM-32, no. 4, pp. 396-402, Apr. 1984.

10 P. G. Howard and J. S. Vitter, "New methods for lossless image compression using arithmetic coding," Proceedings of the Data Compression Conference, Snowbird, Utah, pp. 3-12, 1991.

11 Y. M. F. Lure, C. S. J. Liou, R. J. Stewart, and J. J. Metzner: "Reversible compression of medical images using decomposition and decorrelation methods", Proc. 26th IEEE Annual Asilomer Conf. on Sig. Syst. and Comput., Invited paper, Montery, CA, Nov. 1-3, (1993).

12 O. Baudin, A. Baskurt, F. Dupont, R. Prost, and R Goutte, "Structural analysis and coding of multimodal medical images," Proc. SPIE, Medical Imaging 1993, Image Capture, Formatting, and Display, Vol. 1897, pp. 298 - 304, 1993.

13 R. J. Stewart, Y. M. F. Lure, and C. S. J. Liou, "An approach to achieve maximum lossless compression of medical images", submitted to Data Compression Conference, IEEE Computer Society, Snowbird, Utah, Mar., 29-31 (1994).

14 R. M. Haralick, "Statistical and structural approaches to texture analysis," Proc. IEEE, vol 67, May, 1979, pp. 786-804.

15 M. Unser, "Sum and difference histograms for texture classification," IEEE Trans. on Pattern Analysis & Machine Intelligence," Vol. PAMI-8, no. 1, Jan. 1986, pp. 118-125.

16 C. Sun and W. G. Wee, "Neighboring gray level dependance matrix for texture classification," Computer vison, graphics and image processing, vol. 23, 1983, pp. 341-352.

17 A. Jain and R. Dubes, Algorithms for Clustering Data, Englewood Cliffs, NJ, Prentice-Hall, 1988.

18 J. R. B Cockett, and J. A. Herrera, "Decision tree reduction," Jnl. of the ACM, Vol 37, no. 4, Oct, 1990.

19 J. Ziv, and A. Lempel: "Compression of individual sequences via variable-rate coding", IEEE Trans. Inform. Theory, May 1978, Vol. IT-24, no. 5, pp. 530-536.

Automatic Cataloguing and Characterization of Earth Science Data Using SE-trees

Ron Rymon*
University of Pennsylvania
rymon@linc.cis.upenn.edu

Nicholas M. Short Jr.†
NASA/Goddard Space Flight Center
short@dunloggin.gsfc.nasa.gov

Abstract

In the near future, NASA's Earth Observing System (EOS) platforms will produce enormous amounts of remote sensing image data that will be stored in the EOS Data Information System. For the past several years, the Intelligent Data Management group at Goddard's Information Science and technology Office/930.1 has been researching techniques for automatically cataloguing and characterizing image data (ADCC) from EOS into a distributed database [Cromp *et al.*, 92]. At the core of the approach, scientists will be able to retrieve data based upon the contents of the imagery. The ability to automatically classify imagery is key to the success of contents-based search.

We report results from experiments applying a novel machine learning framework, based on Set-Enumeration (SE) trees [Rymon, 93], to the ADCC domain. Following the design of [Chettri *et al.*, 92], we experiment with two images: one taken from the Blackhills region in South Dakota, the other from the Washington DC area. In a classical machine learning experimentation approach, an image's pixels are randomly partitioned into training (i.e. including ground truth or survey data) and testing sets. The prediction model is built using the pixels in the training set, and its performance estimated using the testing set. With the first Blackhills image, we perform various experiments achieving an accuracy level of 83.2%, compared to 72.7% reported by Chettri *et al.* using a Back Propagation Neural Network (BPNN), and 65.3% using a Gaussian Maximum Likelihood Classifier (GMLC). However, with the Washington DC image, we were only able to achieve 71.4%, compared with 67.7% reported for the BPNN model, and 62.3% for the GMLC.

1 The Problem

In the near future, NASA's Earth Observing System (EOS) platforms will produce enormous amounts of remote sensing image data. An EOS platform will typically carry a number of instruments, each capable of continuously taking measurements on several "channels" of the EM spectrum. Each platform is expected to produce an order of 100 gigabytes of such raw

*This work was supported in part by a graduate fellowship ARO Grant DAAL03-89-C0031PRI while the author was in the University of Pennsylvania. Address for correspondence: Ron Rymon, 5600 Munhall Road #207, Pittsburgh, PA 15217.

†Address for correspondence: Nicholas Short, Code 930.1, NASA/Goddard Space Flight Center, Greenbelt, MD 20771

C-3

data a day; the EOS Data and Information system alone will maintain around 11 petabytes of mostly science image data. A number of previously untackled research problems arise along the various phases of such unprecedented venture: planning observations, transmitting the information to the ground stations, standard product generation and archiving, and finally accessing and analysis by scientists of various disciplines.

Each of these activities can benefit from automated techniques that interpret, analyze, and summarize image data. The Intelligent Data Management (IDM) project of NASA/GSFC/930.1 has developed a paradigm for managing very large databases, called the Intelligent Information Fusion System (IIFS) [Cromp *et al.*, 92]. In addition to providing techniques for retrieval and science data management, the IIFS manages the processing/reprocessing process by first preprocessing image data based on real-time extraction of scientific features for browsing. By providing a glimpse of the contents of the data, scientists can minimize processing/reprocessing rates by prioritizing product generation scheduling. A major part of IIFS, called Automatic Data Characterization and Cataloguing (ADCC), focuses on classification techniques of imagery according to known features (e.g, forestation, urban area, etc) for the generation of browse products and post-retrieval analysis and verification.

While each of these classification tasks is important by itself, and while they share much of their functionality, they represent different tradeoffs between the accuracy of classification and the resources (primarily time) required to attain it. For the purpose of retrieval, real time performance is most important whereas for analysis tasks accuracy is often the top consideration. Algorithms that can be tailored to execute in both the real-time *and* post-retrieval/analysis phase, despite their very distinct accuracy-runtime tradeoffs, can provide a unified framework for simplifying the process.

2 SE-tree-based Classification

About a decade ago, researchers in machine learning (e.g. ID3 [Quinlan, 86]) and statistics (e.g. CART [Breiman *et al.*, 84]) proposed using a *decision tree* as an underlying structure in classification algorithms. A large body of work since then has resulted in significant improvements to algorithms and understanding; thousands of systems and applications followed suit. In the remote sensing domain, for example, [Civco, 89] has shown that decision trees can be built to provide for impressive interpretation accuracy. A new learning framework which generalizes the decision tree-based approach by using a more expressive *Set Enumeration* (SE) tree has recently been presented in [Rymon, 93]. In a recent study, yet to be published, SE-trees were empirically shown to enjoy a particular advantage over decision trees in large domains (for which relatively few examples are available), and in noisy domains. In the remainder of this section we provide a brief and *informal* overview of this approach; the interested reader is referred to [Rymon, 93] for a more in-depth presentation.

2.1 Set Enumeration Trees

SE-trees were first proposed by [Rymon, 92] simply as a way to systematically search a space of sets. As a representation for a decision making procedure, an SE-tree differs from a decision tree in that *multiple* attributes are typically used in each of its branching points. The construction process, and the way in which the SE-tree is used in classification of new instances, are both

very similar to the process in which decision trees are commonly constructed and used. In fact, the construction process generalizes the construction of a decision tree, except that here the corresponding partitions are often *not* mutually exclusive. In fact, one way to view an SE-tree is as an edge-sharing *collection* of numerous decision trees. However, taking advantage of their overlap, an SE-tree is not nearly as large as that collection. Similarly, the SE-tree-based classification algorithm generalizes the one which uses decision trees, except that here a *number* of paths can match a new instance.

Given a set of attributes, or features, a *complete* SE-tree is simply a tree representation of all *sets* of attribute-value pairs. It uses an indexing of the underlying set of attributes to do so systematically, i.e. to *uniquely* represent *all* such sets. Figure 1 depicts a complete SE-tree for the space of partial instantiations of three binary attributes (A, B, and C). Assuming alphabetic indexing, notice that a node is only expanded with attributes alphabetically greater than its own (these attributes are referred to as that node's *View*). Of course, the complete SE-tree is typically too large to be used in practice, and so an algorithm will usually search the most relevant parts of that tree.

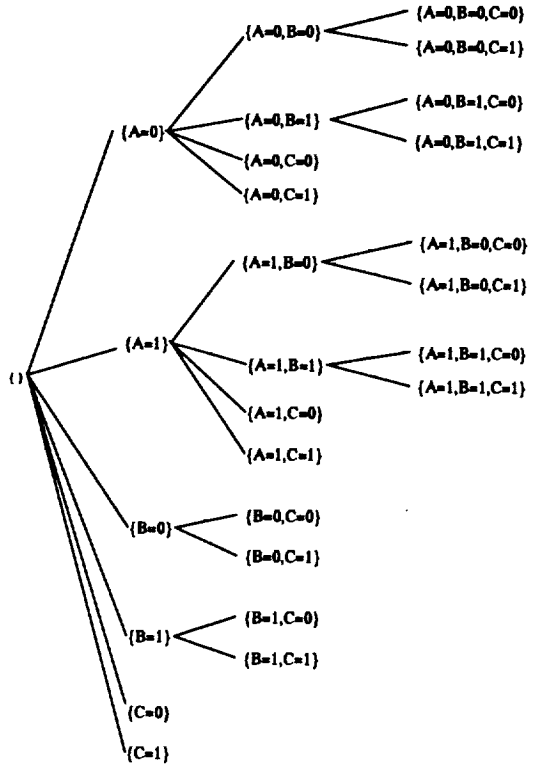


Figure 1: Complete SE-tree for 3 Binary Variables

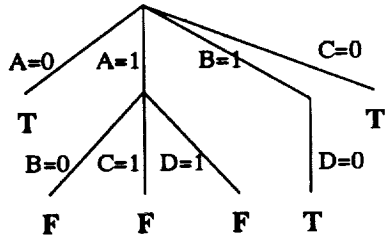
2.2 *SE-Learn* and *SE-Classify*

SE-Learn and *SE-Classify* are, respectively, families of SE-tree-based algorithms for acquiring knowledge from examples (learning), and for using such knowledge to classify new instances. The SE-tree underlies *SE-Learn*'s *search* for classification rules, and is used as an efficient *representation* for these rules, both for the purpose of storing them and for the purpose of subsequently using them in *SE-Classify*.

Like its decision tree counterparts, *SE-Learn* works by recursively partitioning the training set. Unlike decision tree-based frameworks, however, this process stops not only with the discovery of a rule-node, but can also be truncated with the dismissal of a node as not having the potential of leading to rules. Typically, *SE-Learn* starts with all training instances at the SE-tree's root, and first branches on all attribute-value pairs appearing in the training set. According to the SE-tree's structure, it then recurs on a node's remaining training instances, but only with respect to attributes in that node's *View*. *SE-Learn* stops partitioning either when all remaining training instances are equally classified, or by a similar stopping criterion which designates the node's label as a rule (e.g. if there is a great majority for a given class). Rule nodes become leaves; they are retained and labeled with the appropriate class. While searching for rules, *SE-Learn* prunes away nodes which are subsumed by previously discovered rules. Furthermore, its systematic exploration allows it to prune nodes that cannot *lead* to rules, e.g. when the remaining instances only differ in their assignment to attributes *not* in the node's *View*.

Consider, for instance, the following set of instances, and the SE-tree constructed for it by *SE-Learn*:

A	B	C	D	Class
0	0	1	0	T
0	1	1	1	T
1	0	1	0	F
1	1	0	0	T
1	1	1	1	F



Given an SE-tree acquired by *SE-Learn* and a new instance to be classified, the decision procedure *SE-Classify* works by recursively branching from the root of the SE-tree constructed by *SE-Learn* on paths that agree with that instance. Consider the previous example, and a new instance {A=1,B=0,C=0,D=1}. Note that unlike in a decision tree, given its multiple-attribute branching, an SE-tree may include a *number* of such paths. For the new instance, these are the ones labeled with {A=1, B=0}, and {C=0} respectively. *SE-Classify* follows such paths until reaching a class-labeled leaf. Given the potential multiplicity, however, different paths may be labeled with different classes. In our example, for instance, the first rule is labeled with a F whereas the second is labeled with a T. Different variations of *SE-classify* take one of two approaches, or a combination thereof. The first approach involves prioritizing paths (or parts thereof, e.g. some features may be more reliable than others), and consequently the classes labeling their leaves; the algorithm predicts the class labeling the highest priority path. The second approach involves traversing all such paths and using a so-called resolution criteria to determine a prediction from the *combination* of classes. In the remote sensing domain, for example, all rules can be weighted by the reliability or relative relevance of their specific components. Hybrid approaches involve such resolution among the class-labels of a subset of paths with the highest priorities.

2.3 Discussion

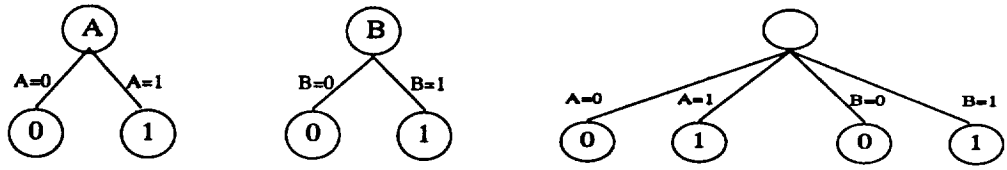
The SE-tree's power as a classification tool draws from the fact that it allows considering a large number of decision procedures in different ways in *each* classification decision. Consider

for example the following (rather small) data set and illustration of the two-binary-attribute domain:

A	B	Class
0	0	0
1	1	1

A \ B	0	1
0	0	?
1	?	1

Four different hypotheses are consistent with this training data, corresponding to any assignment of 0 or 1 to {A=0,B=1} and {A=1,B=0}. In contrast, there are only *two* ID3-style[1] decision trees, depicted in Figure 2(a). The corresponding SE-tree (Figure 2(b)) contains, as subset of its arcs, both trees. As we will soon demonstrate, it can be used to represent *all four* hypothesis



(a) Decision Trees (b) SE-Tree

Figure 2: SE-tree versus Decision Trees

Both *SE-Learn* and *SE-Classify* allow prioritization via user-specified *exploration policies*. Technically, an exploration policy is simply a ranking of sets of attribute-value pairs. In *SE-Learn*, it is used to guide a *best-first* exploration. In *SE-Classify*, it allows implementing *preferences* by prioritizing the paths on which the decision procedure branches.

The role of an exploration policy in *learning* is dual. First, if all minimal rules for the given problem are explored, then the particular order in which nodes are expanded can affect the number of nodes that have to be explored vis-a-vis pruning. In the remote sensing domain, for example, an infra-red channel is known to be a good separator for water-based classification. Thus, relevant problems, branching on such channels first is likely to reduce the tree size. More importantly, if an SE-tree exploration is truncated before all rules are discovered then the exploration policy determines *which* are discovered. The similarity between SE-trees and decision trees allows us to borrow many of the techniques developed for the latter. First, we know that the number of nodes that have to be explored depend on the indexing function used in the SE-tree. A number of techniques developed for attribute selection in decision tree-based frameworks (see, for example, studies by [Mingers, 89a, Buntine & Niblett, 92]) can be used here to select a good indexing function. Second, to reduce exploration, but also to reduce overfitting, we can take advantage of statistically-motivated pruning techniques developed for decision trees (e.g. [Mingers, 89b])

The exploration policy plays an even more important role in *classification*, where it is used to implement *preference* (*bias*). Since training data is often incomplete, a large number of classifiers may be consistent with it, yet differ significantly on new data. The notion of *bias*, the set of preferences guiding the choice among such classifiers has thus received significant attention in the Machine Learning literature (e.g. [Mitchell, 80, Utgoff, 86]). In the remote sensing domain, classification bias can be defined by the particular channel's reliability and/or separability, by the relevance of the feature, and even by the particular data chosen by the scientist for training (since such choice may itself represent some preference). *SE-Classify* allows separating

[1]There are more decision trees, but only these can be generated by an ID3-like procedure.

genuine domain-based preferences from those that merely reflect the particular algorithm used for learning. In many cases, the latter (termed *search bias* [Buntine, 90]) can be eliminated altogether. In variations of *SE-Learn* where exploration is truncated, the exploration policy reflects a learning-phase bias.

Consider again the previous example where we argued that the SE-tree can represent all four hypotheses. Indeed, the particular hypothesis selected in a particular classification session depends on the exploration policy (bias) used. Assuming a classification algorithm which follows arcs by their ranking and predicts the first leaf to be encountered, an OR function[2], for example, can be achieved using an exploration policy which ranks the SE-tree arcs in the following order: B=1 first, A=0 second, A=1 third, and B=0 fourth. The algorithm's response to each possible instance is depicted by the following table:

Instance	Arc followed	Class
{A=0,B=0}	A=0	0
{A=0,B=1}	B=1	1
{A=1,B=0}	A=1	1
{A=1,B=1}	B=1	1

Exploration policies and truncated learning can also be used to trade off accuracy and time, both in learning and in classification. In addition to simple truncation, of particular interest may be a variation of *SE-Learn*, described in [Rymon, 93], that learns *consistent* SE-trees, i.e. ones in which all paths matching a new instance are equally labeled.

Finally, in [Rymon, 93], we show that the SE-tree's indexing function can be dynamically tweaked so that the first nodes to be explored by *SE-Learn* will correspond to those explored by *any given* decision tree-based algorithm. This decision tree will then appear at the leftmost side of the SE-tree. As a result, a variation of *SE-Learn* was implemented which starts off with a given decision tree, and hill-climbs in the performance space. Viewed differently, this result shows once more that a decision tree is a special case of an SE-tree – one in which every node expands with only one new feature and in which every possible instance matches at most one path.

To summarize, the SE-tree can be both viewed and constructed as a generalized decision tree. The similarity between *SE-Learn*/*SE-Classify* and decision tree-based algorithms allows borrowing from the many techniques that were developed for the latter, e.g. discretization algorithms, selection criteria, pruning techniques, etc.

3 Experiments

A prototypical system which implements a variation of *SE-Learn* has been built in the University of Pennsylvania and tested on a number of domains. For the purpose of this paper, we have mainly experimented with Landsat-derived imagery data of South Dakota's Blackhills region. Each pixel in the image corresponds to $79m \times 79m$ on the ground. For every pixel, we have readings from four sensors set to different channel radiances. The readings are each digitized to an integer between 0 and 255. The task is to classify each pixel according to its correct "ground truth", i.e. urban, agricultural, range, forest, or barren. Figure 3 describes the distribution of pixels.

[2]Not modeled by either decision tree.

Class	Name	# pixels	% of total
0	Urban	6676	2.55%
1	Agricultural	42432	16.19%
2	Rangeland	16727	6.38%
3	Forested land	194868	74.34%
6	Barren	1441	0.55%
	Total	262144	

Figure 3: Distribution of Pixels for Blackhills Image

In our experiments, we tried to stay close to the design of [Chettri *et al.*, 92], who had experimented with two classification techniques: Back Propagation Neural Networks (BPNN), and Gaussian Maximum Likelihood Classifier (GMLC). Chettri *et al.* have used the Blackhills and the Washington DC data sets in their experiments. Accuracy results from their experiments, estimated on a holdout from each data set, are presented in the first part of Figure 4. [Campbell *et al.* 89] have also experimented with the Washington DC image, albeit using a more refined geographical classification system.

Experiment description	Accuracy
Back Propagation Neural Network	72.7%
Gaussian Maximum Likelihood Classifier	65.3%
SE-Learn/SE-Classify	
exp. 1 (invq)	47.0%
exp. 1 (vote)	75.5%
exp. 1 (quad)	80.0%
exp. 2 (w/ cluster-based discretization)	83.2%
exp. 3 (w/ four neighbors)	89.8%
exp. 4 (w/ bit-based discretization)	81.6%
exp. 5 (w/ bit disc. & stat-significance pruning)	
ID3 decision tree only	75.9%
SE-tree	78.5%

Figure 4: Comparative Accuracy on Blackhills

In our first experiment with *SE-Learn/SE-Classify*, we have used the Blackhills data set as is, i.e. we associated one attribute with each sensor reading. A fifth attribute was associated with the class, taking 5 values. Learning was pursued to completion, i.e. until all relevant parts of the SE-tree are exposed. A conjunction was defined as a rule if and only if all training instances conforming to it were labeled with the same class. We have experimented with three different *domain independent* weight-based resolution criteria:

1. majority (simple) voting among all matching rules (*vote*);
2. quadratic weighting (*quad*) in which every rule is weighted by r^2 (r denotes cardinality); and
3. inverse quadratic weighting (*invq*) where every rule is weighted by $1/r^2$.

As displayed by Figure 4, there was a great deal of variation in the performance and resolution criteria favoring more specific rules did significantly better.

One problem with the design of the first experiment is that, at least currently, the SE-tree framework handles poorly ordered features. In particular, it is unable to consider *ranges* in such features, and thus has to come up with many more rules than are really necessary. This problem is common to many discrete classification methods. It is common to precede such programs with a process in which all attributes are discretized. Thus, in our second experiment, we added a clustering-based discretization scheme which partitioned the 0-255 range of each sensor reading into a number of intervals. These intervals were then given a name and were input to the *SE-Learn/SE-Classify* program as nominal values. As shown by Figure 4, this has resulted in an improved accuracy. As aside, the difference between the different resolution criteria has almost disappeared, i.e. similar results were obtained using any of the three resolution criteria.

Although the results obtained thus far were already impressive, we tried to milk even more out of the data by allowing the program to look also at a pixel's four neighbors. Given the substantial increase in the size of the training data, we had to add bias in the *learning phase*. In particular, we decided to prefer a pixel's own readings to those of its neighbors. Figure 5 presents accuracy results for ten different tree sizes. We believe that the ultimate improvement in accuracy resulted from the fact that regions tend to be homogeneous, i.e. that a pixel is likely to belong to the same class as its neighbors. The effectiveness of this bias is demonstrated by the temporary *drop* in accuracy, occurring immediately as soon as the algorithm starts considering neighboring pixels. The range of accuracies and their overall correspondence to the tree size (and thus runtime) represents a potential for a dynamic choice of tradeoff. The overall monotonicity suggests that accuracy can be further improved.

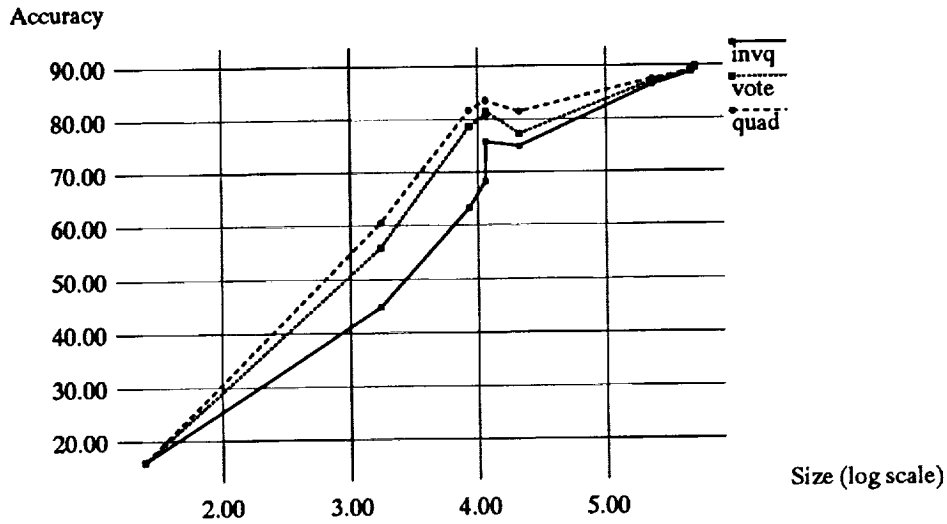


Figure 5: Potential Tradeoff between SE-tree Size and Accuracy

In a fourth experiment, we have discretized the input data by associating a vector of 8 boolean features with each sensor reading, for a total of 32 features per pixel. Again, for lack of time, learning was not pursued to completion. Instead, the program first explored an ID3-based decision tree as the leftmost part of the SE-tree, and then augmented this tree with more parts of the SE-tree in a hill-climbing fashion. However, given the sheer size of the complete SE-tree,

we could not run this process to completion. The result corresponds to the last partial SE-tree we could expose, with a simple voting resolution criterion used in classification.

Finally, in a fifth experiment, using the same discretization as above, we added a pruning procedure in which a conjunction is defined as a rule (and is thus not further refined) as soon as one of the classes becomes statistically significant in it (considering the priors of course). In this case, the ID3-based decision tree (pruned as above) was first explored, and new rules were then added by their relative desirability according to Quinlan's information-gain measure. The first result reported corresponds to the decision tree; the second to an SE-tree consisting of this decision tree plus few thousand rules. The pruned decision tree consists of only 7 rules, compared with several thousand rules in the unpruned decision tree.

Having completed the experiments on the Blackhills image, we turned to the Washington DC image. Here, we only ran an experiment similar to the the latter Blackhills experiment, i.e. attributes were bit-discretized, and statistically insignificant rules were pruned away. Figure 6 presents results from this experiment

Experiment description	Accuracy
Back Propagation Neural Network	67.7%
Gaussian Maximum Likelihood Classifier	62.3%
SE-Learn/SE-Classify	
ID3 decision tree only	67.7%
SE-tree	71.4%

Figure 6: Comparative Accuracy on Washington DC

References

[Breiman *et al.*, 84] Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees.* Wadsworth, Belmont, 1984.

[Buntine, 90] Buntine, W., Myths and Legends in Learning Classification Rules. *Proc. AAAI-90*, Boston, MA, pp. 736-742, 1990.

[Buntine & Niblett, 92] Buntine, W., and Niblett, T., A Further Comparison of Splitting Rules for Decision-Tree Induction. *Machine Learning*, 8(1), pp. 75-86, 1992.

[Campbell *et al.* 89] Campbell W. J., Cromp, R. F., and Hill, S. E., Automatic Labeling and Characterization of Objects Using Artificial Neural Networks. *Telematics and Informatics*, 6(3-4):259271, 1989.

[Cromp *et al.*, 92] Cromp, R. F, Campbell, W. J., and Short, N. M., An Intelligent Information Fusion System for Handling the Archiving and Querying of Terabyte-sized Spatial Databases. *Int'l Space Year Conference on Earth and Space Science Information Systems*, Pasadena CA, 1992.

[Chettri *et al.*, 92] Chettri, S. R., Cromp, R. F., and Birmingham M., Design of Neural Networks for Classification of Remotely Sensed Imagery. *Proc. Goddard Conference on Space Applications of Artificial Intelligence*, Greenbelt, MD, 1992.

[Civco, 89] Civco, D., Knowledge-Based Land Use and Land Cover Mapping, *Proc. of 1989 ASPRS/ACSM*, Baltimore, MD, Vol. 3, pp. 276-91.

[Mingers, 89a] Mingers, J., An Empirical Comparison of Selection Measures for Decision Tree Induction. *Machine Learning*, 3(3), pp. 319-342, 1989.

[Mingers, 89b] Mingers, J., An Empirical Comparison of Pruning methods for Decision Tree Induction. *Machine Learning*, 4(2), pp. 227-243, 1989.

[Mitchell, 80] Mitchell, T. M., The Need for Biases in Learning Generalizations. *Technical Report 5-110*, Rutgers University, 1980.

[Quinlan, 86] Quinlan, J. R., Induction of Decision Trees. *Machine Learning*, 1(1), pp. 81-106, 1986.

[Rymon, 92] Rymon, R., Search through Systematic Set Enumeration. *Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge MA, pp. 539-550, 1992.

[Rymon, 93] Rymon, R., An SE-tree-based Characterization of the Induction Problem. *Proc. International Conference on Machine Learning*, pp. 268-275, Amherst MA, 1993.

[Utgoff, 86] Utgoff, P. E., *Machine Learning of Inductive Bias*. Kluwer Academic, Boston MA, 1986.

N94-35059

VEG: AN INTELLIGENT WORKBENCH FOR ANALYSING SPECTRAL REFLECTANCE DATA

P. Ann Harrison
JJM Systems, Inc.
1225 Jefferson Davis Hwy., Suite 412
Arlington, VA 22202
Tel: (703) 416-8256
FAX: (703) 416-8259

Patrick R. Harrison
U. S. Naval Academy

Daniel S. Kimes
NASA/GSFC Code 923

Abstract

An Intelligent Workbench (VEG) has been developed for the systematic study of remotely sensed optical data from vegetation. A goal of the remote sensing community is to infer physical and biological properties of vegetation cover (e.g. cover type, hemispherical reflectance, ground cover, leaf area index, biomass and photosynthetic capacity) using directional spectral data. Numerous techniques that infer some of these vegetation properties have been published in the literature. A fundamental problem is deciding which technique to apply to the data and then estimating the error bounds on the results. Studies have found that using conventional techniques produced errors as high as 45%.

VEG collects together in a common format techniques previously available from many different sources in a variety of formats. The decision as to when a particular technique should be applied is non-algorithmic and requires expert knowledge. VEG has codified this expert knowledge into a rule-based decision component for determining which technique to use. VEG provides a comprehensive interface that makes applying the techniques simple and aids a researcher in developing and testing new techniques. VEG also allows the scientist to incorporate historical databases into problem solving. The scientist can match the target data being studied with historical data so the historical data can be used to provide the coefficients needed for applying analysis techniques. The historical data also provides the basis for much more accurate error estimates than were previously available. VEG also enables the scientist to try "what-if" experiments on data using a variety of different techniques and historical data sets to do comparative studies or test experimental hypotheses.

VEG also provides a classification algorithm that can learn new classes of surface features. The learning system uses the database of historical cover types to learn class descriptions of one or more classes of cover types. These classes can include broad classes such as soil or vegetation or more specific classes such as forest, grass and wheat. The classes can also include subclasses based on continuous parameters, e.g. 0-30% ground cover. The learning system uses sets of positive and negative examples from the historical database to find the most important features that uniquely distinguish each class. The system then uses the learned classes to classify an unknown sample by finding the class that best matches the unknown cover type data. The learning system also includes an option that allows the user to test the system's classification performance.

VEG was developed using object oriented programming, and the current version consists of over 1500 objects.

Introduction

The intent of this paper is to describe the advanced and novel concepts and features of the VEG system, and to show how VEG contributes to and extends the capabilities of

the scientist. VEG is an intelligent workbench for doing scientific studies of the earth's vegetation using optical reflectance data from sensor platforms. The system is being developed as a NASA/GSFC effort in the Biospherical Sciences branch. The workbench represents the development of a concept originally proposed on a much smaller scale by Abelson and Sussman (1987). Their workbench was intended to provide a tool that integrated a diverse set of concepts into an expressive environment for conducting scientific investigations. The VEG system provides a new and sophisticated intelligent system for the support of analysing spectral reflectance data of vegetation.

Background

The remote sensing community studies spectral data from the Earth's surface to infer physical and biological properties of vegetation. Large quantities of sensor data are collected and integrated to produce knowledge about surface characteristics such as cover type, ground cover, leaf area index, biomass and photosynthetic capacity. Future work using the Earth Observing System (EOS Reference Handbook, 1993) will produce significantly more complex as well as larger volumes of data. Studies of spectral reflectance data contribute critically important ecological information to a variety of scientific work including the effect of forest and natural vegetation clearing on local and regional climates, the relation of vegetation properties to energy and water balance, the relation between environmental parameters governing the energy balance and drought and desertification, and the relation between the absorbed, photosynthetically active radiation and the potential productivity of vegetation systems. The importance of these studies is discussed in detail in Kimes, Sellers and Newcomb (1987).

A central process in analysis is the application of a variety of extraction techniques to the raw spectral data to extract additional information for inferring surface characteristics. The fundamental problem is deciding which techniques to apply to the data, and estimating the error bounds on the results. Studies have found that using traditional, ad hoc approaches, the errors of estimation were as high as 45% (proportion of true value) (Kimes, Harrison & Ratcliffe, 1991; Kimes and Sellers, 1985). Heuristic approaches, promise to overcome the simplicity and lack of flexibility of traditional algorithmic approaches and reduce estimation error by taking advantage of partial knowledge to make decisions about technique choice.

The basic datum being analyzed is directional optical reflectance data. Directional reflectance observations are made and then extraction techniques are used to relate these measurements to vegetation characteristics. Reflectance data can be collected on the ground, from aircraft or from satellites. The nature of this data is such that many decisions as to how to handle a particular data set need to be made at the expert level. The process of analysis is also complex and time consuming, requiring numerous steps and the comparison of new data with a potentially very large database of historical data with known attributes. The VEG workbench was designed to manage these problems.

Overview of VEG

VEG collects in a common format various techniques previously available in a hodgepodge of formats from a variety of different sources. VEG makes these techniques readily available to the scientist in one program. It also provides a rule-based decision tool for determining which technique to choose. It captures expertise in rules about when to use each technique. It captures the priority that should be given to different techniques by a simple weighting scheme. VEG provides a comprehensive interface that makes applying the techniques simple. VEG also incorporates historical databases into the problem solving process, enabling the matching of a target being studied to similar historical data so the historical data can be used to provide the coefficients needed for applying the techniques. The historical data also provides a much more accurate error

estimate than was previously available. VEG provides an interface for entering data from external files and outputting results to files in a variety of different formats. VEG also includes a toolbox which allows the user to browse the system, dynamically plot data, get help and print screen dumps.

The current version of VEG implements three different capabilities: estimation of vegetation parameters, estimation of atmospheric effects and a classification learning system. These capabilities represent the three subgoal categories in the system. The subgoal category "vegetation parameter techniques" enables the scientist to apply various techniques to calculate the surface properties, spectral hemispherical reflectance, total hemispherical reflectance, view angle extension and proportion ground cover. Subgoals in the category "atmospheric techniques" make atmospheric corrections to data. "Atmospheric techniques" allow satellites and aircraft data to be corrected for atmospheric effect to determine what the equivalent ground level measurements would have been. Additional atmospheric techniques allow data collected at ground level to be projected to different atmospheric heights. These atmospheric capabilities are currently being implemented. The "classification learning system" subgoals category enables VEG to learn class descriptions of different vegetation classes and then use the learned classes to classify an unknown sample. The "neural networks" subgoal category provides for analysis using neural or connectionist networks. It is not yet available. Figure 1 shows a decomposition of basic VEG system goals.

VEG was implemented using object oriented programming. The objects in the VEG knowledge base were arranged in a loosely defined hierarchy organized by the major components: databases, control methods, techniques, tools and rules. Within the components, objects are organized in abstraction hierarchies. Separate subclasses hold the objects required by the "estimate vegetation parameter" and "estimate

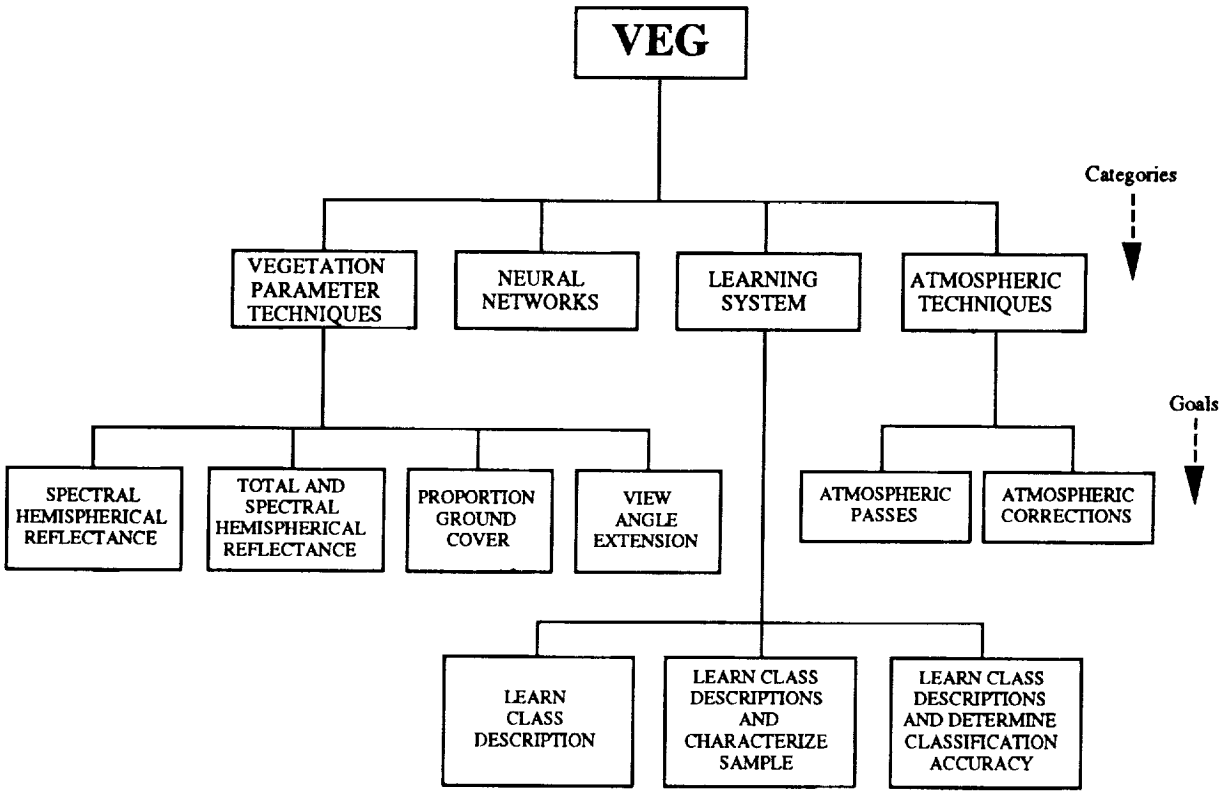


Figure 1: Goal Decomposition of VEG

atmospheric effect" goal categories. The learning system is housed in a separate knowledge base that is loaded only when needed. The full object system with data and rules loaded typically consists of about 1500 objects.

The database components of VEG include various databases used by the system. The most important database subclass contains various sets of typical cover type data which are used to test and demonstrate the VEG system. If VEG is run using new cover type data, additional units are constructed in this subclass to hold the new cover type data. During processing, additional objects are created to store the intermediate and final results of applying various techniques to a cover type sample. These can be inspected or browsed at any time.

All the options in VEG make use of the historical cover type database. This database contains results from experiments by scientists on a wide variety of different cover types. The historical cover type database is maintained externally. It is loaded when needed in a specific application. Currently this is in the form of cases stored in flat files. In the future, it is envisioned that a relational database environment will replace the flat files.

Some of the methods required by VEG are stored in objects. Other methods are stored in files external to the knowledgebase. When the VEG knowledgebase is loaded, these methods files are also loaded. The files contain compiled Common Lisp code for executing steps in processing data and applying the techniques.

Rules are used to determine which techniques to apply to a sample of cover type data. There is a different set of forward chaining rules for each VEG subgoal. In addition, the subgoal proportion ground cover has two sets of rules, one for single wavelength techniques, and one for multiple wavelength techniques. The rules are quite complex. They combine execution of Common Lisp functions with traditional pattern matching. Figure 2 shows an example of a rule. This rule selects the technique 2FULL.1HALF.STRINGS if the data contains two full and one half strings.

VEG also contains a rulebase for ranking the techniques. Currently, the rules in this rulebase implement a simple weighting system. It is anticipated that a more complex rulebase for ranking techniques, incorporating more remote sensing expertise, will be added to VEG in the future.

The rules in VEG are all domain rules rather than control rules. System control is embedded in the window system through the ordering of windows and the constraints on the data input to any window.

VEG is embedded in an extensive, window-driven interface system that provides a variety of screens to enhance dialogue between the scientist and the system. The interface is a key feature of this system. It was designed to focus the scientist on the appropriate level of organization to carry out scientific work without attention to "housekeeping" functions. The interface allows the scientist to interact with VEG and select options at all stages of a run by clicking the mouse over the appropriate menu option. It prevents the user from selecting any step before the prerequisite steps have been carried out. The interface allows a scientist with no knowledge of Common Lisp or the detailed structure of VEG to use the system with ease.

Most operations are controlled using the mouse. The only time that the scientist needs to use the keyboard during a run is if he or she chooses to enter new data manually. When a new value is entered manually, a function is run. If the user has typed in an invalid value, a message is displayed and the value is not retained in the slot. Thus the interface provides validation of the input data. The interface also prevents incomplete data sets from being stored.

```
IF
        (THE CURRENT.SAMPLE.WAVELENGTHS OF
                    ESTIMATE.HEMISPHERICAL.REFLECTANCE IS ?X)
        (THE STRING.OBJECTS OF ?X IS ?NUM)
        (LISP (= (LENGTH ?NUM) 3))
        (LISP (= 1 (COUNT-IF #'(LAMBDA (X)
                                    (EQ 'HALF (GET.VALUE X 'FULL)))
                                ?NUM)))
        (LISP (= 2 (COUNT-IF #'(LAMBDA (X)
                                    (EQ 'FULL (GET.VALUE X 'FULL)))
                                ?NUM)))
THEN
        (LISP (ADD.VALUE ?X 'TECHNIQUES '2FULL.1HALF.STRINGS))))
```

In Plain Text:

If

There is a unit containing data being studied at one wavelength.
The unit contains data which can be characterized as containing 3 strings.
Of these strings, one is a half string and two are full strings.

Then

Add the value 2FULL.1HALF.STRINGS to the TECHNIQUES slot of the unit.

Figure 2: The Rule that Selects the Technique 2FULL1HALF.STRINGS

An interface to an input file of unknown cover type data is available in VEG. The interface enables the user to name the input file and specify the format for the file. Using this format, the input file is read and the cover type data is stored for processing in the system. VEG also provides the user with the option of having the results of processing written to a file and selecting the format that should be used.

The toolbox is an important part of VEG. The user can activate the toolbox at any time during a run. The toolbox allows the user to read a description of the VEG system, browse the units and slots within the VEG system, obtain help about any screen, plot the zeniths, azimuths and reflectance values of reflectance data in two different plots, explore the historical data base and print out a screen dump of the current screen. The toolbox provides a means of managing the levels of abstraction the scientist sees and allows the scientist to deepen his understanding of system functionality.

A help system has also been integrated into VEG. The help system is currently a prototype version of a system that would provide on-line help for a scientist using VEG. It would allow the scientist to get more information about each screen in the VEG interface. It was designed to help the new user of VEG to learn how to operate the system. Since the help system may not be needed by an experienced user, it was configured so that it is loaded only when needed. The first time the user asks for help, the help system is automatically loaded. An interface that allows the scientist to add and modify help messages has also been integrated into VEG. This enables the scientist to evolve the help system over time.

The Subgoal "Spectral Hemispherical Reflectance"

The steps in the subgoal "spectral hemispherical reflectance" are described in this section to illustrate how VEG can be used. When the option "spectral hemispherical reflectance" is selected, the menu shown in Figure 3 is displayed. This menu enables the user to invoke the steps involved in processing target data to estimate

the spectral hemispherical reflectance and estimate the error in the calculation. Before each step is carried out, a check is made to make sure that the prerequisite steps have been carried out. For example, the results cannot be output before the techniques have been executed. If any prerequisite steps have not been carried out, a message is displayed and the user is prompted to complete the prerequisite steps.

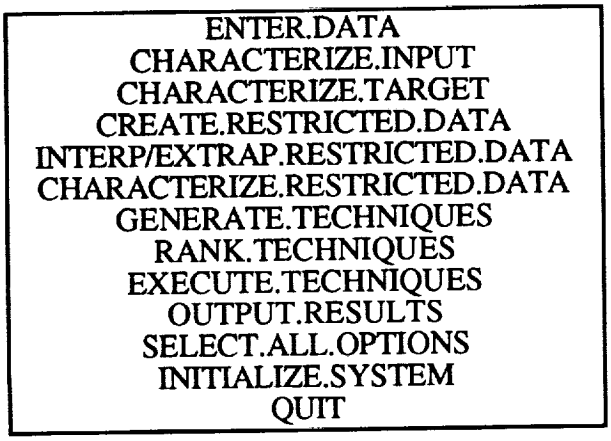


Figure 3: Steps in the Subgoal "Spectral Hemispherical Reflectance"

The first step is to enter the target data. The user can either enter a new, original set of data for an unknown target or select one of a number of samples of target data already stored in VEG. Each set of target data, whether entered by the user or selected from the samples already in VEG, can contain reflectance data at one or more wavelengths. Next, the target data at each wavelength is characterized. Sets of view angles in the same azimuthal plane are identified as "strings." Strings are characterized as full-strings if they contain both forwardscatter and backscatter data and half-strings if they contain either backscatter or forwardscatter data. Next the target is characterized. If the target data does not contain a value for ground cover or leaf area index, a crude estimation of these values is made in this step.

The next step is creating the restricted data set. This step involves selecting a subset of the historical database to be used for generating the coefficients required by the techniques and estimating the error term when various techniques are applied to the target data. The selection of the restricted data set can either be made automatically by the system or manually by the user.

If the user elects to have the restricted data set selected automatically by the system, the database of historical cover types is searched to find the cover types that best match the target. The subset of historical cover types that matches the wavelength of the target is first identified. From this subset, the cover types whose ground cover and solar zenith angle are within ten percent of the values for the target are then identified and pushed onto a list. If the list contains insufficient values, the search is then widened to include cover types whose sun angles and proportion ground cover are within 20 percent of the values for target data. The search criteria are progressively widened until either sufficient cover types have been identified or all cover types whose sun angle and proportion ground cover are within 100 percent of the values in the target have been collected.

The user can also manually select the restricted data set. In this case, a screen is opened. This screen allows the user to enter the maximum and minimum values to be considered for parameters such as height and solar zenith angle. The database of historical cover types is searched to find the cover types that match the criteria entered by the user. The user can then select the matched cover types, enter new maximum and minimum values and match the data again or select a subset of the matched data.

Next, the raw reflectance data for each cover type in the restricted data set is interpolated and extrapolated so that the view angles exactly match at each wavelength the view angles in the target data. The data in the restricted historical data units are characterized using the same methods that were used to characterize the target.

Generating the techniques to be applied to the data is the next step. The techniques can be generated automatically or

selected by the user. If the user elects to have the system generate the techniques, rules are run and the techniques that are suitable for estimating the spectral hemispherical reflectance of the target are identified. If the user elects to choose the techniques manually, a screen containing the names of all the available spectral hemispherical reflectance techniques is opened. When the user left-clicks on the name of a technique, a brief description of the technique is displayed. A function is called to check whether the technique is suitable for the sample. If the technique is not suitable for the sample, an error message is displayed and the technique is deselected. Rules in the "rank techniques" rulebase are run next and the techniques are ranked according to a simple weighting scheme and then displayed in order. The user can select the best one, two or three techniques for each wavelength or pick all the selected techniques.

The techniques are applied to the data at each wavelength in the target. If a technique requires coefficients, the user is asked whether all or half the restricted data set should be used for generating the coefficients and estimating the error. The appropriate coefficient methods are applied as necessary. The techniques are applied to the restricted historical data and the difference between the calculated spectral hemispherical reflectance and the correct value for the spectral hemispherical reflectance stored in the database is calculated. Using the error measurements from several historical cover types, the root mean square error is calculated. This provides an estimate of the error involved in applying the technique to the target data.

In the final step, the results are displayed on the screen. For each technique, the estimate of the spectral hemispherical reflectance, the error estimates and the coefficients are displayed. The screen allows the user to flip between the results at different wavelengths. The user is then asked whether the results should be written to a file. The results for all the VEG subgoals, including the subgoal spectral hemispherical reflectance, can be written to a file.

The Learning System

The learning system provides a tool for classifying new data and for learning new classifications. The learning system uses historical data that represents positive and negative examples to learn classifications. The learned classifications can then be used to classify unknown samples. This is a form of supervised learning first discussed by Mitchell (1982). The theory upon which the learning system was based is discussed in detail in Kimes, Harrison and Harrison (1992).

The learning system provides the user with three different options. In Option 1, the system uses the database of historical cover type data to learn class descriptions of one or more classes of cover types. These classes can include broad classes such as soil or vegetation or more specific classes such as forest, grass or wheat. The classes can also include subclasses based on continuous parameters such as 0-30% ground cover, 31-70% ground cover and 71-100% ground cover. In Option 2, the system learns class descriptions for one or more classes and then uses the learned classes to classify an unknown sample by finding the class that best matches the unknown cover type data. Option 3 allows the user to test the system's classification performance. In this option, the system learns class descriptions for one or more classes and then classifies the appropriate samples in the data base. The percentage of correctly classified samples is then used to summarize the degree of classification accuracy achieved by the learning system.

The first step in Option 1 is to define the training problem. An interface allows the user to enter the solar zenith angle, wavelengths and directional view angles. In order to define the class whose description is to be learned, the user first selects a parameter. In the case of a continuous parameter such as ground cover, the range of possible values is displayed and the user is prompted to enter the maximum and minimum values for the class. In the case of a discrete parameter such as description, the screen displays the possible values of the

parameter and prompts the user to enter the value for the parameter in the class. For example, if the parameter is description, the class might be forest. VEG then checks the validity of the entered data and prompts the user to enter the data again if it is invalid. Additional class parameters can then be defined as necessary. For example, a class might be defined as forest with 70-100% ground cover. The user can then enter data for additional classes such as 31-70% ground cover.

The second step is for the system to learn the class descriptions for the classes that were defined in the previous step. The first step in learning the class descriptions is to generate the training sets. The system searches the historical cover type database and finds the cover types that best match the training problem. A cover type matches the training problem if it has data at all the wavelengths specified in the training problem, its solar zenith is close to the training problem solar zenith, and it has a value for every parameter specified in the class definition. Once a matching cover type has been identified, the values in the slots for each parameter in the class definition are examined. If the cover type data fits the class definition, the name of the cover type is added to the positive training set. Otherwise, it is added to the negative training set. In the first search through the data base, each matching cover type whose solar zenith is within 10% of the training problem's solar zenith is identified and added to the appropriate training set. If insufficient cover types have been found for the training sets, the search is then repeated. In the second search, matching cover types whose solar zenith is within 20% of the training problem solar zenith are identified. The process of increasing the bounds on the solar zenith and searching through the database is continued until either the positive or negative training set exceeds the maximum permissible size, both training sets exceed the minimum permissible size or the bounds have increased to $\pm$ 100%. The learning system is usually run with a minimum training set size of 8 units. If when the search ends either training set is found to be empty, a message is displayed on the screen and the process of learning class descriptions is stopped.

Next, the raw reflectance data from the cover type data in the training sets at the appropriate wavelength is interpolated and extrapolated to match the view angles in the training problem at each wavelength.

Once the training sets have been set up, rules are run in order to determine the set of possible hypotheses that can be constructed for the data in each training set. The left-hand side of each rule tests the view angle data. If the rule fires, the appropriate Common Lisp function is called. Each function generates possible hypotheses to be used in the training problem.

For example, the rule LR.1 fires if the view angle data at a particular wavelength contains at least two view angles. The right-hand side of this rule calls the lisp function TRY-DIRECTION-RELATIONSHIPS which generates direction relationships for every possible pair of view angles in the data and adds these to the list of hypotheses to be tested on the training problem. An example of a direction relationship that might be generated by this function is,

(GREATER-THAN
0.64 (60 180) (30 180)).

This relationship represents the hypothesis that at wavelength 0.64 μm, the reflectance at the view angle (60 180) is greater than the reflectance at view angle (30 180).

When the forward chaining of the rules has been completed, the set of all possible separate hypotheses for each training problem has been generated.

The next step in learning the class descriptions is to determine the discrimination score for each separate hypothesis. Each hypothesis such as (GREATER-THAN 0.64 (60 180)(30 180)) is tested on each sample in the positive and negative training sets. The sample score is 1 if the hypothesis is true and 0 otherwise. The discrimination score is calculated as:

$$\left[\frac{1}{p}\sum_{i=1}^{p} S_i\right] - \left[\frac{1}{n}\sum_{j=1}^{n} S_j\right] \qquad (1)$$

where each sample score is S, S_i is the ith positive sample score, S_j is the jth negative sample score, p is the number of samples in the positive training set and n is the number of samples in the negative training set. Thus a discrimination score of 1 for a hypothesis represents the case where the hypothesis is true for all samples in the positive training set and false for all samples in the negative training set. This represents perfect discrimination. A score of 0 is the break even point where there is no effective discrimination between the positive and negative training sets. A score of less than zero for a hypothesis represents the case where the hypothesis is true for more samples in the negative training set than in the positive training set. In this case, the converse of the hypothesis would yield a positive discrimination score. For each hypothesis such as (GREATER-THAN 0.64 (60 180)(30 180)) two separate scores are calculated. The order of the elements is re-ordered and two scores such as:

(((((GREATER-THAN
(60 180)(30 180)) T) 0.64)) 0.4) (2)

and

(((((GREATER-THAN
(60 180)(30 180)) NIL) 0.64)) -0.4) (3)

are reported. In this example, the score (((((GREATER-THAN (60 180)(30 180)) T) 0.64)) 0.4) means that the hypothesis that the reflectance at angle (60 180) is greater than the reflectance at angle (30 180) for the wavelength 0.64 μm produced a discrimination score of 0.4. The discrimination score in (2) is calculated directly by testing the hypothesis (GREATER-THAN 0.64 (60 180)(30 180)) on all the data in the positive and negative training sets. The discrimination score in (3), -0.4, is calculated as minus one multiplied by the discrimination score in (2). Scores such as (2) and (3) are calculated for each hypothesis.

The next step in the learning of class descriptions is to construct compound hypotheses. A compound hypothesis is composed of the combination of two or more individual hypotheses. The idea is that the interactions between various individual hypotheses may account for more variance (be more predictive) than any individual hypothesis. All the individual hypotheses are considered as potential parts of compound hypotheses, and not just the best single hypothesis.

Before compound hypotheses are constructed, heuristics are used to reduce the set of hypotheses for each training problem by removing any hypothesis that could not be combined with another hypothesis to form a compound hypothesis with a discrimination score better than the current best score. For this reason, every hypothesis whose positive training set score is less than or equal to the current best score for the problem is removed from the list of hypotheses. Hypotheses that do not discriminate or that score zero for the negative training set are also removed from the list of hypotheses. At the end of this step, the list of single hypotheses of each training problem contains only those hypotheses that could potentially be combined with other hypotheses to form a compound hypothesis with a discrimination score greater than the current best score for the problem.

The list of single hypotheses may contain in excess of fifty hypotheses, even after it has been reduced. The number of possible compound hypotheses for some training problems is immense. The problem of dealing with such a large number of potential compound hypotheses was the subject of much effort. Several alternative strategies were experimented with before a successful solution to the problem was found. The first attempt was to implement a breadth-first search. Compound hypotheses that had been investigated were stored on an explored list. Each time a compound hypothesis was investigated, all possible combinations of the hypothesis and other hypotheses were constructed and stored on an unexplored list. Checks were made to prevent duplication of compound hypotheses

on the unexplored list and to prevent the same hypothesis from being investigated more than once. This involved sorting all the separate hypotheses within a compound hypothesis into a standard order so that comparisons could be made. This strategy was rejected because it was slow and the system frequently crashed because it ran out of memory. The second attempt was to implement a heuristic search with a depth bound. This strategy was also very limiting since, in many cases, the system ran out of memory even before the search of compound hypotheses consisting of only two separate hypotheses was completed.

It was decided to try a completely different approach. The new approach was a "Generate and Test" approach. This strategy involved testing some compound hypotheses that the previous version would have recognized as not being possible solutions. However, the new approach had a much reduced memory requirement compared with the previous version because explored and unexplored lists were not kept. All possible compound hypotheses were generated systematically in a way that made duplication impossible. This also greatly reduced the processing time because no checks for duplication needed to be made. This approach was successful. Tests showed that learning class descriptions can take as long as twenty three hours, but no test has failed because of memory problems. The user has the option of interrupting the learning at any time and using the intermediate results.

Unless the best score for the problem is 1.0, the learning system constructs and calculates the discrimination scores for all the hypotheses at level 2 (i.e. compound hypotheses each containing two separate hypotheses). Each compound hypothesis is generated and its discrimination score is immediately calculated. If the score is better than the previous best discrimination score for the problem, the hypothesis and score are recorded, replacing the previous values. If the score equals the best discrimination score for the problem and is better than the best discrimination score for single hypotheses for the problem, the compound hypothesis and score are recorded in addition to the previous best score. If the best discrimination score for a single hypothesis is equaled by a compound hypothesis, the compound hypothesis and score are not stored. Once the level 2 search has been completed, a check is made to determine whether a 10% improvement in the best discrimination score for the problem was achieved by searching at level 2 compared with considering only the single hypotheses. If this level of improvement has occurred, the level 3 hypotheses for the problem are constructed and processed. If a 10% improvement has not occurred, no further compound hypotheses are constructed or tested for that training problem. Constructing and testing hypotheses a level at a time continues until insufficient improvement is achieved or the user interrupts the processing. The user can also control the level of compound hypothesis testing by setting a depth bound. Depth in this context refers to the number of hypotheses in the combination. This step completes the learning of class descriptions.

The final step in learning system Option 1 is to output the results. The solar zenith angle, wavelength(s) with associated view angles and the class definition for the first class to be reported are displayed. The names of the cover types in the positive and negative training sets are displayed along with the best discrimination hypothesis scores for the class. If more than one training problem has been investigated, the user can flip between the classes.

In Option 2, the user either enters a new sample or selects a sample from the set of examples already stored in VEG. The user then enters one or more training classes. This class data is used together with the solar zenith, wavelength(s) and view angles from the sample to define the training problem(s). The system learns the class descriptions for the training problem(s) as in option 1. The sample is then classified to determine the degree to which it matches each of the defined classes and to identify the class that it best fits.

Target data is entered or selected from the samples stored in VEG as in the subgoal "Spectral Hemispherical Reflectance". The

values of the directional view angles, solar zenith and wavelength from the target are used so that the training problem matches the target. In order to complete the definition of the training problems, the user must then enter the class parameters and values and store the data as in option 1. Class descriptions are then learned as in Option 1.

The class descriptions are used to classify the unknown sample by testing to determine which class has the most evidence that the sample belongs to that class. The relative score that is computed to determine whether a sample belongs to a particular class is:

$$\text{Score} = 1 - \frac{E_{Opp}}{E_{Sup}} \qquad (4)$$

or

$$\text{Score} = -1 + \frac{E_{Sup}}{E_{Opp}} \qquad (5)$$

where E_{Sup} is the sum of the discriminating scores of instances in the class solution that are true for the sample data, and E_{Opp} is the sum of discriminating scores that are false for the sample data. Equation (4) is used if the supporting evidence is greater than the opposing evidence. Otherwise, equation (5) is used. These scoring concepts are from Wharton (1987). For each class, the score given by equation (4) or (5) is calculated. The target is identified as belonging to the class that had the highest score according to equation (4) or (5). The final step in Option 2 is to output the results.

The purpose of Option 3 is to allow the user to test or validate the new classes defined using option 1 or 2. New classes can also be defined and tested using this option. Class data is entered for one or more classes. The system then learns the class descriptions as in options 1 and 2. The cover types in the positive training set for each class are then classified according to the learned classes. If the best class for the cover type is the class for which the cover type is a member of the positive training set, the cover type is considered to have been correctly classified. This process is repeated for all the cover types in the positive training sets of all the training problems. The classification accuracy achieved by the learning system is calculated as the number of correctly classified cover types divided by the total number of cover types in the positive training sets of all the training problems.

As in Options 1 and 2, the final step in Option 3 is to output the results. In addition to the data displayed in these options, the cover types that were both correctly and incorrectly classified as belonging to the class are listed together with the performance score for all the classes.

Summary

The VEG system provides a workbench supporting remote sensing scientists doing analysis of spectral reflectance data. VEG was designed using object oriented programming techniques for ease of development, data management and to facilitate continuing system evolution. VEG collects together in one program a variety of analysis techniques that were previously only available in different formats from separate resources. The current version of VEG has over 1500 objects including multiple rulebases, historical databases, graphics tools, browsers, analysis tools, interface objects and functions. It was designed so that it can be distributed across several machines. The database object could be, for example, managed remotely from the rest of the system.

VEG encourages creative investigation by accommodating a variety of repeated tests under different conditions. The interface acts as a research assistant in that it tells the user which combination of data is essential to a particular query, it prevents the system from accepting invalid data, and it ensures that intermediate steps in processing data are carried out in the correct order. Each time new capabilities are added to VEG, the interface is expanded as well. The interface both prompts the user and validates the input. It helps the scientist navigate through the system, reminding him or her when a step has been neglected or a value is out of range or of the wrong type.

The flexibility of the system allows the scientist a platform to conduct any number of explorations of a large body of reflectance data in a very short period of time. What took days in the past can now be accomplished in minutes. This means that the scientist can be much more productive and expansive in his/her thinking than would have been allowable without the time contraction and complexity management that this system provides.

The learning system provides a tool for classifying new data and for learning new classifications. The learning system uses historical data that represents positive and negative examples to learn classifications. The learned classifications can then be used to classify unknown samples. This is a form of supervised learning.

VEG was developed as a NASA/GSFC effort in the Biospherical Sciences branch. It is now being used by remote sensing scientists. It has proved to be a highly useful tool supporting scientific investigation as described by Kimes, Harrison and Ratcliffe (1991), Kimes and Holben (1992), Kimes, Harrison and Harrison (1992), Kimes, Irons and Levine (1992), Kimes and Deering (1992), Kimes, Kerber and Sellers (1993), and Kimes, Harrison and Harrison (1994).

Acknowledgment

This work was funded by the NASA Remote Sensing Science Program in Code SEP03 at NASA Headquarters.

References

Abelson, H. and Sussman, G.J. (May 1987). The Dynamicist's Workbench: 1 Automatic Preparation of Numerical Experiments. *MIT AI Memo 955.*

Asrar, G., and Dokken, D. J. (Eds.). (1993). EOS Reference Handbook. Earth Science Support Office Document Resource Facility: NASA.

Kimes, D. S. and Deering, D. W. (1992). Remote Sensing of Surface Hemispherical Reflectance (Albedo) Using Pointable Multispectral Imaging Spectrometers. *Remote Sensing of Environment,* 39, 85-94.

Kimes, D.S., Harrison, P.R. and Ratcliffe, P.A. (October 1991). A knowledge-based expert system for inferring vegetation characteristics. *International Journal Remote Sensing*, 12(10), 1987-2020.

Kimes, D. S., Harrison, P. R., and Harrison P. A. (1994). Extension of Off-Nadir View Angles for Directional Sensor Systems. *Remote Sensing of Environment,* (Submitted).

Kimes, D.S., Harrison, P.R. and Harrison, P.A. (March 1992). Learning Class Descriptions from a Data Base of Spectral Reflectance with Multiple View Angles. *IEEE Transactions on Geoscience and Remote Sensing*, 30(2), 315-325.

Kimes, D. S. and Holben, B. N. (1992). Extracting Spectral Albedo from NOAA 9 AVHRR Multiple View Data Using an Atmospheric Correction Procedure and an Expert System. *Remote Sensing of Environment,* 13, 275-289.

Kimes, D. S., Irons, J. R., and Levine, E. R. (1993). Learning Class Descriptions from a Data Base of Spectral Reflectance of Soil Samples. *Remote Sensing of Environment,* 43, 161-169.

Kimes, D. S., Kerber, A. G., and Sellers, P. A. (1993). Spatial Errors in Creating Hemispherical Reflectance (Albedo) Maps from Directional Reflectance Data. *Remote Sensing of Environment,* 45, 85-94.

Kimes, D.S. and Sellers, P.J. (1985). Inferring hemispherical reflectance of the Earth's surface for global energy budgets from remotely sensed nadir or directional radiance values. *Remote Sensing of Environment*, 18, 205-223.

Kimes, D.S., Sellers, P.J. and Newcomb, W.W. (August 1987). Hemispherical

Reflectance Variations of Vegetation Canopies and Implications for Global and Regional Energy Budget Studies. *Journal of Climate and Applied Meteorology,* 26(8), 957-972.

Mitchell, T.M. (1982). Generalization as Search. *AI,* 18, 203-226.

Wharton, S.W. (1987). A Spectral-Knowledge-based Approach for Urban Landcover Discrimination. *IEEE Transactions on Geoscience and Remote Sensing*, GE-25, 272-282.

A STATISTICAL INFERENCE APPROACH FOR THE RETRIEVAL OF THE ATMOSPHERIC OZONE PROFILE FROM SIMULATED SATELLITE MEASUREMENTS OF SOLAR BACKSCATTERED ULTRAVIOLET RADIATION

N.L. BONAVITO[1], C. L. GORDON[2], R. INGUVA[3], G. N. SERAFINO[4] and R. A. BARNES[5]

ABSTRACT

NASA's Mission to Planet Earth (MTPE) will address important interdisciplinary and environmental issues such as global warming, ozone depletion, deforestation, acid rain and the like with its long term satellite observations of the Earth and with its comprehensive Data and Information System. Extensive sets of satellite observations supporting MTPE will be provided by the Earth Observing System (EOS), while more specific process related observations will be provided by smaller Earth Probes. MTPE will use data from ground and airborne scientific investigations to supplement and validate the global observations obtained from satellite imagery, while the EOS satellites will support interdisciplinary research and model development. This is important for understanding the processes that control the global environment and for improving the prediction of events. In this paper we illustrate the potential for powerful artificial intelligence (AI) techniques when used in the analysis of the formidable problems that exist in the NASA Earth Science programs and of those to be encountered in the future MTPE and EOS programs. These techniques, based on the logical and probabilistic reasoning aspects of plausible inference, strongly emphasize the synergetic relation between data and information. As such they are ideally suited for the analysis of the massive data streams to be provided by both MTPE and EOS.

To demonstrate this, we address both the satellite imagery and model enhancement issues for the problem of ozone profile retrieval through a method based on plausible scientific inferencing. Since in the retrieval problem, the atmospheric ozone profile that is consistent with a given set of measured radiances may not be unique, an optimum statistical method is used to estimate a "best" profile solution from the radiances and from additional apriori information. This method includes a first guess profile and an estimate of its

1. High Performance Computing Branch, Space data and Computing Division, NASA Goddard Space Flight Center, Greenbelt, MD 20771; (301) 286-4079. Email: c0nlb@charney.gsfc.nasa.gov

2. SYMBIOTIC TECHNOLOGIES Inc., Riverdale, MD 20737; (301) 779-2449 and Laboratory for Atmospheres, NASA Goddard Space Flight Center, Greenbelt, MD 20771; (301) 286-0714 Email: PACF::TOMS$25A: [Gordon]

3. University of Wyoming, Department of Physics and Astronomy, Laramie, WY;(307) 766-6150 and SYMBIOTIC TECHNOLOGIES Inc., Riverdale, MD 20737; (301) 779-2449. Email: Ramarao@corral.uwyo.edu

4. Goddard Distributed Active Archive Facility, NASA Goddard Space Flight Center, Greenbelt, MD 20771; (301) 286-3021. Email: Serafino@eosdata.gsfc.nasa.gov

5. Mantech, Inc., and Laboratory for Hydrospheric Processes, NASA Goddard Space Flight Center, Greenbelt, MD 20771; (301) 286-0501. Email: RBarnes@gsfcmail.nasa.gov

variance, the estimated errors in the measurements, and correlations between profile variance and errors of measurement at different levels. The apriori information provides a constraint on which of the solutions consistent with the measured radiances is to be accepted.

A Bayesian analysis of this problem shows that while the data may fully specify the likelihood of a profile, the apriori information is often dismissed as not as fully cogent as the data. In Bayesian estimation, a balance is found between these two in order to ensure that a unique solution can be selected from within the maximum likelihood of feasible solutions. In addition, since the number of levels over which the ozone is distributed is greater than the number of measured radiances, then the problem of inferring the profile from these measured radiances is an ill-posed one. However, the problem is not only ill-posed, it is also nonlinear and since the transfer function is itself dependent on the profile, the information which is passed from the profile plane to the data plane is expressed as a Fredholm integral equation of the first kind. Ozone retrieval thus appears well suited to a statistical inference analysis that encompasses both logical and probability based reasoning.

In this application, a maximum entropy based Bayesian method is introduced which fully utilizes the evidence of prior information and makes logical assignments of numerical values to probabilities from the measured data. A nonlinear transfer function which includes a single scatter model, and a given climatological profile are convolved in order to model twelve solar backscattered ultraviolet (SBUV) radiances. These range from 255.7 to 339.9 nm. The model radiances and the radiative transfer function are then used as input to both the optimum statistical and maximum entropy methods so as to compare the retrieved profiles with the given one. In the maximum entropy approach, both the data values and apriori information are used as constraints on the entropy. This yields a nonlinear equation for the retrieved profile, and the results obtained are seen to compare favorably with the corresponding analysis provided by the standard optimal statistical estimation procedure.

In this environment, we demonstrate the power of inductive inferencing to identify the source of data and then to accurately infer beyond the given data. These considerations are important in the technology of artificial intelligence. In the mammalian brain for example, the inferencing process in which new information or patterns are discovered and from which predictions are made is implicit in the ability called learning. Most raw data reaching the brain is noisy, incomplete and the product of the convolution of several nonlinear sources. How the brain deconvolves these signals and learns from them remains a mystery. We present results of how two powerful methods of inductive inference are used to accomplish this.

INTRODUCTION

Developing a comprehensive understanding of how the Earth functions requires global observations on a sustained, consistent basis for a decade or longer. These observations must provide both a characterization of the state of the whole planet and detailed measurement of its regional variations. They must also enable quantification of the processes that govern the Earth system. Remote sensing of the Earth's environment from space provides the only truly global perspective available. Making the full set of observations goes well beyond the capabilities of any single satellite however, and many of the detailed measurements can only be made in situ. Such a massive network of global observations is planned for the MTPE program. Extensive sets of satellite observations will be provided by both EOS and the Earth Probes in order to support the MTPE.

Among the several mission objectives of the Earth Observing System is included that of understanding the structure, state variables, composition, and dynamics of the atmosphere from the ground to the mesopause. Since remote sensing of the atmospheric composition and profiling by satellites first began, it has become a major technique for the analysis of planetary atmospheres. As a consequence, many sophisticated methods for deriving atmospheric parameters from satellite measured radiation have been developed. To date most methods attempt to deduce a best estimate of the state of the atmosphere from the given measurements, where the intensity and spectral distribution of the latter are assumed to depend on the atmospheric state in a known way. In the problem of the retrieval of the atmospheric ozone profile for example, previous models of the profile and the knowledge of the behavior of radiative transfer are combined with measured data including total ozone, to reach conclusions about the ozone profile. This is a form of deductive analysis in which we classify the solutions as profiles and then require the algorithm to infer the most likely one based on given information. By deductive is meant the special implication of drawing a particular inference from a generalization.

To estimate the profile from the data, we include that phase of plausible reasoning in artificial intelligence (AI) known as inductive inference. By inductive inference we mean the arrival at a conclusion by using available evidence to reason from a part to a whole or from the individual to the universal. A common problem that arises in all data processing is that of how to handle incomplete information. The situation is further complicated if there are several such datasets originating from different sources. What is required to address this is a method that will not only perform multiple source processing of incomplete data, but will also induce inferences from the data. When an inference is made beyond the observational data, a logical relationship between the data and the inference must be expressed. This relation is in a generalized logic, which is not necessarily deductive, and from which inference is neither deductively proved nor disproved from the data. It assesses the support for the inference given the data, but the essential feature is that this support can be of many different degrees. For example, many instances of an event happening, with no exception, in given circumstances, are better evidence than one instance that the event will happen the next time the circumstances occur. This relation between a set of data and a conclusion is called a probability, and the subject is essentially what is called a many valued logic. Generally speaking, probability theory is the system of reasoning applicable in the absence of certainty. This is also known as inductive logic. As such, a probability expresses a degree of reasonable belief. In ordinary logic, a fixed set of postulates is given at the start, and all propositions asserted later are consequences of this set. In probability theory both the data and the proposition considered are subject to alteration, and it is therefore necessary to keep the data explicit. This relation is usually written in the form

$$P(q|p) = a$$

(read the probability of q, given p, is a), where a is the number that expresses the degree of confirmation. A fundamental development of the theory of probability has been provided by Keynes (1929), in which he contends that the above relation expresses an extended logic or a logic of probable inference. It is defined as a relation between a hypothesis and a conclusion, corresponding to the degree of rational belief and limited by the extreme relations of certainty and impossibility. In this sense then, classical deductive logic would reduce to a special case of the more general development since it would fall within the domain of the limiting relations. As a consequence, certainty would be a special case of probability since the latter cannot be based entirely on classical logic. Using this as a basis, Cox (1946) employs the algebra of symbolic logic to derive the rules of probability from

primitive notions which are independent of the frequency concept. In effect he determines the rules or relations of reasonable expectation consistent with symbolic logic.

In terms of both utility and decision modeling, the processing of radiance data for profile estimation represents a vast and realistic class of problems ideally suited for the introduction of inductive inferencing. Generally speaking, data processing can be considered as an operation in which N numbers can be determined from K observations. For $N < K$, any data regarded as exact values, often lead to a mutual inconsistency in the process of determining the best values of N from K. If $N = K$, then a unique solution exists. However, deconvolution under these conditions will lead to unstable solutions if several of the data values contain almost the same information about the conclusions. This in effect leads to an $N > K$ condition and the problem is then ill-posed since there are many conclusions consistent with the same data. This is the case for the profile retrieval. There are two frequently used methods for addressing this. One is model fitting and the other is the addition of data from other sources so as to get $N = K$. In the first case $N < K$ and the problem is changed to one of parameter fitting. Here an answer is obtained whether the assumption is true or false. If the model is not known to be correct, then we have essentially constructed one of the infinity of conclusions that fits the data. When $N = K$, one can assume that the solution is unique and then use one of several standard mathematical approaches to determine it. If however, several of the data points contain the same information, then the problem may again be ill-posed. An example of this can be found in power spectral estimation, which involves a Fourier transformation of data between two canonically conjugate spaces, such as position and momentum. Since there is no data included beyond the range of measurements, these are in effect considered to be zero. The unmeasured Fourier momentum components however are not zero. This assumption which causes a discontinuity in the maximum measured momentum value leads to large oscillations in position space. To overcome this, Fast Fourier transform (FFT) techniques, employ a smoothing of the data by a time domain window. However, the design of these windows are not based on the true spectrum, so that immediate consequences of this are sidelobe leakage in the transfer function of the smoothing window and a limit on the resolution. For a time series of data covering an interval Δt, the energy of the process defining this data will be constrained within this time interval according to the Heisenberg Uncertainty Principle. In addition, the Fourier transform of this time series function confines the energy to a bandwidth $\Delta f \geq (\Delta t)^{-1}$. Consequently, the best resolution attainable is $\Delta f = (\Delta t)^{-1}$. This is because the function is assumed to be zero outside of the interval in which it is given. If the function can be extended or continued in some physically realistic manner, then the spectral frequency resolution will be considerably higher than $(\Delta t)^{-1}$. For a segment of data of a stationary time series which is short compared to the time series itself, the spectral estimation method of Burg (1967) extends this short data sequence to that of a complete series through inductive inferencing employing the maximum entropy principle.

In this paper, we first review the problem of the ozone profile retrieval and then briefly describe Bayesian and maximum entropy concepts. We present results based on a maximum entropy/Bayes algorithm using radiance data generated from a given climatological ozone profile. These results are compared with those of a classical method known as the optimal statistical solution, using this same generated radiance data.

STATEMENT OF THE PROBLEM

In 1934, Gotz et al. (1934) were able to use measurements of diffusely transmitted solar ultraviolet radiation to infer the main features of the atmospheric ozone profile. Since this classic work, there has been extensive analysis on the problems of inferring atmospheric profiles from measurements of solar irradiance backscattered by the atmosphere (Twomey, 1963, 1965, 1966; Twomey and Howell, 1963, 1967; Mateer, 1964, 1965). The possibility of deducing the ozone profile was first suggested by Singer and Wentworth (1957), and the first mathematical examination of the problem was made by Twomey, (1961). He showed that by using a single scatter atmospheric model, and by expressing the mass of ozone above a given pressure level as an explicit function of the atmospheric pressure, the spectral energy distribution of the backscattered radiance was a Laplace transform of the ozone profile. This method was used in some of the earliest work on evaluating measurements of backscattered radiation. The retrieval of the ozone profile from satellite measurements of the solar ultraviolet radiation backscattered by the earth and its atmosphere, is usually divided into two parts: that of the high level profile above 25-30 km, and below this, the inference of the low level profile. In the high level case, a single scatter model is usually adequate to determine backscattered intensity accurately. The corresponding wavelengths here are at 2975 Å and shorter. For wavelengths that penetrate the ozone layer and are backscattered appreciably within the troposphere, multiple scattering calculations are essential and the effects of aerosol scattering as well as cloud and ground reflections become important. A considerable amount of apriori statistical information about the low level ozone profile is available, whereas relatively few reliable data are available for the high level profile.

The inference of atmospheric profiles from radiance measurements usually involves the inversion of an integral equation of the form

$$\int K(x,y)f(x)dx = g(y). \qquad (1)$$

The g(y) are the radiance measurements specified at various values of y, K(x,y) is the appropriate kernel, and f(x) is a function of the unknown atmospheric profile. In matrix form, Equation (1) can be written in the form

$$\mathbf{Af} = \mathbf{g} \qquad (2)$$

where **A** is the matrix that transforms from the f(x) profile plane into the g(y) observation plane and which also allows for the amplitude transmission of differential spatial scales from the f(x) plane to the g(y) plane. Equations such as (l) in which the kernel is also a function of the desired variable, are called Fredholm integral equations of the first kind (Courant and Hilbert, 1953; Fox and Goodwin, 1953; Fox, 1953, 1962, 1964; Phillips, 1964; Tricomi, 1957). In practice, the following approach is used: For a plane parallel atmosphere, the backscattered radiance I, in the satellite nadir direction, with a solar zenith angle θ and a wavelength λ, can be written

$$I(\lambda,\theta) = F_o(\lambda)(3\beta_\lambda/16\pi)(1+\cos^2\theta)\int_0^1 \exp[-(1+\sec\theta)(\alpha_\lambda X_p+\beta_\lambda p)]dp \qquad (3)$$

where

$F_o(\lambda)$ = extraterrestrial solar irradiance

β_λ = atmospheric scattering coefficient $(atm)^{-1}$

α_λ = ozone absorption coefficient $(atm\text{-}cm)^{-1}$

and

X_p = amount of ozone above pressure p (atm) in atm-cm.

Equation (3) is considered the starting point for the retrieval of the vertical atmospheric ozone profile. In addition to being ill-posed, it is also ill-conditioned in the sense that there are many solutions which exactly satisfy an integral equation slightly perturbed from the original starting conditions.

In the process of inverting Equation (3), additional apriori constraint information is employed to help reduce the problem to one of estimation. A thorough discussion of this is given by Rodgers (1976). The apriori constraints, are sometimes called 'virtual measurements' since they contain information used in the construction of the profile. These can be derived from the physics, from mathematical restrictions on the solution, or from other independent information. The apriori information used in the optimum statistical method also includes a "first guess" profile obtained from the best available ozone climatology. The latter and its variances and covariances are taken as a function of latitude, time of year, and the total ozone. The radiances that such a profile yields when convolved with a radiative transfer function, is then calculated and the differences between these and the measured or direct radiances are then used to provide a new set of profile values. It is expected that the successively iterated results are more consistent with both the measurements and the first guess profile. The application of this method also requires an assessment of the uncertainty or variance in the measurements and statistical apriori profile information. The former is characterized by errors of measurement and requires covariances in the errors of measurements to determine how dependent the errors at one wavelength are on the errors at another. For the apriori information, the corresponding variances and covariances are obtained in the development of the climatology. A complete description of an inversion algorithm which utilizes the optimum statistical method is given by Fleig et al (1990). This approach proceeds as follows: The backscattered radiation given by Equation (3) is first written in terms of the ratio of backscattered to incident radiation. A single scatter representation for the radiative transfer function is introduced, and the ratio is linearized by expanding in a first order Taylor series about a first guess profile. The problem takes the form of Equation (2) where A is now independent of f. The partial derivatives called the weighting functions, are obtained from the ozone profile of the previous iteration and a solution by inversion is obtained in an iterative fashion using apriori and error information. A "best solution" is arrived at when the rms differences between the measured and estimated radiances is minimized.

THE INTRODUCTION OF PLAUSIBLE INFERENCE

An important concern in data processing is that of identifying the data with its source. This may not always be an easy task. The inclusion of inductive inference methods then become not only an attractive option but also a necessary one (Pearl, 1988). A sound method of plausible inference should ideally consist of a strong interaction between logical and probabilistic reasoning. In the profile retrieval problem for example, one attempts to induce the profile from the data with a minimum of bias. To accomplish this requires two items: a prior probability for each of the possible classifications, and the values of the

conditional probabilities of the attributes that define the classes. This is the probability of seeing the data given the class and is called the likelihood of the data. Cox (1979) and Horvitz (1986) provide a thorough and in depth discussion on this. They maintain that with the satisfaction of a certain set of specific conditions, the standard "axioms" of probability theory including Bayes' Theorem (Bayes, 1763; Jeffries, 1983) will then follow logically and can be uniquely defined.

Fundamentally, Bayes' Theorem means calculating the posterior conditional probability

$$P(H_i|DI) = P(H_i|I)\ P(D|H_iI) / P(D|I) \qquad (4)$$

where H_i represents an hypothesis in a sequence of hypotheses $(H_1,.....,H_i,.....,H_n)$, which form a complete set and whose truth one wishes to judge. D is a set of data, and I is whatever prior information one has in addition to the data.The inference can then be summarized such that if H_i is the desired profile, then the best estimate of H_i, in light of the data and any apriori information, is given by that profile which maximizes this posterior probability. Bayes' Theorem thus relates this probability that we require to the two others, one of which can be computed directly. Here $P(H_i|I)$ is the prior probability and represents the state of knowledge (or ignorance) about the profile before there is any data. This prior state of knowledge is modified by the data through the likelihood function or conditional probability $P(D|H_iI)$. This quantity indicates how likely it is that a particular data set would have been obtained from a given (trial) hypothesis. In a typical classification problem, the prior and likelihood terms will compete as to the number of classes present with the likelihood preferring the largest number possible and the prior preferring the least. The conditions which provide a number acceptable to both, will also yield the highest value of the posterior probability. If an experiment is performed and new data D occurs, then a reevaluation is required of the hypothesis H_i in order to calculate the new conditional probability (the left hand side of Equation (4)). With the continued occurrence of data D in repeated experiments, we tend to believe more in the hypothesis H_i at the expense of believing less in the others. The prior probability can be "well-behaved" as is the case whenever this function possesses a single maximum. It can also be "badly behaved" in the sense of having many local maxima. This is usually the case when the data and the desired variable are nonlinearly related. In such circumstances, techniques involving simulated annealing methods are sometimes used to avoid producing local subsidiary solutions (Kirkpatrick, Gelat & Vecchi, 1983). These are usually of little help whenever many almost equally probable solutions are present.

Since the functional form for the likelihood of the data depends essentially on the nature of the source producing the data, then the posterior probability will inherit much of this complex topology. An example of this is found in the restoration of the blurred and noisy image of the binary stellar system R Aquarii, provided by the Hubble Space Telescope Faint Object Camera, (Bonavito et al.,1993). Here the image suffered from both spherical aberration and detector saturation and was characterized by sharp peaks of intensity within data cells immersed in a dim background. Datasets such as these are subject to noise governed by Poisson statistics which are then modeled in the likelihood function.

For complex systems requiring extensive calculations, Bayesian networks show some promising developments (Pearl, 1988; Charniak & McDermott, 1985). To determine the posterior distribution of a Bayesian network, one must specify the prior probabilities of what are termed root nodes (or nodes with no predecessors) on an AI graph. It is also necessary to specify the conditional probabilities or likelihood of all of the nonroot nodes

(evidence or data), given all possible combinations of their direct predecessors. Bayesian networks allow one to calculate the conditional probabilities of the nodes in the network given that the values of some of the nodes have been observed. They are calculated from a small set of probabilities relating only to neighboring nodes. The nodes can be considered as random variables representing various states of affairs. In realistic cases, the networks may consist of thousands of nodes which are evaluated many times as new evidence comes in. What changes then is the conditional probability of the nodes given the new data. The ability of the networks to greatly reduce the complete specification of a probability distribution in complex systems using built-in independence assumptions, now makes extensive Bayesian analyses realizable.

THE MAXIMUM ENTROPY METHOD

Maximum entropy has its roots in the work of Boltzmann (1877) and Gibbs (1875) near the latter part of the last century and in the work of Shannon (1948). It has to do with drawing inferences from incomplete information. Fundamentally, it states that any inferences made concerning the outcome of any natural process should be based upon the probability distribution which has the maximum entropy permitted by the data taken during observation of the process. Here the data is defined as ensemble averages,

$$d_k = \sum_{j=1}^{n} P_j A_{kj}, \qquad 1 \le k \le m, \tag{5}$$

where A_{kj} defines the nature or physics underlying the measured quantities, and the P_j, the distribution upon which the ensemble averages are imposed as constraints. Then as shown by Gibbs (1875) and Jaynes (1957), using the the method of Lagrange multipliers, with the partition function,

$$Z(\lambda_1,\ldots,\lambda_m) = \sum_{j=1}^{n} \exp\Big(-\sum_{k=1}^{m}\lambda_k A_{kj}\Big) \tag{6}$$

the maximum entropy distribution is,

$$P_j = \frac{1}{Z(\lambda_1,\ldots,\lambda_m)} \exp\Big(-\sum_{k=1}^{m} \lambda_k A_{kj}\Big), \quad 1 \le j \le n. \tag{7}$$

The Lagrange multipliers λ_k are obtained from

$$\frac{\partial \ln Z}{\partial \lambda_k} + d_k = 0, \qquad 1 \le k \le m, \tag{8}$$

a set of m simultaneous equations for m unknowns. Any other distributions allowed by the constraints (5), will necessarily have entropy values less than those determined by Equation (7). The fact that P_j is a positive quantity has important implications in all areas of signal processing. There are as many Lagrange multipliers as there are equations of constraint, and these constitute the disposable parameters of the minimally prejudiced probability distribution. They are to be so adjusted as to satisfy the given data. From this, one may conclude that maximum entropy is the appropriate method to 'reason' from the microscopic to the macroscopic. Thus, if one wishes to consider the expectation values for the measured data values given by Equation (5), as entities for which the sum of the probabilities is equal to one, then the corresponding measured values can be said to introduce an element of logical reasoning into the problem of plausible inference. In this sense, they help to determine the consequences of the model for the given constraints. On the other hand, one can also consider that probabilistic reasoning, which enters through Equation (7), is required to interpret the plausibility of the model.

ESTIMATION OF THE ATMOSPHERIC OZONE PROFILE

The problem that we address in this paper is defined as follows: We convolve a known ozone profile at a specified latitude, time of day and solar zenith angle, with a given radiative transfer function using twelve ozone absorption and twelve atmospheric scattering coefficients. This produces twelve model SBUV radiance data values. Using these simulated data values together with the SBUV estimated total ozone and the radiative transfer function, the task is to retrieve the above known (Given) ozone profile used in the convolution.

Let us summarize some of the key issues pertaining to the retrieval problem. We first note that there are more levels over which to distribute the total ozone than there are measured data values (the ill-posed problem). The transfer function is also non-linear and is itself a function of the desired profile. This gives rise to an expression for the backscattered radiation that is in the form of a Fredholm integral equation of the first kind. These integral equations are very difficult to solve and often times unwarranted assumptions are imposed in order to handle them. The problem is also ill-conditioned in that there are many possible solutions which exactly satisfy this integral equation whenever the original starting conditions are slightly perturbed.

The problem is then formulated in terms of the atmospheric pressure. This is possible since the altitude above the surface (except for minor local barometric fluctuations) and the ozone amount distribution, are each be expressed parametrically as a function of atmospheric pressure. It is also useful to choose atmospheric pressure as the independent variable, since atmospheric pressure, not altitude, has a direct influence upon the scattering of the ultraviolet solar radiation.

Measurements of backscattered ultraviolet solar radiation are made at a small number, m of wavelengths, so that in order to facilitate calculation, the atmosphere is divided into n layers, where n is greater than m. A large number of layers is sometimes used to obtain a smooth curve representing the amount of ozone in each layer. In what follows, x_j represents the amount in the jth layer and T is the total amount of ozone in all of the layers.

To adapt the maximum entropy method to the problem of profile retrieval requires the identification of the probability distribution (Equation (7)), with an appropriate profile parameter such as the fraction of total ozone received at a particular level,

$$f_j = x_j / T. \tag{9}$$

From this, the distribution can be written as

$$P_j = \frac{f_j}{\sum_{j=1}^{n} f_j} \tag{10}$$

As a consequence, P_j can then be replaced by f_j where $f_j \geq 0$ at each level. This positivity constraint is guaranteed by the exponential in the maximum entropy solution.

It is convenient to describe the observed radiances of Equation (3) in terms of a quantity Q_λ defined as the ratio of incident to backscattered radiation

$$Q_\lambda = \int_0^{p_N} \exp(-[(\hat{\alpha}_\lambda X_p + \hat{\beta}_\lambda p)]dp) \tag{11}$$

where $\hat{\alpha}_\lambda = \alpha_\lambda (1 + \sec\theta)$ and $\hat{\beta}_\lambda = \beta_\lambda (1 + \sec\theta)$. Equation (11) is of the form of a Fredholm integral equation of the first kind and an approximate maximum entropy solution for this type of equation as well as those of the second kind and the Wiener-Hopf type have been developed by Mead (1986) and Papanicolaou (1984). In their approach, generalized moments are introduced into the integral equation and the problem is converted into an equivalent one in which the informational entropy is maximized using these moments as constraints. Rather than utilizing this approach however, we proceed as follows: The integral in Equation (11) is discretized by dividing the atmosphere into n layers

$$Q_\lambda = \sum_{j=1}^{n} A_{\lambda j} g_{j\lambda} \tag{12a}$$

where

$$A_{\lambda j} = [\exp(-\hat{\beta}_\lambda p_j)] W_j \tag{12b}$$

$$W_j = 0.5\ (\Delta p_j + \Delta p_{j+1}), \quad \text{for} \quad j = 1,2,....,n-1$$

$$= 0.5\ \Delta p_n, \quad \text{for} \quad j = n \tag{12c}$$

and

$$g_{j\lambda} = \exp(-\hat{\alpha}_\lambda \sum_{k=1}^{j} x_k). \tag{12d}$$

Here x_k is the ozone amount in the kth layer. From this, one can then define

$$\Delta p_j = p_j - p_{j-1}, \qquad (12e)$$

where p_j is the pressure at the bottom of the jth layer. Equation (12a) is a nonlinear equation for the ozone layer distribution $\{x_1, x_2, x_3, \ldots, x_n\}$. Maximizing the informational entropy subject to the normalization

$$\sum_{j=1}^{n} f_j = 1 \qquad (13)$$

and the m constraints Q_λ, where $\lambda = 1,2,\ldots,m$, yields

$$f_j = 1/Z \exp(\Gamma_j). \qquad (14)$$

Here $Z = \sum_j \exp(\Gamma_j)$, $\Gamma_j = \sum_{k=1}^{N} \sum_{\lambda=1}^{M} C_\lambda \gamma_\lambda A_{\lambda k} g_{k\lambda}$ and $C_\lambda = \hat{\alpha}_\lambda T$.

The γ_λ are the Lagrange multipliers which couple the constraints Q_λ.

RESULTS AND DISCUSSION

All of the data used in this problem including that which comprise the curves shown as Given and Guess in Figure 1, and that which was used to evaluate the maximum entropy and optimum statistical technique profiles, were provided by the Atmospheric Chemistry Branch of the Laboratory for Atmospheres at the Goddard Space Flight Center. The Rayleigh scattering and ozone absorption coefficients which define the spectroscopic character of this particular radiative transfer function are shown in Table 1. The left hand column are the twelve wavelength values for which twelve corresponding data values of Q_λ given by Equation (12a) were produced during the convolution process. The value for the solar zenith angle was taken to be 69 degrees.

Table 1
Absorption and Scattering Coefficients

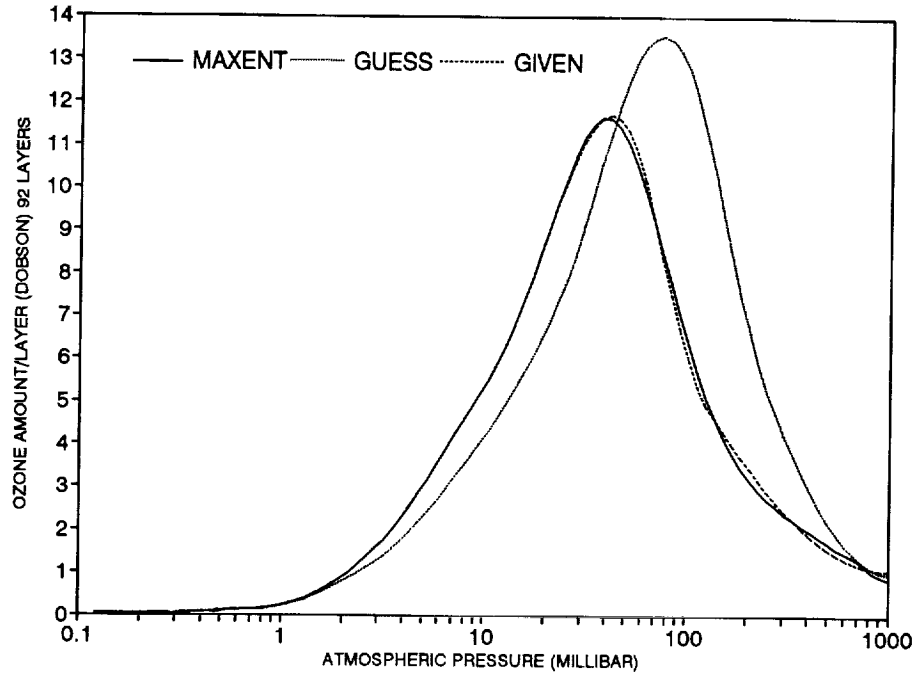
Wavelength (nm)	Ozone Absorption Coefficient (atm-cm)-1	Rayleigh Scattering Coefficient (atm)-1
255.7	309.7	2.4573
273.6	169.9	1.8131
283.1	79.88	1.5660
287.7	48.33	1.4597
292.3	27.82	1.3627
297.6	13.66	1.2605
302.0	7.462	1.1831
305.9	4.281	1.1194
312.9	1.632	1.0198
317.6	0.8684	0.9527
331.3	0.1397	0.7956
339.9	0.0248	0.6864

The ozone distribution for the curve labeled Given was obtained from twelve larger layers of ozone known as Umkehr layers. These Umkehr layers are in turn derived from an algorithm which is based on climatology information. The twelve values are then cubic spline interpolated to 92 layers to yield what is shown as the Given curves of Figures 1(a) and (b). These 92 values were then used to obtain the convolution of a single scattering radiative transfer function given by Equation (3). The total ozone was obtained by the summation of the ozone amount at each of the 92 levels. To convert to Dobson units, these are multiplied by 1000. The curves shown as the Guess on Figures 1(a) and (b), are obtained by changing the day number, latitude and total ozone in the above.

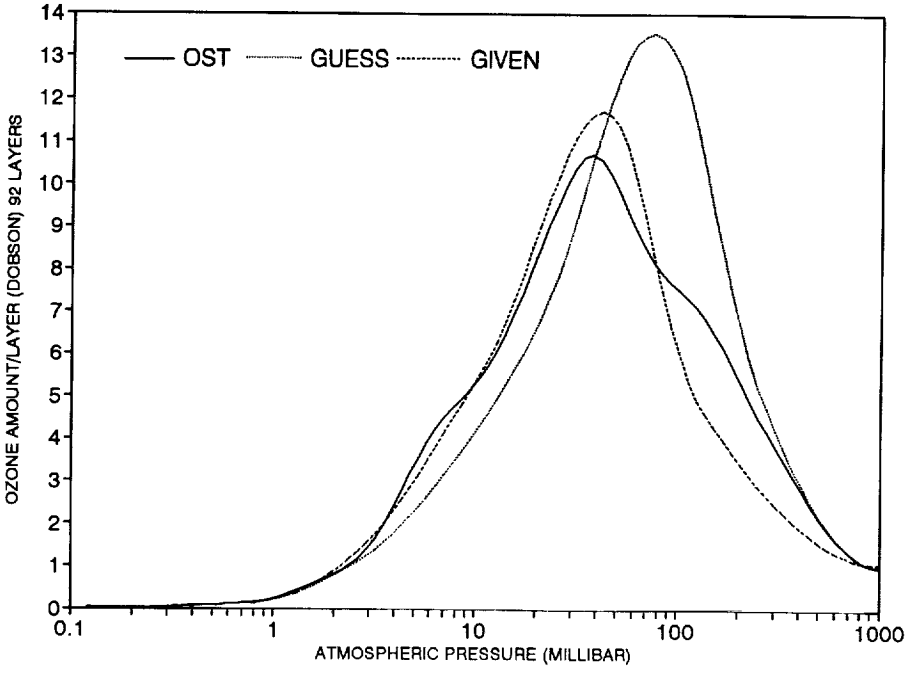
Figure 1(a) shows the maximum entropy retrieved profile. It is clear from these results that this inversion is very close to the Given profile, that is, the one used as the known profile in this example. This agreement is almost exact at all pressure values from below 1 mb up to 1 atmosphere. Figure 1(b) depicts the profile retrieved by the optimum statistical technique for this same Given profile.

This example has allowed us to demonstrate the power of inductive inference methods not only to identify correctly the most likely source of a dataset but also to accurately "predict" new information. In Bayesian analysis the sample probabilities are used to induce an hypothesis which most likely will identify with the data source. In this way, Bayesian statistics involves learning something about the assumptions by looking at the results. This provides a quantitative way to evaluate the probabilities of different assumptions, given the data. This is important in science for example, where there are often competing hypotheses for the explanation of some natural phenomenon. Going back into the unknown, using the observations, is what characterizes Bayesian statistics. In this sense it uses data to test hypotheses. Maximum entropy on the other hand, uses the model identified with the data source to make inferences about the data samples. This cannot be done in classical statistics.

The process of updating knowledge by introducing new data is a basic one in the animal ability called learning. This is complicated by the fact that raw data reaching the mammalian brain is more likely than not the result of the convolution of several nonlinear sources which may have generated datasets that are noisy and incomplete as well.



(a)



(b)

Figure 1: Ozone retrieval by (a) Maximum Entropy and (b) Optimum Statistical Technique (Courtesy of Symbiotic Technologies, Inc.)

The ability to deconvolve these signals and learn from them has fascinated experimental psychologists for more than a century. A recent trend by members of this community has been to pursue the idea that individuals solve particular kinds of problems by making specific inferences (deductive) using rough guidelines that keep track of conclusions compatible with the information at hand and along with relevant prior knowledge. The best fit between the premises of a problem and the acceptable conclusion is judged to be plausible. Psychologists have also focused on ideas which may be involved in forming decisions out of incomplete or ambiguous pieces of information. It is under these conditions however that the human brain often falls prey to what is called "cognitive illusions." More recently, the ability "to understand" has occupied the attention of scientists and engineers engaged in the field of machine learning. The studies surrounding learning in the brain has split along the two lines of thought called behaviorism and cognitivism. These center about the question of whether learning is a matter of behavioral patterning by reinforcement or the storage and use of knowledge. The early behaviorists considered learning as automatic and machinelike. They observed that if a particular response to a particular stimulus pays off for an organism, then the response is likely to be repeated and the probability that the response will be further repeated will be increased by further rewards. It holds that behavior begins as essentially random activity, but connections are strengthened between stimuli and response when the latter are followed by a satisfying result. Known as reinforcement, this is said to strengthen a response, thereby making it more probable. Complementing this concept is that of cognitivism which holds that from the process of reinforcement, information is retained and is confirmed or not by experience, resulting in learning. With the addition of information to random neural systems and with the development of expectations about how certain goals can be achieved, both perspectives can be viewed as the psychological analogues to self-organization. In a very similar fashion, this can also be viewed as a Bayesian description of learning.

REFERENCES

Bayes, T. (1763). Philosophical Transactions of the Royal Society of London, 330-418.

Boltzmann, L. (1877). Uber die Bezehung Zwischer dem Zweiten Haupsatzender Mechanischen Warmtheorie und der Wahrscheinlichkeitsrechnung, Respectivede Satzenuber das Warmegleichgewicht. Wien. Bericht, 76, 373-395.

Bonavito, N. L., Dorband, J. E., & Busse, T. (October, 1993). Maximum Entropy Restoration of Blurred and Oversaturated Hubble Space Telescope Imagery. Journal of Applied Optics, 32, No. 29, 5768-5774.

Burg, J.P., (1967). Maximum Entropy Spectral Analysis. Proc. 37th Meeting, Soc. Explor. Geophys. Oklahoma City, OK.

Charniak, E., & McDermott, D. (1985). Introduction to Artificial Intelligence. Reading, MA.: Addison-Wesley.

Courant, R., & Hilbert, D. (1953). Methods of Mathematical Physics. Vol. I. New York and London: Interscience.

Cox, R. T. (1946). Probability, Frequency and Reasonable Expectation, American Journal of Physics, 9, 1-13.

Cox, R. T. (1979). Of Inference and Inquiry-an Essay in Inductive Logic: The Maximum Entropy Formalism. L. Levine and M. Tribus (Editors). Cambridge, MA: MIT Press.

Fleig, A. J., McPeters, R. D., Bhartia, P. K., Schlesinger, B. M., Cebula, P., Klenk., K. F., Taylor, S. L., & D. F. Heath. (January 1990). *Nimbus 7 Solar Backscatter Ultraviolet (SBUV) Ozone Products User's Guide,* NASA Reference Publication 1234.

Fox, L. (1964). *Introduction to Numerical Linear Algebra.* Oxford: Clarendon Press.

Fox, L. (1962). *Numerical Solution of Ordinary and Partial Differential Equations.* Oxford: Pergamon.

Fox, L., & Goodwin, E. T. (1953). The Numerical Solution of Non-singular Linear Integral Equations. *Philosophical Transactions of the Royal* Society, A 241, 501.

Gibbs, J. W. (1875). *Heterogeneous Equilibrium.* New Haven: Conn. Academy of Science. Reprinted (New York: Longman, Green and Co., 1928); reprinted (New York: Dover, 1961).

Gotz, F. W. P., Meetham, A. R., & Dobson, G. M. B. (1934). The Vertical Distribution of Ozone in the Atmosphere. *Proceedings of the Royal Society of London,* Ser. A. 145, 416-446.

Horvitz, E. J., Heckermann, D. E., & Langlotz, C. P. (1986). Framework for Comparing Alternate Formalisms for Plausible Reasoning. *Proceedings of the 5th National Conference on Artificial Intelligence.* Philadelphia, PA. 210-214.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review,* 106, 620-630 and 108, 171-190.

Jaynes, E. T. (1982). On The Rational of Maximum Entropy Methods. *Proceedings , IEEE,* 70, No. 9, 939-952.

Jeffreys, H. (1983). *Theory of Probability.* Oxford University Press, Fourth Edition.

Keynes, J. M. (1929). *A Treatise on Probability,* London: Macmillan.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Science,* 220, 671-680.

Mateer, C. L. (1964). A Study of the Information Content of Umkehr Observations. Phd. Thesis, University of Michigan.

Mateer, C. L. (1965). On the Information Content- of Umkehr Observations. *Journal of the Atmospheric Sciences,* 22, 370-381.

Mead, L. R. (1986). Approximate Solution of Fredholm Integral Equations by the Maximum Entropy Method. *Journal of Mathematical Physics,* 27, 2903-2907.

Mead, L. R., & Papanicolaou, N. (1984). *Journal of Mathematical Physics,* 25, 2404.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 2nd Ed. San Mateo, CA.

Phillips, D. L. (1962). A Technique for the Numerical Solution of Certain Integral Equations of the First Kind. *Journal of the Association of Computing Machinery, 9*, 84.

Rodgers, C. D. (1976). Retrieval of Atmospheric Temperature and Composition from Remote Measurements of Thermal Radiation. *Reviews of Geophysics and Space Physics*, 14, 609-624.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27, 623-656.

Singer, S. F., & Wentworth, R. C. (1957). A Method for the Determination of Vertical Ozone Distribution from a Satellite. *Journal of Geophysical Research*, 62, 299-308.

Tricomi, F. G. (1957). *Integral Equations*. New York and London: Interscience.

Twomey, S. (1961). On the Deduction of the Vertical Distribution of Ozone by Ultraviolet Spectral Measurements from a Satellite. *Journal of Geophysical Review*, 66, 2153-2162.

Twomey, S. (1963). On The Numerical Solution of Fredholm Integral Equations of the First Kind by the Inversion of the Linear System Produced by Quadrature. *Journal of the Association of Computing Machinery*, 10, 97-101.

Twomey, S. (1965). The Application of Numerical Filtering to the Solution of Integral Equations Encountered in Indirect Sensing Measurements. *Journal of the Franklin Institute*, 279, 95-109.

Twomey, S. (1966). Indirect Measurements of Atmospheric Temperature Profiles from Satellites. II. Mathematical Aspects of the Inversion Problem. *Monthly Weather Review*, 94, 363-366.

Twomey, S., & Howell, H. B. (1963). A Discussion of Indirect Sounding Methods with Special Reference to the Deduction of Vertical Ozone Distribution from Light Scattering Measurement. Monthly Weather Review, 91, 659-664.

Twomey, S., & Howell, H. B. (1967). Some Aspects of the Optical Estimation of Microstructure in Fog and Cloud. *Journal of Applied Optics*, 6, 2125-2131.

U. S. Standard Atmosphere. (1976). National Oceanic and Atmospheric Administration, National Aeronautics and Space Administration, United States Air Force. 39.

N94-35061

Application of Artificial Neural Networks in Hydrological Modeling : A Case Study of Runoff Simulation of a Himalayan Glacier Basin

A.M. Buch A. Narain P.C. Pandey
E-Mail : ambuch@sac.ernet.in

Remote Sensing Applications Group
Space Applications Centre (ISRO)
Ahmedabad 380-053 (INDIA)

Abstract

The simulation of runoff from a Himalayan Glacier basin using an Artificial Neural Network (ANN) is presented. The performance of the ANN model is found to be superior to the Energy Balance Model and the Multiple Regression model. The RMS Error is used as the figure of merit for judging the performance of the three models, and the RMS Error for the ANN model is the least of the three models. The ANN is faster in learning and exhibits excellent system generalization characteristics.

Introduction

Central Asia, the main regime of mountain glaciation in the Himalayas, is notable for its diversity of hydrological and meteorological conditions. As the largest ice-sheet outside the polar regions, it nourishes some of the largest river systems, like the Indus, the Ganges, etc. These mountain rivers have as their source the liquid run-off from melting snow and ice as well as liquid precipitation into their drainage basins. However, for most of the river systems in the Central Asia, the water from melting is the main source of nourishment.

The study of the glaciated mountainous areas presents many difficulties, due to both lack of information on the water balance elements and the great variety and complexity of natural conditions of these areas. The structure of natural zones of ablation (and consequently the water balance) is determined by geographical position and the peculiarities of orography, the absolute and relative height of mountains, the circulation processes and radiation regime, the exposition of slopes, the geological structure, etc. [7]

There exist many empirical methods of simulating and predicting glacier melt runoff,

including time series modeling and actual physical parametric modeling. One of the most widely studied and viable method is the Energy-Balance Model. This method has been advocated by various researchers for operational runoff forecasting from glacierized basins. [3], [4], [10]

The Study Area

The Chhota Shigri glacier is a medium size valley glacier situated in the Lahaul and Spitti District of Himachal Pradesh (India). The glacier lies between latitudes 77-29N and 77-33N and longitudes 32-11E to 32-18E, covering an area of about 19.39 sq. km. Geologically this area is confined to the Rohtang gneissic complex of the Central Crystalline of the higher Himalaya. Fig. 1 (a) shows various features of the glacier and Figure 1 (b) shows relevant hydrogeomorphological parameters. [1]

The Energy Balance Model

Marks and Dozier (1992) have carefully analyzed the various types of heat fluxes and their relative influence over the snow melt run-off process in a small alpine basin. They have clearly demonstrated the dominance of solar radiation heat flux and two other important fluxes, viz. the sensible heat flux, and the latent heat flux in the runoff calculations. This logic can be applied to other glaciated areas as well. Since the solar radiation is easily measured and the heat flux is easily

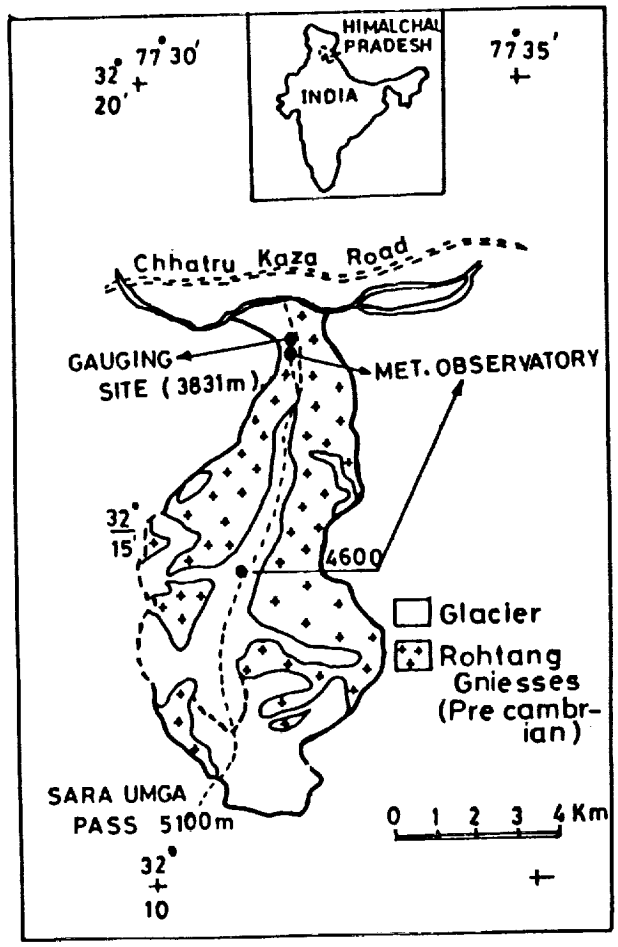


Figure 1 (a). Glacier Location Map

Total Area of the Glacier	19.39 sq. km.
Accumulation Area	18.15 sq. km.
Average Height of Acc. Area	5010 Mts.
Ablation Area	1.24 sq. km.
Average Height of Abl. Area	4133 Mts.
Accumulation Area Ratio	0.91
Maximum Glacier Height	5595 Mts.
Snout Height	3840 Mts.
Orientation	NNE
Average Gradient	176 Mts./km.
Length of Glacier	10.00 km.

(adopted from Dhanju and Buch, 1989)

Figure 1 (b). Hydrogeomorphological Parameters

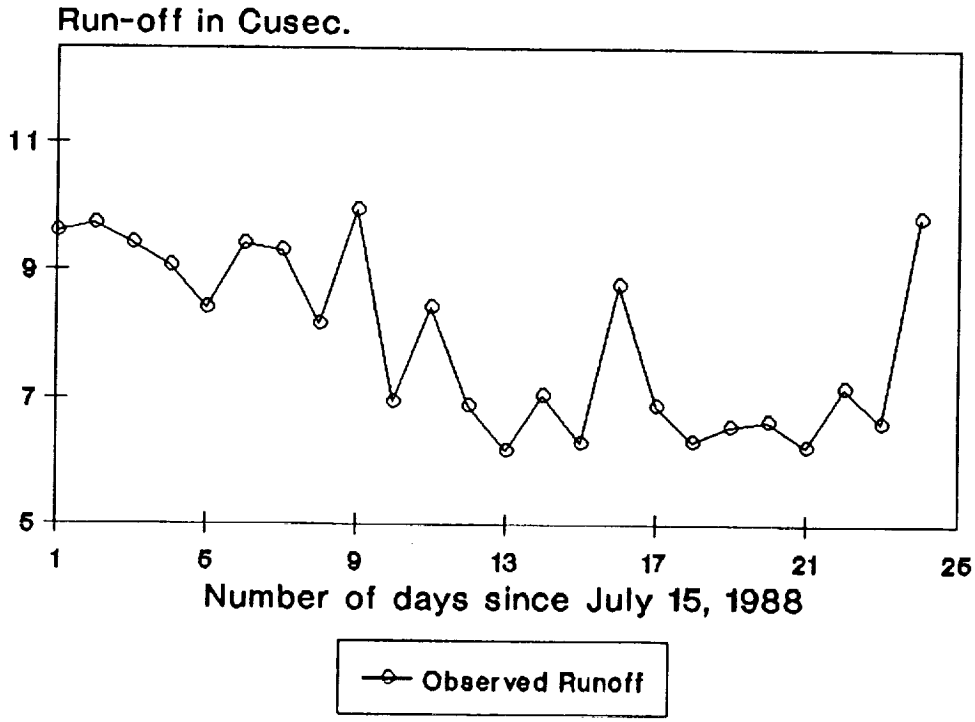


Figure 2. Hydrograph of Chhota Shigri Stream

modeled for a given topography, it should be possible to reliably model the runoff from glaciated areas.

The energy budget for a melting glacier can be expressed as

$$Qm = Qn + Qh + Qe + Qr \text{ --- (1)}$$

Where

Qm = Energy flux actually available for melting the ice.

Qn = Net radiation flux (in Watts / Sq. Meter)

Qh = Sensible heat flux (in Watts / Sq. Meter)

Qe = Latent heat flux (in Watts / Sq. Meter)

Qr = Sensible heat flux from rain. (This term is negligible in our case as no rain was observed during the period of this study.)

The net radiation flux is calculated as

$$Qn = Qsw + Qlw \quad (2)$$

Where

Qsw = Shortwave radiation balance measured at the observatory, by net radiometer.

Qlw = Long wave radiation budget at the surface, modeled using the algorithm suggested by [13].

The turbulent fluxes, viz. The Qh and Qe were determined using the standard aerodynamic formulae and the roughness length was assumed to be constant at 5mm for this experiment.

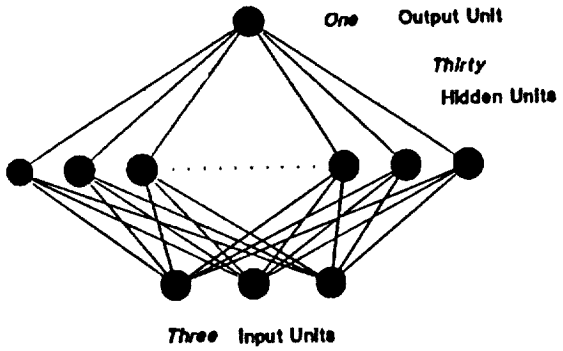


Figure 3. Neural Network Architecture used for Glacier Run-off Simulation

The hydrometeorological data and the net radiation flux data, along with corresponding runoff, were collected in situ during the Chhota Shigri Glacier Expedition in July - August, 1988. The approach to this glacier is possible only during the ablation season, as the high mountain passes remain snow-covered for the rest of the period. This is the major bottle-neck in collecting long-term melt-discharge records for these areas. The Hydrograph of the Chhota Shigri melt-stream is shown in Figure 2.

The data are sparse and call for a technique that can generalize the latent system configuration to improve their applicability to runoff calculations. One such recent technique that possesses excellent generalization characteristics in addition to very good learning capability is the Artificial Neural Network technique.

Modeling Physical Systems and Processes using Artificial Neural Networks

Many of our research efforts in the field of Remote Sensing remain directed towards understanding and developing mathematical models that can simulate dynamic physical processes in oceans, atomosphere and the environment in general. The spacecrft-based sensors provide much-needed details about various parameters that govern these processes. However, due to limited knowledge of the physical processes in the environment, and the inherent noise in many geophysical data, environmental systems often cannot be accurately represented through numeric values describing their physical properties and interactions, but rather are subjected to categorisation into broad classes. [9]

The modelers have been investigating many different techniques to exactly simulate the processes from the previous knowledge, available in terms of historic data (very good survey of recent tachniques is provided by Tsonis and Elsner, 1989). Many efforts to model dynamic systems have been undermined by the system's inherent chaoticity in the data. One solution is to model the temporal variability of a parameter rather than the parameter per se. [2]

Neural Network Architecture used for Runoff Simulation

Recently, there has been a great surge in the application of multi-layer-feed-forward Artificial Neural Networks to diverse system identification problems [2], [11], [5]. In

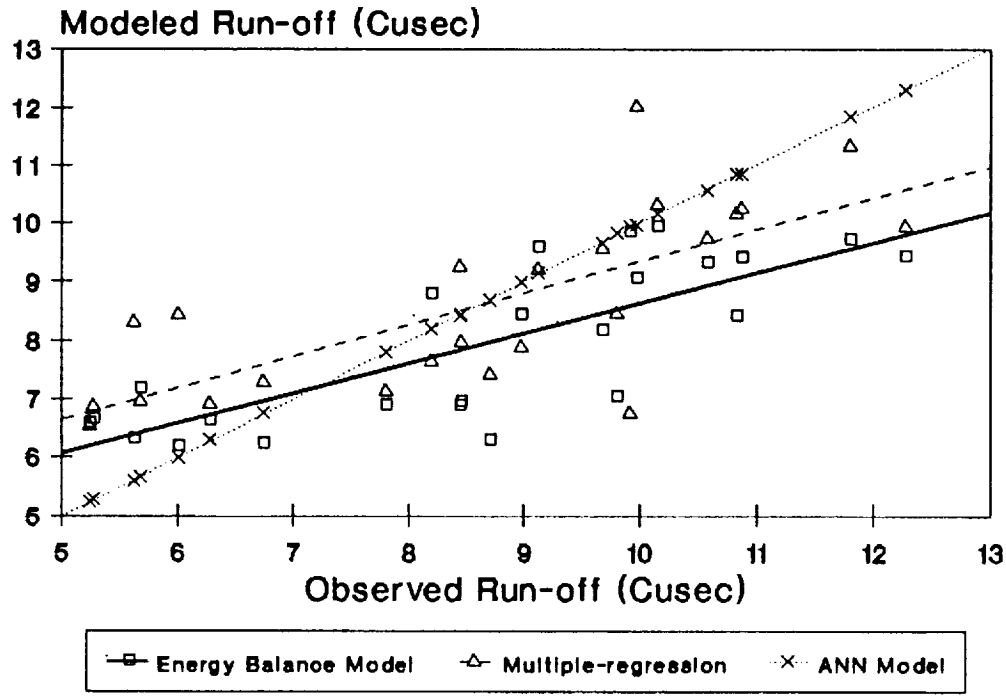


Figure 4. Comparative Model Performance

this study, the Fast-Back-Error-Propagation algorithm has been used. [6]

The network for run-off prediction comprises 3 layers, viz. the input layer, the hidden layer and the output layer. The input layer consists three input neurons corresponding to three heat fluxes viz. the Solar Radiation Heat Flux, the Sensible Heat Flux and the Latent Heat Flux. The hidden layer has 30 nodes, and the output layer consists of one neuron corresponding to the modeled run-off value. The schematic diagram is shown in Figure 3.

Generalisation Performance of ANN Model

Generalisation is a measure of how well the network performs on the actual problem once the training is complete.

The standard method for measuring the generalisation characteristics of a given model is called the method of 'Cross Validation'. The method splits the data set into two subsets, viz. the training data set and the testing data set. The learning is performed using the training data set and the network performance is evaluated using the test data set. Unless the training set is large enough, the performance of the network on training data is not likely to be an accurate measure of its performance on future or unknown data.

To achieve a statistically significant result, several independent splits are required and the average of the results is accepted as the overall

Model	RMS Error
Energy Balance Model	1.46
Multiple Regression Model	1.54
ANN Model (1391 Iterations)	0.06

Figure 5. Table Showing Modeling Errors

performance of the network. While the cross validation method is a widely accepted method, it is extremely time-consuming in the case of ANN as lenghty training times are required for each independent partition of the data set.

In the case of the Himalayan glacier basins, the hydrometeorological data collected in situ are sparce, as the mountain passes leading to the glacier snout remain open for a very short period towards the end of summer. Hence, all of the data sets have to be used for meaningful training. Thus there is a need for alternate reliable methods for predicting the generalisation performance of the network without having a test data set.

An alternative technique which requires far fewer computations is called the 'Predicted Squared Error' technique. This technique relies on statistical methods to derive an expression for the generalisation performance of a system as a function of its performance on the training data set, the number of free parameters in the system and the size of the training data set. [5]

$$PSE = MSE+(2*Nw/Np)*(r2) \quad (3)$$

Where

PSE = Pridicted Squared Error. (For Future)

MSE = Mean Squared Error of the training data set.

Nw = Number of free parametres in the model (120 in our case).

Np = Number of training patterns for the model (24 in our case).

(r2) = Variance of the noise determined by the formula :

$$(r2) = abs[(p/(p-Nw))*MSE] \quad (4)$$

Where

p = Number of patterns used for training the network.

The MSE in the ANN model for the Chhota Shigri Runoff Model is estimated at 0.0036. Hence by substituting the values in eq. 4 and eq. 3 we get the PSE as 0.0126, which is quite acceptable as model error considering the paucity of data.

Conclusions

Figure 4 depicts the resultant calculated runoff from the energy balance model against two other models, viz. the Multiple-Regression Model, the Energy Balance Model and the ANN model. (See Figure 5)

It can be clearly seen from the

comparative performance of the three different models that the trained ANN model assumes the flow values that are closest to in situ measured values. (The Correlation Coefficient between the measured runoff and the runoff simulated by ANN model is of the order of 0.9998) Currently, we are investigating the application of different ANN models to diverse hydrological and meteorological simulation and forecasting problems.

The technique of system modeling with ANN holds very good promise. However, it requires rigorous research and simulations before any operational model can be developed.

Acknowledgements

The authors are grateful to Mr. N.D. Sen, School of Environmental Sciences, Jawaharlal Nehru Univesity, New Delhi, for providing Hydrometeorological Data and Energy Balance Model output for the Chhota Shigri Glacier.

References

(1) Dhanju, M.S. and Buch, A. (1989) "Remote Sensing of Himalayan Glaciers.",Proc. National Meet on Himalayan Glaciology, June 1989, pp. 193-213.

(2) Elsner, J.B. and Tsonis, A.A. (1992) " Non-linear Prediction, Chaos and noise.", Bulletin of the American Meteorological Society, Vol. 73, No. 1, pp. 49-60.

(3) Gottlieb, L. (1980) "Development and application of a run-off model for snow covered and glacierized basins.", Nordic Hydrology (II), pp. 225 - 272.

(4) Hay, J.E. and Fitzharries, B.B. (1988) " A comparison of energy balance and bulk aerodynamic approaches for glacier melt." Journal of Glaciology, Vol. 34,pp. 145-153.

(5) Hush, D.R. and Horne, B.G. (1993) " Progress in Supervised Neural Networks", IEEE Signal Proc. Meg., Vol. 10, No. 1, pp.8-39.

(6) Karayiannis, N.B. and Venetsanopoulous, A.N. (1992) "Fast Learning Algorithms for Neural Networks.",IEEE Tras. on circuits and Systems-II Analog and Digital Signal Processing, Vol. 39, No. 7, pp. 453-474

(7) Lvovich, M.I. and Tsigelnaya, I.D. (1971), "A method of studying the water balance and estimating the water resources of glacial mountain areas.", Proc. of the Moscow Symposium, 1971.IASH, pp. 229-232.

(8) Marks, D. and Dozier, J. (1992), " Climate and Energy exchange at snow surface in the Alpine Region of Sierra Nevada, 2, Snow Cover Energy Balance.", Water Resources Research, Vol. 28, No. 11, pp. 3043-3054.

(9) Maslanik, J.A. and Schweiger, A.J. (1989) "Classification of Merged AVHRR and SMMR arctic data with Neural Networks.", PE & RS, Vol. 55, No.9, pp. 1331-1338.

(10) Moore, R.D. and Owens, I.F. (1984),"Controls on advective snow-melt in a maritime alpine basin.", Journal of Climate and Applied Meteorology.", Vol. 23, pp. 135-142.

(11) Tsang, Leung., Chen, Zhengxiao., Oh, Seho. et al. (1992), "Inversion of Snow Parameters from Passive Microwave Remote Sensing Measurements by Neural Network trained with a Multiple Scattering Model.", IEEE Trans. on Geoscience and Remote Sensing, Vol. 30, No. 5, pp. 1015-1024.

(12) Tsonis, A.A. and Elsner, J.B. (1989), "Chaos, non-linear attractors and weather.", Bulletin of the American Meteorological Society, Vol. 70, pp. 545-547.

(13) Upadhyay, D.S., Chaudhary, J.N. and Katyal, K.N. (1983) "An ampirical model for prediction of snowmelt runoff in Sutlej." Proc. of Ist Nat. Symp. on Seasonal Snow cover, V. II, pp. 129-142

Monitoring, Control, and Diagnosis

Intelligent Resources for Satellite Ground Control Operations

Patricia M. Jones
University of Illinois at Urbana-Champaign
Department of Mechanical and Industrial Engineering
1206 W. Green St., Urbana IL 61801
217-333-3938 (Voice)
217-244-6534 (Fax)
pmj@ux1.cso.uiuc.edu (email)

Keywords: knowledge-based spacecraft command and control; intelligent user interfaces

ABSTRACT

This paper describes a cooperative approach to the design of intelligent automation and describes the Mission Operations Cooperative Assistant for NASA Goddard flight operations. The cooperative problem solving approach is being explored currently in the context of providing support for human operator teams and also in the definition of future advanced automation in ground control systems.

INTRODUCTION

The increasing sophistication and complexity of satellite ground control operations requires new approaches to the design of ground control systems. One approach to providing intelligent assistance to operators in real-time control tasks is via an intelligent cooperative problem solving system. Unlike the traditional expert system that queries the user, detects problems, and offers advice, the cooperative problem solving approach advocates providing a variety of flexible intelligent resources to the human operator (Woods, 1986; Jones, 1991; Jones and Mitchell, 1993).

A theory of human-computer cooperative problem solving proposed by Jones (Jones, 1991; Jones and Mitchell, 1993) provides high-level design guidelines for the development of intelligent cooperative automation. These principles advocate human authority, mutual intelligibility, openness and honesty, multiple perspectives, and the management of trouble. In short, the human operator(s) should retain locus of control in interaction and decision making, and the intelligent automation should be obvious, inspectable, unambiguous, provide multiple views of the situation, and provide support for varying levels of help and expertise.

The rest of this paper will focus on the modeling, representation, and architecture of an existing prototype cooperative problem solver system and current research on defining new requirements and architectures for intelligent cooperative support.

THE MISSION OPERATIONS COOPERATIVE ASSISTANT

A prototype cooperative problem solver system, the Georgia Tech Mission Operations Cooperative Assistant (GT-MOCA, hereafter referred to as MOCA), was developed for the context of NASA Goddard real-time flight operations and was experimentally evaluated with ERBS and COBE flight analysts in the context of a high-fidelity real-time interactive simulation of MOCA provides an interactive normative operator model, messages, and interactive graphics to be utilized in conjunction with the operator's existing work environment. In particular, the interactive normative model provides a visualization of the content and structure of current expected activities and allows operators to query this representation and also delegate activities to MOCA.

MOCA is based on the operator function model (OFM) (Mitchell, 1987) and the OFMspert architecture (Rubin, Jones, and Mitchell, 1988). The operator function model is a heterarchic-hierarchic network model that specifies activities at various levels of abstraction and the actions needed to successfully accomplish those activities. Activities and actions are nodes in the OFM network, and arcs represent system triggering events or the successful completion of activity. The OFM is implemented as a blackboard architecture that forms the basis of intent inferencing (Rubin, Jones, and Mitchell, 1988). Here, the basic OFMspert components as implemented in MOCA and the MOCA-specific Cooperative Problem Solver component are described.

Figure 1 illustrates the overall software architecture. The Controlled System Interface class parses information from the controlled system (in this case, a high-fidelity interactive simulation) and sends appropriate messages to other OFMspert components.

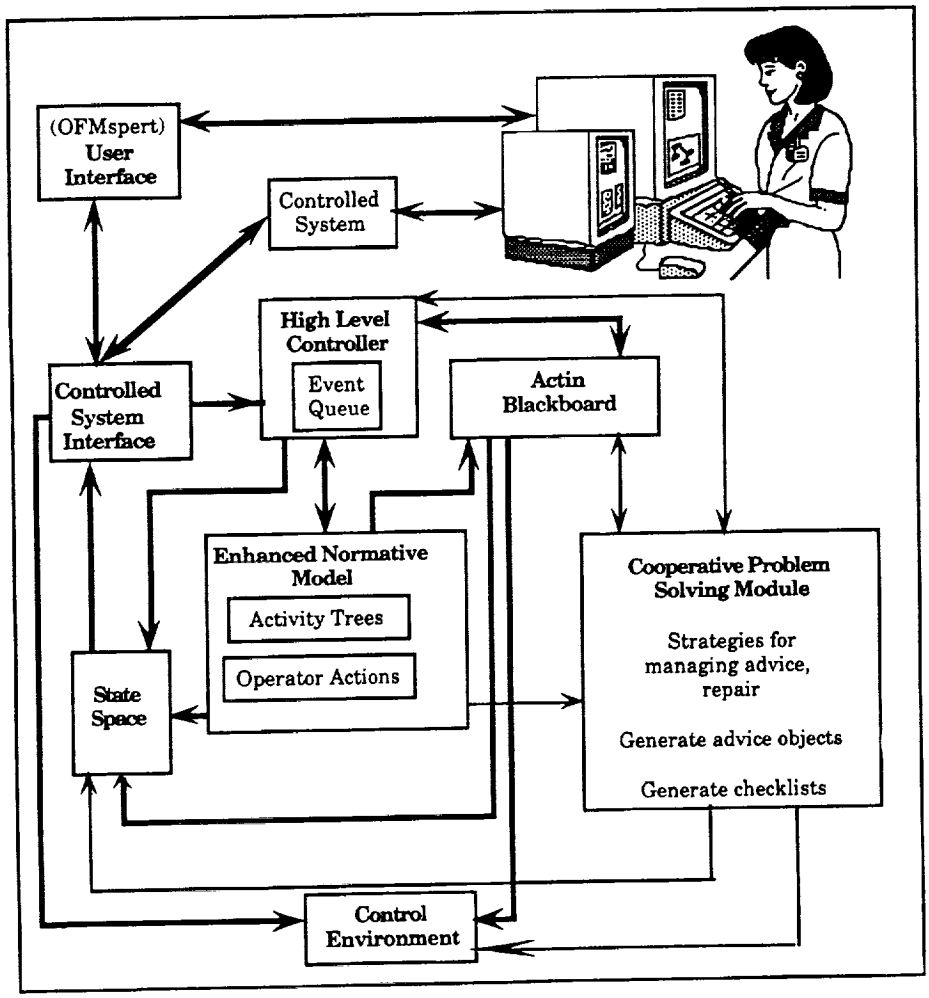


Figure 1. The MOCA architecture.

The State Space class represents the current state of the controlled system. Thus, many of the classes that are fully implemented in the simulation itself (e.g., classes to represent spacecraft components) are partially replicated here to provide a modular, efficient representation of the current system state. The Control Environment class represents the simulation user interface as a collection of DisplayPanel objects, each of which has a name,

a character string denoting what information that panel contains, and an integer denoting whether or not the panel is currently displayed.

The EnhancedNormativeModel class encapsulates the knowledge of the operator function model for a particular domain (here, real-time flight operations), where the model is represented as activity trees (organized functions, subfunctions, and tasks) and actions. In general, the Enhanced Normative Model contains tables of all these structures and member functions that are used to instanciate and schedule the removal of activity trees and actions on the blackboard. Most of this information is file-driven; at the beginning of the program, the Enhanced Normative Model reads in files that give the structure of the activity trees (i.e., function-subfunction-task relationships), their supporting actions, and the structures of all the nodes in the operator function model. The different information and formats in each file correspond to four classes to represent that information. The class ActivityTreeStruct represents the model-driven information (i.e., function-subfunction-task activity trees) of the operator function model. The ActivityTreeStruct class also encapsulates the alert message associated with each tree. The class ModelTreeStruct is a subclass of ActivityTreeStruct that also includes the actions associated with each task. The class ConnectionStruct encapsulates intent inferencing information; i.e., the names of actions are associated with the names of the tasks which those actions support. Also, the ConnectionStruct class contains members to represent the "what is" and "how to" information associated with each action. Finally, the NodeStruct class is used to represent ActivityNodes. This class represents a node's level, name, type, purpose, and enabling event as character arrays. This information is used by Enhanced Normative Model methods for "instanciating" a particular tree.

The Blackboard class of MOCA is essentially the same as previous implementations of the OFMspert blackboard (Rubin, Jones, and Mitchell, 1988; Chronister, 1990). The blackboard data structure has four levels: functions, subfunctions, tasks, and actions. The control structures consist of three lists of events: the clock events list, the problems list, and the events list. The clock events list is a time-sorted list of events to be done at prespecified times. The problems list is a list of unconnected actions. The events list is a list of current events to be processed. The knowledge sources are member functions for processing events to maintain and update the blackboard. Some extensions were made to the representation of blackboard nodes. The abstract superclass ActivityNode now also contains a member denoting the names of the supernodes that this node can connect to, because subfunctions and tasks as well as actions can be posted and connected to nodes at a higher level. ActivityNode also contains a new member that describes the enabling event for its posting (e.g., the Monitor function's enabling event is that telemetry data have begun to arrive). The subclasses of ActivityNode are FunctionNode, SubfunctionNode, TaskNode, and ActionNode. The ActionNode class includes "what is" and "how to" information.

Part of the structure of the Cooperative Problem Solver class is given in Table 1. It represents message-sending paths with other OFMspert components, the history and current focus of allocated activities and communicative acts, the current checklist of actions to be displayed or performed, and an organized collection of declarative information used in the performance of allocated bookkeeping activities. The Cooperative Problem Solver class has a number of member functions for the creation and maintenance of these structures. Besides the behaviors of creating, adding, deleting, and finding items on the various lists, the Cooperative Problem Solver has a number of key functions that are summarized in Table 2.

As noted in these tables, significant classes associated with the Cooperative Problem Solver represent communicative knowledge in the form of *communication knowledge objects* (in a similar spirit to context spaces (Reichman, 1985). Class CommunicationKO represents the communicative act's name, the time it was initiated, the time it was ended, its initiating and terminating conditions (as described in the main thesis text), its history of status changes, purpose, priority, and content. Class CommunicationKO has three

subclasses that represent specialized types of communicative acts. Class AdviceReminderKO has an additional member that represents the feedback structure generated by blackboard assessments. Class LimitViolationKO has additional members to represent the status and current value of the associated spacecraft parameter (whose name is

Table 1. Partial structure of Class Cooperative Problem Solver

Member	Description
giveAdvice	Boolean variable Denotes whether or not to present advice/reminder messages to the user. This is the variable that is set when the user clicks on the "Give Advice" checkbox on the MOCA main menu.
allocatedFunctions allocatedSubfunctions allocatedTasks	Pointers to an AllocatedActivity object These members of class CPS denote the heads of linked lists of AllocatedActivity objects that correspond to the functions, subfunctions, and tasks allocated to MOCA.
currentConversational-Context	Pointer to a CommunicationKO object Denotes the communicative act which is the current focus of processing by MOCA.
currentActivity	Pointer to an ActivityKO object Denotes the activity which is the current focus of processing by MOCA.
currentAdvice	Pointer to an AdviceReminderKO object Denotes the advice which is the current focus of processing by MOCA.
currentChecklist	Array of 10 character pointers (character strings) Denotes the current dynamically-generated checklist to be displayed to the user and/or to be performed by MOCA.

Table 2. Significant member functions of class CooperativeProblemSolver

Member function	Description
propagateResponsibility	When the user allocates an activity node to MOCA, its responsibility is set to "MOCA". This effect propagates downward such that the responsibility of subnodes is also set to "MOCA". Propagation also occurs upward such that supernodes' responsibility is either "shared" (if other subnodes of the supernode are still allocated to the human) or "MOCA" (if all other subnodes have also been allocated to MOCA).
makeRespCommAct	Given an activity node allocated to MOCA, this function returns an associated CommunicationKO that represents the communicative act of "echoing" that act of delegation.
manageCommunication	This function is the heart of MOCA's management of communicative acts. Every CommunicationKO that is created and added to the appropriate list is passed as an argument to this function. Currently, manageCommunication sets the CommunicationKO as the currentConversationalContext, and decides, based on the purpose of this act, which function to call to generate a message to the user as shown in the next box.
generateAdvice generateAlert generateConfirmation generateExplanation generateDetails generateAcknowledgment generateLimitViolation-Message generateDataDropout-Message generateLimitSummary-Message	The functions called from manageCommunication if the currentConversationalContext's purpose is, respectively, to provide ADVICE on missing, out of order, or late actions; ALERT the user that certain functions, subfunctions, and/or tasks were posted on the blackboard; CONFIRM that MOCA has performed an action in response to an activity delegation; EXPLAIN how to do an action on the dynamic checklist display; ELABORATE on "what is" an action on the dynamic checklist display; acknowledge that MOCA received the request to perform an activity (ACTIVITY_MANAGEMENT); or provide domain-specific alert messages. This function places the currentConversationalContext's content in the appropriate textlist display.
generateDynamicChecklist	Given an activity node for which the user requests delegation or a checklist, this function queries the EnhancedNormativeModel to find the actions that constitute the successful fulfillment of that activity.
generateSpecialChecklist	Given an activity node for which the user requests delegation or a checklist, this function examines context-specific information to generate the appropriate checklist of actions. For example, if the user requests a checklist for the TRPBK subfunction, this function examines the current support characteristics to decide if the appropriate control actions involve Tape Recorder 1 or Tape Recorder 2 (i.e., the list may be TR1STBY and TR1PBK, or TR2STBY and TR2PBK).

assigned as the name of the LimitViolationKO). Class DataDropoutKO has the same structure of a CommunicationKO, but is treated differently; its name represents the type of data loss (e.g., "fwd link") and its start and end times represent the start and end times of the data loss.

INTELLIGENT SUPPORT FOR ACTIVITY MANAGEMENT

MOCA provides the basis for further architectures to support human-computer cooperative problem solving. MOCA's limitations,however, included its lack of integration between the resources for cooperative problem solving and its lack of support for planning (Jones and Mitchell, 1993). Currently we are building the Intelligent Support for Activity Management (ISAM) architecture which addresses these issues (Jones, 1993a, 1993b, Jones and Goyle, 1993; Jones, Patterson, and Goyle, 1993). In particular, activities are represented more completely by Activity Objects that explicate knowlege of priorities, resources, constraints, and temporal relationships between activities. Furthermore, the context of activity -- including the current state of the user's "information space" (e.g., displays), current state of the controlled system, and evolving status of artifacts that both guide activity and are the result of activity -- is explicitly captured and represented by objects as well. This architecture is currently under development; a high-level conceptual overview is provided in Figure 2 below.

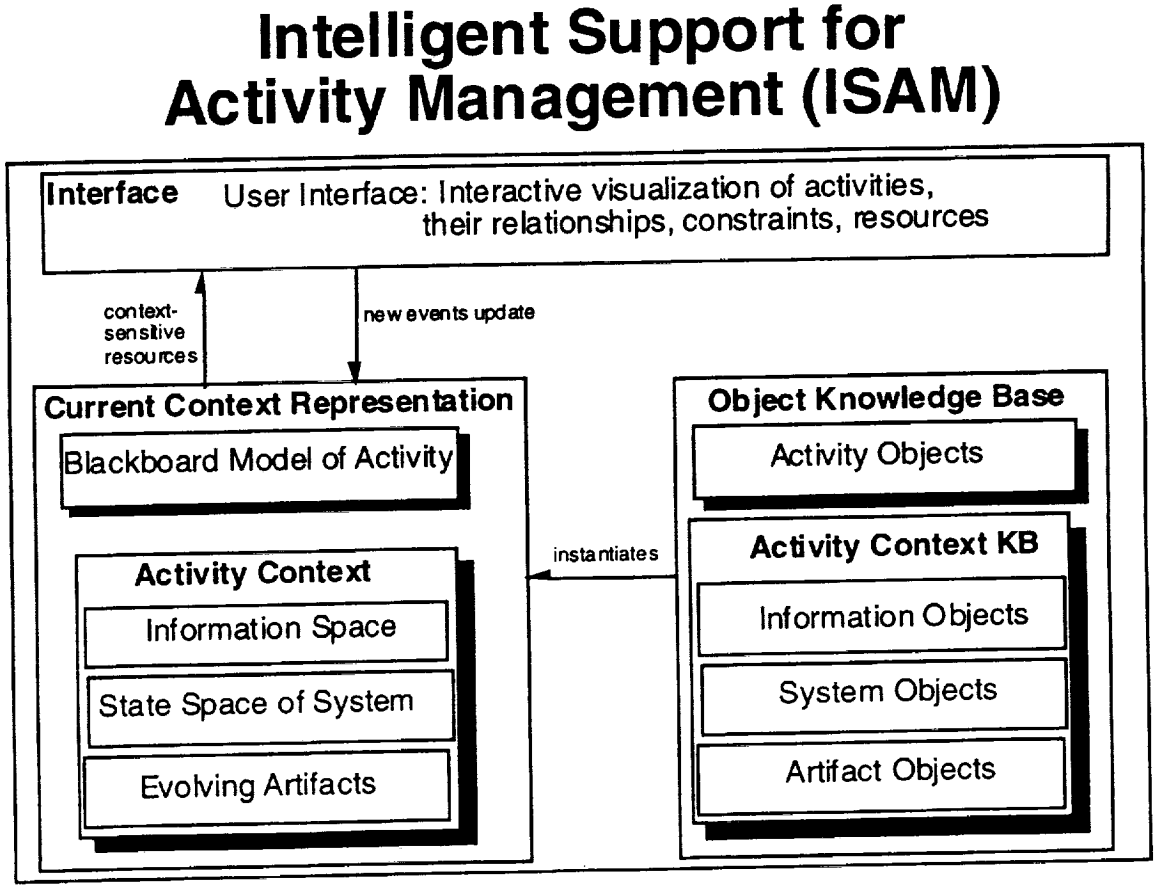


Figure 2. ISAM Architecture.

ACKNOWLEDGMENTS

This research is supported by grants from NSF (IRI92-10918) and NASA Goddard Space Flight Center (NAG5-244).

REFERENCES

Chronister, J. A. (1990). MS thesis, School of ISYE, Georgia Institute of Technology.

Jones, P. M. (1993a) Cooperative support for distributed supervisory control: Issues, requirements, and an example from mission operations. *Proceedings of the ACM Intenational Workshop on Intelligent User Interfaces*, 239-242, Orlando, FL January 1993.

Jones, P. M. (1993b). Cooperative work in mission operations: Analysis and implications for computer support. Manuscript in preparation.

Jones, P. M. (1991). Human-computer cooperative problem solving in supervisory control. PhD dissertation, School of ISYE, Georgia Institute of Technology.

Jones, P. M. and Goyle, V. (1993). A field study of TPOCC mission operations: Knowledge requirements and cooperative work. EPRL-93-05, Engineering Psychology Research Laboratory, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign.

Jones, P. M. and Mitchell, C. M. (1993). Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent operator's associate. Manuscript accepted for publication to *IEEE Transactions on Systems, Man, and Cybernetics.*

Jones, P. M. and Mitchell, C. M. (1991a). A mechanism for knowledge-based reminding and advice-giving in the supervisory control of complex dynamic systems. *Proc. of the 1991 IEEE International Conference on Systems, Man, and Cybernetics.*

Jones, P. M. and Mitchell, C. M. (1991b). Cooperative interaction in the supervisory control of complex dynamic systems. *Proc. of the 1991 IEEE International Conference on Systems, Man, and Cybernetics.*

Jones, P. M., Mitchell, C. M., and Rubin, K. S. (1988). Intent inferencing with a model-based operator's associate. *Proceedings of the Sixth Symposium on Empirical Foundations of Information and Software Sciences* (249-258). Atlanta,GA.

Jones, P. M., Mitchell, C. M., and Rubin, K. S. (1990). Validation of intent inferencing by a model-based operator's associate. *International Journal of Man-Machine Studies, 33*, 177-202.

Jones, P. M., Patterson, E. S. and Goyle, V. (1993). Modeling and intelligent aiding for cooperative work in mission operations. *Proc. of the 1993 IEEE International Conference on Systems, Man, and Cybernetics*, Le Touquet, France.

Mitchell, C. M. (1987). GT-MSOCC: A research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17,* 553-570.

Reichman, R. (1985). *Getting computers to talk like you and me.* Cambridge, MA: MIT Press.

Rubin, K. S., Jones, P. M. and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics, 18*, 618-637.

Woods, D. D. (1986a). Cognitive technologies: The design of joint human-machine cognitive systems. *The AI Magazine, 6*, 86-92.

Vista Goes Online: Decision-Analytic Systems for Real-Time Decision-Making in Mission Control

Matthew Barry
Propulsion Systems Section
NASA/Johnson Space Center DF63
Houston, TX 77058

Eric Horvitz (PI), Corinne Ruokangas, Sampath Srinivas
Information and Decision Sciences
Rockwell International Science Center
444 High Street
Palo Alto, CA 94301

Abstract

The Vista project has centered on the use of decision-theoretic approaches for managing the display of critical information relevant to real-time operations decisions. The Vista-I project originally developed a prototype of these approaches for managing flight control displays in the Space Shuttle Mission Control Center (MCC). The follow-on Vista-II project integrated these approaches in a workstation program which currently is being certified for use in the MCC. To our knowledge, this will be the first application of automated decision-theoretic reasoning techniques for real-time spacecraft operations.

We shall describe the development and capabilities of the Vista-II system, and provide an overview of the use of decision-theoretic reasoning techniques to the problems of managing the complexity of flight controller displays. We discuss the relevance of the Vista techniques within the MCC decision-making environment, focusing on the problems of detecting and diagnosing spacecraft electromechanical subsystem component failures with limited information, and the problem of determining what control actions should be taken in high-stakes, time-critical situations in response to a diagnosis performed under uncertainty. Finally, we shall outline our current research directions for follow-on projects.

1 Introduction

The Vista project is a collaborative research and development effort between the Palo Alto Laboratory of the Rockwell Science Center, the Rockwell Space Operations Company, and NASA/Johnson Space Center to develop techniques for reducing the cognitive load on operators responsible for monitoring and controlling complex physical systems. In particular, the project has centered on the use of decision-theoretic approaches for generating diagnostic assistance and for directing computer programs to display the most relevant information in a decision context.

Last year, we developed and demonstrated

a prototype Vista-I decision-support and display-management system for Space Shuttle Orbital Maneuvering System (OMS) burn monitoring and control activities. This prototype system provides propulsion system flight controllers with diagnostic decision support by reasoning under uncertainty about alternative problems, and by prioritizing them according to probability and criticality [1].

This Vista-I prototype stimulated efforts to continue this work by extending the reasoning models and porting the techniques to MCC-class workstations, culminating with certification of the software for mission operations. To accomplish these efforts, the Vista team this year developed the Vista-II system. This system improves the Vista-I uncertainty models, supplements them with utility models, and captures the prototyped display-management features and techniques within an X-windows-based workstation program connected to the MCC telemetry data streams. The resulting program currently is undergoing final development and verification and validation testing prior to certification.

2 Description

The proper management of uncertainty in decision-making is critically important in high-risk operations endeavors like manned space flight. The Space Shuttle OMS performs many critical maneuvers (commonly called *burns*) during every mission, including orbit insertion and deorbit, rendezvous target phasing and orbital plane adjustments, deployed-satellite and collision-avoidance separations, and contingency propellant dumps. Therefore, it is vitally important that correct OMS diagnoses and operations decisions be made promptly when subsystem faults occur during these maneuvers.

The set of possible faults is known *a priori*, as are the valid responses to any combination of these failures. Since the OMS subsystem is well-transduced, the fault detection and diagnosis tasks are rather straightforward for an experienced flight controller; a less-experienced flight controller, however, may have a bit more uncertainty about fault signatures and correct response actions. However, any flight controller faces significantly more difficult decision-making tasks when a prior failure of the spacecraft instrumentation or data processing subsystems has rendered many of the primary OMS sensors inoperative. Our program-embedded uncertainty models handle this often-encountered situation by using whatever information is available in the current situation, including secondary sensors and prior probabilities. Moreover, prior problems within the OMS subsystem may increase the difficulty of diagnosing multiple faults; the uncertainty models handle these situations in an elegant manner because they calculate the probability distribution over all faults.

In Vista applications, we use uncertainty models to calculate the probability distributions over the set of possible faults based on observed sensor data. We use these probability distributions in conjunction with utility models to determine which course of action to recommend. Both of these models affect the automated selection of adaptive displays which the program provides to flight controllers for making the final diagnosis and response decisions. Sections 2.1 and 2.2 describe the uncertainty and utility models, respectively, and section 3 describes the displays and display-management techniques we've built into the Vista-II system.

2.1 Uncertainty Model

Automated reasoning systems often require representations of uncertainty about the world. These models often employ Bayesian inferencing techniques to calculate condi-

tional probabilities over a collection of hypotheses given some evidence. They are especially applicable to fault detection and diagnosis problem domains in which multiple faults may occur or in which only a limited amount of evidence is available. Vista systems employ these models within larger decision-theoretic models representing uncertainty and utility in decision-making processes. In Vista systems we apply these uncertainty models to the usual problems of fault detection and diagnosis, but we also apply them to the problem of automatically controlling the presentation of information to the user given uncertainty about the world.

Vista systems use *belief network* models to calculate the probability distributions over a set of possible faults for the OMS rocket engines and their associated propellant distribution systems and sensors. Belief networks are computational models which represent probabilistic influences among observations (evidence) and possible explanations for these observations (conclusions or diagnoses).[1] In the OMS burn monitoring and control program specifically, we use belief networks to represent the probabilistic influences among telemetered readings from OMS pressure, temperature, quantity, and valve position sensors against a collection of possible faults or explanations which best describe these observations. Figure 1 depicts a compact representation of the OMS burn network.

Each belief network node contains conditional probabilities based on the conditional probabilities of its ancestors. We enter observations from the world as certain evidence in certain leaf nodes. The inference engine propagates this evidence, using Bayes' Rule, to all of the other nodes in the network. Extracting the resulting values of features within designated fault nodes we obtain the conditional probability distribution for given exhaustive set of faults. The program uses this fault probability distribution to update and manage displays and as input into the utility model.

[1]These belief networks, sometimes referred to as *causal probability networks*, are special forms of more general *influence diagrams* [2].

2.2 Utility Model

For automated decisions about the best action to take under uncertainty, it is important to employ a representation of the value of alternative outcomes. Having access to the values of alternative outcomes allows for the selection of fault-response actions that have the highest expected utility. In the Vista-II system we employ a *utility model* to calculate the value of alternative outcomes based on the fault probability distribution. We display the distribution of these values over all of the alternative actions and assume that the flight controller will select the action with the maximum expected utility. Section 3 describes these displays.

The Vista-II utility model determines the value of alternative outcomes by calculating the scalar product of the fault probability distribution vector with an action-specific, utility-weighting parameter vector. We have experimented with various sets of weighting parameters. The set currently in place reflects a single-attribute model which describes the "right response" or "gut feeling" gleaned from experienced flight controllers. Essentially, these parameters reflect the utility of selecting action A in response to each possible fault F. We have also constructed more specific multi-attribute utility models which can provide the weighting vector elements by performing a linear combination of decision attributes. These decision attributes include measures such as the importance of achieving maneuver targets (based on criticality), the risk of damage to spacecraft subsystem components, the per-

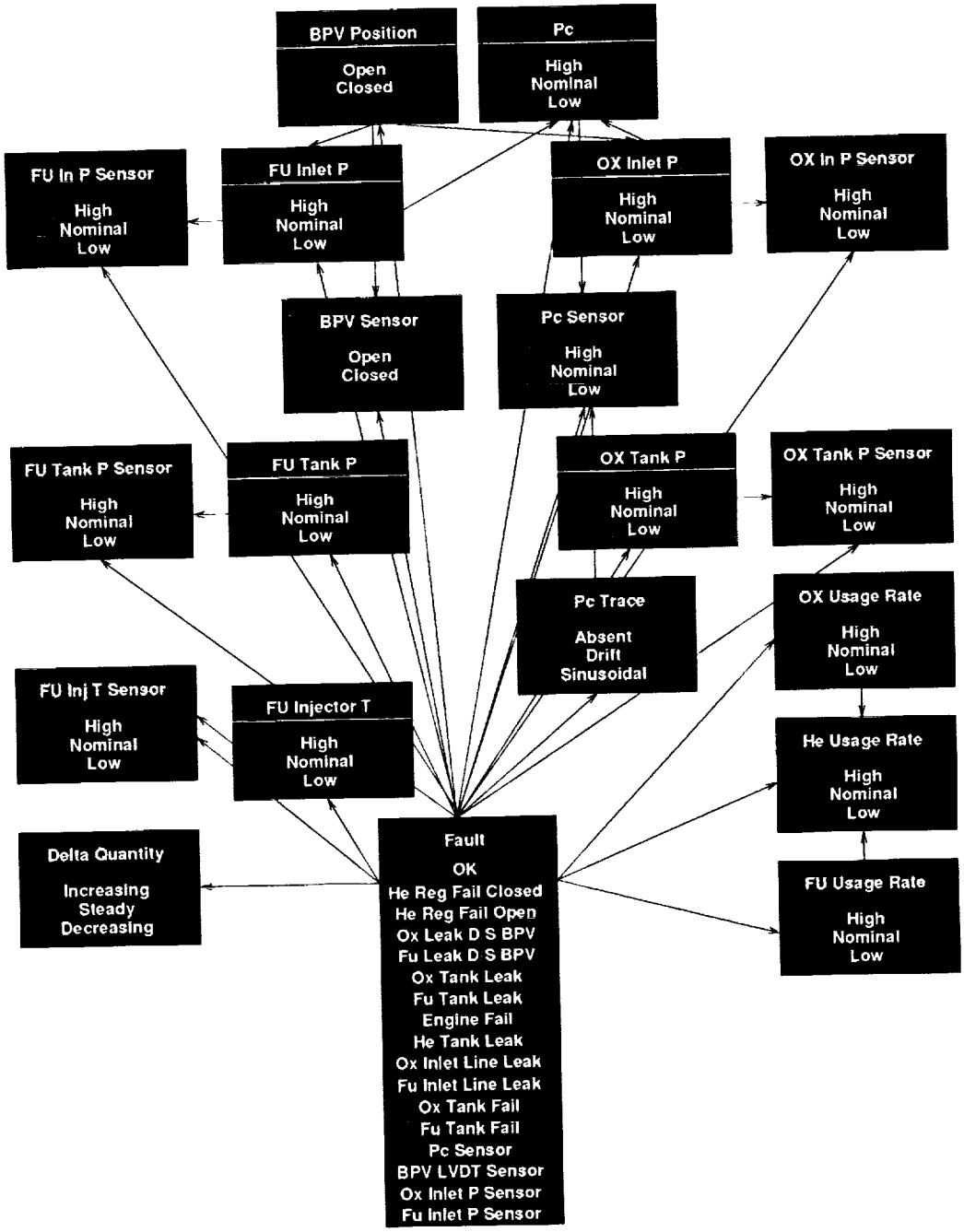


Figure 1: The belief network for OMS burn monitoring. Arcs represent probabilistic influences between the nodes. Grayed titles denote evidence nodes.

formance capabilities available from backup systems, and the potential impact to mission objectives. These multi-attribute utility models will be particularly important in distributed decision-making applications because they provide a way to account for disparate degrees of contribution from independent subsystems toward common decision attributes.

The model we have implemented in the Vista-II system provides the flight controller with a utility value distribution over four possible actions. These actions correspond to doing nothing ("continue"), terminating the burn ("stop"), or selecting a backup burn configuration ("engine-fail downmode" and "propellant-fail downmode"). Since the expected utility of executing these actions in response to a fault is context-dependent, the utility model employs a different set of weighting parameters for each user-selected context. Section 3 describes the mechanism by which the user can select the context. As the fault probability distribution changes according to the uncertainty model, the utility model changes the distribution over these possible actions and the program shows this distribution on the displays.

3 Implementation

The Vista-II application has been realized in a working program on MCC-class workstations. These workstations run the Unix operating system and the X-Windows System, and use the OSF/Motif window manager. The OMS burn monitoring program was written in the C language using the OSF/Motif programming style and widget set and the Hugin API inference engine for the belief networks.[2] Owing to our lack of access to a commercial product performing utility modeling, we have coded the utility models by hand. In this section we describe some of the implementation techniques, display-management philosophies, and design details found in this program.

First, since the two OMS engines are functionally identical, but provide unique sets of sensor values, we use a copy of the belief network for each engine and change the engine-specific sensor value evidence nodes according to the appropriate sensor names and locations. The belief network developer assigns to each fault given in the "fault" node an associated "group" name, which we use to collect related faults into named groups in order to summarize these faults on a smaller display. The OMS burn monitoring program loads these two belief networks at run time. Once loaded and initialized, the program constructs some of the Vista displays automatically based on the contents of the designated "fault" node in the network. The program then cyclically gathers telemetered sensor values, translates analog values into qualitative values (such as `low`, `nominal`, or `high`), then installs these qualitative values as certain evidence in the sensor nodes. If the value of any evidence node has changed since the last data cycle, the program runs the belief network inference engine to compute the probability distributions over all of the possible values in each of the other nodes. The program then uses these new probability distributions to update and select the appropriate displays.

Next, we draw a distinction between two sorts of displays built into this program: *fixed* displays and *adaptive* displays. The fixed displays essentially are conventional flight controller displays showing spacecraft subsystem configurations, current sensor values, internal

[2]Unix is a trademark of AT&T. The X-Windows System is a trademark of Massachusetts Institute of Technology. OSF/Motif is a trademark of Open Software Foundation, Inc. Hugin API is a trademark of Hugin Expert A/S.

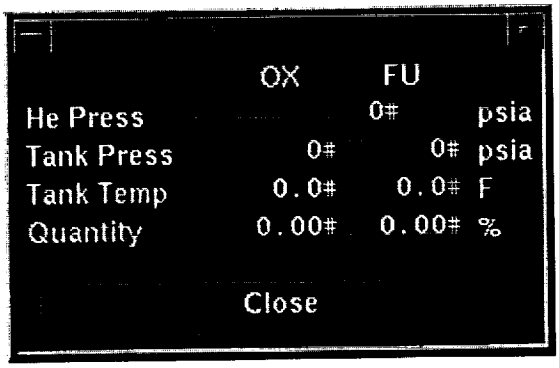


Figure 2: Left OMS summary display. Only a small sampling of information about the Left OMS is available to the user from this display.

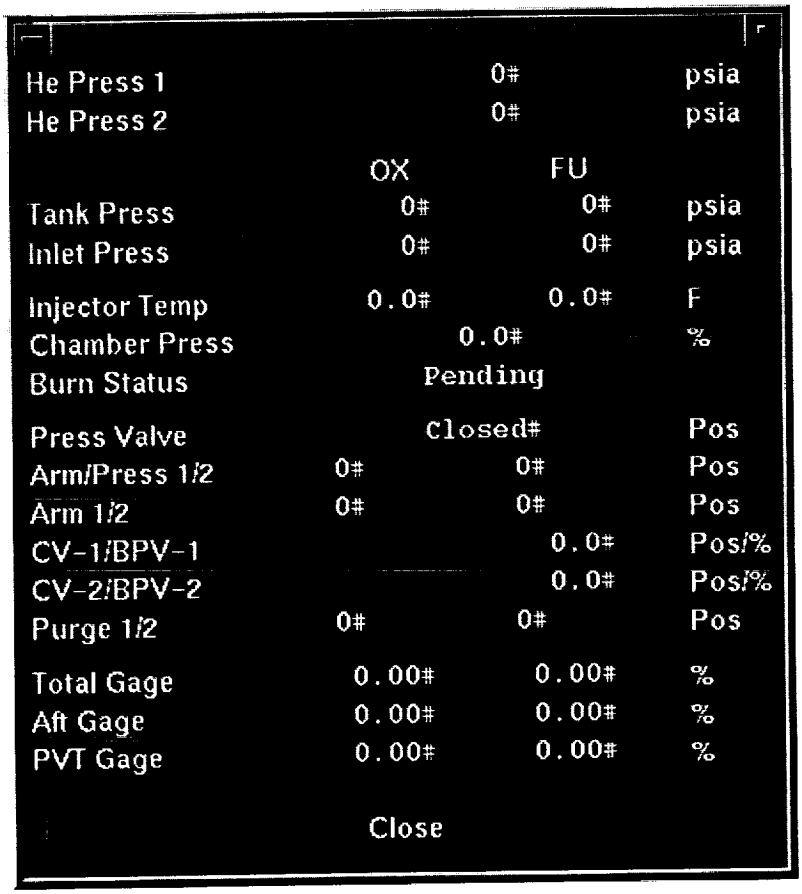


Figure 3: Left OMS detailed display. All of the Left OMS sensor and calculation data is available to the user from this display.

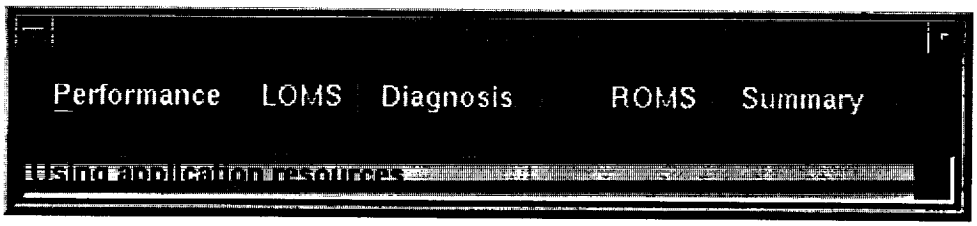


Figure 4: OMS burn program palette menu. The user may select any of the program's displays from this menu, thereby overriding automatic controls over the presented displays. This menu contains pulldown menus for "n-of-many" selections and option menus for "one-of-many" (mutually exclusive) displays.

computation results, mission status information, and so on. These displays are "fixed" because they're compiled into the program. Some of the fixed displays pertain to various levels of "granularity" or detail; these range from showing an overview sampling or summary of important information, to showing every bit of detailed information. In order to manage the "real estate" on the screen, and thereby manage the cognitive load on the user, the program employs the Vista models to select which degree of detail is suitable for display: it chooses the summary displays when there isn't much of interest in the current decision context from one series of displays, but chooses the detailed displays when crucial information from these displays is necessary to make the best-informed decision. If necessary, the program will "shrink" the irrelevant displays and "enlarge" the relevant displays by selecting among the fixed displays in each series. Of course, there may be some information overlap in each level of granularity. Figures 2 and 3 show a summary and detailed display for the Left OMS subsystem. Since we allow the user to override any of these automatic display selections, the program also provides a "palette" menu from which to select any of the displays made available by the program. Figure 4 shows the palette for the OMS burn program.

The adaptive displays provide the users insight into the probability and utility distributions calculated by the inference engine. These displays are "adaptive" in the sense that the program builds them automatically, based on external information, so that various configurations of the displays may be used for different stages of development or by different users. Specifically, the program constructs these displays from information contained within the belief networks; since there are a pair of belief networks for any complete OMS burn model, the program actually builds two sets of displays. First, the program builds a "detailed" diagnosis display which lists all of the possible faults provided by the model. We use a histogram representation to convey the probability distribution over these faults; initially, the distribution corresponds to the *a priori* probabilities of occurrence. Second, the program builds a "summary" diagnosis display which lists all of the fault groups encountered in the fault list. It is assumed that each possible fault is a member of one and only one fault group. Again, we use a histogram representation to convey the probability distribution over the fault groups. As the program acquires and processes telemetry data, the inference engine will determine new probability distributions which the program will present to the user by changing the magnitudes of the appropriate graph elements. Figures 5 and 6 show examples of these displays. These two displays represent the "granularity" offered into

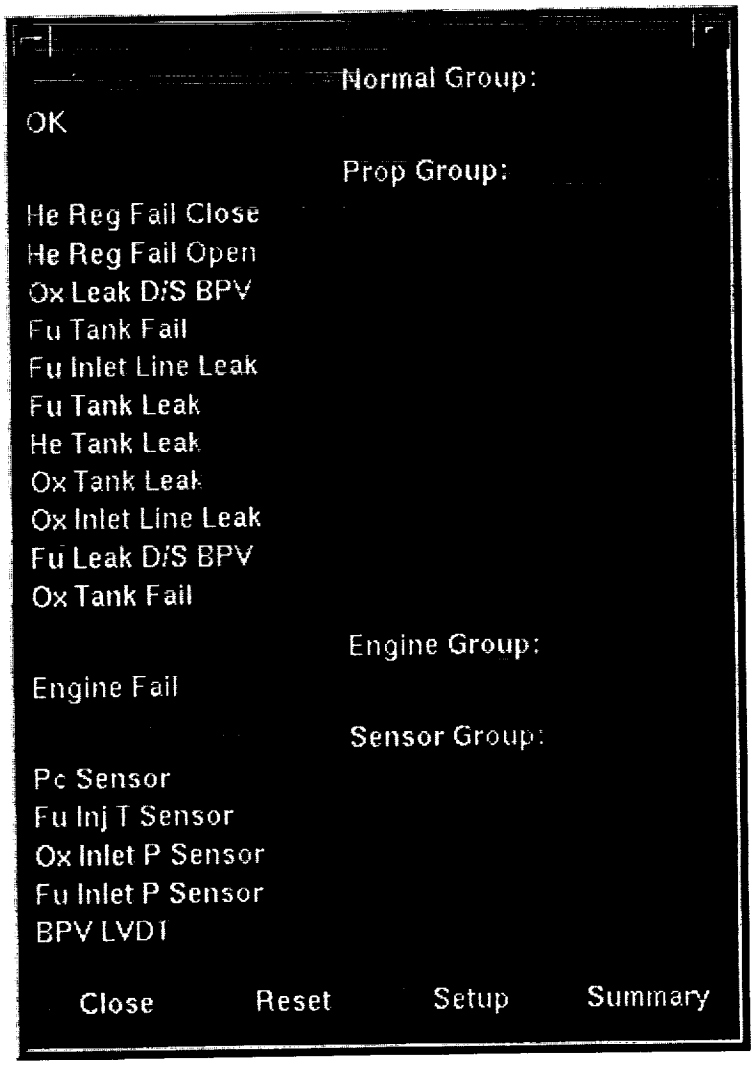


Figure 5: A "detailed" diagnosis display. Each entry in the histogram represents the relative probability for the named fault.

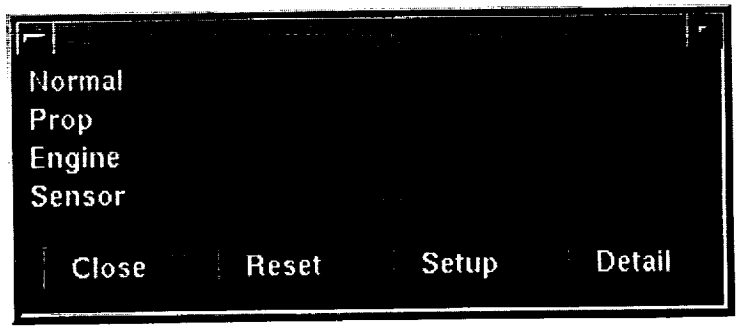


Figure 6: A "summary" diagnosis display. Each entry in the histogram represents the summation of the probabilities for all faults in the named group.

the diagnosis information. Since the summary diagnosis display consumes less screen space than the detailed diagnosis display, it is meant to be used as the primary diagnosis display when the probability of any fault is low. The program will automatically replace the summary display with the detailed display when the probability of any fault exceeds some threshold. We shall describe below a built-in feature which enables the user to adjust this threshold. To override these automatic controls, the displays also provide convenient push-buttons to increase or decrease granularity. There is also a push-button to invoke the "setup" dialog, which we describe below.

Another adaptive display is the *action-selection* display. Since the belief networks do not contain information for the utility models, the program builds this display based on information contained in a user-controlled file. This file contains certain actions and utility model parameters necessary to build the display. Once again, there is one action-selection display for each OMS. Figure 7 shows the action-selection display for the OMS burn monitoring program. Since the number of actions is small in this application, there is only one level of granularity among the action-selection displays.

The "setup" dialog box provides the user with some control over the behavior of the inference and display-management functions (see figure 8). The three option menus provide the user with a mechanism for selecting the context of the OMS burn, such as whether the burn is critical, whether a minimum burn target must be satisfied, and what performance capabilities remain in redundant systems in the event of a failure of the primary system. The configuration of these menus affects the parameters used by the utility model. The "auto-display threshold" slider bar enables the user to select the fault probability value above which the program will automatically present the detailed diagnosis display (for all faults other than "ok"). The "auto-freeze threshold" slider bar enables the user to select the fault probability above which the program will cease to update the probability and utility distributions and displays. This feature disables updates to faults which manifest themselves in a dynamic fashion, presenting evidence convincingly initially (with high probability), then appearing to change into a different signature. Since the initial signature best represents the real problem, we may choose to disable further calculations after exceeding a certain confidence threshold.

Finally, adopting the Vista philosophy on screen real-estate management, the OMS burn program can control the placement of most of these displays automatically. For example, the program will place the Left and Right OMS summary and detailed displays adjacent to each other if a companion display is already visible on the screen. It will also substitute the mutually exclusive displays at the same screen location. These automatic placements override the window manager's controls over window placement. If a companion display is not visible, the program will defer placement to the window manager, which then employs the user's default geometry settings or interactive placement resources. These automatic controls provide convenient display-management techniques which minimize distraction of the user during crucial decision-making contexts.

The OMS burn belief network and utility models capture a tremendous amount of flight controller expertise. The belief networks were developed in direct consultation with flight controllers, and accurately represent the probabilistic reasoning performed by these flight controllers during real-time MCC operations. The *a priori* probabilities for the uncertainty models and the utility parameters for the utility models were derived from

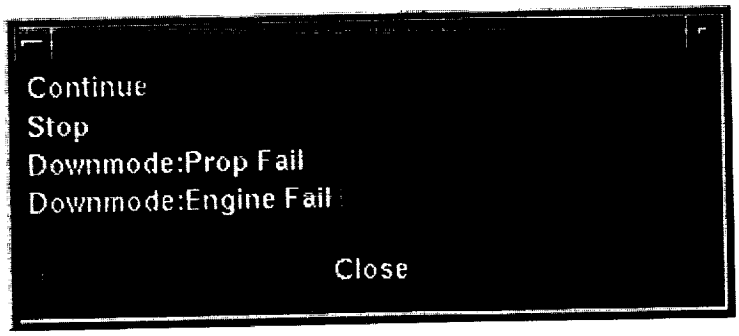


Figure 7: An action-selection display. Each entry in the histogram represents the relative utility for the named action in the current burn context.

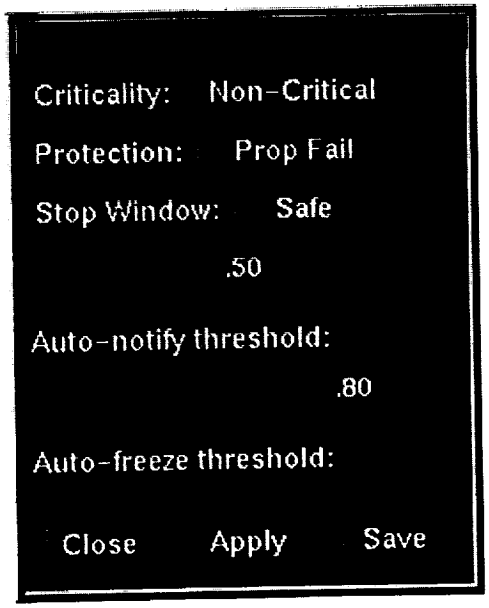


Figure 8: The OMS burn program "setup" dialog box. Slider bars enable the user to set thresholds for display-management functions. Option menus enable the user to establish the burn context for the utility model.

the results of surveys of all of the flight controllers responsible for OMS burn monitoring. We have found that these model parameters have worked extremely well during rigorous tests of this new program.

4 Future Work

The Vista-I and Vista-II systems have been very successful, particularly in demonstrating the usefulness of these decision-theoretic approaches to decision-making and display-management in real-time operations. These successes have generated many interesting ideas we intend to pursue as we enhance the models and reasoning techniques. Many of these ideas will be pursued during next year's Vista-III project.

Using collaborating Vista models, we are experimenting with a distributed expert system approach to group decision-making applications. Using the *information sharing protocol* developed at JSC [4], we distribute the probability and utility distribution results from various Vista models across a network to other flight controllers whose systems may be affected by the operations of another system. Such a multi-agent application is especially useful for prioritizing a serial list of actions to be forwarded to the astronauts. This approach is also interesting for the deployment of adaptive multi-attribute utility models.

We are also experimenting with the integration of empirical *sensor importance measurements* derived by the *selective monitoring* (SELMON) project at the NASA Jet Propulsion Laboratory (JPL) [5]. These measurements often provide additional intuitive representations of sensor observations as evidence for the sensor nodes in the belief networks, particularly when the dynamic behavior of a sensor is important information.

A focus on sensor importance can also be made from a strictly probabilistic or statistical standpoint. One interesting application of these techniques lies in determining the diminished confidence in the latest sensor reading over time. Another similar application can determine the information content of a particular display, enabling the program to suggest a fixation on that display if it isn't currently visible.

Finally, we are developing new implementation techniques to facilitate the integration of uncertainty models within workstation programs. These implementation techniques include display-management protocols interacting with the window manager, new frameworks of interacting objects to facilitate display construction, and possibly new X-compatible widgets which hide all of these implementation details from the programmer.

References

[1] Horvitz, Ruokangas, Srinivas, and Barry. *A Decision-Theoretic Approach to the Display of Information for Time-Critical Decisions: The Vista Project.* Proceedings of Fourth Annual Workshop on Space Operations Applications and Research (SOAR '92), NASA/Johnson Space Center, Houston, TX, 1992.

[2] Howard and Matheson. "Influence Diagrams," in *The Principles and Applications of Decision Analysis.* Howard and Matheson (eds.), Strategic Decisions Group, Menlo Park, CA, 1981.

[3] Andersen, Olesen, Jensen, and Jensen. *Hugin - A Shell for Building Bayesian Belief Universes for Expert Systems.* Proceedings of the Eleventh International Congress on Artificial Intelligence, 1989.

[4] Barry, Scott and Weismuller. *A Distributed Computing Model for Telemetry*

Data Processing. Submitted to the Goddard Conference on Space Applications of Artificial Intelligence, NASA/Goddard Space Flight Center, Greenbelt, MD, 1994.

[5] Doyle, Chien, Fayyad, and Wyatt. *Focused Real-Time Systems Monitoring Based on Multiple Anomaly Models.* Seventh International Workshop on Qualitative Reasoning, Eastsound, WA, 1993.

N94- 35064

Mission Evaluation Room Intelligent Diagnostic and Analysis System (MIDAS)

Author: Ginger L. Pack/ER221

Position/Title: MIDAS Project Manager

Affiliation: NASA (ER22)
Address: Lyndon B. Johnson Space Center
Houston, TX 77058

Phone: (713) 483-1515
FAX : (713) 483-3204

Co-Authors:
1. Jane Falgout
2. Joseph Barcio
3. Steve Shnurer
4. David Wadsworth
5. Louis Flores

Affiliation: Lockheed Engineering & Sciences Corporation

Address: ESPO C19
PO Box 58561

Phone : (713) 483-2056

ABSTRACT

The role of Mission Evaluation Room (MER) engineers is to provide engineering support during Space Shuttle missions, for Space Shuttle systems. These engineers are concerned with ensuring that the systems for which they are responsible function reliably, and as intended. The MER is a central facility from which engineers may work, in fulfilling this obligation. Engineers participate in real-time monitoring of shuttle telemetry data and provide a variety of analyses associated with the operation of the shuttle.

The Johnson Space Center's Automation and Robotics Division is working to transfer advances in intelligent systems technology to NASA's operational environment. Specifically, the MER Intelligent Diagnostic and Analysis System, (MIDAS) project provides MER engineers with software to assist them with monitoring, filtering and analyzing Shuttle telemetry data, during and after Shuttle missions. MIDAS off-loads to computers and software, the tasks of data gathering, filtering, and analysis, and provides the engineers with information which is in a more concise and usable form needed to support decision making and engineering evaluation. Engineers are then able to concentrate on more difficult problems as they arise.

This paper describes some, but not all of the applications that have been developed for MER engineers, under the MIDAS Project. The sampling described herewith was selected to show the range of tasks that engineers must perform for mission support, and to show the various levels of automation that have been applied to assist their efforts.

INTRODUCTION

The purpose of the MIDAS project is to provide MER engineers with real time intelligent software to assist them with monitoring, filtering and analyzing Shuttle telemetry data, during and after Shuttle missions, and to capture the expertise held by highly experienced engineers. This is accomplished by applying advanced automation and intelligent systems to the tasks required to perform engineering evaluation.

The MER engineers perform a variety of task in support of Shuttle missions. They continuously monitor and evaluate the performance and health of their systems, provide engineering analysis

support to Mission Operations personnel, and are responsible for certifying the integrity of orbiter systems for subsequent flights. Historically, engineers have had to rely upon several screens full of telemetry data, displayed as ASCII text and numerical values, to understand the state and operating condition of their system. Information in this form is crammed onto displays that engineers must monitor and interpret, (Figure 1). Several of these displays exist for each subsystem. Engineers visually monitor these displays during missions, to obtain an understanding of the behavior of their systems. Only one display at a time may be shown. This arrangement precludes the engineer from noticing changes in information that are not contained on the display being viewed. The selection of screens available for viewing is controlled by Mission Operations personnel. Consequently, during missions engineers must often "chase around" the displays which contain the information in which they are interested. This further complicates the duties that engineers must perform, in maintaining the health of the systems for which they are responsible. Additionally, engineers use manual methods to perform analysis. Events are logged by hand, and associated analysis is performed using pencil and paper. This mode of operation can be tedious, time consuming, and vulnerable to the errors that result from distraction and boredom.

TECHNICAL APPROACH

The first task to be done in automating the MER, was to analyze the current methods of operating, and identify problems and deficiencies. A questionnaire was developed to learn where engineers were spending their time, which tasks were labor intensive, boring, repetitive, and just hard to do within current processes and methods. Discussion of these issues involved knowledge engineering to capture system expertise, and determine the kinds of analysis with which engineers needed help. It was learned that engineers wanted assistance in recognizing trends that were reflected in the data, and evaluating the potential impact on hardware. They also needed to capture the data and perform identification and diagnosis of problems before a critical state is reached. Additionally, engineers wanted help in capturing and analyzing in-flight data to check out and verify the health of the systems for future flights. Finally, they needed relief from the task of visually monitoring and manually gathering the data needed to perform various analyses. Computers and software could provide for these needs, and enable the engineers to focus their efforts on problems as they arise versus investing so much effort gathering and sifting through large amounts of data to evaluate system health and performance. Using the questionnaire as a guide, developers work with engineers to identify troublesome areas and to reach agreement on which tasks are to be built into software.

Automation requirements are developed through close interaction with Shuttle subsystem engineers. User feedback is solicited and incorporated at every phase of product definition and development. Developers work closely with users while developing the architectural design, and while the implementing the design. Frequent product reviews are also conducted with the user to ensure appropriateness and accuracy of results. User involvement in all phases of development results in products that are familiar to users and fit well to the user's needs. It also ensures user "buy in" to the technology, since they help to define the implementation of the technological solution.

ACCOMPLISHMENTS

Discrete Log

The Discrete Monitor and Logging program (figure 2), was created to relieve engineers of the continuous eyes of monitoring on system data. Previously, this task was done by having a person watch a screen full of data (figure 1), noting the occurrence of certain events, and writing them down on a sheet of paper. The Discrete Monitor and Logging program reads the shuttle telemetry stream and automatically detects changes in the state of discrete parameters. The changes are

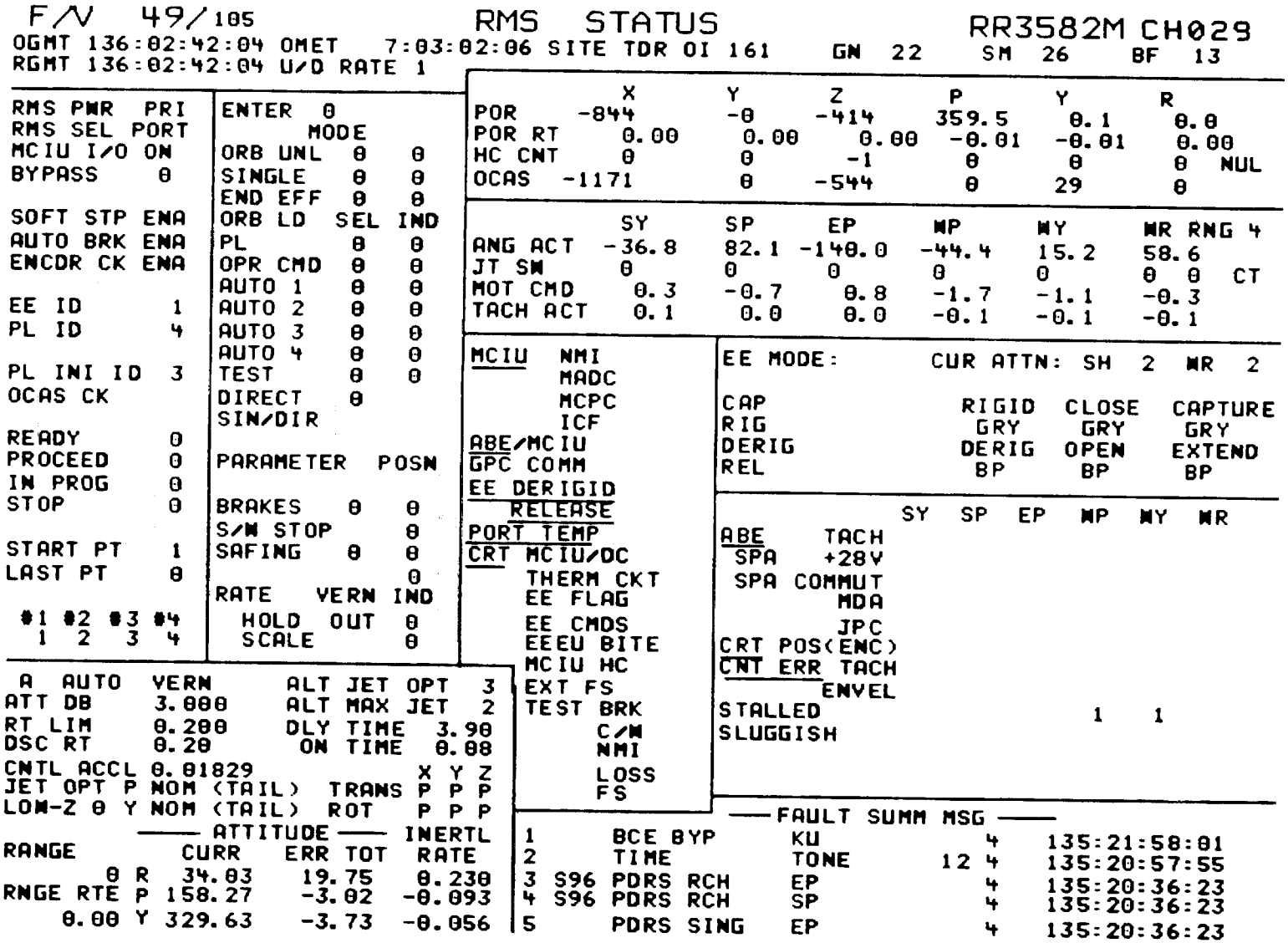


Figure 1: Data Display

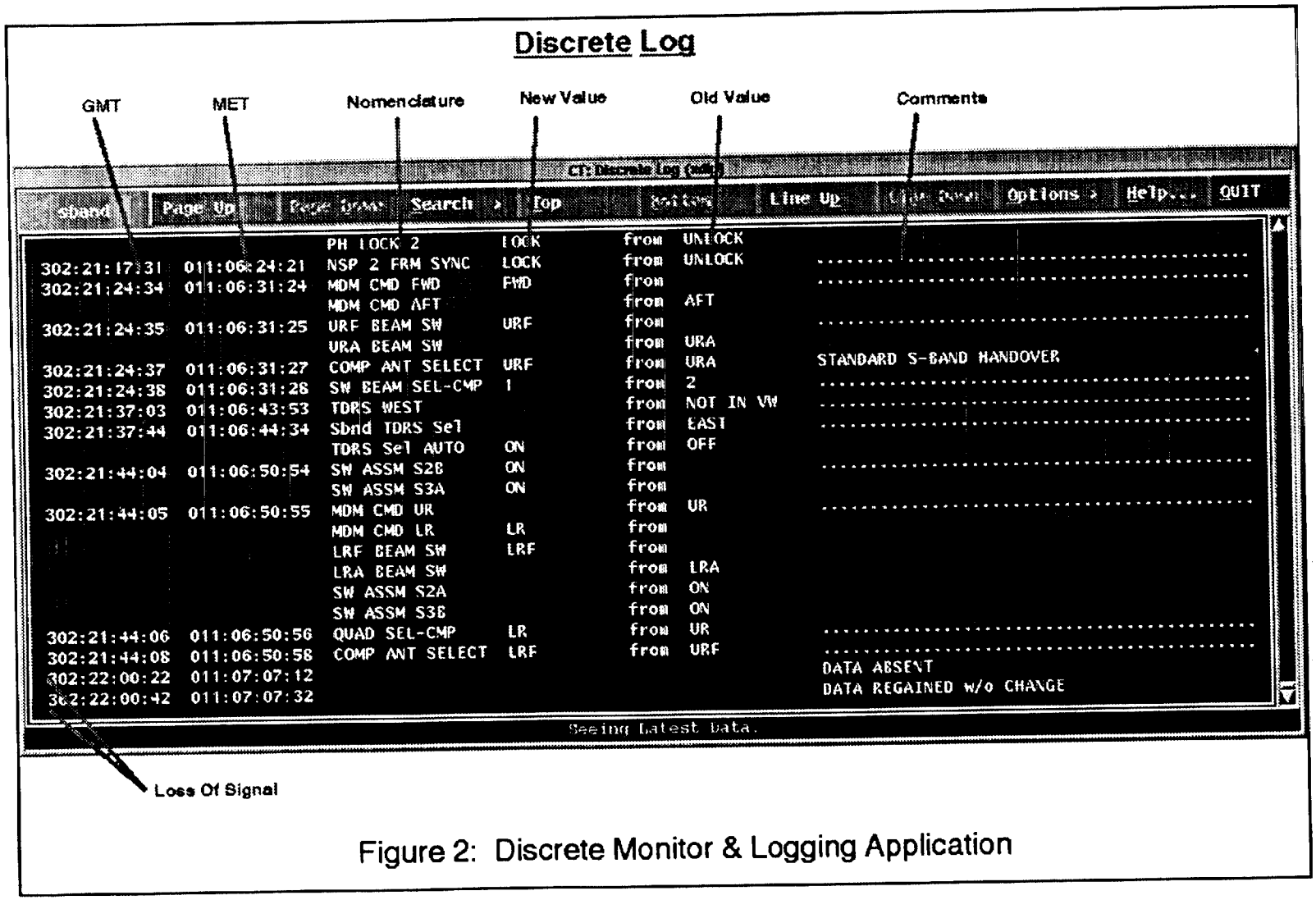


Figure 2: Discrete Monitor & Logging Application

logged and written to the screen, and to a file that may be translated to ASCII format for further manipulation by the user. The entries consist of a description of the changed event, the Greenwich Mean Time (GET) and Mission Elapsed Time (MET) of the change, the previous state, the current state, and a comment field. The event descriptor and state changes are entered in English. Comments may be entered from a template or edited by the user. The log entries are painted yellow when data has changed during a period of signal loss. The first version of the log, logged each change without regard to how long or how often the parameter had changed or was changing. Consequently, parameters that "fluttered" on and off quickly filled up the log. This was taken care of by adding a module that would wait for the data to stabilize before reporting that the data had changed.

Analog Plot

A companion application to the Discrete Monitor and Logging program is the Analog Plot program (figure 3). The Analog Plot program graphically depicts system behavior that is described by analog values. The plots appear as real time strip charts but are defined by conditional logic among the analog values. Thirty minutes of data are shown. The plots are scrolled every ten minutes. Plots are organized in "families" that are made up of pages. These pages can be combined in mix and match fashion to facilitate visual comparison of data. A "snap shot" of interesting data can also be saved and re-loaded for comparison against other data.

Ku-band Automated Self-test Analyst

The Ku-band Automated Self-Test Analyst (KASTA) is an expert system that detects the start of the Ku-band self-test, and interprets the test results. The self test is done prior to deployment of the shuttle's Ku-band antenna. The purpose of the self-test is to ensure that the Ku-band antenna is healthy and will operate correctly when used. The self-test consist of several sub-tests whose outcome depends on the success or failure of previous tests. The pass/fail condition of different combinations determine the success or failure of the entire Ku-band self-test, and consequently whether the antenna is fit for use. The expertise for analyzing this test was held by the Ku-band subsystem manager, requiring her to be available for every Ku-band self-test. KASTA allows non-expert personnel to evaluate the self-test.

KASTA evaluates the tests that determine the ability of the Ku-band antenna to achieve proper position within allowed time, verifies the ability of the antenna to track properly, accounts for and manages exceptions to self-test procedure, and recognizes non-significant "failures". KASTA uses telemetry to identify initiation of the self-test, and begins to monitor each test according to pass/fail criteria contained in the knowledge base. Telemetry values are evaluated against rules that determine whether the conditions of the tasks are met, and explanation is provided to support conclusions made by KASTA.

Considerable thought was given to how this information should be conveyed to the user. The application interface provides feedback to the user regarding which task or subtask is in progress, and where in the self test procedure the task falls. Other types of information also had to be communicated through the user interface. The status of each task, task outcome, and the data associated with each task, such as task duration, and parameters checked, had to be presented to the user. Additionally, details concerning the failure of any task to complete had to be communicated. Finally, explanation and supporting data for KASTA's decisions, had to be made available.

The objectives of the display were to strive for easily comprehensible presentation of the various kinds of information, and to provide a highly intuitive display in terms of comprehension of data and in terms of human-computer interaction. These goals were met by grouping like information, arranging information in table format, and using color only to convey specific information. Also, groups of information were placed into windows and labeled using names that clearly described the

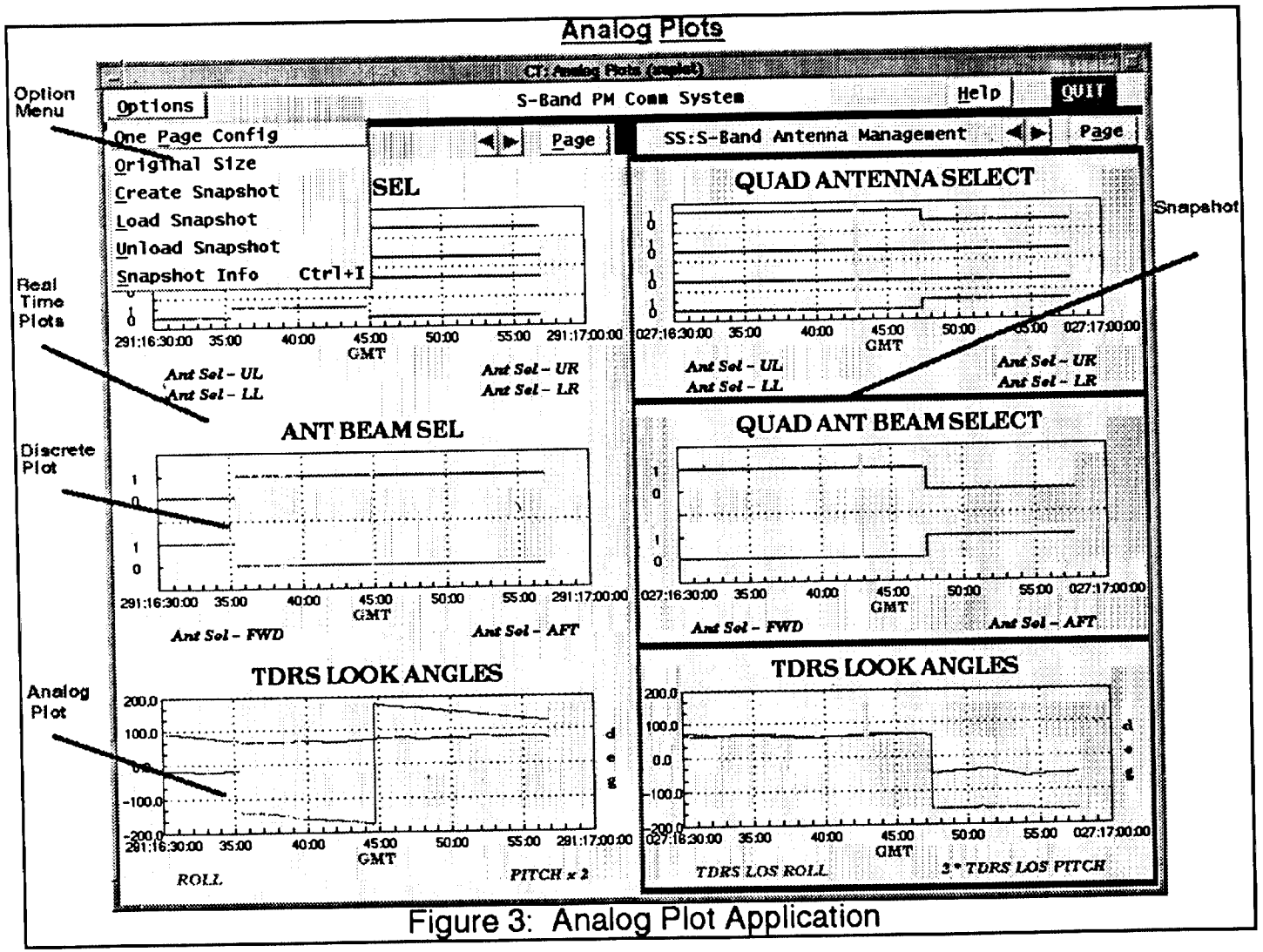


Figure 3: Analog Plot Application

content of the window. The result was a well-organized and highly intuitive display that users could understand and interact with, while having little or no formal instruction (figure 4).

The most important information was placed in the most prominent position of the display area, the center. Supporting data was arranged around the center. The center of the display contains the self test tasks and detailed information pertaining to the tasks. The SELF TEST TASKS window, lists each of the tasks that comprise the Ku-band self-test. The task currently being executed is highlighted in this window. A red or green box indicates whether the task passed (green), or failed (red).

The REAL TIME DATA window is located to the left of the display center, and lists the telemetry and corresponding values that are associated with the self-test. Telemetry corresponding to a task that is in progress, is highlighted while the task proceeds. To the right of the center the SELF TEST STATUS window contains information concerning which task is in progress, time elapsed for the task, start time for the task, and mission time given as Greenwich Mean Time (GMT) and Mission Elapsed Time (MET). Below this window, is a MESSAGES window that allows KASTA to inform the user which tasks have failed.

Users may also review all of the data that KASTA used to determine the outcome of any task. After the self test procedure has completed, the data that was used to determine the outcome of any task is accessible in two ways. The user may review only the data important to a particular task, or review all of the data important to the entire self test procedure.

KASTA performs its analysis by cycling through five distinct phases for every second of telemetry data. The first two phases check the incoming data to prevent KASTA from making decisions using unreliable data. An important benefit of this approach is that KASTA's knowledge base always operates on a time-homogeneous set of data. No analysis rules are allowed to fire until the phase in which data acquisition takes place is finished. This promotes consistency for KASTA's analysis. The five analysis phases are data acquisition, post data acquisition, self test progress update, task progress update, and task outcome analysis.

KASTA's data acquisition strategy first checks the quality of the data before passing the data on. An overall quality indicator determines how well the frame count transmission was received, and is associated with each second's worth of data. If the quality indicator is not 100% that second of data will not be used. An individual quality status indicator is associated with each telemetry value.

During the post data acquisition phase, the telemetry is submitted to more rigorous quality checks. Conditions checked include recency of data, degraded data quality, and bad status associated with the quality indicator or the timestamps, or questionable timestamps associated with the data. If any of these conditions are met for the frame of data (one second of data), none of the remaining phases will be executed. Conversely, if the data passes screening, the next phase of KASTA's analysis is entered. This phase is referred to as the "self test progress phase", and contains the rules that determine when the self test starts and completes. Also included are the rules that determine at any time, how far into the self test procedure, the current analysis is. The third phase of analysis determines how a specific task being executed is progressing. This is the "self test progress update". This phase contains the rules that determine the identity of the current task, and how far into that task the analysis is, at a given time. The outcomes of the tasks are determined during the "task outcome analysis phase". This set of rules specify the pass/fail criteria for the self test, and summarizes the results of all the self test tasks and sub-tasks. KASTA cycles through these phases as long as the self test is in progress. When the self test has completed, the data acquisition and post data acquisition phases will continue to be executed to provide updates to the REAL TIME DATA window.

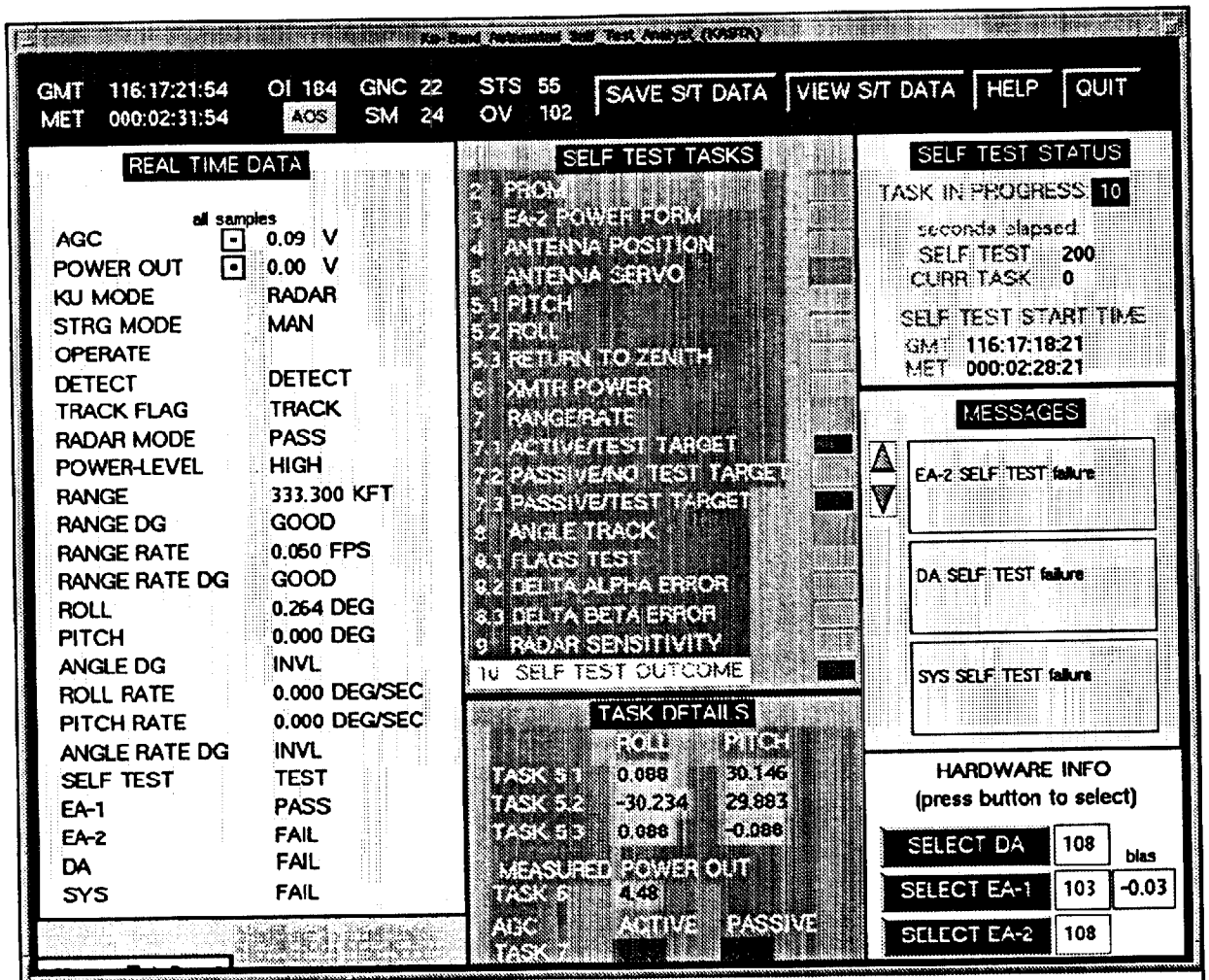


Figure 4: Ku-Band Automated Self Test Analyst (KASTA)

KASTA is written in Gensysms G2 expert system shell. The logic used to analyze the self test in real-time is implemented using a combination of rules, objects, and G2 functions. KASTA is used by MER engineers and, the Integrated Communications Officers (INCOs), who are responsible for Shuttle mission support.

RMS Monitor

Since April, 1992, the Remote Manipulator System (RMS) Direct Drive Test (DDT) Monitor has supported missions whenever a flight includes use of the Shuttle robotic arm. The DDT Monitor (figure 5), was constructed to monitor the direct drive test of the RMS joint motors. The six joints of the Shuttle robotic arm are moved by six motors that are individually tested before the arm is deployed. The motors are tested by driving each of them in the forward (positive) and reverse (negative) position. The DDT Monitor automatically detects the initiation of the direct drive test, and plots the results on a screen, as the test occurs. The plot is a visual image of the motor tachometer output, and should produce an expected profile when the motor is operating properly. A warning is issued if the motors are driven beyond 20 seconds a time limit that, if exceeded, invalidates the test. However the application continues to plot the profile. The profile can be saved to an historical file, then retrieved and overlaid against current or other historical data for comparison of current behavior to past behavior. If more that one test is done for a joint motor, the application identifies the subsequent tests and assigns an appropriate number. The plotted image of the test is color coded to correspond with the appropriate number test.

Automated In-flight Check-out

The purpose of the in-flight check out is to gather and analyze data needed to verify the health and performance of shuttle systems for subsequent shuttle flights. Performing checkout during flights helps to eliminate the amount of ground testing that must be done between shuttle flights. Existing methods of performing in-flight checkout required engineers to manually gather data throughout the mission, and to pour over large amounts of historical data in search of the information needed to assess and evaluate the in-flight checkout requirements. The Automated In-flight Checkout application automatically gathers the data that is used to conduct in-flight tests, evaluates the data against requirement pass/fail criteria, summarizes the results of the tests, and produces the In-flight Checkout Report. Additionally, when an in-flight checkout requirement is not met, the user is alerted of the discrepancy, and given an opportunity to acknowledge the event and classify it, or to acknowledge it without assigning any classification. The classifications are "failed" for failed requirements that cannot be explained, and "non-problem failure", for failures that can be explained, and require no action. Each reported requirement is associated with a window that when opened, explains what conditions and data caused the test to fail (figure 6).

PRSD Trend Analysis

The PRSD Tank Quantity application assists subsystem personnel in detecting leaks in the shuttle oxygen and hydrogen fuel tanks, a problem that has proved to be quite challenging. The application calculates and reports the total accumulated average kilowatt and amperage for the last 24 hours of the mission and, for the accumulated mission time. The H2 and O2 tank quantity differences between known consumption demands and measured depletion is calculated and reported over the most recent 24 hours and for the accumulated mission time. A prediction of time remaining at the current usage rate, and at the average amperage used over the accumulated mission time is calculated and reported. A two day reserve is included in the prediction, and reserve quantities for the H2 and O2 tanks are also reported. (see figure 7)

Theoretically, it should be possible to detect leaks by comparing the measurements provided by tank sensors, to a calculated consumption that can be obtained from the demands of known system loads. However, the sensor reading are affected by several other factors. Side-effects from thruster

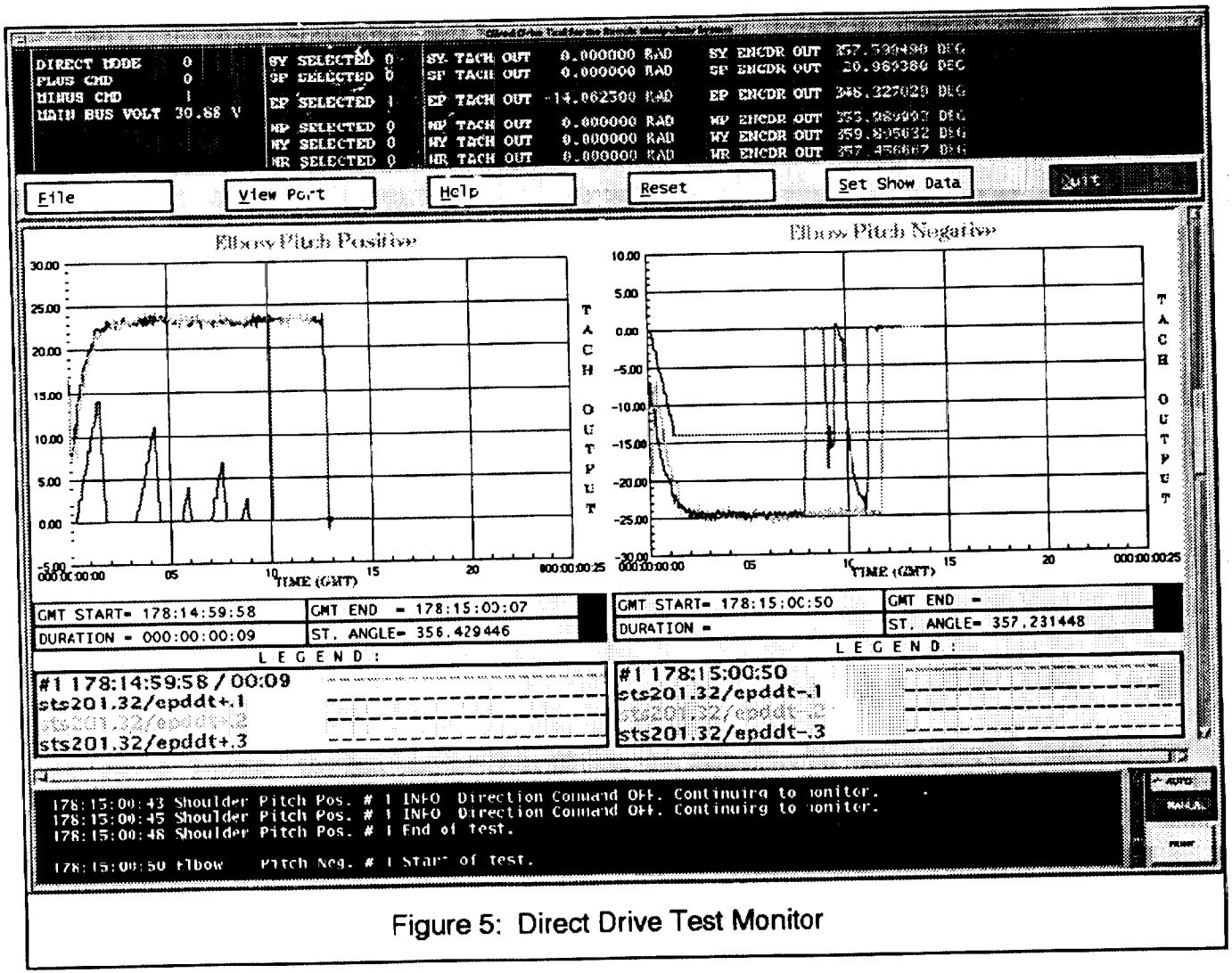


Figure 5: Direct Drive Test Monitor

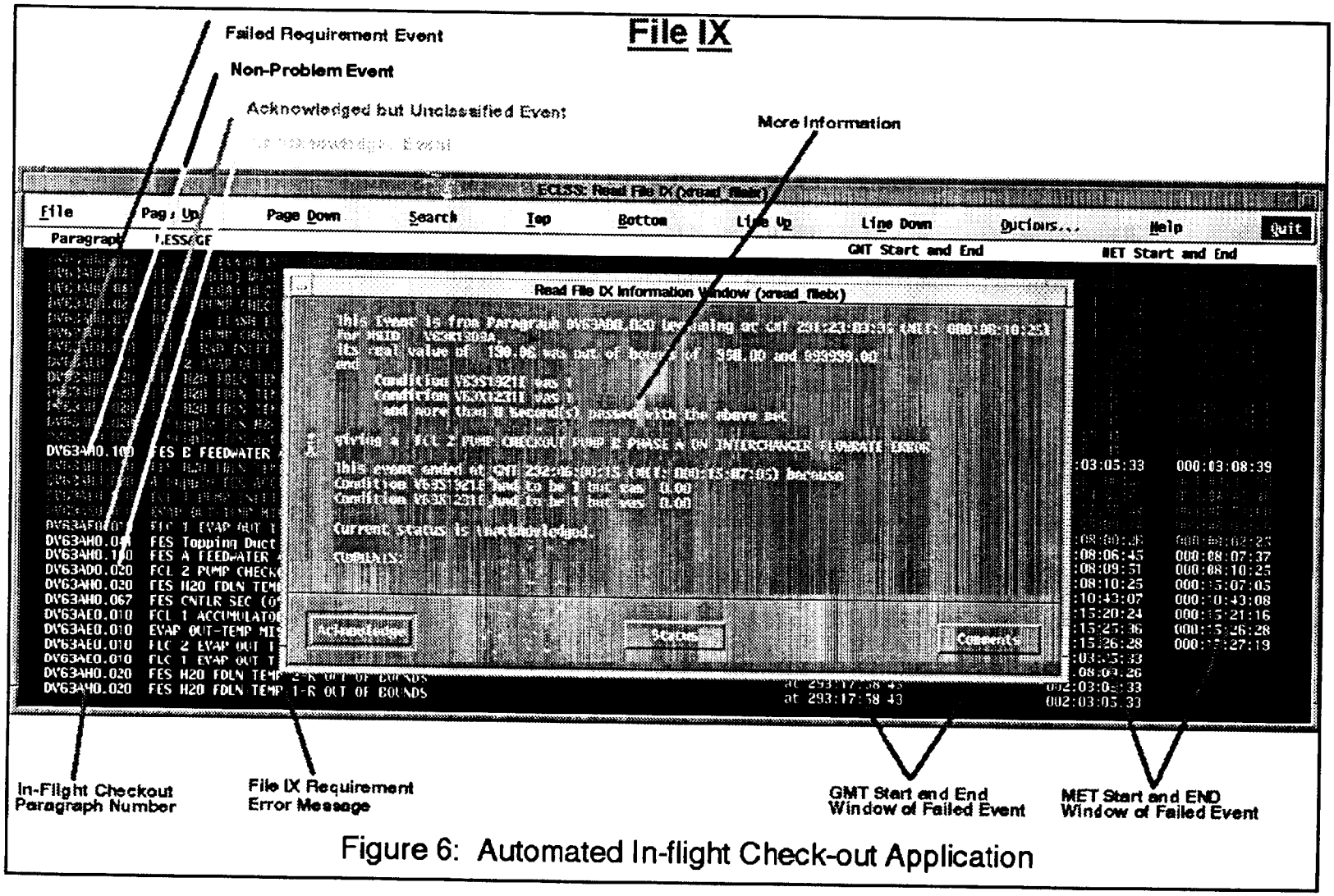


Figure 6: Automated In-flight Check-out Application

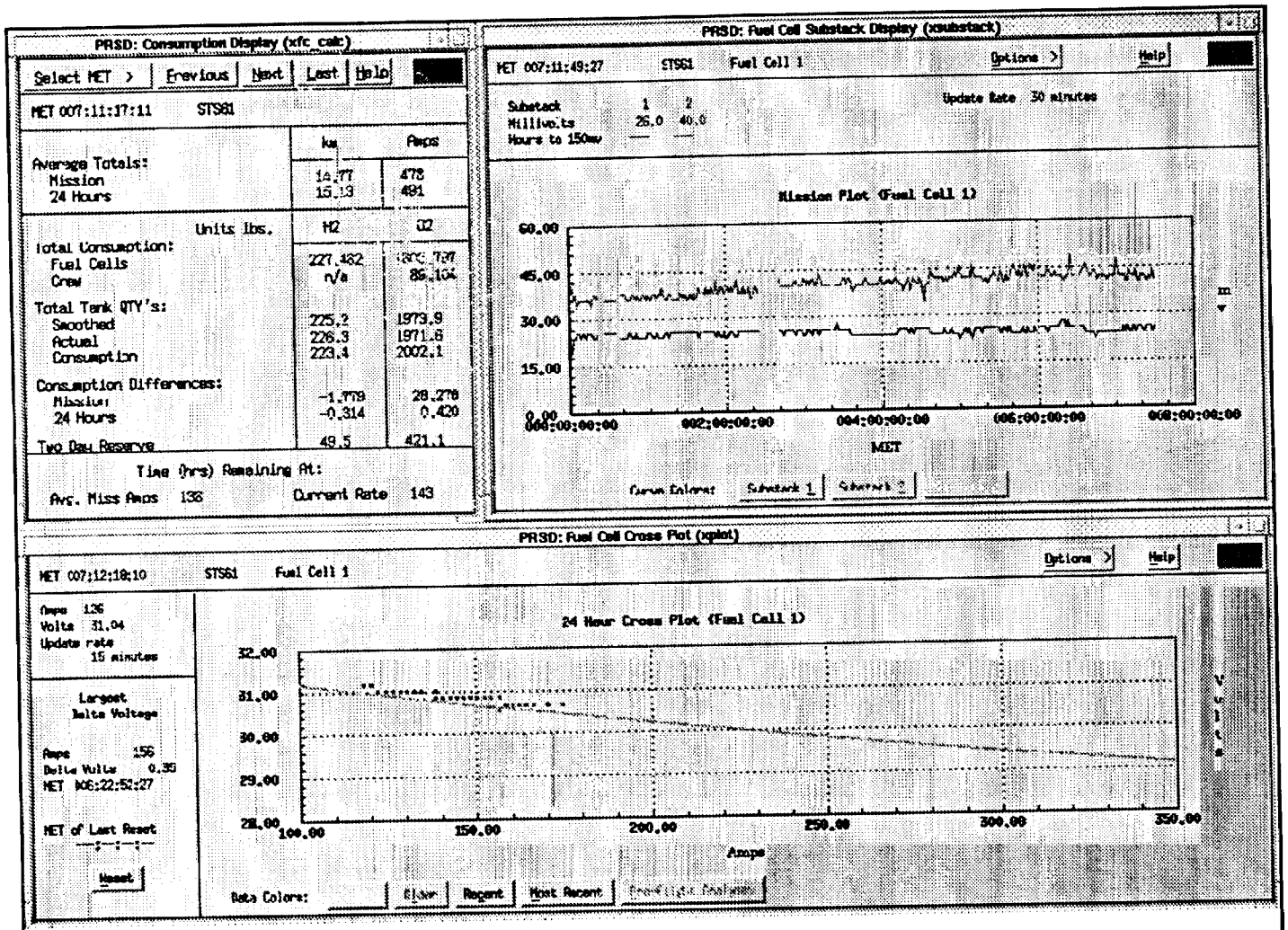


Figure 7: PRSD Trend Analysis Application

burns, on/off cycling of mixer heaters, and tank content stratification and destratification, render the sensor readings unreliable for positively determining quantities. Consequently, before any progress could be made to detect leaks, the first task was to attempt to accurately determine the amount of fuel in the tanks.

The Tank quantity measurements consists of 1) the raw measured total tank quantity of O2 and H2 throughout the flight, and 2) the calculated consumption of O2 and H2 based on the power provided by the fuel cells. If the measured tank quantity starts dropping faster than the calculated consumption there may be reason to suspect a leak in one of the tanks. Compensation for the variability in the measured sensor values is obtained by applying an exponential smoothing algorithm to the sensor readings. The smoothed values are plotted against the calculated consumption. A confidence interval is calculated to provide an indication of how well the calculated value tracks the adjusted measured value. A significant divergence between the two measurements provides some indication of a possible leak that should be investigated further.

The measured tank quantity data contains variability introduced at the sensor. One of these is caused by the heater cycle turning on and off to maintain the tank's pressure at a level such that it will become the active tank. These variations have an oscillation-like characteristic with a period of roughly one to two hours. In order to facilitate the comparison of measured quantity to calculated quantity, an exponential smoothing technique was selected to compensate for these variations. The effect of the exponential smoothing technique is similar to that of a moving average in which the data is weighed heavier toward the leading edge and becomes insignificant in the distant past. Furthermore, since real time telemetry data contains small gaps, a technique which can handle irregularly spaced data and which was published by D.J. Wright [1] was selected. This technique is an extension of a double exponential smoothing adapted to data occurring at irregular time intervals.

Since the reactant tank quantity data has a trend component (the quantity is decreasing), a double exponential smoothing technique is required. Double exponential smoothing separately smooths both the quantity estimate and the trend value. The smoothed quantity estimate is obtained by using the following equation:

$$\hat{\mu}_{n+1} = \hat{\mu}_n + (t_{n+1} - t_n)\hat{\beta}_n$$

where

$\hat{\mu}_n$ = the smoothed quantity estimate at time t_n, and

$\hat{\beta}_n$ = the smoothed estimate of the trend at time t_n.

The smoothed telemetry quantity measurements may be compared to the quantity estimated based on a separate mathematical model of oxygen and hydrogen consumption. This model uses the voltage and current telemetry data to compute estimated reactant consumption based on the chemical reaction equations for the fuel cell. This consumption estimate contains noise as well and is exponentially smoothed using the same technique used to smooth the telemetry quantity data.

As an aid for the comparison of the two measurements, a confidence level calculation testing that the two slopes are statistically equal is computed. The trend estimates from the mathematical model and the telemetry quantity data are used as input parameters to calculate a t-statistic which represents the probability that the two slopes are equal. This gives a number between 0% and 100% which reflects the degree of certainty that the two slopes are the same. This number is used to alert controllers of anomalous conditions in the fuel cell sub-system.

The next phase of this effort consists of investigating the use of learning algorithms and pattern recognition techniques to identify and classify trends that are reflected in the data.

FY 94 PLANS

PRSD Trend Analysis

Work in FY 94 will continue the trend analysis of the PRSD data. Pattern recognition, neural networks, and learning systems will be investigated and applied to this domain. These technologies are being applied to four tasks that have been identified for this area; recognition and prediction of destratification in the oxygen tanks, prediction of stratification in the oxygen tanks and recognition of the stratification signature, oxygen/manifold leak detection and trend analysis, and hydrogen tank/manifold leak detection and trend analysis.

The resulting drastic pressure drop that occurs when oxygen tank contents are destratified, (the remixing of a fluid that is non-uniformly dense (stratified), can be confused with large leaks that would also be indicated by a sharp decline in tank pressure. The ability to determine when and how much of the tank contents become stratified, is an important aspect of PRSD failure analysis. It is also important to know when remixing of the contents has occurred. The extent of existing tank content stratification, and the type of thruster burn that is to occur can be used to predict the probability of destratification, and assess its severity. The signature for an impending destratification is identified by a drop in pressure of as much as 300 psi within 15 minutes, caused by the cooling induced by mixing high density (cool) liquid near the tank walls, with the hot, low density liquid near the tank heater. This may occur during a maneuver after a long period of no acceleration of the vehicle. The severity of the destratification is affected by the distance of the tank form the center of gravity, and the type and direction of the thruster burn. Over 30 recorded instances of destratification will be used to train a system to identify the signature of destratification, anticipate the probability of an impending destratification, and assess the severity of tank destratification.

A slow, gradual decrease in measured tank quantity that would indicate a small leak, can also be the result of content stratification. Another aspect of PRSD trend analysis is to recognize the signature of stratification, and asses to which degree the contents have become stratified. Stratification affects the reliability of the sensor reading, an effect that will also have to be quantified. Numerable data points on stratification are available and will be used to train the system to recognize the stratification signature.

ATCS Diagnostic System

The Active Thermal Control System consists of several subsystems that together provide cooling for the shuttle. The ATCS Diagnostic System is to be comprised of diagnosis modules for the Flash Evaporator, Ammonia Boiler, Radiator, Freon Coolant Loops.

The first diagnostic system to be developed is that of the FES. The shuttle's Flash Evaporator System (FES) provides heat rejection of shuttle thermal loads. Poor instrumentation and too frequent failures have made this a prime area of investigation for automated diagnosis. The concept for this application was completed in FY93, and development is proceeding with knowledge capture of system expertise, and development of detailed requirements. The FES Diagnostic System will detect, identify, and diagnose failures that could cause the FES to stop working, a condition known as "FES shutdown". Seven categories of failures have been identified that would lead to FES shutdown; 1) restricted Freon flow to the FES, 2) Freon leakage, 3) water spray valve module failure, 4) steam duct failure, 5) accumulator failure, 6) controller

failure, and 7) FES supply water heater failure. The FES diagnostic system will address each class of failures.

SUMMARY

MIDAS applications have provided engineers and flight controllers at the Johnson Space Center, with software tools that reduce the amount of work involved in supporting missions, and that capture vanishing system expertise. Midas applications off-load to computers and software, the tasks of data gathering, filtering and analysis, perform automated diagnosis of system problems, and allow engineers to quickly and accurately identify and resolve problems. MIDAS applications have positively affected MER mission support by allowing subsystem managers to reduce MER mission support staffing, and by allowing subsystem personnel to share diminishing corporate knowledge and responsibility for their systems. Additionally, MIDAS applications have eliminated tasks previously required of humans, supplied previously unavailable data and analysis, and eliminated the need for some post-mission analysis. Some applications such as the Discrete Monitor and Log, Analog Plots, and File IX applications are re-usable and have been cloned to quickly distribute these capabilities to other systems, thereby increasing the cost effectiveness of development.

MIDAS Applications are written in the "C" programming language using X-windows version 11, Release 5 (X11R5), and the Motif widget set. Also used is Gensym's G2 expert system shell. The applications currently run on the SUN (OS 4.1.3 version of UNIX) Sparc1+ and Sparc2 machines, Masscomp workstations, and DEC-stations.

ACKNOWLEDGEMENTS

The MIDAS Project is sponsored by the Automation and Robotics Division, Intelligent Systems Branch, with funding from the Orbiter Projects Office, Code VA of the Johnson Space Center. The NASA Project Manager for this task is Ginger L. Pack/ER2. Programming support is performed by the NASA Engineering Support Contractor, Lockheed Engineering and Sciences Corporation (LESC). The programmers for this task are Jane Falgout, Joe Barcio, Bryan Kubena, and Charles Smith, all of LESC. The technical lead is Louis Flores of LESC.

REFERENCES

1. D. J. Wright, FORECASTING *IRREGURLARLY SPACED DATA: AN EXTENSION OF DOUBLE EXPONENTIAL SMOOTHING,* Comput. & Indus. Engng, Vol. 10. No. 2, pp 135-147, 1986.

2. *KU-BAND RADAR SELF TEST APPLICATION REQUIREMENTS,* April 1993

3. Steve Schnurer, *TREND ANALYSIS: APPLICATION EVALUATION AND RECOMMEDATION*, November 1993

Qualitative Model-Based Diagnosis Using Possibility Theory*

Cliff Joslyn†‡

Abstract

The potential for the use of possibility theory in the qualitative model-based diagnosis of spacecraft systems is described. The first sections of the paper briefly introduce the Model-Based Diagnostic (MBD) approach to spacecraft fault diagnosis; Qualitative Modeling (QM) methodologies; and the concepts of possibilistic modeling in the context of Generalized Information Theory (GIT). Then the necessary conditions for the applicability of possibilistic methods to qualitative MBD, and a number of potential directions for such an application, are described.

Possibility theory is being developed as an alternative to traditional theories of uncertainty. While possibility is logically independent of probability theory, they are related: both arise in **Dempster-Shafer evidence theory** as **fuzzy measures** defined on **random sets**; and their distributions are **fuzzy sets**. Together these fields comprise the new field of **Generalized Information Theory**.

Possibilistic processes, which generalize interval analysis, are based on a set of partially overlapping intervals, resulting in non-additive weights on a set of alternatives. Thus they are suitable for **qualitative modeling** methods, which inherently require loose representations of uncertainty, and typically involve **interval analysis**.

Qualitative methods are appropriate for modeling complex systems, such as spacecraft, where the interaction among the large number of parts and varying environmental conditions results in the possibility of unpredictable behavior and long-run departure from established steady-state domains. Therefore it is hypothesized that possibilistic methods may be useful for qualitative model-based diagnosis and trend analysis of spacecraft systems.

1 Model-Based Diagnosis

The **model-based** approach to systems **diagnosis** (MBD) [17] is based on the premise that knowledge about the internal structure of a system can be useful in diagnosing its failure. In MBD, a software model of the system, given inputs from the real system, generates and tests various failure hypotheses.

A typical MBD approach (derived from some of the standard literature [2, 7, 16]) to diagnosing a spacecraft (here described as some internal **system** whose **sensor** measurements output to a **telemetry** stream) is shown in Fig. 1.

An **alarm** is a report that some observed system attributes have departed from nominal, and entered error, conditions, usually by exceeding some threshold values; a **prediction** is a report that some system attributes should be in certain states; a **fault hypothesis** is a list of system components which may

*Prepared for the 1994 Goddard Conference on Space Applications of Artificial Intelligence.

†Graduate Fellow, Systems Science, SUNY–Binghamton, 327 Spring St. # 2, Portland ME, 04102, USA, (207) 774-0029, cjoslyn@bingsuns.cc.binghamton.edu, joslyn@kong.gsfc.nasa.gov

‡Supported under NASA Grant # NGT 50756.

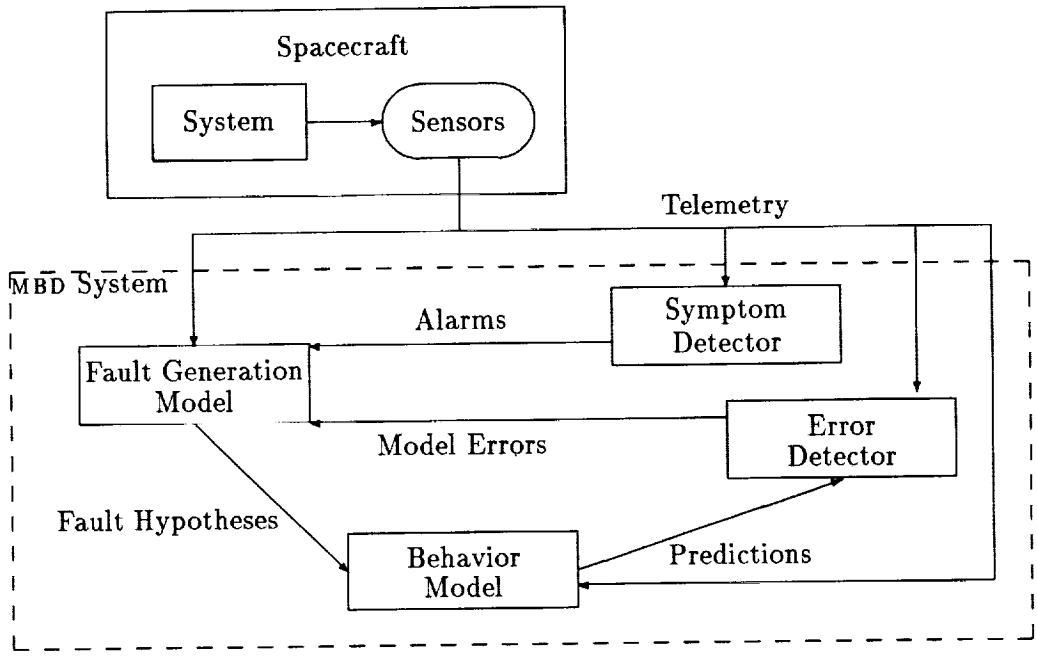


Figure 1: A typical model-based diagnostic system.

have failed; and an **error** is a report of a discrepancy between predicted and measured system attribute states.

The overall MBD system then involves two distinct spacecraft models. The **fault generation model** (FGM) takes inputs from telemetry, alarms, and errors, and either produces anew, or modifies existing, fault hypotheses. The **behavior model** takes inputs from telemetry and fault hypotheses, and outputs predictions. These are then corroborated against telemetry to produce errors. The fault hypotheses act to modify the behavior model so that it predicts system behavior as if the hypothetical system components had actually failed.

Both models can be difficult to construct, typically involving delicate tradeoffs among accuracy, precision, and tractibility. But the FGM, as the heart of the MBD approach, is particularly complex and involved. The FGM could be, for example, an inversion of the behavior model (as for Dvorak and Kuipers [7]) or a decision tree (as for Shen and Leitch [31]). Through backwards reasoning a variety of subsets of components can be identified, any of which are consistent with the given telemetry and alarms.

Filtering is the process by which error output is used to prune the set of fault hypotheses. If the prediction of the behavior model as modified by a particular fault hypothesis produces errors, then that fault hypothesis is not retained. As the system is monitored over time, further observations narrow the class of fault hypotheses. Achieving the null set indicates model insufficiency. But if the overall MBD system stabilizes to a non-empty set of fault-hypotheses, then these are advanced as possible causes of the failure.

2 Qualitative Modeling

Qualitative modeling (QM), usually considered a part of artificial intelligence, can be broadly described as the attempt to deliberately model systems at a high level of abstraction from the actual systems

themselves. Of course this approach produces models which are less precise than they might be, but with the tradeoff of potentially greater tractibility and accuracy (the less you say, the better your chance of being right).

QM methods can be useful when there is only a poor model of the original system, or when there are missing or incomplete data. This can happen when systems are incompletely specified, when they have parameters or states which aren't always known with certainty, or when complexity makes detailed prediction difficult. For more information about QM in general, see the anthologies edited by Bobrow [1], and Fishwick and Luker [10], and survey articles by Fishwick [8] and Guariso, Rizzoli and Werthner [14].

There are a variety of broad approaches within QM, which come under the names of naive physics, qualitative physics, qualitative simulation, qualitative reasoning, qualitative dynamics, etc. There are also a number of specific methods, including bond graphs, causal loop modeling, natural language modeling, "lumped" state space models, and inductive approaches.

In this paper the QM methods of most interest are those which use uncertainty distributions on state variables, and mixed interval- and point-valued dynamical systems. In models using uncertainty distribution methods the uncertainty about some attribute is represented mathematically by weights on all possible values. The set of weights, as a distribution, acts as a meta-state in the space of all possible distributions, and functional equations relating these meta-states produce predictions about the distribution meta-state at future times. Models using probability distributions are familiar as Markov processes and other kinds of stochastic models, and these have correlates in possibility theory (see Sec. 3.2). These methods are actually semi-qualitative, since the numerical representation of the distribution adds a quantitative component.

In an interval-valued dynamical modeling system like QSIM [28], a precise point-valued dynamical system of differential or difference equations is replaced by a homomorphic interval-valued process. Typically qualitative variables are identified within certain intervals, some relatively unconstrained (for example $x \in [0, \infty)$), and some constrained by landmark values (for example $x \in [x_{\min}, x_{\max}]$).

Qualitative variables are then generally related in three ways:

Functional: For example, if $y = M^-(x)$ then y is a monotonically decreasing function of x, so that if $x \in [0, x_{\max})$ then $y \in (-\infty, 0]$ or $y \in (M^-(x_{\max}), 0]$.

Arithmetic: Standard mathematical operations can also be represented qualitatively, for example if $x \in [0, x_{\max}]$ and $y \in (-\infty, 0]$, then $xy \in (-\infty, 0]$, but $x + y$ is unknown.

Dynamic: Change of state is represented by qualitative magnitude and direction. Qualitative differential relations link directions with magnitudes, for example given $y = dx/dt$, then

$$x \text{ increasing} \rightarrow y \in (0, \infty), \qquad x \text{ decreasing} \rightarrow y \in (-\infty, 0).$$

Of course, determinative results may not be available in such a qualitative model. For example, we saw above that $x + y$ could be any value in $(-\infty, \infty)$. Similarly, the existence of landmark values leads to uncertainty as to whether a landmark has been crossed. To account for each possibility, two alternatives must be branched off. Therefore in general, QM systems have a tree of possible system behaviors, and external factors (heuristics or other constraints) may be required to prune that tree.

QM has been applied to MBD to produce qualitative model-based diagnostic systems. For example, in the approach of Dvorak and Kuipers [7], model predictions are intervals of possible system state values. Stochastic methods, for example Bayesian networks [12] and Markov processes [13], have been

used extensively in MBD applications. And recently Shen and Leitch [31, 32] have advanced the FuSim method for qualitative MBD which uses fuzzy arithmetic (see Sec. 4.2).

3 Possibility Theory

Possibility theory was originally developed in the context of fuzzy systems theory [34], and was thus related to the kinds of cognitive modeling that fuzzy sets are usually used for. More recently, possibility theory is being developed as a new form of mathematical information theory complementing probability theory in the context of Generalized Information Theory (GIT). The author is developing possibility theory based on consistent random sets, and also an empirical semantics for possibility, including possibilistic measurement procedures and applications to the modeling of physical systems.

3.1 Possibilistic Mathematics in GIT

Table 3.1 summarizes the primary formulae of probability and possibility theory in the context of GIT and random set theory. These are only briefly explained here; see [5, 24, 26, 27] for more information.

Given a finite universe $\Omega := \{\omega_i\}, 1 \leq i \leq n$, the function $m: 2^\Omega \mapsto [0,1]$ is an **evidence function** (otherwise known as a **basic probability assignment**) when $m(\emptyset) = 0$ and $\sum_{A \subseteq \Omega} m(A) = 1$. Denote a random set generated from an evidence function as $\mathcal{S} := \{\langle A_j, m_j \rangle : m_j > 0\}$, where $\langle \cdot \rangle$ is a vector, $A_j \subseteq \Omega, m_j := m(A_j)$, and $1 \leq j \leq N := |\mathcal{S}| \leq 2^n - 1$. Denote the **focal set** as $\mathcal{F} := \{A_j : m_j > 0\}$ with **core** $\mathbf{C}(\mathcal{F}) := \bigcap_{A_j \in \mathcal{F}} A_j$ and **support** $\mathbf{U}(\mathcal{F}) := \bigcup_{A_j \in \mathcal{F}} A_j$.

The **plausibility** and **belief** measures on $\forall A \subseteq \Omega$ are

$$\mathrm{Pl}(A) := \sum_{A_j \cap A \neq \emptyset} m_j, \qquad \mathrm{Bel}(A) := \sum_{A_j \subseteq A} m_j.$$

Since they are dual, in that $\mathrm{Bel}(A) = 1 - \mathrm{Pl}(\overline{A})$, in general only plausibility will be considered below. The **plausibility assignment** (otherwise known as the **one-point coverage function**) of $\mathcal{S}$ is $\vec{\mathrm{Pl}} = \langle \mathrm{Pl}_i \rangle := \langle \mathrm{Pl}(\{\omega_i\}) \rangle$, where

$$\mathrm{Pl}_i := \sum_{A_j \ni \omega_i} m_j. \tag{1}$$

$\vec{\mathrm{Pl}}$ is a **fuzzy subset** of Ω that can be mapped to an equivalence class of random sets on Ω.

Under certain conditions the evidence values m_j and the plausibility assignment values Pl_i are mutually determining. Then $N \leq n$, and $\vec{\mathrm{Pl}}$ is a **distribution** of $\mathcal{S}$. When $N = n$ (there are exactly as many focal elements as there are elements of the universe), then the indices j on the focal elements A_j, and the i on the universe elements ω_i are equivalent, and it may be useful to use one or the other interchangeably. This is the case in the two rightmost columns of the table.

When $\forall A_j \in \mathcal{F}, |A_j| = 1$, then $\mathcal{S}$ is **specific**, and $\Pr(A) := \mathrm{Pl}(A) = \mathrm{Bel}(A)$ is an additive **probability measure** with **probability distribution** $\vec{p} = \langle p_i \rangle := \vec{\mathrm{Pl}}$ and additive normalization $\sum_i p_i = 1$ and operator $\Pr(A) = \sum_{\omega_i \in A} p_i$.

$\mathcal{S}$ is **consonant** ($\mathcal{F}$ is a **nest**) when (without loss of generality for ordering, and letting $A_0 := \emptyset$) $A_{j-1} \subseteq A_j$. Now $\Pi(A) := \mathrm{Pl}(A)$ is a **possibility measure** and $\eta(A) := \mathrm{Bel}(A)$ is a **necessity measure**. Since results for necessity are dual to those of possibility, only possibility will be discussed in the sequel.

As Pr is additive, so Π is **maximal**:

$$\forall A, B \subseteq \Omega, \quad \Pi(A \cup B) = \Pi(A) \vee \Pi(B),$$

Table 1: Summary of probability and possibility in GIT.

	RANDOM SET	DISTRIBUTIONS: $i \leftrightarrow j$	
		Probability	Possibility
Focal Set	Any	Singletons: $A_i = \{\omega_i\}$	Nest: $A_i = \{\omega_1, \ldots, \omega_i\}$
		$\{\omega_i\} = A_i$	$\{\omega_i\} = A_i - A_{i-1}, A_0 := \emptyset$
Structure	None	Partition	Total order
Belief	$\mathrm{Bel}(A) = \sum_{A_j \subseteq A} m_j$	$\Pr(A) := \mathrm{Bel}(A)$	$\eta(A) := \mathrm{Bel}(A)$
Plausibility	$\mathrm{Pl}(A) = \sum_{A_j \cap A \neq \emptyset} m_j$	$\Pr(A) := \mathrm{Pl}(A)$	$\Pi(A) := \mathrm{Pl}(A)$
Relation	$\mathrm{Bel}(A) = 1 - \mathrm{Pl}(\overline{A})$	$\mathrm{Bel}(A) = \mathrm{Pl}(A) = \Pr(A)$	$\eta(A) = 1 - \Pi(\overline{A})$
Distribution	$\mathrm{Pl}_i = \sum_{A_j \ni \omega_i} m_j$	$p_i := \mathrm{Pl}_i = m_i$	$\pi_i := \mathrm{Pl}_i = \sum_{j=i}^{n} m_j$
		$m_i = p_i$	$m_i = \pi_i - \pi_{i+1}, \pi_{n+1} := 0$
Measure		$\Pr(A \cup B) = \Pr(A) +$ $\Pr(B) - \Pr(A \cap B)$	$\Pi(A \cup B)$ $= \Pr(A) \vee \Pr(B)$
Normalization		$\sum_i p_i = 1$	$\bigvee_i \pi_i = 1$
Operator		$\Pr(A) = \sum_{\omega_i \in A} p_i$	$\Pi(A) = \bigvee_{\omega_i \in A} \pi_i$
Nonspecificity	$\sum_j m_j \log_2 \lvert A_j \rvert$	0	$\sum_{i=2}^{n} \pi_i \log_2 \left[\frac{i}{i-1}\right]$ $= \sum_{i=1}^{n} (\pi_i - \pi_{i+1}) \log_2(i)$
Strife	$-\sum_j m_j \log_2 \left[\sum_{k=1}^{n} m_k \frac{\lvert A_j \cap A_k \rvert}{\lvert A_j \rvert}\right]$	$-\sum_i p_i \log_2(p_i)$	$\sum_{i=2}^{n} \pi_i - \pi_{i+1} \log_2 \left[\frac{i^2}{\sum_{j=1}^{i} \pi_j}\right]$ $< .892$
Semiring	$\langle \oplus, \otimes \rangle$	$\langle +, \times \rangle$	$\langle \vee, \sqcap \rangle$
Marginals		$p(x) = \sum_y p(x, y)$	$\pi(x) = \bigvee_y \pi(x, y)$
Conditionals		$p(x, y) = p(x \lvert y) \times p(y)$	$\pi(x, y) = \pi(x \lvert_{\sqcap} y) \sqcap \pi(y)$
		$p(x \lvert y) = p(x, y)/p(y)$	$\pi(x \lvert_{\sqcap} y) \in [\pi(x, y), 1]$
		$\forall y, \sum_x p(x \lvert y) = 1$	$\forall y, \bigvee_x \pi(x \lvert_{\sqcap} y) = 1$
Process		$\mathbf{P} := [p(x \lvert y)]$	$\boldsymbol{\Pi} := [\pi(x \lvert_{\sqcap} y)]$
		$p' = p \cdot \mathbf{P}$	$\pi' = \pi \circ \boldsymbol{\Pi}$
		$p'(x) = \sum_y p(y) \times p(x \lvert y)$	$\pi'(x) = \bigvee_y \pi(y) \sqcap \pi(x \lvert_{\sqcap} y)$
Concepts		Division among distinct hypotheses	Coherence around certain hypotheses
		Frequency	Capacity
		Chance	Ease of attainment
		Likelihood	Distance, similarity

where $\vee$ is the maximum operator. As long as $\mathbf{C}(\mathcal{F}) \neq \emptyset$ (this is required if $\mathcal{F}$ is a nest), then $\vec{\pi} = \langle \pi_i \rangle := \vec{\mathrm{Pl}}$ is a **possibility distribution** with maximal normalization $\bigvee_i \pi_i = 1$ and operator $\Pi(A) = \bigvee_{\omega_i \in A} \pi_i$. Also define the core and support of the possibility distribution

$$\mathbf{C}(\pi) := \{\omega_i : \pi(\omega_i) = 1\} = \mathbf{C}(\mathcal{S}), \qquad \mathbf{U}(\pi) := \{\omega_i : \pi(\omega_i) > 0\} = \mathbf{U}(\mathcal{S}).$$

Nonspecificity and **strife** are two **uncertainty measures** which are defined on random sets. They measure respectively the possibilistic and probabilistic aspects of the uncertainty or information represented in the random set. They achieve the forms shown in the table for the possibilistic and probabilistic special cases. Note that in the probabilistic case the uncertainty collapses to stochastic entropy, while in the possibilistic case the strife is bounded above by a small number.

Stochastic and possibilistic processes are defined on their respective distributions when two operators $\oplus$ and $\otimes$ are available such that $\langle \oplus, \otimes \rangle$ form a semiring ($\otimes$ distributes over $\oplus$), and $\oplus$ is the operator of the distribution. For probability, $\oplus = +$, and $\langle +, \times \rangle$ is the unique semiring.

For possibility, $\oplus = \vee$, and there are many semirings of the form $\langle \vee, \sqcap \rangle$, where $\sqcap$ is a triangular norm (monotonic, associative, commutative operator with identity 1 [5]). $\wedge$ (the minimum operator) and $\times$ are two of the more popular norms, as is $0 \vee (x + y - 1)$. Conditional possibility is not always unique, depending on the norm used. The formulae for marginal, joint, and conditional probability and possibility (which is dependent on $\sqcap$) are then shown in the table, as is the next state function for a stochastic and possibilistic process.

3.2 Possibilistic Models

Probability and and possibility almost never coincide (only for distributions of the form $\langle 0, \ldots, 1, \ldots, 0 \rangle$). Semantically, probability and possibility theory are also related to very different concepts [20]. Probability is inherently additive, and is thus concerned with the dispersal or division of knowledge over a set of distinct hypotheses, and so with concepts related to frequency.

But possibility is inherently *non-additive*. It is concerned with the *coherence* of knowledge *around* a set of certain hypotheses (the core $\mathbf{C}(\pi)$), and thus with *ordinal* concepts related to capacity. Where probability makes very strong constraints on the representation of uncertainty (additivity), possibility makes only very *weak* constraints. The maximum relation is a very weak operator, and there is a choice of many norms to use, some of which are strong, and others of which are also weak.

So possibilistic models are appropriate where stochastic concepts and methods are inappropriate, including situations where long-run frequencies are difficult if not impossible to obtain, or where small sample sizes prevail. This is true in **reliability analysis**, for example, where failures and system entry into non-nominal behavior domains are very rare; and **trend-analysis**, where even though observations are made over a long time, the state variables of concern change only very slowly, and new domains of behavior are only very rarely seen. In these cases the weakness of the possibilistic representation is matched by the weak evidence available.

A general modeling relation is shown in Fig. 2, where: t, t' are former and subsequent times; $W = \{w\}$ are the states of the world; $M = \{m\}$ are the states of the model; $o: W \mapsto M$ is the measurement function; $r: W \mapsto W$ is the movement of "reality"; and $f: M \mapsto M$ is the modeling prediction function.

Reality is presumably given, or at least operates on its own without our help. So to make a valid model we are required to provide the measurement and prediction functions so that the diagram commutes. Measurement is used both to set the initial conditions of the model and to corroborate later measurements against the model state, while prediction is used to produce the future model state.

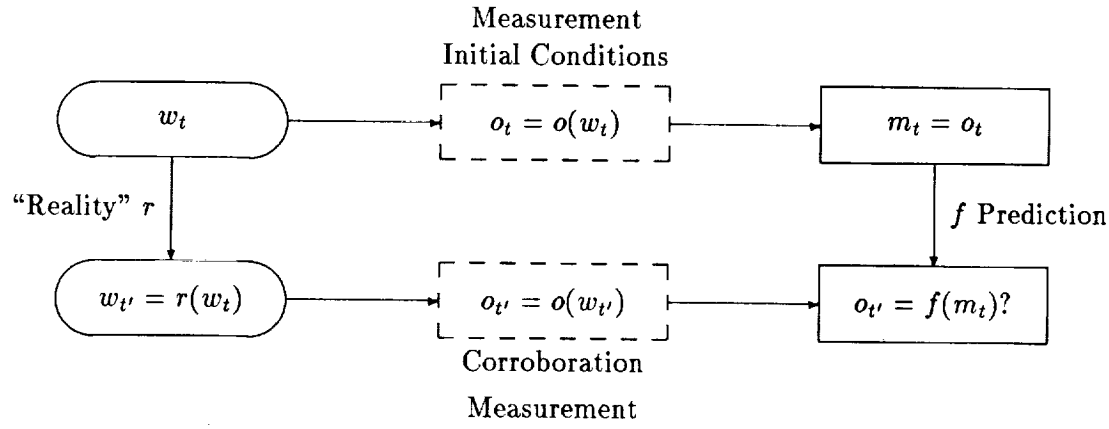


Figure 2: A general modeling relation.

In a stochastic model, $M = \{\vec{p}\}$, where $\vec{p}$ is a probability distribution of the state of the world. And in a possibilistic model $M = \{\vec{\pi}\}$, where $\vec{\pi}$ is a possibility distribution of the state of the world. Once the final time possibility distribution is achieved, then a **possibilistic Monte Carlo method** [24] is required to select a final outcome.

So a possibilistic model requires possibilistic measurement and prediction procedures. Possibilistic prediction procedures will be based on the possibilistic processes briefly outlined in Sec. 3.1. For example, the author has defined **possibilistic automata** as possibilistic Markov processes [22] which generalize non-deterministic automata.

Possibilistic measurement methods have also been developed by the author [18, 21]. The essential requirement is the collection of the frequency of occurrence of subsets or intervals which are partially overlapping. If the core of the observed intervals (their global intersection) is nonempty, then (1) will yield an empirical possibility distribution.

An example is shown in Fig. 3. On the left, four observed intervals are shown. The bottom two occur with frequency 1/2, while each of the upper two have frequency 1/4. Together they determine an empirical random set. The step function on the right is the possibilistic histogram derived from (1).

There are a variety of well-justified continuous approximations of a possibilistic histogram. Two examples are shown in the figure. The rising diagonal on the left is common to both. The two falling continuous curves on the right are distinct to each. The parallelogram form marked π^* is one of the most commonly used continuous approximations, but it must be noted that this is only one possibility among many, including smooth curves. This approach to possibilistic measurement generalizes to n intervals and to the continuous case.

The core, here $\mathbf{C}(\vec{\pi}) = [1.5, 2]$, being nonempty, is included in the support $\mathbf{U}(\vec{\pi}) = [1, 4]$. If the core were empty, then $\bigvee_i \mathrm{Pl}_i < 1$, so that $\vec{\mathrm{Pl}}$ would not be a possibility distribution. In this case, possibilistic normalization procedures, which have also been developed by the author [19], would be required.

4 Possibility Theory as a Qualitative Modeling Method

Mathematical possibility, in both theory and applications, is still in the basic research phase, just out of its infancy. For example, the axiomatic basis for possibility theory and the properties of possibility distributions on continuous spaces are still being defined, and the semantics of possibility in physical

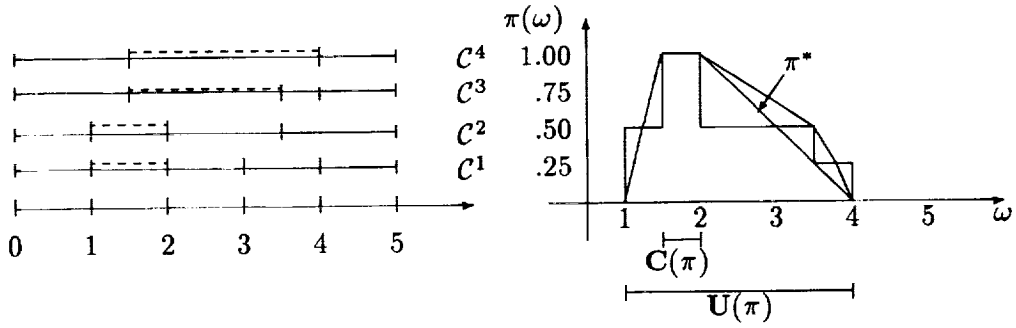


Figure 3: (Left) Four example observed intervals. (Right) The possibilistic histogram and two continuous approximations.

systems has been considered only by very few. But there are many reasons why it can be hoped, and even expected, that possibility theory can come to play an important role in QM in general, and in the application of QM to MBD in particular.

Hamscher et al. have noticed some of the weaknesses of stochastic methods for MBD.

> It is usually assumed that reliable failure statistics will be available, but this is in fact rare in practice. What is needed ... is a way of working with likelihoods that could be specified ordinally rather than quantitatively. [17, p. 452]

This is *exactly* what possibility theory provides, a non-additive, ordinal approach to QM which hybridizes interval-valued dynamics and uncertainty distribution methods.

4.1 Possibility Theory and Interval Analysis

Possibility theory can be used as a generalization of interval analysis. A **fuzzy interval** is a convex possibility distribution on $\mathbb{R}$ where

$$\forall x, y \in \mathbb{R}, \quad \forall z \in [x, y], \quad \pi(z) \geq \pi(x) \wedge \pi(y).$$

This is the case for the measured possibility distributions from Sec. 3.2, as shown in the example in Fig. 3. As illustrated in Fig. 4, under these conditions, π can be represented as a set of nested intervals weighted by their possibility values, where $\pi^0 := \mathbf{U}(\pi)$ as a special case, and

$$\forall \alpha \in (0, 1], \quad \pi^\alpha := \{x \in \mathbb{R} : \pi(x) \geq \alpha\}, \qquad \alpha_1 \geq \alpha_2 \rightarrow \pi^{\alpha_1} \subseteq \pi^{\alpha_2}.$$

A standard interval $[a, b] \subseteq \mathbb{R}$ is a special case, where

$$\pi(x) = \begin{cases} 1, & a \leq x \leq b \\ 0, & x < a \text{ or } x > b \end{cases}, \qquad \forall \alpha \in [0, 1], \quad \pi^\alpha = [a, b].$$

For a fuzzy interval π where $\exists! x \in \mathbb{R}, \pi(x) = 1$, then π is a **fuzzy number**. **Fuzzy arithmetic** [25] generalizes mathematical operations such as addition and multiplication from interval arithmetic [30] to fuzzy numbers.

4.2 Possibility Theory and Fuzzy Theory

As mentioned above, possibility theory was originally developed by Zadeh [34] in the context of fuzzy sets and fuzzy logic. For Zadeh, a possibility distribution simply *was* a fuzzy set by another name, and

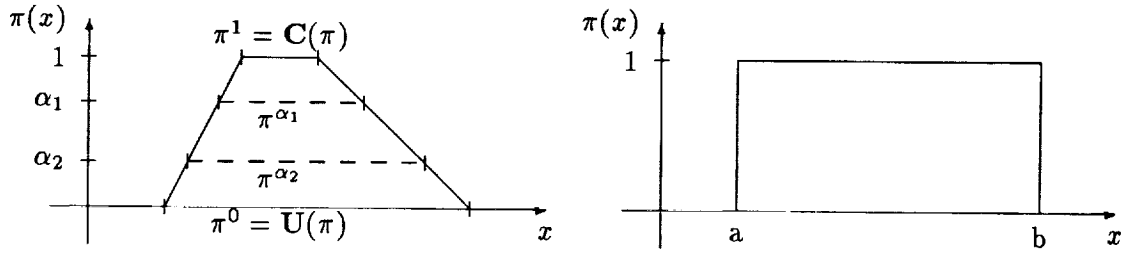


Figure 4: (Left) A possibility distribution as a collection of weighted intervals. (Right) The special case of a crisp interval.

thus possibilistic information theory was *strictly* related to fuzzy information. This view is less than adequate for a variety of reasons. While it is certainly true that a possibility distribution is a fuzzy set, in GIT there are *many* structures which are legitimately fuzzy sets, including *probability* distributions. The author provides a full discussion elsewhere [24].

Nevertheless, fuzzy theory and possibility theory do share a number of points in common. In particular, fuzzy intervals and numbers *are*, in fact, possibility distributions. Thus QM methods which use fuzzy arithmetic as discussed in Sec. 4.1 are essentially possibilistic. An example is the recent work of Sugeno and Takahiro [33].

These methods are also very popular in applications. Fuzzy arithmetic has been used as a QM method for MBD, for example by Shen and Leitch [32] and Fishwick [9]. They use the standard methods of fuzzy control systems, where a set of overlapping fuzzy intervals divide a quantity space into a few linguistic values like "large positive" and "small negative". These fuzzy sets are not measured properties of the system being modeled, and are dependent on the heuristic specification of the system modeler. Thus they are essentially modeling the cognitive state of some human expert, rather than directly modeling the system in question.

This contrasts sharply with the possibilistic processes discussed in Sec. 3.2. First, they are cast strictly within the context of mathematical possibility theory (including possibilistic processes) specifically, rather than fuzzy theory generally. Also, they are based on measurement of the system in question.

5 A Possibilistic Approach to MBD

At both the general level and in some specific ways, there are areas of MBD for which it is appropriate to consider a possibilistic approach.

5.1 Possibilistic Symptom and Error Detection

Typically the symptom and error detectors simply compare the measured value against a crisp interval of nominal or predicted values [7]. This is inadequate because the resulting cutoff from nominal to error condition is essentially arbitrary. It is natural to use a fuzzy interval to generalize this, measuring either prediction errors or fault symptoms as the possibilistic distance of the telemetry from the predicted or nominal system state respectively.

Consider a measured value x compared against an error fuzzy interval of the form of Fig. 4. Such a possibility distribution could be the output of the behavior model, for instance, and would then serve as input to the error detector. Then $\pi(x)$ is the strength of the error or alarm raised. When $x \in \mathbf{C}(\pi)$,

then $\pi(x) = 1$ and there is no alarm. When $x \notin \mathbf{U}(\pi)$, then $\pi(x) = 0$ and the alarm is complete. In between, an intermediate alarm is raised.

Even in situations where crisp thresholds are acceptable, they may be dynamic, varying as a result of changing system and environmental conditions. Doyle et al. consider the situation of an earth-orbiting spacecraft as it proceeds through sunlight and shadow.

> Impingent solar radiation changes the thermal profile of the spacecraft, as does the configuration of currently active and consequently, heat-generating subsystems on board. Thresholds on temperature sensors should be adjusted accordingly. A particular temperature value may be indicative of a problem when the spacecraft is in shadow or mostly inactive, but may be within acceptable limits when the spacecraft is in sunlight or many on-board systems are operating. [3]

This situation is shown in Fig. 5. Assume a variable, say the temperature t of a given component, must be kept in a critical range as the spacecraft moves in and out of daylight. As it does so, the range shifts as shown in the upper figure, where the transition periods begin at a change in sunlight, and continue to thermal equilibrium. For simplicity, assume that that interval is sampled uniformly six times during the orbital day, twice each for daylight D_i, night N_i, and transition period T_i. The possibilistic histogram for the possibility $\pi(t)$ of t holding a value at any given time and a parallelogram approximation are shown.

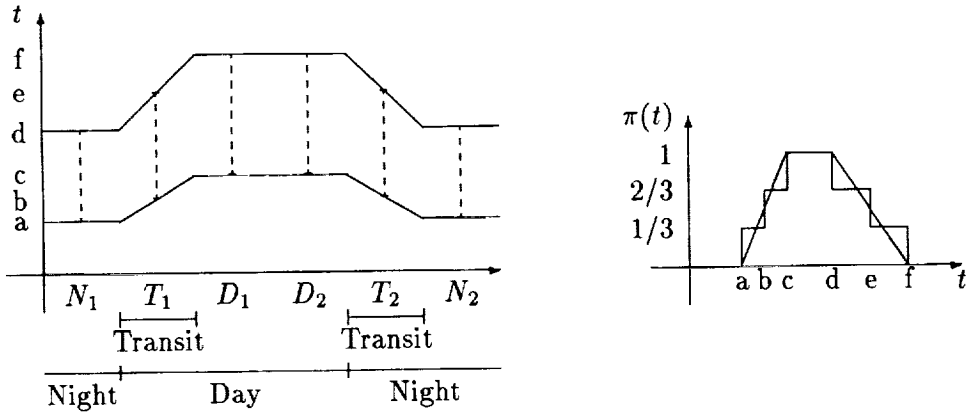


Figure 5: (Left) Variable critical range of a component through a day-night cycle. (Right) Its possibilistic histogram and a parallelogram approximation.

A combination of these two approaches is also possible, where instead of a crisp interval changing over time, rather a whole possibility distribution itself changes with time.

5.2 Sensor Modeling

Although the MBD system contains two models, the behavior model and the FGM, as a whole, it is itself also a model of the spacecraft. As such, it is dependent on its inputs from measurement, and thus on the sensor output of the spacecraft. Thus there are modeling issues in MBD concerning the sensors themselves.

When modeling complex systems, sensor data may be sparsely distributed, with missing observations, and sometimes very small samples sizes. As argued elsewhere by the author [20], these are important

conditions for the inapplicability of stochastic methods, and when they hold, possibilistic methods should be considered.

In this respect, there is strong support in the literature for the idea that possibility, as distinct from probability, has a role to play in QM. For example, Luo and Kay observe

> When additional information from a sensor becomes available and the number of unknown propositions is large relative to the number of known propositions, an intuitively unsatisfying result of the Bayesian approach is that the probabilities of known propositions become unstable. [29]

While Durrant-Whyte takes a typical statistical approach, he also notes

> A robot system uses notably diverse sensors, which often supply only sparse observations that cannot be modeled accurately. [6]

Dvorak and Kuipers make a similar observation in the context of model-based monitoring.

> All measurements come from sensors, which can be expensive and/or unreliable and/or invasive. Monitoring is typically based on a small subset of the system parameters, with limited opportunity to probe other parameters. [7]

5.2.1 Data Fusion

Possibilistic measurement as outlined in Sec. 3.2 is predicated on the observation of subsets or intervals which are partially overlapping. It is therefore imperative to consider the source of these intervals. But traditional measurement methods do not in general yield overlapping intervals. Rather the purpose in designing a good sensor is to produce distinct outcomes, perhaps intervals with some uncertainty, but still disjoint, forming equivalence classes.

But overlapping intervals *may* result from the combination of data from *different* instruments which measure the same system attribute, either directly or indirectly. Thus random set theory in general and possibility theory in particular is significant when considering the problem of **data fusion** in MBD [15, 29].

Hackett and Shah discuss data fusion in general, including indirect measurements, and the Dempster-Shafer (that is, random set) approach.

> Every sensor is sensitive to a different property of the environment; in order to sense multiple properties, it is necessary to use multiple sensors. A system using multiple sensors that sense a single property can be used. [15]

Dubois, Lang, and Prade [4] have also considered the data fusion problem using possibilistic logic.

Indirect Measurements First consider the situation where measurements of a component are not made directly, but rather knowledge of the state of the component is only gained indirectly by inference from the outputs of sensors of other components. Doyle et al. [3] offer an example from jet aircraft: low engine thrust can be indicated by either low exhaust temperature or low turbine rotation speed, or both.

This situation is illustrated in Fig. 6. Here component A is not monitored. Its state can only be inferred from the sensors D and E, which monitor components B and C, and which in turn are causally

connected to A. Each of the intervals reported by D and E individually is distinct and disjoint. But since the knowledge of A provided by D and E is mediated by B and C, together they may indicate that A exists in two different, possibly overlapping, intervals.

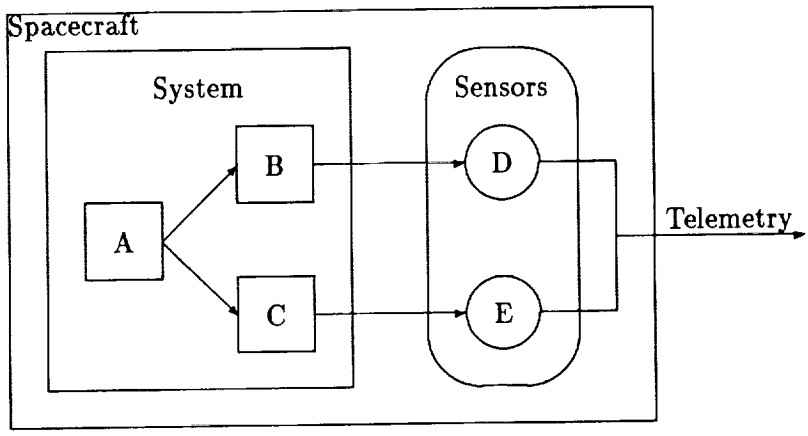


Figure 6: Indirect measurements of the state of a spacecraft component.

So as the amount of sensor "penetration" (sensor/component ratio) drops, standard measurement methods yielding frequency distributions may become less tenable, leaving only observations of random sets.

Redundant Measurements Alternatively, a system component may be monitored redundantly by multiple instruments. If these sensors are identical, and identically calibrated, then the result will simply be as if there was a time-series of observations from a single instrument. But if they are mutually discalibrated, either out of phase, or scale, or both, then the intervals reported from each instrument may overlap.

If the sensors measure distinct modalities (e.g. pressure and temperature) of a single component, then a process of **registration** [15] is required to derive a report from one in the modality of the other, or two new reports from each in a third modality. In any event, the argument here is very similar to the one above in the case of indirect measurements, and possibly overlapping intervals my result.

5.2.2 Sensor Failure Modeling

As mentioned above, in MBD data are not only combined from disparate sensors, they are also sometimes incomplete, degraded, or missing altogether. Even when standard (disjoint) observations are made, under these conditions there is the potential for the application of GIT and possibility theory.

First, in GIT standard measurements are represented as singleton sets $\{\omega_i\}$, where each $\omega_i \in \Omega$ may indicate a disjoint interval. In a Bayesian or stochastic approach, sensor failure is represented by a uniform distribution over each of the $\{\omega_i\}$, again dividing our ignorance among a set of disjoint choices.

But in GIT, a missing observation is represented, more accurately, as an observation of the *entire* universe Ω. While this does not result in a specifically possibilistic situation, neither does it result in a frequency or probability distribution. This is discussed more fully by the author elsewhere [24].

For a simple example, assume that a system with three states $\Omega = \{a, b, c\}$ is observed at ten uniformly distributed times, with a and c each seen twice, b seen three times, and three cases where the sensor made no report. These final three cases must be recorded as observations of Ω, and the specific observations

replaced with observations of the singleton sets $\{a\}, \{b\}$, and $\{c\}$ respectively. The overall empirical random set is then

$$\mathcal{S} = \{\langle\{a\}, 1/5\rangle, \langle\{b\}, 3/10\rangle, \langle\{c\}, 1/5\rangle, \langle\Omega, 3/10\rangle\}$$

with plausibility assignment $\vec{\text{Pl}} = \langle 1/2, 3/5, 1/2\rangle$, which is neither an additive probability distribution nor a maximal possibility distribution.

When sensor data are not missing, but rather degraded, compromised, or suspect in some way, a confidence weighting on each sensor's output is naturally not additive: our confidence about the sensors is not divided among them, since all could be perfect or any number of them could be in any state of degradation. Instead it is natural to represent this confidence as a possibility distribution on each sensor's output. Again, an observation in the core indicates complete confidence, while one outside the support indicates complete sensor failure.

Representation of a graduated degree of sensor failure allows a corresponding graduated degree of confidence in model predictions. The need for this has been noted by Fulton.

> When we detect a broken sensor, great difficult arises if we continue diagnosing other failures, because typical rule-based systems do not degrade gently when sensors fail (because the mapping is dependent on a complete and accurate set of sensor data). [11]

5.3 Possibilistic Models Proper

In the sequel, the term **system model** will refer to the FGM or the behavior model generally. So finally, it is useful to consider possibilistic methods applied directly to the system models themselves, constructing them as possibilistic processes such as possibilistic automata, and not as fuzzy arithmetic systems as discussed in Sec. 4.

Input to these systems may or may not be proper possibility distributions, since both crisp (standard) intervals and point values are special cases of possibility distributions. But if they are, then it was discussed how telemetry, alarms, and errors can be possibilistically weighted. A possibilistic FGM then would be responsible for producing as its output a set of fault hypotheses which are possibilistically weighted for input to the behavior model. This would in turn generate model prediction errors with possibilistic weights.

A system model which is a possibilistic automata can also be cast as a possibilistic Markov process. As such, its key component is its **transition matrix** $\boldsymbol{\Pi}$, essentially a vector of conditional possibility distributions as discussed in Sec. 3.1 and shown in Table 3.1. Each conditional possibility distribution represents the possibility, for a given input, of transiting from one system state to another.

The semantics of this transition matrix in a system model is understood in terms of a subsystem-level model where the conditional possibilistic weight indicates a non-additive coupling or relatedness among subsystems. This could be, for example, the efficiency of the subsystem, as in the approach of Doyle et al. [3]. Or, when considering the system model as a causal graph, as in the approach of Hall et al. [16], the weights indicate the degree of causal connectivity between subsystems.

Thus in the possibilistic approach a system model is essentially a possibilistic network, where nonadditive, possibilistic weights are placed on the arcs of a causal graph. The corresponding network appears similar to a Bayesian network, but the mathematics is possibilistic, not stochastic. It has been shown by the author [22] that such nonadditive possibilistic processes are actually the valid generalizations of nondeterministic processes, where stochastic networks are not.

6 Conclusion and Future Directions

We have considered possibility theory as a qualitative modeling method in the context of GIT (probability theory, Dempster-Shafer evidence theory, random set theory, and fuzzy theory). We have also defined the key concepts of model-based diagnosis, and have considered, in the context of spacecraft diagnosis, the potential for the application of possibility theory to MBD in terms of symptom and error detection, data fusion, sensor failure modeling, and nonadditive causal graphs.

It should be emphasized that this work is still in the basic research phase. Mathematical possibility theory is still being developed, and most of the key concepts in possibilistic modeling (for example, possibilistic measurement and automata) have only been defined in the past year. The application of possibility theory to the modeling of physical systems and the semantics of possibility in empirical contexts is being considered only by a few.

This work points to many future directions for research, including computer-based implementation of possibilistic models proposed by the author [23], and continued exploration of the conditions for the application of possibilistic modeling to spacecraft systems.

Acknowledgments

This work has been completed as part of a PhD degree in Systems Science from SUNY-Binghamton, and was supported under Grant # NGT 50756 from NASA-Goddard. I would like to thank Walter Truszkowski, the entire Code 520 group, as well as the University Affairs office at Goddard for their continuing help and strong support over the last three years.

References

[1] Bobrow, Daniel G ed.: (1985) *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge MA

[2] Davis, R and Hamscher, W: (1992) "Model-Based Reasoning: Troubleshooting", in: *Readings in Model-Based Diagnosis*, ed. W Hamscher et al., pp. 3-24, Morgan Kaufman, San Mateo CA

[3] Doyle, RJ; Sellers, SM; and Atkinson, DJ: (1989) "Focused Context-Sensitive Approach to Monitoring", in: *Proc. 11th Int. Joint Conf. on Artificial Intelligence*, Detroit

[4] Dubois, Didier; Lang, Jèrôme; and Prade, H: (1992) "Dealing with Multi-Source Information in Possibilistic Logic", in: *Proc. 10th European Conf. on Artificial Intelligence*, pp. 38-42, Vienna

[5] Dubois, Didier and Prade, Henri: (1988) *Possibility Theory*, Plenum Press, New York

[6] Durrant-Whyte, HF: (1987) "Consistent Integration and Propogation of Disparate Sensor Observations", *Int. J. Robotics Research*, v. **6**:3, pp. 3-24

[7] Dvorak, D and Kuipers, B: (1992) "Model-Based Monitoring of Dynamic Systems", in: *Readings in Model-Based Diagnosis*, ed. W Hamscher et al., pp. 249-254, Morgan-Kaufmann, San Mateo CA

[8] Fishwick, Paul A: (1989) "Qualitative Methodology in Simulation Model Engineering", *Simulation*, v. **52**:3, pp. 95-101

[9] Fishwick, Paul A: (1991) "Fuzzy Simulation: Specifying and Identifying Qualitative Models", *Int. J. of General Systems*, v. **19**, pp. 295-316

[10] Fishwick, Paul A and Luker, Paul A eds.: (1990) *Qualitative Simulation Modeling and Analysis*, Springer-Verlag, New York

[11] Fulton, Steven L: (1990) "Introduction to Model-Based Reasoning", *AI Expert*, v. **1/90**, pp. 48-55,

[12] Gertler, Janos J and Anderson, Kenneth C: (1992) "Evidential Reasoning Extension to Quantitative Model-Based Failure Diagnosis", *IEEE Trans. on Systems, Man, and Cybernetics*, v. **22**:2, pp. 275-289

[13] Grossmann, Wilfried and Werthner, Hannes: (1993) "Stochastic Approach to Qualitative Simulation Using Markov Processes", in: *IJCAI '93*

[14] Guariso, Giorgio; Rizzoli, Andrea; and Werthner, Hannes: (1992) "Identification of Model Structure via Qualitative Simulation", *IEEE Trans. on Systems, Man, and Cybernetics*, v. **22**:5, pp. 1075-1086

[15] Hackett, JK and Shah, M: (1990) "Multi-Sensor Fusion: A Perspective", in: *Proc. 1991 Int. Conf. on Robotics and Automata*, pp. 1324-1330, Cincinnati

[16] Hall, Gardiner A; Schuetzle, James; and La Vallee, D et al.: (1992) "Architectural Development of Real-Time Fault Diagnositc Systems Using Model-Based Reasoning", in: *Proc. 1992 Goddard Conf. on Space Applications of AI*, ed. Steve Rash, pp. 77-86, NASA Goddard, Greenbelt MD

[17] Hamscher, W; Console, Luca; and Kleer, Johan de eds.: (1992) *Readings in Model-Based Diagnosis*, Morgan-Kaufman

[18] Joslyn, Cliff: (1992) "Possibilistic Measurement and Set Statistics", in: *Proc. NAFIPS 1992*, v. **2**, pp. 458-467, NASA

[19] Joslyn, Cliff: (1993) "Empirical Possibility and Minimal Information Distortion", in: *Fuzzy Logic: State of the Art*, ed. R Lowen and M Roubens, pp. 143-152, Kluwer, Dordrecht

[20] Joslyn, Cliff: (1993) "Possibilistic Semantics and Measurement Methods in Complex Systems", in: *Proc. 2nd Int. Symposium on Uncertainty Modeling and Analysis*, ed. Bilal Ayyub, pp. 208-215, IEEE Computer Soc.

[21] Joslyn, Cliff: (1993) "Some New Results on Possibilistic Measurement", in: *Proc. NAFIPS 1993*, pp. 227-231, Allentown PA

[22] Joslyn, Cliff: (1994) "On Possibilistic Automata", in: *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, in press

[23] Joslyn, Cliff: (1994) "Object-Oriented Architecture for Possibilistic Models", to be presented at the 1994 Computer-Aided Systems Theory Conference

[24] Joslyn, Cliff: (1994) *Possibilistic Processes for Complex Systems Modeling*, SUNY-Binghamton, Binghamton, NY, PhD dissertation

[25] Kaufmann, A. and Gupta, M.M.: (1985) *Introduction to Fuzzy Arithmetic*, Reinhold, New York

[26] Klir, George: (1993) "Developments in Uncertainty Based Information", in: *Advances in Computers*, v. **36**, ed. M. Yovitz, pp. 255-332, Academic Press

[27] Klir, George and Folger, Tina: (1987) *Fuzzy Sets, Uncertainty, and Information*, Prentice Hall

[28] Kuipers, BJ: (1989) "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge", *Automatica*, v. **25**, pp. 571-585

[29] Luo, RC and Kay, MG: (1989) "Multisensor Integration and Fusion in Intelligent Systems", *IEEE Trans. on Systems, Man, and Cybernetics*, v. **19**, pp. 901-931

[30] Moore, RM: (1979) *Methods and Applications of Interval Analysis*, in: *SIAM Studies in Applied Mathematics*, SIAM, Philadelphia

[31] Shen, Qiang and Leitch, Roy: (1991) "Diagnosis Continuous Dynamic Systems By the Use of Qualitative Simulation", in: *Proc. Third International Conf. on Control*, v. **2**, pp. 1000-1006, Edinburgh

[32] Shen, Qiang and Leitch, Roy: (1993) "Fuzzy Qualitative Simulation", *IEEE Trans. on Systems, Man, and Cybernetics*, v. **23**:4, pp. 1038-1061

[33] Sugeno, Michio and Yasukawa, Takahiro: (1993) "Fuzzy-Logic-Based Approach to Qualitative Modeling", *IEEE Trans. on Fuzzy Systems*, v. **1**:1, pp. 7-31

[34] Zadeh, Lotfi A: (1978) "Fuzzy Sets as the Basis for a Theory of Possibility", *Fuzzy Sets and Systems*, v. **1**, pp. 3-28

AN INTELLIGENT CONTROL SYSTEM FOR FAILURE DETECTION AND CONTROLLER RECONFIGURATION

SAROJ K. BISWAS

Intelligent Systems Application Center
Department of Electrical Engineering
Temple University
Philadelphia, PA 19122
Phone: (215)-204-8403, FAX: (215)-204-6936
E-mail: SBISWAS@TEMPLEVM.BITNET

ABSTRACT

We present an architecture of an intelligent restructurable control system to automatically detect failure of system components, assess its impact on system performance and safety, and reconfigure the controller for performance recovery. Fault detection is based on neural network associative memories and pattern classifiers, and is implemented using a multilayer feedforward network. Details of the fault detection network along with simulation results on health monitoring of a dc motor have been presented. Conceptual developments for fault assessment using an expert system and controller reconfiguration using a neural network are outlined.

I. INTRODUCTION

With the increased demand for reliability and safety, there is a need for an intelligent restructurable control system which has the ability to detect a system fault as early as possible and restructure the controller in the event of a failure. In addition such intelligent control systems must operate in real time or near real time so as to be able to predict faults before their actual occurrence. The intelligent control systems should also be able to classify a fault in terms of its type and severity for the human supervisor or *an expert system* supervisor monitoring the performance of the system.

In this paper we propose an Intelligent Restructurable Control System consisting of three functional modules or subsystems: a) a Fault Detection and Isolation module, b) a Fault Assessment and Strategy Development module, and c) a Control Reconfiguration module. The fault detection subsystem utilizes a neural network that detects and classifies faults by associating the faults with patterns of sensor data. Fault assessment and strategy development require analyses of linguistic phrases and decision making so that an expert system is the best tool for the implementation of this module. Since control reconfiguration must be carried out in real time or near time, suitable table lookup or parallel computational facility is essential for controller reconfiguration. Thus this intelligent control system integrates three of the emerging technologies that have become synonymous with intelligent control, namely, neural network, expert system, and parallel processing.

The paper is organized as follows: Section II presents the intelligent restructurable control system architecture which is followed by development of the FDI neural network and simulation results in Section III. The paper is concluded with some remarks in section IV.

II. AN INTELLIGENT RESTRUCTURABLE CONTROL SYSTEM

A restructurable control system must have the capability to accommodate automatically for any unanticipated failure of system components, and to reconfigure the controller by way of a real time redesign of the control system. The key feature that is of fundamental concern in this respect is that the response time is limited. The complete restructurable control problem may be broken into three distinct but interrelated problems:

1) Fault detection and isolation (FDI),
2) Assessment of the FDI results and strategy development
3) Control reconfiguration, and

In this paper we propose an Intelligent Restructurable Control System as shown in Figure 1 consisting of three functional modules or subsystems: a) a fault detection and isolation module, b) a fault assessment and strategy development module, and c) a control reconfiguration module. Once the controller is redesigned following a failure, implementation of the controller must be carried out through available hardware/software. The following subsections outline the underlying principles of operation of each of the above modules.

2.1 Fault Detection and Isolation

Two key requirements for the fault detection system are:

a) It must operate in real time or near real time

b) It must classify the fault in terms of its type and severity.

The proposed diagnostic neural network methodology for failure detection is based on the analyses of patterns of plant sensor data. In this approach, a fault is conceptualized as a mapping or association of a pattern of the process data (measured through sensors) to a fault condition. Since neural networks are known to perform best as pattern classifiers, a neural network fault identifier is used in this FDI subsystem.

2.2 Fault Assessment and Strategy Development

Once the FDI subsystem has detected and identified a failure, an assessment of the FDI results must be made. As one may expect, this task may require analysis of linguistic phrases and decision making; thus an expert system is required in this subsystem. The expert system accepts information from the fault identification (FDI) neural network, and classifies the fault as 1) a survivable fault, or 2) a catastrophic fault. In the event of a survivable fault, the expert system will prescribe immediate notifications to the Control Reconfiguration subsystem.

2.3 Control Reconfiguration

The Control Reconfiguration subsystem actually redesigns the controller for the faulty system. For high speed dynamic systems, computation time for control redesign is an important factor in choosing the method to be used. A two stage control reconfiguration strategy is proposed in this paper: 1) a fast inner loop **stabilizing controller** for immediate stabilization of the faulty system implemented through neural network, and 2) a more accurate outer loop **performance recovery controller** implemented through vector/parallel processor. The inner loop controller stabilizes the system and brings it to safety. The outer loop controller recovers as much of the prefault system performance as possible [10].

In this paper we present the details of the fault detection and isolation subsystem. Research for implementation of the other two subsystems is in progress.

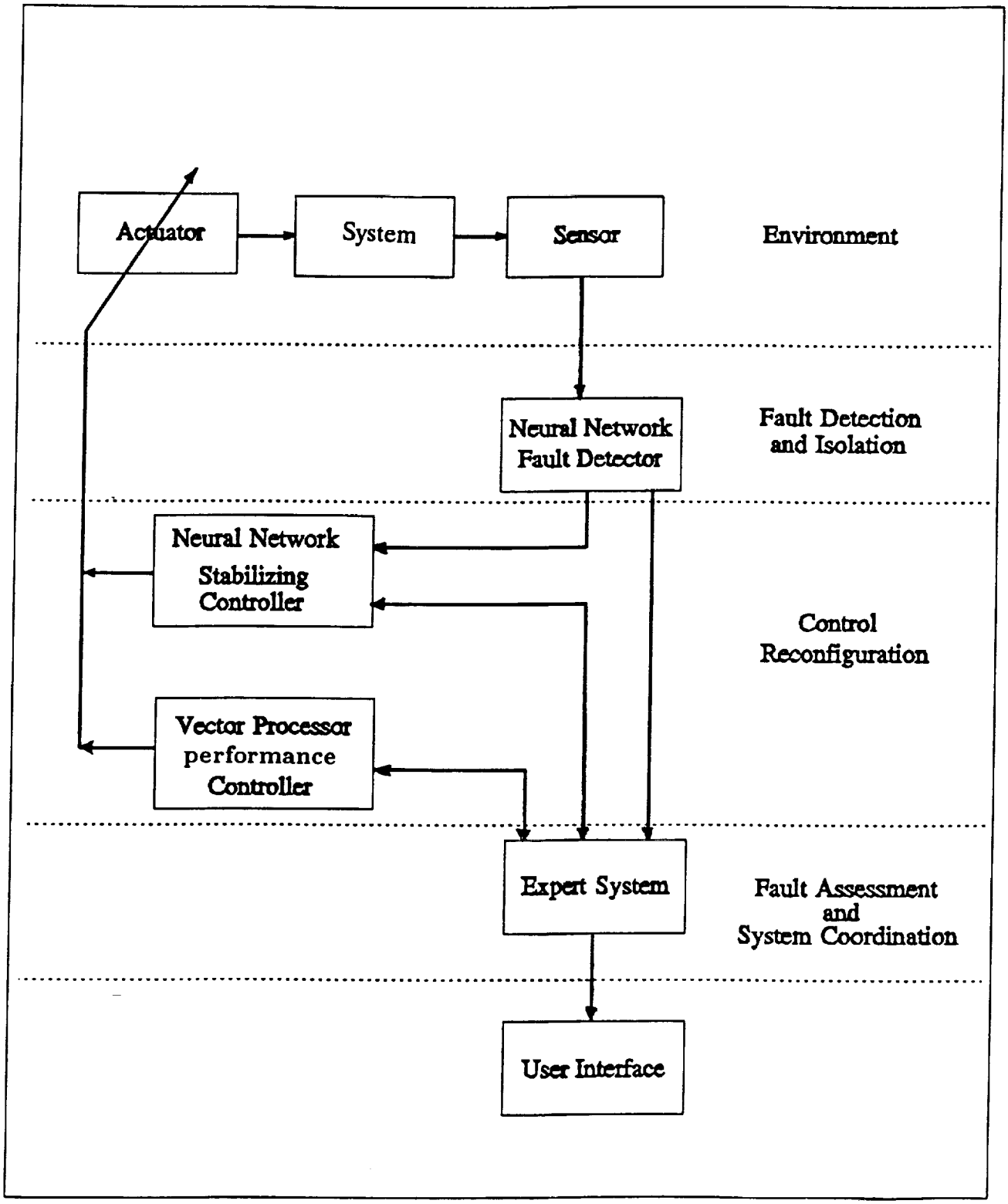


Figure 1. Architecture of the Intelligent Restructureable Control System

III. FAULT DETECTION AND ISOLATION

3.1 Background

Traditionally, human evaluation has been used as a primary means of fault detection, however this is grossly inadequate and prone to misjudgments. The earliest methods of failure detection have been based on limit violation and/or trend violation [1] of sensor data. Since human evaluation could be unrealiable, analysis of sensor data [2,3] to extract certain characteristic features of the process has been suggested. These analyses include computation of process efficiency, estimation of frequency spectrum, and the autocorrelation function, etc. requiring the use of Kalman filters, observers, and FFTs.

Another method of identifying a fault is by measurement or estimation of parameters [4,5,6,7] that govern the input-output mapping of the process. Least square estimation, Kalman filter [5], maximum likelihood estimation [6] are carried out to estimate the unknown parameters. However, this method suffers from lack of uniqueness which may lead to false alarms and/or wrong diagnosis of faults.

The use of neural network as an effective alternative for failure detection and classification to overcome the inherent drawbacks of the traditional methods has become a subject of current research interest. An application of neural networks for failure detection has been reported in [8,9] for the detection of incipient fault in a single phase induction motor.

3.2 Neural Network Based Failure Detection and Isolation

This section presents the proposed Failure Detection and Isolation (FDI) subsystem based on artificial neural network. The primary function of the FDI system is fault identification and determination of the *type of fault* and its *level of severity*. Fundamentally, a fault can be considered as a mapping or association from fault scenarios to patterns of sensor data. Neural networks are known to perform best as associative memories and pattern classifiers, and hence is used as the basic building block of the FDI system.

3.2.1 Fault Representation

One of the functions of the FDI subsystem is to classify a fault in terms of its type and its level of severity. The types of faults that may occur in a dynamical system are very much system specific. Levels of severity can be classified in several discrete conditions such as severe fault, mild fault, normal operation etc. In any case we can designate typical fault conditions in terms of a set, say $F_1, F_2, \cdots F_n$. This set of faults includes both *the type of fault* as well as *the severity of the fault*. For example: F_1 may represent mild (level of severity) damage of a steam valve (type of fault) while F_2 may represent a more severe version of the same fault or a different kind of fault altogether.

3.2.2 Network Architecture

Since the Failure Detection and Isolation system is based on associative memories and pattern classifiers, a multilayer feedforward network is considered as the basic network topology for the FDI system. The network is arranged into an input layer, an output layer, and one or more hidden layers. The input layer receives sensor data information from the process, and the output layer provides the fault diagnostic information.

The number of neurons in the input layer is equal to the number of sensors that carry out the measurement of various quantities in the process. The neurons in the input layer have linear input-output mapping. The neurons in the hidden layer(s) have a sigmoidal input-output map. The number of neurons in the

hidden layer(s) must be sufficiently large so that the fault library (or the stored fault scenarios) of the network is of an acceptable size for a given system.

The number of neurons in the output layer is equal to the number of bits of the binary number corresponding to the number of faults that must be identified by the FDI network. The neurons in the output layer have sigmoidal input-output mapping so that each neuron can take discrete values 0 or 1. The on-off status of the neurons in the output layer is translated to a fault code or fault number corresponding to certain type of fault at certain level of severity. For example, consider a system in which 16 faults $F_1, F_2, \cdots F_{16}$, are to be identified. Then the output layer must have 4 nodes.

3.2.3 Training

Training of a neural net is the process by which the values of the weights are determined based on historical fault data. For the purpose of training it is essential that historical or simulated fault data be available representing various types of faults along with their possible levels of severity. Training of the FDI network may be done using the back propagation [11,12] algorithm or other adaptive training algorithms [13]. Also, pruning algorithms can be used for the purpose of reducing the number of interconnection of the neurons in the network.

3.2.4 Example: Fault Monitoring of a DC Motor

The Failure Detection and Isolation (FDI) system described above has been used for monitoring of the health status of a dc motor. The state of the dc motor can be described in terms of three state variables, such as speed of rotation, armature current, and temperature. These are also physically measurable variables through appropriate sensors. The control input to the motor is the armature voltage.

It was desired to monitor the performance the motor in terms of five different types faults, such as

a) Faulty Controller
b) Partially shorted winding
c) Excessive bearing friction
d) Motor overload
e) Blocked ventilating system

at three levels of severity, such as

a) Mild
b) Moderate
c) Severe

Thus corresponding to the five different types of faults at three levels of severity, a total of 15 possible (single) faults must be identified along with one normal operating state. Each of these 16 faults are then assigned a unique binary number ranging from 0000 to 1111 with 0000 being the *no fault* and 1111 representing a severely faulted motor. Clearly this requires 4 nodes in the output layer of the FDI network. Since there are three measurable signals available from the motor, the input layer has 3 nodes, one for each sensor signal. The FDI network was assumed to have two hidden layers with eleven nodes in each layer. The number of nodes in the hidden layers were determined after some trial and error simulation.

In order to train the FDI neural networks to give correct responses a large data base of fault data was generated through computer simulation of the motor dynamics given in the appendix. Training of the network was completed after several training sessions. Figure 2 shows reduction of the rms error of the output nodes of the network as a function training runs. Convergence of a few of the synaptic weights are shown Figure 3.

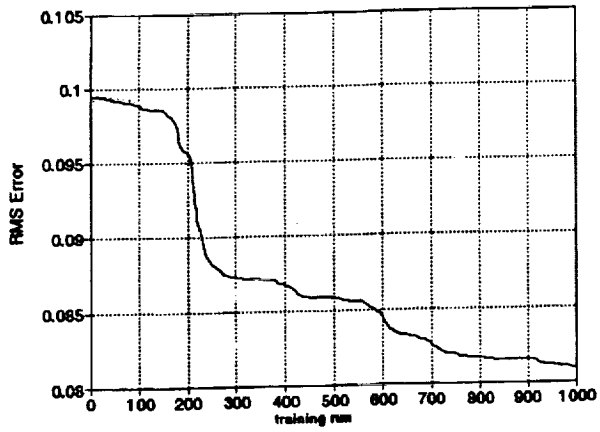


Fig. 2 Convergence of Training

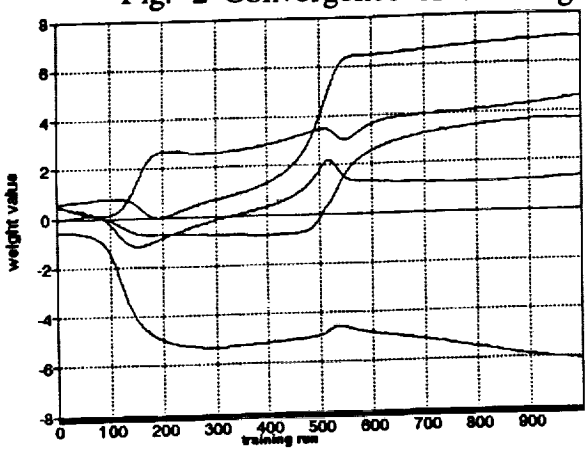


Fig. 3 Convergence of Synaptic Weights

After the completion of training, the FDI system was tested to monitor the health status of the motor. Various faults were simulated on the motor, and the fault data from the motor was applied to the input layer of the FDI network. The network compares the input pattern with the fault patterns it has been trained to recognize. The network was able to recognize the faults with 95% accuracy.

IV. CONCLUSIONS

We present the architecture of an intelligent restructurable control system that automatically identifies the occurrence of a fault, and restructures the controller for performance recovery.

The failure detection (FDI) subsystem is implemented using a neural network. The FDI network identifies the type of fault as well as its level of severity. A major advantage of this neural network based method is that it identifies faults that are characterized by changes in the numerical values of process variables as well as faults that are characterized by changes in certain physical attributes of the process.

APPENDIX

The dynamic model of a dc motor driving a load is given by

$$\frac{dI_a}{dt} = \frac{1}{L_a}V_a - \frac{R_a}{L_a}I_a - \frac{K_b}{L_a}\omega$$

$$\frac{d\omega}{dt} = \frac{K_i}{J_m}I_a - \frac{B}{J_m}\omega - \frac{1}{J_m}T_{\text{load}}$$

$$\frac{dT}{dt} = \frac{R_a}{C_{\text{Th}}}I_a^2 - \frac{1}{R_{\text{Th}}C_{\text{Th}}}T + \frac{1}{R_{\text{Th}}C_{\text{Th}}}T_{\text{ambient}}$$

where

R_a, L_a : Armature resistance, Inductance
I_a : Armature current
ω, T : Motor Speed, Temperature
K_i, K_b : Constants
C_{Th} : Thermal capacitance
J_m, B : Motor Inertia, friction coefficient
T_{Load} : Motor Load
R_{Th} : Rotor Surface heat transfer coefficient

VI. REFERENCES

[1] Isermann R., "Failure Detection Methods for the Supervision of Technical Processes", *Process Automation*, Vol. 1, pp. 36-44, 1981.

[2] Saedtler E., "Hypothesis Testing and System identification Methods for On-line Vibration Monitoring of Nuclear Power Reactors", *5th IFAC Symp. on Identification and System Parameter Estimation*, 1979.

[3] Williams M.M. and R. Sher, "*Progress in Nuclear Energy*, Vol. 1, Pergamon Press, Oxford, 1979.

[4] Mehra R.K. and J. Peschon, "An Innovation Approach to Fault Detection and Diagnosis in Dynamic Systems" *Automatica*, Vol. 7, 1971.

[5] Montgomery R.C. and A.K. Caglyan, "A Self Reorganizing Digital Flight Control System for Aircraft", *AIAA 12th Aerospace and Science Meeting*, Washington, 1974.

[6] Willisky A.S. and H.L. Jones, "A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems subject to Abrupt Changes", *Proc. IEEE Conf. on decision and Control*, Phoenix, Arizona, 1974.

[7] Pau L.F., *Failure Diagnosis and Performance Monitoring*, Marcel Dekker, New York, 1981.

[8] Chow M. and S.O. Yee,"Application of Neural Networks to Incipient Fault Detection in Induction Motors", *Journal of Neural Network Comput.*, Vol. 2, pp. 26-32, 1991.

[9] Chow M., Peter M. Mangum, and S.O. Yee, "A Neural network Approach to Real-Time Condition Monitoring of Induction Motors", *IEEE Trans. on Industrial Electronics*, Vol. 38, pp. 448-453, 1991.

[10] N. Palumbo, S.K. Biswas and B.P. Butz, "Recovery of Close-to-Nominal Pre-Fault Performance Using the Pseudo-Inverse Eigenstructure Assignment Method", *1992 American Control Conference*, 1992.

[11] Lippmann R.P., "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[12] Kohonen, T., An Introduction to Neural Computing, *Neural Networks*, Vol. 1, pp. 3-16, 1988.

[13] Helferty J.J., Biswas S.K. and Maund Mark,"Experiments in Adaptive Control using Neural Networks", *Proc. IEEE International Conf. on Systems Engineering*, Pittsburgh, PA, August, 1990.

Knowledge Engineering

Telecommunications Issues of Intelligent Database Management for Ground Processing Systems in the EOS Era

Joseph D. Touch

USC / Information Sciences Institute (touch@isi.edu)

ABSTRACT[1,2]

Future NASA Earth science missions, including the Earth Observing System (EOS), will be generating vast amounts of data that must be processed and stored at various locations around the world. Here we present a stepwise-refinement of the Intelligent Database Management (IDM) of the Distributed Active Archive Center (DAAC - one of seven regionally-located EOSDIS archive sites) architecture, to showcase the telecommunications issues involved. We develop this architecture into a general overall design. We show that the current evolution of protocols is sufficient to support IDM at Gbps rates over large distances. We also show that network design can accommodate a flexible data ingestion storage pipeline and a user extraction and visualization engine, without interference between the two.

1: Introduction

In addition to its manned space program, NASA runs telemetry-gathering missions. Among the celestial bodies studied is the Earth. Current and future Earth science missions (including EOS) will generate enormous amounts of data. This data must be archived in an accessible manner to be useful for analysis. EOS in particular will generate a continuous stream of 11.5 Mbps, which isn't notable except that the stream is relentless over the life of the satellite (about 5-10 years), resulting in 5.2 Gigabytes of data per hour, or 45 Petabytes (10^15 bytes) per year.

This data is processed prior to storage to facilitate access, and retrieved and converted into a useful form; these functions comprise the EOS Data Management Software System (DMSS), which is an example of a more general concept called Intelligent Database Management (IDM). Here we present an overview of the telecommunications issues of IDM, which involves data ingestion, storage, fusion, and rendering. Data ingestion is the processing of data prior to storage; data fusion is the combining of various streams of stored data to form a composite information base suitable for direct rendering. The components of IDM for EOS are distributed globally over large distances (over 2000 miles) and bandwidth (1 Gigabit/second). Thus telecommunications issues, including latency reduction, high bandwidth protocols, and distributed resource allocation are a fundamental component of IDM.

Here we present a stepwise-refinement of the Distributed Active Archive Center (DAAC - one of seven regionally-located EOSDIS archive sites) architecture. We also discuss how current protocols are sufficient to support the IDM DAAC. We describe a network design that accommodates both a flexible data ingestion storage pipeline and a user extraction and visualization engine.

The most abstract description of DAAC is a set of continuous satellite data input streams (between 16 Mbps and 26 Kbps, totalling 25 Mbps average, 164 Mbps peak), and a 200-500 Mbps sporadic user visualization stream, with low BW user commands. Internally, the input and output are related only by storage, i.e., the input stream archiving and output stream generation are independent. We partition the continuous input archive stream (ingestion) from the user command and visualization streams (extract), both of which operate on the data store. There also may be multiple ingestion and extraction streams per DAAC. The general design proposed uses separate subnetworks of heterogeneous processors - one for ingestion and the other for extraction. The processors and subnetworks form a dynamically-configurable dataflow engine, where subnetwork partitioning inhibits interference and provides recon-

1. This paper is based on a consulting report the author prepared while at the University of Pennsylvania. A version of this report will appear as part of the "HPCC Data Management white paper," of the NASA GSFC Information Science and Technology Office (ISTO). - Code 930.1.

2. This research was partially sponsored by the Advanced Research Projects Agency through Ft. Huachuca Contract No. DABT63-91-C-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Army, the Advanced Research Projects Agency, or the U.S. Government.

figurability. We show that the telecommunications aspects of IDM can be managed by this physical resource partitioning.

More importantly, we show that existing protocols, or existing proposals to evolve these protocols, are sufficient to support IDM. There is a growing controversy in protocol research involving the use of existing protocols for high speed (Gbps) wide area (2,000+ mile) environments. There are several protocol issues involved, including soft real-time delivery (i.e., jitter control), guaranteed bandwidth (reservation), and accommodation of high bandwidth-delay product links. Issues under investigation (without evolutionary solutions) include hard real-time delivery (scheduled delivery constraints), and methods for latency reduction. IDM data processing (both ingestion and visualization) requires isochronous data transfer, i.e., controlled jitter in transmission and processing. Fortunately, the data collection is automated and uses loosely-coupled feedback from ground control, rather than from user visualization. Latency reduction affects only the user visualization control loop. The user-perceived latency is likely to be affected more by the extraction processing latency than by the propagation latency (typically 100 ms). Existing evolutionary modifications to existing data transport protocols accommodate soft real-time transfer (RTP), bandwidth reservation (ST-II, RSVP), and high bandwidth-delay product links given continuous data streams (TCP extensions for LFN's).

Some of the initial documents of the IDM project have described various aspects of the project, but none has considered the specific telecommunications aspects within this project, or the impact of those issues on the other design considerations. This document section is an effort to augment those discussions sufficiently to suggest a design of the telecommunications system, from which other design criteria implications will be readily evident.

2: DMSS Background

The DMSS project is described by a set of documents that address the overall structure of the EOS and EOSDIS projects, information management, query derivation from user directives, data modeling, internal processing, and database and computation scaling of the IDM DAAC system. None yet includes a discussion of the telecommunications issues involved or of the implications of those issues on other design criteria. This is partly because telecommunications research is relegated to other projects of the HPCC effort, and because the telecommunications issues may not require original research.

One document describes the scaling issues of the database and processing components of the project, but admits that the processing load cannot be accurately determined a priori [12]. This indicates that a scalable processing solution is required, one in which dynamic load configuration is possible.

Others describe the static issues of database and visualization access to the EOSDIS [5], or the data distribution and archiving requirements [6]. There are dynamic corollaries to these static issues that describe the reconfiguration of the system in a flexible way.

Management considerations mandate a relatively centralized facility or small set of facilities (the DAACs) [7]. Relieving a centralized load requires a distributed facility, provided that the data distribution is not orthogonal to the geographic configuration. Because the DAAC facilities are geographically distributed, processing within the DAAC should occur at a MAN or LAN scale. The partitioning of data into functional and operational sets among the DAACs indicates that inter-DAAC access, i.e., processing requiring the participation of more than a single DAAC, would be unlikely at first.

Data modeling is described using spatial, spectral, temporal, etc. characteristics [1]. This includes a description of the effects of the variation in access method on the storage organization. These descriptions can be easily augmented to include communications and dynamic post-processing costs, so as to describe the telecommunications effects on data organization as well. The only difficulty is that the telecommunications costs are distributed, whereas the access method frequencies proposed are local to a particular DAAC component.

Finally, the high-performance processing of satellite data before initial archiving requires the use of specialized equipment and systems [2], [9]. The individual board design of these components and the functional decomposition into processing elements is a scalable solution [9]. The board integration currently relies on existing technologies for system design (VME/VSB), rather than on true networking of components. The resulting pipeline permits chaining of processing elements within a single processing node (i.e., VME/VSB backplane), but not among different backplanes. Further, components of a single system can be used to process at most a single data stream, thus prohibiting an ultimately flexible design.

3: Observations

There are other observations that affect the overall telecommunications recommendations as considered herein. These include the software issues regarding protocols and support, and topological issues.

3.1: Telecommunications software issues

Many of the software issues are already being considered at various levels of the EOSDIS effort. The software can be partitioned into 5 main areas: front-end user interface intelligence and expert systems, back-end satellite data processing before archiving, archival processing for data management, and extraction processing execution. Each of these areas has implications for the telecommunications organization, but the extraction processing is especially influential.

The front-end user system involves expert systems [10], connectionism [8] or neural networks [3], and support for user visualization tools. All of these are user-level front-end issues, and can require sharing of scheduling information at the user-access level. This indicates a network of user-support systems, with loose coupling of state information on the availability of back-end resources, and other competing front-end sessions.

The back-end satellite processing system involves the use of specialized hardware [2], [9]. Each of these systems can be independent, as it processes a specific data stream from a given satellite individually.

The data processing for archiving can also be independent because of the partitioning of the databases among the DAAC sites [7]. Data reorganization is presumed to occur within operational units of the DAACs [1].

The extraction processing, however, has not been thoroughly considered in relation to the system design [12]. Current distributed systems design indicates a back-end network of dynamically-allocated processing elements, which can be configured according to the extraction processing needs of the users. Such a system is exhibited by a networked version of the back-end satellite processing system, with a few modifications (see below). The goal is a back-end network for the servicing of the user-level requests, according to the decompositions suggested by the expert systems at the front-end.

There are other services that are of use, especially in the software portion of the system. Current protocol technology is sufficient to support the data rates and characteristics of the large linear streams of information indicated by satellite measurements. These include TCP/IP, for streaming data transfer. Other options include remote evaluation (late-binding RPC), and the conventional remote procedure call (RPC). Conventional RPC requires sending the data to a remote site and retrieving the results, a mechanism that describes the dynamic allocation of processing components but requires a central controller to scatter and gather the data, creating a communications bottleneck that existing networks cannot support. Late-binding RPC permits the processing components to transmit the results along to subsequent RPC's, rather than requiring collection of the results at the originator of the first RPC. This permits a dynamic pipeline to be created within an existing telecommunications paradigm.

3.2: Network topology and protocol

Most of the telecommunications issues might be relieved by a LAN implementation. Exceeding LAN scale incurs a sharp decrease in data transmission rates. Existing routing, broadcast, and network management implementations also favor LAN scales. One solution is to distribute the access load among the components of a LAN, particularly existing high-speed LANs such as FDDI (100 Mbps) or FDDI-2 (200 Mbps). The FDDI protocol does not scale beyond a 100 meter diameter, but this is sufficient to support the locality of an individual DAAC.

The bandwidth requirements of this network (1 terabit/day) average out to a continuous 11.6 Megabits/second [12]. Conventional LAN technology (i.e., Ethernet) supports 10 Mbps, but only to a maximum of 80% load, i.e., 8 Mbps [11]. This theoretical maximum assumes a single source station on a network; competition among multiple sources decreases this to 60% (6 Mbps) [11]. Even a lightly loaded Ethernet is therefore unsuitable for even a single hop in the data path from satellite to disk storage.

New fiber optic LAN technology (FDDI) supports rates of 100-200 Mbps, large enough to support several simultaneous hops of the data stream if that stream is buffered and averaged over a 24-hour period (requiring 1 Terabit of tape delay). Burst data characteristics are not understood at this time, either within the EOSDIS system or in telecommunications in general, and so are not factored into any solutions.

3.3: Gigabit protocol issues

There are some relevant gigabit protocol issues that affect the design of the DMSS telecommunications system. These include protocol optimizations, rate control methods, and lightweight protocols.

Protocol optimizations are useful in the processing of stream data at gigabit rates; these include methods of header prediction in TCP and factoring frequent header cases out of the protocol stack. Other optimizations in the implementation of TCP have shown the operation of this protocol at rates near 400-700 Mbps, easily supporting both the satellite ingestion and user visualization components of this system.

Rate control methods provide processing adjustment to reduce queuing requirements, and reduce resulting jitter in the packet flow. 'Stop-and-go queuing', 'Leaky bucket', and 'Virtual Clock' are all similar methods for rate adjust-

ment. However, none are included in currently stable transport protocol implementations as yet.

Another protocol of interest, especially in the Data Ingestion operation, is the XTP protocol. XTP is a lightweight protocol that is designed to be implemented in VLSI hardware.

Other methods of achieving high performance protocols are not required here. These include other lightweight protocols, such as protocols that support fast RPC, or low latency transactions, or methods to reduce protocol complexity. The frequency of transactions in the DMSS system is not high enough to warrant these new protocols.

3.4: Suggestions for processing node scalability

One component of the DMSS system, the processing node architecture, is currently based on the VME/VSB bus interconnection [2]. A more flexible solution would use high-speed LAN interconnection methods, such as crossbar switches, for "backplane" communication among processing components.

One suggestion for possible research in this area would be the development of a VME/VSB virtual backplane, one which would permit the arbitrary interconnection of board-level components but exhibit a conventional VME/VSB interface. This crossbar-backplane would permit multiple FDDI interfaces per system aggregate, and thus multiple LAN loops among systems, permitting a more flexible implementation. This latter solution could be implemented incrementally, after the initial implementation phase of the design.

4: Stepwise-refinement

A first step toward the understanding of the communications structure is the stepwise refinement of the system design. These steps are based on the given NASA documentation, and general principles of system design.

The easiest implementation would be a LAN. The problem is that there are two load issues: direct query response load (computation and retrieval of actual data), and meta-data issues such as scheduling, load evaluation, and security access. Security issues are best solved by a physical partitioning of the network, with a coordinated set of controlled access points. The external access points comprise a set of nodes that interact through a separate meta-data LAN. Precomputed plans are sent through the controlled access point to the inner LAN for execution of the extraction.

Data ingestion occurs on the way into the inner data storage LAN, but does not use the outer security/scheduling LAN for access, since satellite-originated paths are presumed secured at the source. Further, the process of ingestion need not alter the meta-data storage until archived inside the inner LAN.

The best way to understand these observations is to see their evolution and extraction from the existing characteristics of the DMSS system. Here we present a step-wise refinement of the telecommunications structure. We also present a description of the data flow and meta-data flows of the system, all to finally define the characteristics of the system sufficient to indicate a design.

4.1: Step 1 - the most general description

The most general description of the EOSDIS system is as an operational entity. Consider the DAAC as a 'black-box', with inputs and outputs [6] [7]. Inputs are comprised of the satellite data stream and the user queries. The output is the user visualization stream. The size of the arrows and lines is representative of the qualitative relative bandwidth requirements.

The input satellite data stream of 1 Terabit/day averages out to 11.5 Mbps (Figure 1). The user commands require negligible bandwidth, both because of their small textual content and their sporadic nature. The user visualization estimate is based on a 1000x1000 pixel display, changing at a rate of 24 frames/second (movie-quality video), at a depth of between 8 bits/pixel and 24 bits/pixel. This results in a session bandwidth of between approximately 200 and 600 Mbps, or 8-24 Mbits per frame.

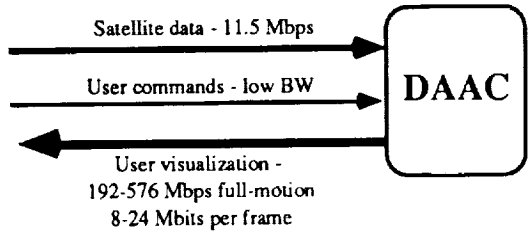


FIGURE 1. Step 1: System input/output

The input satellite stream thus requires a T-3 signal line (45 Mbps), assuming the 1 Terabit/day rate can be smoothed to per-second equivalent. The user commands can be accepted over conventional modem/dialup lines. The raw visualization stream requires SONET STS-12 rates, which are unlikely to be available for user deployment in the time-frame of this project. A lossless intraframe compression at these rates may be available, and would result in a 20-40 Mbps stream, which could be supported by FDDI LAN technology (100 Mbps). Lossy compression, such as JPEG, can further reduce this requirement to the Ethernet LAN realm at approximately 4-8 Mbps.

4.2: Step 2 - partition processing / data

The next step in this refinement involves the partitioning of the system component into data and processing components (Figure 2). The DAAC design is readily partitioned in this manner. This partition is modeled after the so-called 'Von-Neumann' computer architecture design.

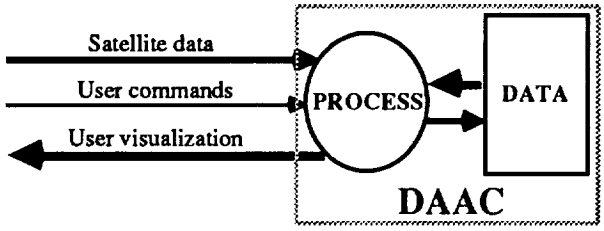


FIGURE 2. Step 2: Von Neumann decomposition

The processing requirements of this diagram are described in [12]. At this point, the processing and communications requirements are not sufficiently specified to determine the design, as was noted in the NASA analysis [12]. At this point, it is evident that this partitioning is not optimal, because the satellite data input stream and user visualization streams are largely independent, yet are processed in a single entity.

4.3: Step 3 - separate input / output streams

The next step in the refinement includes the description of the physical and algorithmic components. The processing is partitioned into ingestion, command processing, and extraction components (Figure 3). The ingestion portion occurs in specialized hardware [2]. The command processing translates user input into algorithms for extraction, which are executed in the extraction component [3]. By this diagram, the I/O intensive components are the ingestion and extraction [4], but there is substantial computation involved in the translations done by the command processing as well [8], [10].

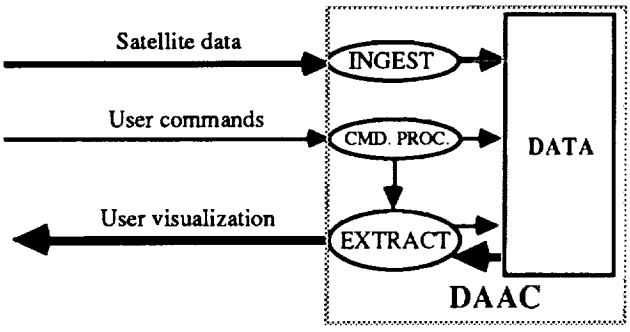


FIGURE 3. Step 3: Internal input/output streams

User commands therefore interact with the extraction process, but not the ingestion, which can be relegated to a separate component. Further, because the database is responding largely to command information (vs. data) from the command and extraction interfaces, the database might benefit from a partitioned internal structure, so that data input into an unorganized archive component can be isolated from the extraction access bandwidth requirements.

4.4: Step 4 - Separate into physical components

The final step in the refinement is the addition of partition information from the known implementation of existing components. The ingestion engine is known to be composed of a number of pipeline stages with separate control and pipeline communication paths [2], [9] (Figure 4). The database is also known to be composed of metadata indicating the semantic modeling of the data structure, and an auxiliary processing element to monitor this modeling, in addition to the data itself [1].

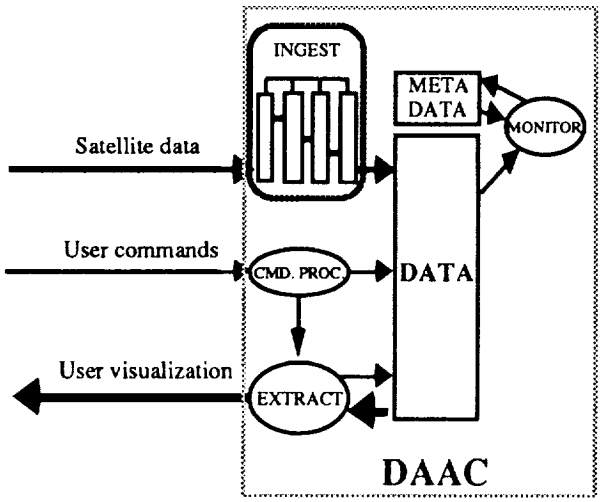


FIGURE 4. Step 4: Internal physical components (as already specified)

4.5: Step 5 - Replicate interior components

This final step in the refinement indicates that the user interface issues can be considered independent of the satellite data ingestion procedures. The decomposition does not yet indicate the systems issues involved, because only single users and satellite streams are indicated. We can augment the structure further by adding multiple copies of each entity, to denote how the replicated components interact.

We do not replicate the database component of the DAAC because we consider user requests within a single DAAC only. This is reasonable because the data of the DAACs is partitioned based on expected use and semantic content. Merging of data streams may occur, but is expected to be managed by the merging of independently delivered visualization streams from a number of independent DAAC sources.

Figure 5 denotes the interaction between the command processing elements in a scheduling capacity. The extraction processes are shown as independent, because determining overlapping computation is intractable and of little benefit given independent user control. The satellite processing streams are independent, but the number of pipeline stages is flexible and may be allocated from an aggregate rather than within dedicated system sets.

The decomposition shown here indicates the components of interaction and the bandwidth characteristics between them. It is also useful to view the data streams by semantic partitioning, in data flow and meta-data flow diagrams within the same structure.

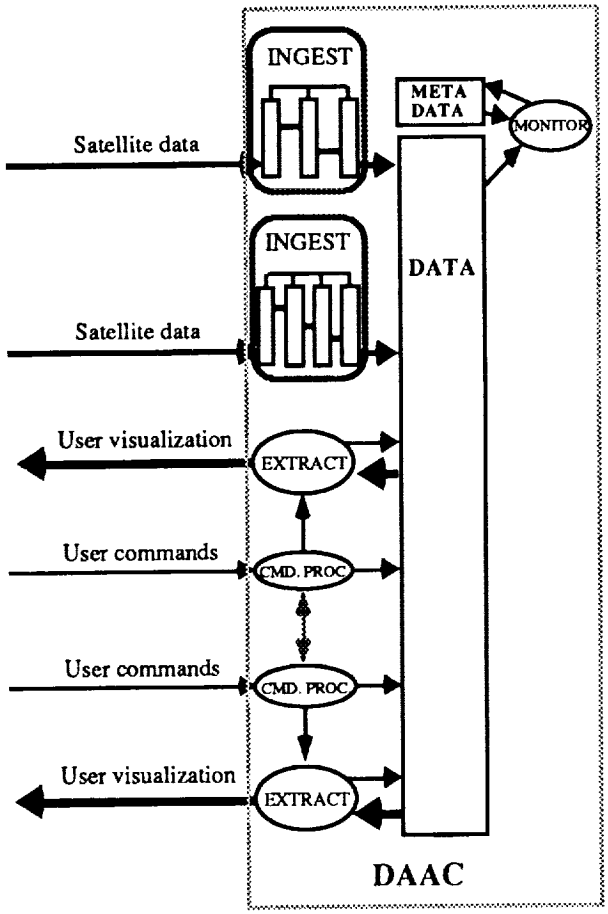


FIGURE 5. Step 4: Internal physical components (as already specified)

4.6: Data flow diagram

The data flow of the system can be described by two diagrams: one indicating the satellite data, the other indicating the archived data. The control operations specified by the user input are considered meta-data flows, shown later.

The satellite data flow consists of 1 Tbps streams pipelined through archival processing systems (Figure 6). If these systems are statically specified, the existing design of fixed-pipeline configuration will suffice [2]. If the satellite processing components are dynamically allocated, a network must be established among the elements. We assume here that these processing stages are largely static because of the data that would be lost during any reconfiguration. Thus, the satellite data flows represent fixed interconnections beyond the underlying dynamic network design. Further, scalability is provided by the addition of separate processing systems for additional satellite data streams in an independent fashion with linear cost. The streams can be compressed from the satellite to the pipeline processor, but the inter-process bandwidth requirements do not necessitate intermediate compression prior to storage.

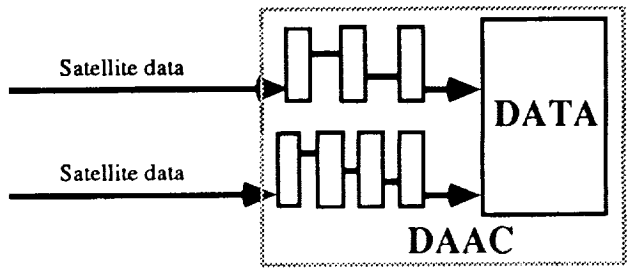


FIGURE 6. Satellite data flow (11.5 Mbps) - fixed interconnection

As a side-bar, we note that compression of archived data may have unanticipated effects on the communication load of the user visualization processing, as well as affecting retrieval. Extraction of particular information is complicated by stream encoding because it may require the decoding of a large section of data to locate a particular item, especially if the encoding destroys the key information. Effort should be made to avoid this if random access is required.

Further, a variable bit-rate encoding may cause fluctuating loads on the storage and extraction processes. While the storage process may be able to accommodate this fluctuation, the output data fluctuation will generate variable bit-rate streams to the extraction processors, which will require jitter control to permit stream merging. Recent research has also indicated that variable bit-rate streams

can cause interference effects in networks, even when the streams do not directly compete for resources.

The user visualization may require an arbitrary amount of pipelining, merging, and interleaving of extracted archive information (Figure 7). Whereas the individual streams are independent to permit independence in user control, the allocation of resources to the extraction processes is necessarily highly dynamic. The resources of extraction processes should be part of a dynamically reconfigurable network so additional resources can be added for additional functionality or scale of service.

Processing the streams usually occurs in the uncompressed domain, so any compression should occur at the final stage before user output. As a result, compression cannot be effectively used to reduce internal network load. The resulting interaction requires a very high bandwidth, very high connectivity network, such as BISDN (i.e., ATM).

If the user visualizations are restricted to conventional resolutions (500x500 at 1-8 bits, rather than 1000x1000 8-24 bits deep), the data streams are reduced from 200-600 Mbps to 6-50 Mbps, at full-motion 24 frames/second. While these streams cannot be accommodated in even a single Ethernet hop, a modified FDDI ring can be used.

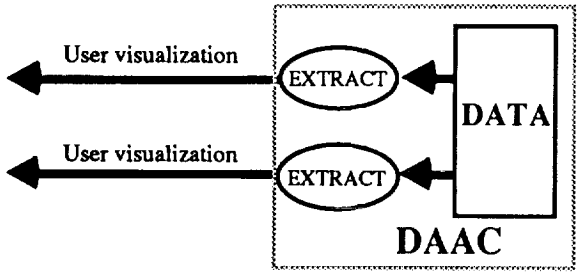


FIGURE 7. User visualization flow (200-600 Mbps raw / 5-12 Mbps compressed)

Consider the dual-ring FDDI. Each level of the ring can accommodate 100 Mbps. There are some recent protocol systems which permit the utilization of multiple segments of the ring simultaneously; this would permit sequences of processors on the ring to be configured as a pipeline, and the output would be collected on the other ring. The result would permit redistribution and configuration of extraction processing resources within the ring.

If the visualization stream is not full-motion or full-color, the bandwidth required would be reduced even further. Also, it is not clear at this time whether the full bit-rate is required during extraction or is the result of data stream merging costs, the latter of which could be transmitted to the user in a repeating loop.

4.7: Meta-data flow diagram

The other flows denote the control streams. Some of these streams are user-specified control, and others are control between components of the system (Figure 8). These are not high-bandwidth paths that dominate the network design, but are communication paths which must be provided, at least transitively, by the interconnection topologies.

These streams include: interaction among the pipeline elements, monitoring of archive access for dynamic reorganization, user commands, extraction commands, database retrieval commands, and communication between the command processors for distributed resource allocation.

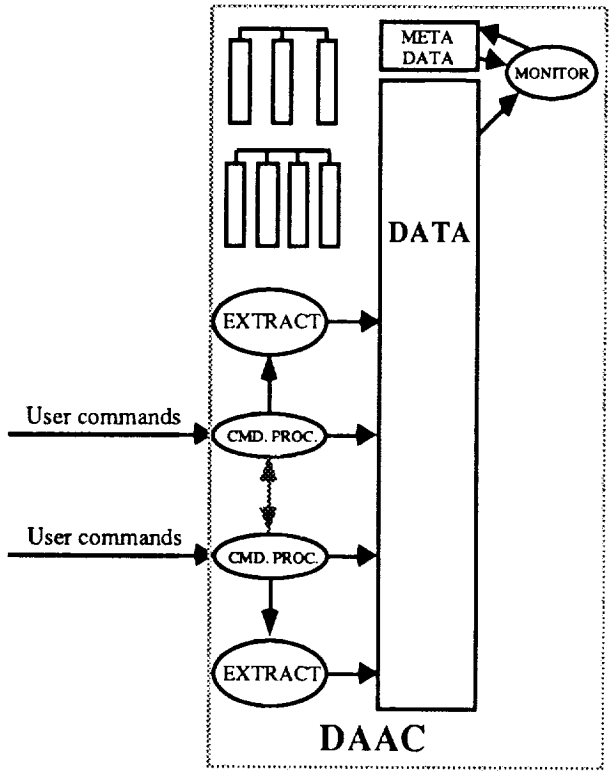


FIGURE 8. Meta-data (control, management) flow (low bandwidth)

4.8: Indicated design

The following are the recommendations for the design of a network for a DAAC system that is flexible, scalable, and secure.

There is a multiple ring structure. The rings comprise the data input, query transformation, and data processing and output components of operation. By separating the structure thus, the satellite processing is partitioned from the user-level operations and the query processing is parti-

tioned from the internal extraction operations. The former provides a robust isolation between data input and output and the latter provides a similar isolation of user and data processing. The result is a robust and secure system.

The distribution of satellite processing resources in a high-speed ring (FDDI II) or BISDN network (ATM) provides enhanced pipelining capability, scalability, and dynamic reorganization of resources not afforded by the current, fixed interconnection within individual backplanes [2]. This requires the use of emerging FDDI II protocols supporting the simultaneous use of multiple ring segments, sometimes called 'multiple tokens'. This feature of the network design was emphasized by the meta-data description of the system.

The distribution of user query processing components among a low speed ring or bus (Ethernet, for small distances, or token ring for larger distances) provides links among the command processors to support distributed resource allocation at scheduling time [10], [8]. This interaction was indicated by the step-wise refinement method.

The dynamic allocation of computation elements for extraction processing is similar to the satellite processing ring, i.e., both indicate an FDDI II modified multi-token ring or an ATM switching system (supporting full interconnections via a SONET-rate crossbar or multistage interconnection network). In the case of the extraction processing, the status of processors must be monitored by the query processing network for resource allocation. The idea is that the resources of the high-speed extraction network are allocated 'out-of-band', at scheduling-time, in the query processing network.

The monitoring of the database usage and structure can also occur within the query processing network, because it is a resource re-allocation function.

The result is a system that is composed of three networks: one isolated multi-token FDDI II network or ATM switching system for satellite data processing and a slow query processing network linked to another fast extraction processing network. Security is enforced in the slow query processing network by nature of its physical partitioning from the other two networks.

The general structure is visualized and instantiated with canonical networks in Figure 9. The basic description is as follows. A control console/host computer to each network is assumed.

The satellites are connected to SatNet with T-3 (45 Mbps) lines. The Pipe Processors are as described in the Ingestion portion, modified to provide a network interface, rather than a VME/VSB interface [8]. SatNet is either an FDDI II multi-token ring, or (optimally) an ATM BISDN network, providing full crosspoint interconnection with rates of STS-3 (155 Mbps) to STS-12 (620 Mbps). Until such technology is commercially available, a conventional analog crossbar can be used, because the connections within this network are not frequently modified.

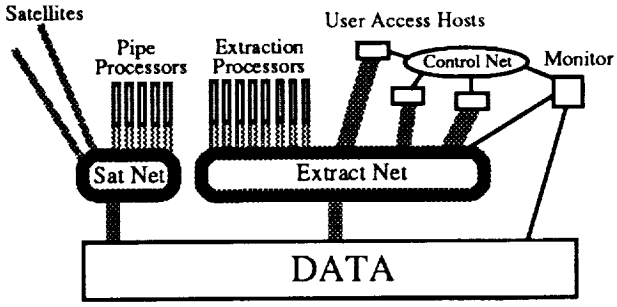


FIGURE 9. Generalized structure of telecommunications of EOSDIS DAAC

The same Pipe Processors can be used as extraction engines, with downloadable programs, or by workstations, as available. The design of ExtractNet supports heterogeneous systems, including supercomputers, workstations, and special-purpose Pipe Processors (as in SatNet).

The extraction processors can be connected to an ATM BISDN switch to implement the ExtractNet component. The ExtractNet should not be implemented with FDDI II or a crossbar, due to the highly dynamic reconfiguration that needs to occur to support varying user-specified extraction processes. The ExtractNet has a high bandwidth link to the database and another to each of the user-host processors. This latter link supports individual visualization streams.

The user access hosts are connected via a relatively conventional token ring, such as FDDI, or even Ethernet. Commands and resource allocation are processed on this network; these are low-bandwidth activities. It is assumed that one user-host will support each user connection because of the bandwidth required per user connection for high quality full-motion video. If still video is used, multiple users can be supported per station.

The ControlNet is used for distributed resource allocation among the user hosts and out-of-band resource allocation of the components of ExtractNet. SatNet allocation can occur off-line because the network is reconfigured only periodically.

A separate monitor host performs low bandwidth computations, such as database restructuring for performance [1]. The design of the database to support dual high-bandwidth ports, or possibly multiple high-bandwidth retrieval ports to ExtractNet, is beyond the scope of this section.

User access is restricted to the ControlNet, where queries are processed within the hosts, or possibly off-loaded into the Extraction Processors of ExtractNet, or a separate high-performance engine connected to ControlNet (like

the monitor). The access control is both physical and logical, so that the user commands are prohibited from utilizing the ExtractNet or SatNet. The interaction is similar to that of RPC, where user commands are decomposed into fixed, preexisting procedures that are pipelined together. User access is as secure as in RPC.

5: Conclusions

Here we have presented an architecture for the DMSS of the IDM DAACs developed by stepwise refinement. We have discussed how existing protocols are sufficient for use in this architecture to support both data ingestion and data fusion and visualization.

The DMSS architecture presented is scalable, partitions the DMSS via gateway access servers, and includes internally replicated processing components. We have also shown a design in which control is distinct from data streams, both logically and topologically.

The architecture we show permits various implementations:

- gateway as authenticator only,
  remainder as centralized server.
- gateway as delegator
  using vector pipelined REV processors.
- gateway as authenticator
  using REV on workstations.

It is this latter approach we feel is most general, scalable, and useful for the architecture of the DMSS IDM DAACs.

6: Acknowledgments

We would like to thank the members of the NASA GSFC Information Science and Technology Office (ISTO). - Code 930.1, notably Nick Short, as well as Bob Cromp, and Bill Campell, for their input into this work. We would also like to thank Jim Chesney of NASA for providing detailed information on the DMSS ingestion hardware. We also thank the members of USC/ISI HPCC Division for their constructive comments on this document.

7: References

[1] Campbell, W., and Short, Jr., N. "Using Semantic Data Modeling Techniques to Organize an Object-Oriented Database for Extending the Mass Storage Model." International Astronautical Federation 42nd Congress, Montreal, October 1991.

[2] Chesney, J., Speciale, N., Horner, W., and Sabia, S. "High Performance VLSI Telemetry Data Systems." Amer. Inst. Aeronautics and Astronautics, Second Int'l. Symp. on Space Information Systems, Pasadena CA, September 1990.

[3] Cromp, R. "Automated Extraction of Metadata from Remotely Sensed Satellite Imagery." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991.

[4] Cromp, R., Campbell, W., and Short, Jr., N. "An Intelligent Information Fusion System for Handling the Archiving and Querying of Terabyte-Sized Spatial Databases." International Space Year Conference on Earth and Space Science Information Systems, Pasadena CA, February 1992.

[5] McDonald, K. "Information Management Challenges of the EOS Data and Information System." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991, pp. 258-267.

[6] Quirk, B., and Thompson, R. "Early-EOS Activities at the Land Processes Distributed Active Archive Center (DAAC)." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991, pp. 339-351.

[7] Ramapriyan, H., and McConaughy, G. "Version 0 EOSDIS - An Overview." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991, pp. 352-362.

[8] Short, Jr., N., and Shastri, L. "The Application of Connectionism to Query Planning/Scheduling in Intelligent User Interfaces." Telematics and Informatics, Vol. 7, Nos. 3/4, 1990, pp. 209-220.

[9] Shi, J., and Grebowsky, G. "A Performance Model for Real-time Packet Processing." Mission Operations and Data Systems Directorate paper, NASA GSFC, 1990.

[10] Short, Jr., N. "A Real-Time Expert System and Neural Network for the Classification of Remotely Sensed Data." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991, pp. 406-418.

[11] Tanenbaum, Andrew S. *Computer Networks*. Second Ed., New Jersey: Prentice-Hall, NJ,1988.

[12] Wharton, S., Chang, H., and Krupp, B. "Sizing the Science Data Processing Requirements for EOS." *Technical Papers ACSM-ASPRS Annual Convention, Vol. 3 - Remote Sensing*, 1991, pp. 478-487.

Group-Oriented Coordination Models for Distributed Client-Server Computing

Richard M. Adler and Craig S. Hughes
Symbiotics, Inc.
725 Concord Avenue
Cambridge, MA 02138

ABSTRACT

This paper describes group-oriented control models for distributed client-server interactions. These models transparently coordinate requests for services that involve multiple servers, such as queries across distributed databases. Specific capabilities include: decomposing and replicating client requests; dispatching request subtasks or copies to independent, networked servers; and combining server results into a single response for the client. The control models were implemented by combining request broker and process group technologies with an object-oriented communication middleware tool. The models are illustrated in the context of a distributed operations support application for space-based systems.

INTRODUCTION

The dominant architecture for distributed systems today is the client-server interaction model. One application, the client, requests a service from a single provider, or server, which performs the desired task and returns the result to the client (cf. Figure 1). Examples of servers include transaction database systems, specialized graphics and numeric processing engines.

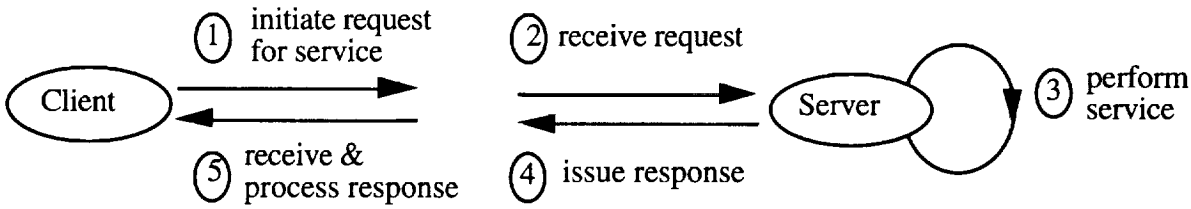


Figure 1. Client-Server model for distributed interaction

As distributed systems grow in complexity and scope, clients often need to interact with multiple servers. For example, a client may issue queries for information that is distributed across multiple, independent databases. Similarly, one application may need to notify various other programs that may be affected by its activities. Such one-to-many interactions are becoming increasingly important in domains such as decision and operations support, concurrent engineering, cooperative workgroups, office automation, and process control.

A special class of one-to-many interactions involves servers that can perform the same task(s), or *functionally redundant* systems. The most obvious form of functional redundancy is exact replication, such as distributing multiple copies of an application or database across a network of computers. Redundancy can also be achieved by replicating functionality through alternative technologies or methods. For example, intelligent advisory systems can be implemented using neural nets, rule-based or model-based reasoning systems. Combining such alternative methods exploits their complementary strengths, and can enhance the completeness, precision, and certainty of service responses.

Functional redundancy facilitates fault tolerance and resource availability, which are important attributes for mission-critical applications. Replicating servers or data resources enhances

reliability by enabling distributed systems to be reconfigured, often automatically, to recover from single point failures. Availability of scarce or heavily-used resources is augmented by allowing concurrent access to multiple distributed copies, for increased performance and convenience.

This paper describes two control models that transparently coordinate distributed interactions between a single client and multiple, possibly redundant servers. Clients access the control functions of these models through a uniform, high-level Application Programming Interface (API). The API enables clients to issue service requests and process responses more or less oblivious to the complex distributed processing that is actually required to satisfy their requests. These coordination models are implemented by combining request broker and process group technologies with an object-oriented communication middleware tool that runs across heterogeneous computing platforms.

The following sections review the basic control requirements for one-to-many client-server computing and enabling software technologies that meet these requirements. The remainder of the paper describes and illustrates the new coordination models.

COORDINATION REQUIREMENTS FOR ONE-TO-MANY INTERACTIONS

Given enabling tools for network communication, distributed control for one-to-one client-server applications is fairly straightforward. Clients initiate interactions by issuing requests. Servers respond, and clients complete the cycle by processing results. The primary source of control complexity arises from detecting errors, such as dropped network links or computer failures, and recovering from such problems in a consistent, predictable manner (Coulouris, 1988).

Introducing multiple servers complicates control requirements significantly. Three basic cases can be distinguished. A client may request a single service to be replicated across functionally redundant servers (Case 1). Alternatively, a client request may decompose into distinct service subtasks, which can be performed separately by different servers. Subtasks for such complex requests may be mutually independent (Case 2) or they may exhibit interdependencies (Case 3). Case 3 dependencies generally entail sequencing or synchronization constraints on how subtasks are performed, and are common in process-oriented or workflow applications. Adler (Adler, 1992a) describes a distributed control model that addresses this class of client-server interactions. This paper focuses on coordination models to support replicated requests (Case 1) and complex requests comprised of independent subtasks (Case 2).

Replication entails distributing a client's request to functionally redundant servers.[1] In contrast, the servers for subtasks obtained by decomposing complex service requests tend to be functionally related but disjoint. In both cases, results from servers must be collected, possibly post-processed, and returned to the client. Service results are post-processed through combinational techniques such as collation, competition, or synthesis. Collation simply collects subtask results into a consistent, uniform format. For example, status codes returned from an update operation on replicated databases might be collected into a list. Competition compares collated results with respect to one or more metrics, selectively filtering results based on their scores. A simple example is the precedence metric, or race competition, which scores results based on their order of arrival. Finally, synthesis combines and reconciles partial results. Examples include: relational join, logical union and intersection operations; voting schemes; merging and reconciling partial segments into global plans or schedules; and rank ordering

[1] For present purposes, replicated requests are assumed to be "simple" in the sense that each server responds to a copy of the request as one discrete action. Also, redundant servers incur additional requirements to coordinate the order in which they process replicated requests if it is necessary to maintain consistent state across interactions, as in mirrored transaction databases.

diagnostic hypotheses using weighted frequencies of candidates derived by complementary fault isolation methods.

The burdens of control for one-to-many interaction models can be allocated to: the client; the servers; combinations of the two; or to independent distributed coordination structures. The first three "hardwired" approaches lead to basic system engineering problems, such as limited reusability, maintainability, and extensibility across different clients and client-server associations. Accordingly, we adopted the fourth strategy, by establishing autonomous coordination models or engines. These engines, called Server Groups, function as generic, system-level servers for implementing one-to-many interactions in a distributed application. The next three sections review the distributed computing technologies that were synthesized to create two distinct Server Group engines: process groups, request brokers, and communication middleware.

Table 1. One-to-Many Client-Server Relationships

Request type	Subtasks	Server Functionality	Coordination Tasks
replicated	•	redundant	duplicate request, route requests, synchronize state, combine results, handle errors
complex	independent	distinct	decompose request, route subtasks, combine results, handle errors
complex	dependent	distinct	decompose request, sequence subtasks to reflect dependencies, route subtasks, combine results, handle errors

PROCESS GROUPS

A *process group* consists of a collection of processes, typically a set of applications, that jointly provide one or more services on a continuous basis (Liang, 1990; Kaashoek, 1993). Group processes are typically distributed across networked computers, operate in disjoint address spaces, and communicate via message-passing mechanisms. A group is *closed* if communication is restricted to members of the group; otherwise, the group is *open*. Interactions between a group and an external process (or group) are called *intergroup* communication. Interactions among members of a group constitute *intragroup* communication (cf. Figure 2).

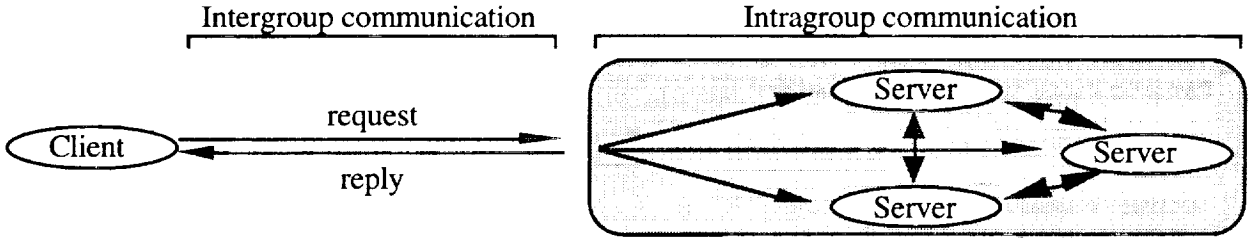


Figure 2. Open Process Group Architecture for One-to-Many Client-Server Model

Process groups provide a natural control framework for organizing collections of servers that are functionally related or redundant, mediating all intergroup and intragroup interactions between applications playing client roles and those acting as servers. The utility of process group models for supporting complex and replicated service requests hinges on the concept of *group transparency*, which simply means that a client may treat a group's service as if it were provided by a single server.

In particular, all group members are identified by a single group name, which acts as a logical address for all intergroup and intragroup communication. Clients need not track either group

membership (naming transparency), nor the host computers for individual group members (location transparency). These properties are especially attractive for interacting with groups whose membership is dynamic and/or mobile. Groups collect and post-process service results produced by member servers over extended time intervals (reply-handling transparency). Finally, clients need not deal with partial failures (fault-handling transparency), in that group interfaces either conceal single point member faults or indicate total failure. In short, group transparency fully conceals the distributed architecture and distributed behavior internal to a process group from clients.

REQUEST BROKERS

Simple, static client-server applications tend to support a limited number of distinct services, such as file, print, and database sharing. In such systems, it is relatively straightforward for client programs to keep track of the services that are available and which servers support those individual services.

In larger distributed systems, it becomes difficult to sustain this "hardwired" strategy, in which each client assumes responsibility for knowing about available services, their associated servers, and interfaces. Maintaining and extending such knowledge for individual clients is particularly cumbersome for distributed systems such as CAD or CASE frameworks, which evolve continually as server applications are enhanced, added or replaced. The notion of a service request "broker" was developed to provide clients with global, system-level support for tracking available services and servers. This strategy has been promoted by the Object Management Group, whose Common Object Request Broker Architecture (OMG/CORBA) specifies a standard for distributed object management systems (OMG, 1991).

A request broker is basically a control mechanism that mediates interactions between client applications requesting services, and server applications capable of responding to such requests. All applications that belong to a distributed system register the services that they support, their locations, and their client interfaces with the broker. Typically, the broker maintains a directory to store this information. Once this registration process is complete, any application that requires a service requests it from the broker. The broker identifies an appropriate server for the requested service using its registration directory, forwards the request, and relays back the response to the client (cf. Figure 3). (Note: OMG divides these broker and directory functions between the Object Request Broker and Object Trader Service.)

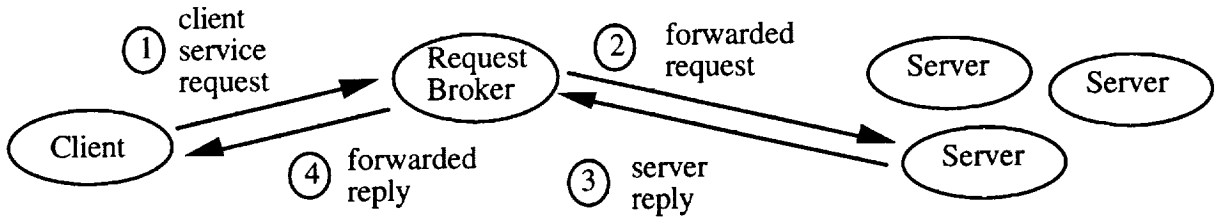


Figure 3. Broker model for forwarding client requests

The request broker architecture frees client applications from having to know where and how to obtain particular services. A client application only needs to know the names for the particular services it expects to request and how to use the request broker API to request those services. Unfortunately, current models are restricted to one-to-one interactions between a single client and a single server. The Server Group eliminates this restriction, extending the broker model to support one-to-many interactions.

OBJECT-ORIENTED COMMUNICATION MIDDLEWARE

Communication middleware refers to software tools that insulate application developers from the complexities of network programming. Middleware generally encompasses some form of systems-level software, or "kernel," together with high-level developer tools (Adler, 1992b). Kernels interface directly to the network protocols that enable communication between remote computers, such as TCP/IP and NetBIOS. Developer tools generally consist of API libraries, which direct the kernel to establish connections between specific computers and to exchange information between the desired applications. The API conceals system-level dependencies across heterogeneous computers, operating systems, and protocols.

Message-passing middleware enables applications to interact by exchanging high-level messages, as opposed to the function call approach embodied in Remote Procedure Calls (Corbin, 1991). A messaging middleware kernel consists of message queues, queue management services, and transport services (i.e. network drivers). Client applications use a messaging API to post request messages to the outbound queue of their local kernel. The kernel transparently transports the message to the inbound queue of the remote kernel, from which the remote server can retrieve it. The server replies to the client through a similar process. This model is convenient for simple client-server exchanges. However, it provides inadequate support for coordinating the complex one-to-many interactions described earlier. As a result, developers must construct suitably extended control apparatus on top of the messaging middleware, and integrate it into their applications.

The present work builds on NetWorks!, an object-oriented middleware tool that provides application connectivity across heterogeneous PC, workstation, and mainframe computing platforms. NetWorks! addresses the shortcomings of conventional messaging tools by adding a scheduler to the kernel and introducing active objects called *Agents*. An Agent consists of: (1) standard program code, such as C or C++; (2) calls to the high-level NetWorks! API library; and (3) calls to application APIs. The scheduler delivers inbound messages to the relevant Agents, which interact with local applications by injecting message data or commands. The scheduler also interacts with Agents to send messages outbound from local applications to remote kernels and Agents (cf. Figure 4). The NetWorks! API enables developers to program these messaging interactions between applications, Agents, and the kernel. NetWorks! provides an intuitive middleware implementation framework for client-server and peer-to-peer interactions because Agents can initiate and react to both request and response messages.

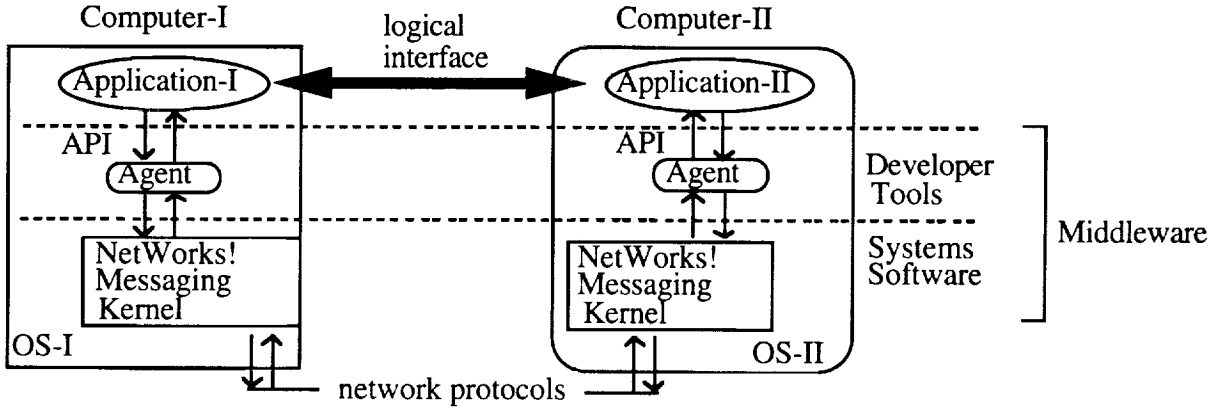


Figure 4. Conceptual Architecture of the NetWorks! middleware tool

In essence, Agents represent active participants in distributed interactions, interfacing with, but distinct from both applications and messaging kernels. As such, they constitute a separate locus

of control that can be used not only to integrate applications, but also to integrate control models for coordinating distributed application Agents. Equally important, NetWorks! Agents can reuse and selectively specialize behavior from other Agents, through object-oriented inheritance.

The NetWorks! Developer Services Library (NDSL) exploits both of these properties. The NDSL consists of a set of predefined Agents that integrate various distributed control models. Each such Agent supports a high-level, service-oriented API that conceals the underlying NetWorks! messaging API. Client applications use the NDSL API to issue service requests and retrieve responses oblivious to the Agents, messaging kernels, and complex distributed processing required to satisfy their requests. The next two sections describe the NDSL Server Group Agents, which were developed specifically to coordinate one-to-many client-server interactions.

THE SERVER GROUP COORDINATION ENGINE

The NDSL Server Group coordination model derives directly from process group technology. Client applications communicate requests to a Server Group via the high-level NDSL API. The Server Group then coordinates the activities of individual group members to provide the requested service (cf. Figure 5). The Server Group is the source of all intragroup communication for replicating or decomposing client requests, and the target of all server responses. The Server Group combines server responses as necessary, returning a single reply to the client.

The Server Group Agent exploits inheritance by reusing a parent NDSL model called the Service Request Manager (SRM). The SRM provides a conventional request broker capability for coordinating one-to-one client-server interactions. The Server Group selectively specializes inherited SRM broker object behaviors to support one-to-many client-server interactions. In particular, extensions were made to:

- the SRM API for registering and requesting services.
- the SRM broker directory, for describing functionally redundant servers, decomposing complex services, and combining results from multiple servers.
- the broker control model for dispatching requests and collecting server replies.

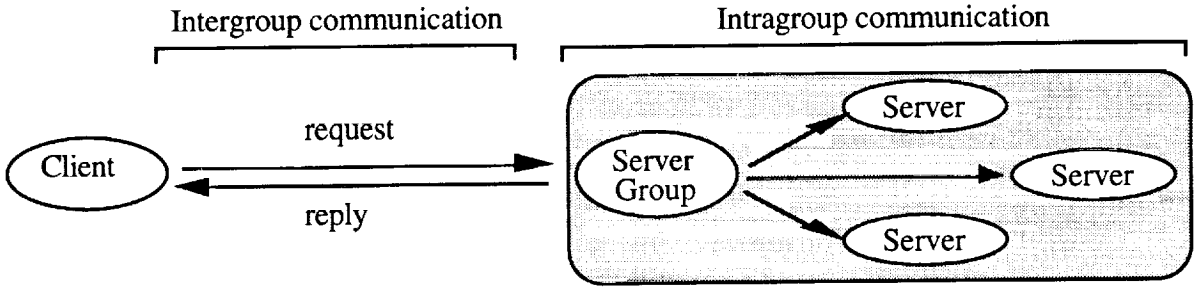


Figure 5. Server Group Architecture for functionally redundant servers

Server Group API

Standard request brokers presuppose that individual services are supported by single, unique servers. Simple directory lookup methods suffice to match client requests against registered servers. One-to-many client-server interactions require added capabilities for describing functionally redundant servers and determining which servers must be invoked to support (complex) requests. The Server Group reflects the first requirement by extending the SRM with a group-oriented registration API. Servers use this API to register as group members with a Server

Group. Membership entries coexist in the Server Group directory with service entries registered via the SRM broker API. The group-oriented API functions include:

- SGMGroupCreate — creates a group with the specified name and attributes.
- SGMGroupJoin — registers an application or Agent server.[2]
- SGMGroupLeave — removes an individual server from a group.
- SGMGroupSuspend — suspends a server from the active membership list so it is not used in current group requests, but preserves its directory entry.
- SGMGroupRejoin — cancels a server's suspension from a group, and reinstates that group member for active service.
- SGMCreateProfile — assigns server profile attribute values for a group member (used in conjunction with SGMGroupJoin).
- SGMGroupMsgSend — multicasts a message to all group members. Multicast is a selective variant of broadcasting, which automatically sends a message to all nodes specified by a membership list rather than to all network nodes.

Server Group Directory and Client Request API

The Server Group extends the SRM broker directory and API to enable multiple servers to register the same service capability. To accommodate functional redundancy, a Server Group directory entry specifies not only the service name, a server, and its location, but also a *server profile* data object. A server profile specifies that server's attributes in providing a service, such as relative speed, completeness, and precision. Server profiles are identical for servers that are exact replicas, but differ for redundant but complementary servers. Attributes take integer values, ranging from 1 (minimal) to 5 (maximal). For example, a rule-based diagnostic system might be assigned values of 3 for relative speed and 3 for completeness vs. values of 1 (slow) and 5 (very complete) for a competing model-based reasoning system. The value 0 indicates that an attribute is not relevant for a given service and should be ignored. Server profiles are extensible, allowing new attributes to be defined to meet application-specific requirements.

Client APIs for request brokers such as the SRM invoke a service by specifying the desired service name and appropriate call parameters. For example, a database query would specify "DBMS-query" as the service type and "Corporate-DB" and "select * from dept where..." as the call arguments. To support redundant servers, the Server Group client API adds a request profile object, which specifies: (1) the maximum number of servers desired; (2) the maximum difference to tolerate between server and request profiles; and (3) an embedded server profile. Thus, the server profile is used both by the Server Group to model the attributes of a server in providing a particular service, and by clients to specify the desired attributes of potential servers for carrying out that service.

Server Group Router

A conventional request broker routes a request to the server specified by the directory entry for the requested service type. The Server Group must also support replicating client requests to

---

[2] Both applications and Agents can register with an SRM or Server Group. An Agent can be viewed as an application with one exportable function (which activates the Agent). Registering an application enables clients to invoke service functions within that application. For example, a spreadsheet might register, and thereby export its recalculation function. The SRM or Server Group creates and stores handles to registered functions, which it uses thereafter to invoke the desired services for clients. The primary tradeoff is that Agents are useful in virtue of inheritance-based reusability of behaviors, whereas performance is better if application services are invoked directly rather than indirectly, via Agents.

send to functionally redundant servers, and decomposing complex service requests into requests for independent subservices. It is straightforward to extend a standard router to support request replication. However, a front-end preprocessor must be added for handling complex requests. The Server Group router algorithm: (1) decomposes the requested service if necessary; (2) extracts candidate servers from directory; (3) filters candidates with respect to an internal match algorithm; and (4) dispatches request task(s) to surviving server candidates.

The Server Group router only invokes the preprocessor (Step 1) on requests for complex services. Complex services are identified by the fact that the Server Group itself is registered as their server. To support decomposition, the server registration API requires a function pointer to be supplied for complex services. The executable code for that procedure must be co-resident with the Server Group. The procedure's input consists of the service entry from the directory corresponding to the complex service type specified in the client's request. The procedure returns a list of directory entries that correspond to the constituent subservices for that composite service.

In Step 2, the Server Group router reuses SRM lookup functionality to extract candidate servers from its directory based on the client's requested service type. In Step 3, the router executes a match algorithm, which filters redundant servers for the client's request. The match algorithm computes a differential between attribute values for the server profiles specified in the client request profile and the directory service entry. Next, it sorts server candidates in order of increasing differentials, eliminating all candidates whose differentials exceed the client's specified threshold. Finally, the algorithm returns any remaining servers up to the limit specified by the client request profile. For simple requests, the Server Group router then dispatches the given request to all surviving servers (step 4). For complex service requests, the router loops through Steps 2 through 4 for the list of subservices produced in Step 1.

This algorithm adds the absolute values of the differences between corresponding attribute values for the client request profile's embedded server profile and the directory service entry's server profile. Next, it sorts these server candidates with respect to increasing order of differential totals. It then eliminates all candidate servers whose differential totals exceed the threshold (if specified). The algorithm returns all surviving candidates up to the limit specified by the client request profile. For simple requests, the Server Group router then dispatches the given request to all surviving servers (step 4). For complex service requests, the router loops through Steps 2 through 4 for the list of subservices produced in Step 1. The overall control model is summarized graphically in Figure 6.

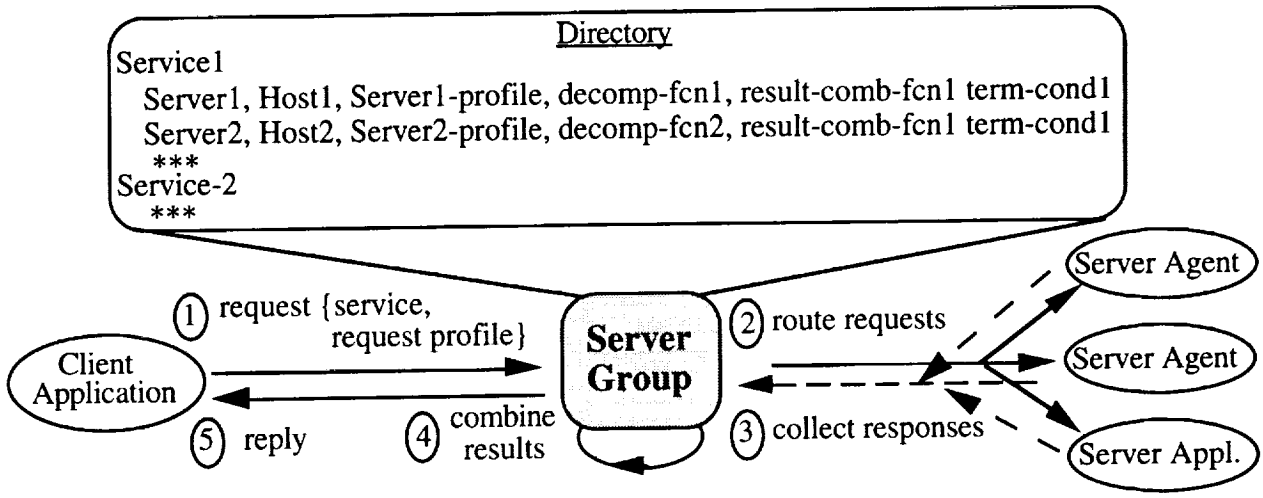


Figure 6. Overview of control sequence for Server Group coordination model

Server Group Processing of Service Results

A conventional request broker routes the single response resulting from a service request back to the requesting client. The Server Group requires more elaborate control mechanisms to collate and process responses from multiple group members. Specifically, the Server Group must know: (1) how to collect all responses for a given client request (which the Server Group may have replicated or decomposed); (2) when to stop waiting for responses for the client request; and (3) how to process the responses that have been collected up to that termination point.

With regards to issue (1), The Server Group relies on unique transaction identifiers generated by NetWorks! to label and automatically track all messages dispatched to support a given distributed interaction. With regards to (2), the Server Group uses a termination test predicate to determine when to stop collecting responses. The choice of termination predicate for a service depends largely on its desired result combination behavior. For example, all possible responses should generally be collected before applying algorithms that synthesize results. Alternatively, competitive result combination functions tend to collect the first N arrivals, or the first N arrivals that satisfy some application-specific conditions. With regards to (3), once all relevant responses are collected for a given client request, the Server Group applies the result combination procedure for the relevant service. Similar to decomposing complex services, the termination test and result combination functions are specified as pointers via the service registration API, with the caveat that the executable codes reside on the Server Group host computer.[3]

REQUESTS FOR RELIABLE, REPLICATED SERVICES

The NDSL Reliable Server Group provides a generic engine for the two phase commit protocol to support replicated databases and other transaction-oriented applications that obtain reliability through replicated servers (Ceri, 1984). The Reliable Server Group inherits most of its capabilities from the basic Server Group, specializing just two behaviors. First, the router omits the candidate filtering process; client requests are automatically multicast to *all* servers registered to support the relevant service. The Reliable Server Group then combines responses as per the standard Server Group control model. Responses indicate either success or failure (due to dropped communication links, failed host processors, or errors generated during server processing of requests). Second, the Reliable Server Group initiates another round of messages to the servers before responding to the client. Specifically, it multicasts a Commit message if all servers acknowledged success, causing them to commit their service actions as permanent transactions. Otherwise, it sends an Abort message to undo or "rollback" any temporary state changes. The two sets of messaging interactions are depicted in Figure 7. In order to exploit this reliable interaction model, server Agents or applications must be designed to support the second sequence of messages, which is absent from the standard Server Group.

---

[3] The Server Group filters redundant servers based on client request profiles, so it cannot determine in advance which servers will be selected. Consequently, result combination and termination test functions must be identical across all servers that register a given service.

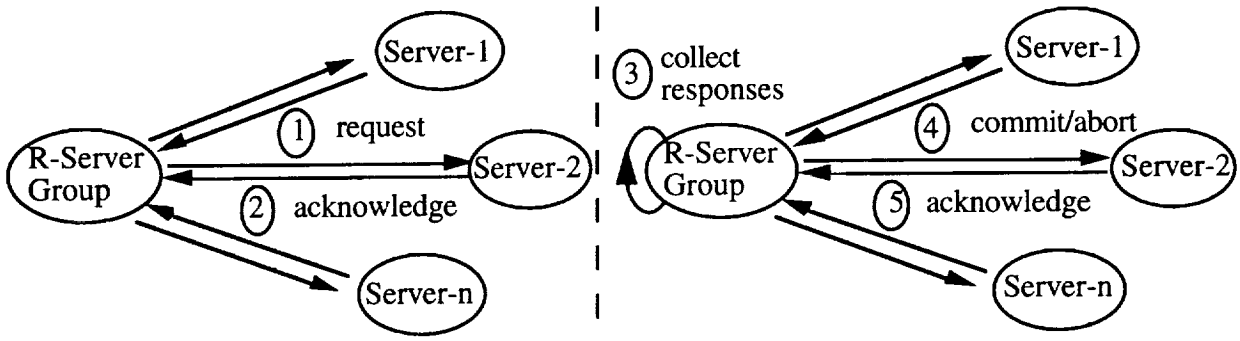


Figure 7. Two phase commit coordination model of the Reliable Server Group

DISCUSSION

Aside from the literature on process groups, research on cooperative coordination models has been most active in the context of Distributed Artificial Intelligence or DAI. Most DAI research focuses on specific types of coordination architectures for cooperating, intelligent servers, such as contract nets, blackboard architectures, and negotiation models (Bond and Gasser, 1988; Gasser and Huhns, 1989). Server Groups differ from such systems in several respects. First, Server Groups provide a centralized control module to manage passive servers, whereas DAI models tend to distribute control among autonomous servers. Second, the NDSL Server Groups were designed to support relatively coarse grained interactions among member servers, whereas DAI servers solve problems through more intensive, fine grained interactions. Third, Server Groups possess the flexibility to support radically different control behaviors for specific services simultaneously, whereas DAI models typically establish a uniform, global protocol tailored to specific (classes of) applications.

The NDLS Server Group is subject to two potential drawbacks common to centralized control models, degraded performance under heavy traffic loading, and reliability limitations due to a critical, single point of failure. The Server Group uses the NetWorks! non-blocking (asynchronous) messaging capability to minimize overheads due to group-based communication and server processing. Request decomposition tends to have minimal impact on performance except in systems with intensive request traffic and/or real-time constraints. The most serious problem arises from result combination behaviors that are computationally intensive, such as relational joins on large data sets or detecting and resolving conflicts across partial plan or schedule segments. Such processing can hold up or block brokering of lower overhead client requests. (It must be noted that decentralized models incur different, but comparable overheads in providing equivalent distributed communication and coordination functions.)

Performance can be improved significantly by isolating processing-intensive result combination behaviors, using distributed architectures that integrate SRMs with Server Groups (cf. Figure 8). All client requests are addressed to an SRM, which brokers non-group tasks and routes requests for group services involving compute-intensive processing to a remote, dedicated Server Group. The Server Group acts as a single logical server with respect to the SRM, isolating computation and concealing the group-based processing. Multiple Server Groups can be introduced as appropriate. Such hybrid configurations illustrate the power of the NDSL building block approach to combine the design simplicity of centralized coordination models with the concurrent processing advantages of distributed control architectures.

As for fault tolerance, the SRM and Server Groups are currently being enhanced for greater reliability via an automated recovery design based on standard checkpointing and message-logging techniques (Strom, 1985). Control extensions to support fault tolerant transparency are

also being investigated, involving automated detection and management of group member server failures (Birman, 1993).

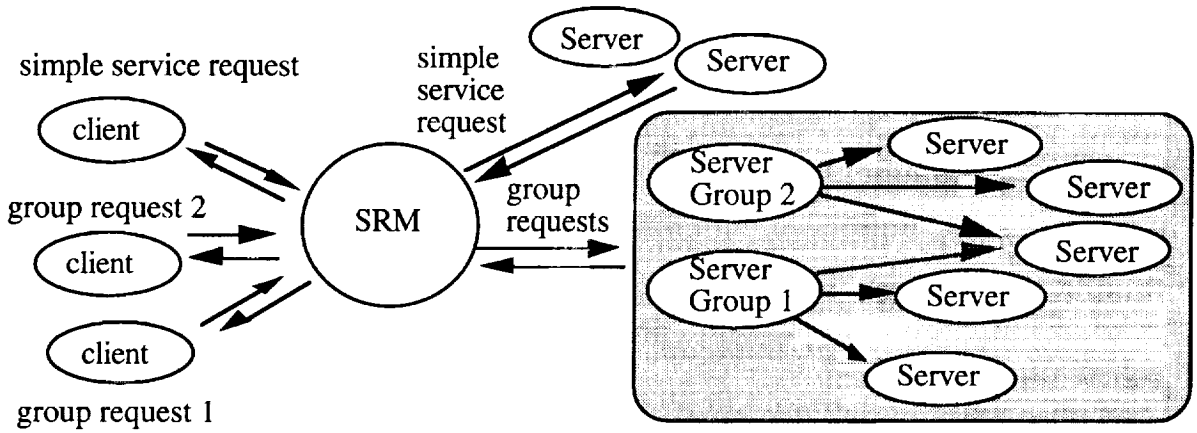


Figure 8. Hybrid Distributed Coordination Model

EXAMPLE APPLICATION

A prototype NDSL Server Group was built to simulate group-based coordination of rule-based systems for distributed operations support of the space station (Ringer, 1991; Walls, 1990). These applications automate fault detection, isolation, and recovery functions (FDIR) for the primary power generation and distribution subsystem (PGDS) and module power subsystems (MPSs). PGDS manages the supply of power to MPSs, which manage power consuming components within Laboratory and Habitation modules. The FDIR expert systems must interact cooperatively to reflect the architectural interfaces between their target systems. The FDIR-Server Group coordinates these interactions. Two additional FDIR systems that do not actually exist, a model-based reasoner and a neural net, were simulated to introduce functional redundancy of FDIR services for an MPS (cf. Figure 9).

The Server Group API was used to register the FDIR servers for the PGDS and MPS and their respective services. A complex service for system-wide FDIR was also registered, which decomposes into diagnostic service requests to the rule-based FDIR systems for both the PGDS and the MPS. The client request API was then used to:

- invoke the FDIR system to diagnose a problem specific to the PGDS.
- invoke the functionally redundant FDIR systems to diagnose a problem in the MPS. Different client request profiles were specified to invoke different numbers and kinds of FDIR servers for the MPS.
- invoke the system-level FDIR service to isolate a problem that could originate in either the MPS or PGDS.

This last test demonstrates a global management capability, in which reports of anomalies trigger coordinated system-wide FDIR activity. (Appendix A lists a partial execution trace for this particular Server Group control test.) Server Group architectures are attractive for complex applications such as distributed operations support because of their modularity and extensibility. New capabilities, such as FDIR applications for different MPSs, can be integrated incrementally, without having to modify (the knowledge bases of) existing group members. All knowledge concerning subsystems and their interrelationships can be isolated within the Server Group, via service decomposition and result combination behaviors. These behaviors, if they are sufficiently complex, may themselves be implemented as system-level intelligent applications.

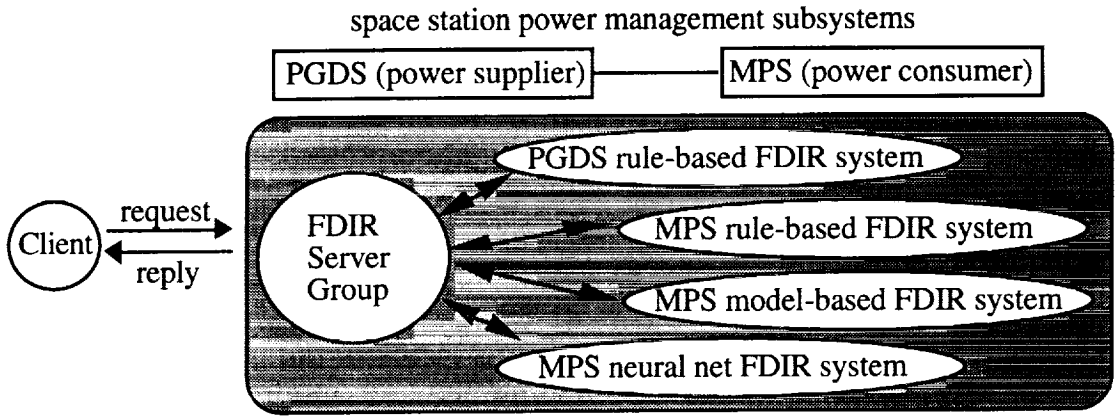


Figure 9. Demonstration scenario for the NDSL Server Group

CONCLUSIONS

The NetWorks! Server Group provides a generic, object-oriented engine for coordinating distributed one-to-many client-server interactions. Major application areas include distributed decision and operations support, data analysis, concurrent engineering, process control, and office automation. The Server Group reuses and specializes a one-to-one client-server request broker model. The extensions adapt process group transparency concepts to exploit functionally redundant servers and to manage client requests for complex services. High level APIs are used to specify server attributes, client request profiles, and request management behaviors specific to particular services. The Reliable Server Group extends the Server Group's model for managing replicated servers by incorporating a two phase commit protocol for reliability. Both models rely on a modular, communication middleware substrate for transparent application connectivity.

The NDSL Server Groups and related NetWorks! tools insulate developers and end-users from the distribution and heterogeneity of networked applications and their computing platforms. They also minimize the complexities associated with designing, implementing, maintaining, and extending the apparatus for coordinating one-to-many interactions among networked clients and servers.

ACKNOWLEDGMENTS

The NetWorks! Server Group technology described in this paper has been developed with funding from the NASA Small Business Innovative Research Program under NASA contracts NAS8-39343 and NAS8-39905.

REFERENCES

R. M. Adler. (1992a) "Coordinating Complex Problem-Solving Among Distributed Intelligent Agents." *Telematics and Informatics*. Vol. 9. Nos. 3&4. pp. 191-204.

R. M. Adler. (1992b) "Object-Oriented Tools for Distributed Computing." *NASA Proceedings for Technology 2002 Conference*. NASA CP-3189. pp. 146-155.

K. Birman. (1993) "The Process Group Approach to Reliable Distributed Computing." *Communications of the ACM*. Vol. 36. No. 12. pp. 36-53.

A.H. Bond and L. Gasser. eds. *Readings in Distributed Artificial Intelligence*. Morgan-Kaufmann. San Mateo, California. 1988.

S. Ceri and G. Pelagatti. (1984) *Distributed Databases, Principles and Systems.* McGraw-Hill. New York.

J. R. Corbin. (1991) *The Art of Distributed Applications: Programming Techniques for Remote Procedure Calls.* Springer-Verlag. New York.

G. Coulouris and J. Dollimore. (1988) *Distributed Systems: Concepts and Design.* Addison-Wesley, Reading, Massachusetts.

L. Gasser and M. Huhns. eds. (1989) *Distributed Artificial Intelligence, volume II.* Morgan-Kaufmann. San Mateo, California.

M. F. Kasshoek, A. S. Tanenbaum, and K Versteop. (1993) "Group communication in Amoeba and its applications." *Distributed System Engineering.* Vol. 1. pp. 48-58.

L. Liang, S. Chanson, and G. Neufeld. (1990) "Process Groups and Group Communi-cations: Classifications and Requirements." *IEEE Computer.* Vol. 23. No. 2. pp. 56-66.

Object Management Group and X/Open. (1991) "The Common Object Request Broker: Architecture and Specification." OMG Document No. 91.12.1 (revision 1.1), OMG, Framingham, Mass.

M. Ringer, T. Quinn, and A. Merolla. "Autonomous Power System Intelligent Diagnosis and Control." (1991) *Proceedings of the 1991 Goddard Conference on Space Applications of Artificial Intelligence.* NASA CP-3110. pp. 153-168.

R. Strom and S. Yemini. (1985) "Optimistc recovery in distributed systems." *ACM Transactions on Computing Systems.* Vol. 3. No. 3. Aug 1985. pp. 204-226.

B. Walls, D. Hall, and L. Lollar. (1990) "Augmentation of the Space Station Module Power Management and Distribution Breadboard." *Proceedings 4th Workshop on Space Operations Applications and Research (SOAR '90).* NASA CP-3103. pp. 355-361.

APPENDIX A

--------- Partial Listing of FDIR SERVER GROUP Directory--------

service	systemlevel-powersys-fdir	labmodule-powersysfdir
server-agent	fdir-grp	rulelm
server-sys	markov	hertz
server-profile	completeness 4	completeness 3
	certainty 4	certainty 3
	speed 2	speed 3
decomp-algorithm	sys-fdir-decomp	nil
term-condition	nil (default)	nil
result-comb-alg	combine-sys-fdir-results	combine-mps-fdir-results

GROUP-MEMBERS
(nntwklm hertz) (mbrlm markov) (rulelm hertz) (pgds markov)

--------- TRACE OF CLIENT REQUEST TO DIAGNOSE SYSTEM-LEVEL PROBLEM --------
Client Requesting System-Level Diagnosis for Power Mgmt. System...

FDIR-GRP Filtering Group Candidates for Service Type systemlevel-powersys-fdir...
Client Request Profile:
Max Servers: 10000 Discrim. Threshhold 10000 *;;;; defaults - use all candidate servers*
Completeness 0 Precision 0 Certainty 0 Timeliness 0 *;;; ignore all server attributes*
Candidates: ((systemlevel-powersys-fdir fdir-grp))
Applying Decomposition Algorithm.... *;;; Invoke Decomposition*
Substituting services (primary-powersys-fdir labmodule-powersys-fdir) *;;; Subtask servers*
Dispatching service systemlevel-powersys-fdir for SGROUP execution...
Symptoms: ("PDCU-B Bus-A LC1 RPC 3" "PDCU-B Switch RBI.3/1 Power 2.38")
Routing primary-powersys-fdir task to Agent APGDS... *;;;Send to subserver 1*
Routing labmodule-powersys-fdir task to Agent ARULELM... *;;;Send to subserver 2*
...
Agent APGDS Responding to Request for Service primary-powersys-fdir... *;;; PGDS Server*
injecting symtom data into PGDS ... diagnosing... diagnosis completed.
APGDS extracting diagnostic conclusions from PGDS:
Leakage-path high-to-low transmission-line RPC.3/6 load
Posting result to FDIR-GRP...
...
Agent ARULELM Responding to Request for Service labmodule-powersys-fdir... *;;; MPS Server*
injecting symtom data into RULELM ... diagnosing... diagnosis completed.
ARULELM extracting diagnostic conclusions from RULELM:
Low-impedance-short cable below switch
Low-impedance-short switch output of switch
Low-impedance-short switch input of a lower switch
Posting result to FDIR-GRP...
...
FDIR-GRP Responses collected for systemlevel-powersys-fdir request
Applying Result Combination Algorithm COMBINE-SYS-FDIR-RESULTS...
Conclusions: *;;;; result combination algorithm - logical union operation*
PGDS "Leakage-path high-to-low transmission-line RPC.3/6 load"
RULELM "Low-impedance-short cable below switch"
"Low-impedance-short switch output of switch"
"Low-impedance-short switch input of a lower switch"

N94-35069

A Distributed Computing Model for Telemetry Data Processing

Matthew R. Barry,*
Kevin L. Scott and Steven P. Weismuller†
Systems Division DF6
NASA/Johnson Space Center
Houston, TX 77058

Abstract

We present a new approach to distributing processed telemetry data among spacecraft flight controllers within the Control Centers at NASA's Johnson Space Center. This approach facilitates the development of application programs which integrate spacecraft-telemetered data and ground-based synthesized data, then distribute this information to flight controllers for analysis and decision-making.

The new approach combines various distributed computing models into one hybrid distributed computing model. This model employs both client-server and peer-to-peer distributed computing models cooperating to provide users with information throughout a diverse operations environment. Specifically, it provides an attractive foundation upon which we are building critical real-time monitoring and control applications, while simultaneously lending itself to peripheral applications in playback operations, mission preparations, flight controller training, and program development and verification.

We have realized the hybrid distributed computing model through an *information sharing protocol.* We shall describe the motivations that inspired us to create this protocol, along with a brief conceptual description of the distributed computing models it employs. We describe the protocol design in more detail, discussing many of the program design considerations and techniques we've adopted. Finally, we describe how this model is especially suitable for supporting the implementation of distributed expert system applications.

1 Introduction

The past approach to spacecraft data distribution in the Mission Control Center (MCC) used a centralized computing model. Mainframe computers processed the incoming telemetry and trajectory data, applied various computations to this data, then distributed this information to hundreds of flight controller displays. This approach worked well for many years, but its cost, inflexibility and resistance to change have become unappealing. Meanwhile, advancing workstation and networked, distributed computing technologies which overcome these limi-

*Rockwell Space Operations Company, 600 Gemini Ave., R20A-4, Houston, TX, 77058.

†Unisys Space Systems, 600 Gemini Ave., U04D, Houston, TX, 77058.

tations grew more compelling. Accompanying these advancements were new software development packages, man-machine interfaces, and artificial intelligence technologies. The application potential of these exciting technologies inspired flight controllers to assume more interactive roles in the development of computational tools which enhance their spacecraft operations capabilities. The desire to implement these enhancements encouraged a revolution in the MCC computing architecture.

The MCC began the transition from centralized to distributed computing several years ago. The current MCC employs *both* configurations simultaneously, running workstation and mainframe applications side-by-side. As the MCC evolves into the Control Center Complex (CCC) for dual Space Shuttle and Space Station operations, the workstations and network will become the principal computing platform.

The current dual-configuration mode of operations in the MCC has afforded flight controllers an opportunity to prototype new operations applications. Until recently these efforts have focused on the development of real-time monitoring and fault detection and diagnosis applications for individual flight control disciplines. These efforts have been successful in demonstrating the enhanced capabilities provided by the distributed computing platform. Moreover, they have encouraged the pursuit of distributed applications which span several flight control disciplines. These distributed applications exploit networking capabilities to provide inter-user information sharing.

The new distributed computing model we describe herein was created to support and encourage this inter-user information sharing. The *information sharing protocol* (ISP) was designed to provide elegant computing techniques which enable workstation applications to communicate with each other. This protocol also supports inter-host communications on a heterogeneous platform complying with industry-standard operating systems and networking protocols. Consequently, the ISP provides users with a consistent communications interface for all of the various operations, training, and development platforms they use in everyday business; lack of such an interface has caused many problems in the past.

There were several motivations that led to the development of the ISP. First, there has been a strong desire to reduce the manpower required for real-time Space Shuttle operations. This reduction in manpower necessitates the development of enhanced computer programs which can perform many of the real-time data monitoring, fault detection and diagnosis, and planning tasks previously performed by flight controllers. These developments have also spawned many projects pursuing "intelligent systems" which meet or exceed the flight controller's reasoning capabilities. Many such systems have already been deployed in the MCC, but there are many more of these intelligent systems appearing on the horizon. The potential for these systems to capture precious spacecraft operations expertise is a clear motivating factor in our work. Second, the increasing number of workstation programs being developed to enhance operations has been accompanied by a significant growth in software development and maintenance costs. The development of the ISP was meant to restrain this growth by providing a package of distributed computing tools which support the basic needs of all flight control disciplines. These tools reduce both development and testing costs. Third, the use of workstations outside of the MCC has increased tremendously over the last few years. Flight controllers now find themselves performing many of their mission preparation, training, and software development tasks in facilities lo-

cated in their offices or laboratories around the center. Since many of the computer programs they use must work in all of these facilities, there is a strong movement toward standardization. This movement is encouraging the use of industry-standard hardware and commercially-developed software to reduce development and maintenance costs. The ISP promotes these standards while providing a layer of customizable telemetry data distribution tools for program development in a heterogeneous facilities environment. Finally, drawing from each of these individual motivations, we envision a tremendous utility in the deployment of distributed expert systems. The ISP was designed to provide a mechanism for these expert systems to communicate with each other.

2 Distributed Computing

A *distributed computing* architecture is a programming model in which computing resources can be allocated in various ways. Typically this involves a network of high-performance workstations providing information to a user. There are two predominant models of how to distribute the available computing resources. A *client-server* computing model refers to a special case of distributed computing in which an application is divided into separate client and server processes. The client process makes requests of the sever process. The server process receives requests from one or more clients and performs some action on their behalf. Servers can provide resource sharing by servicing many clients. Frequently, the client and server pieces are running on different machines; however, this need not be the case. In a *peer-to-peer* computing model, the distributed components of the application act as independent equals, communicating with one another to accomplish a common goal.

The distributed computing model realized in the ISP reaps the benefits of both of these models. At a global level, the goal of the application is to distribute telemetry data to flight controllers. In the client-server role, the primary purpose of the server part of the application is to process telemetry data from a certain data source. The clients request this processed data from the servers. In the peer-to-peer role, the servers allocate computing resources and communicate with one another to provide their clients with data collected from several data sources. The combination of these models provides a system of cooperating agents which supports a wide variety of potential applications.

The user assigns each server to process a portion of data available from the data source. The server provides this data to its clients on a change-only basis; *i.e.* a client will receive data only when it changes from its previous value. Since there are a variety of data sources available, we have written a collection of server programs each of which has been uniquely outfitted to process a particular data source. The data sources include two real-time data streams and various sorts of recorded telemetry files. The server preprocessing includes reading data from the data source, performing noise filtering and other preprocessing on the data, determining what data has changed since the last cycle, and then distributing the changes to the clients. Each server is responsible for any time synchronization and polling tasks.

The user may run clients that receive data from the server and generate new results to supplement the telemetry data. Therefore, the servers also perform the task of reading and distributing this client-synthesized data. Clients that synthesize data for other clients are said to be *publisher* clients, while clients that request any sort of data are said to be *subscriber* clients. Since each server "owns" a

portion of the data source, the servers can interact in the peer-to-peer role to acquire data for clients which subscribe to data owned by other servers. In this situation, one server becomes a subscriber of another server. Along with resource allocation and data distribution, this technique provides a way to enforce data security.

Although there are unique servers for each data source, the data source is transparent to a client. Using the ISP, a client receives data the same way regardless of where the data originated. This feature provides a high degree of portability between computing platforms, and insulates the clients from the nuances of the data source. Essentially, this means that a client program can be deployed against a variety of data sources *without change*. This data-source independence simplifies application program development efforts and reduces the costs associated with software testing and certification.

3 Design Considerations

This section describes a few of the considerations we have made in the design of the ISP. Most of these considerations reflect the motivations established above, while others reflect the specific nature of the telemetry data processing problem. We discuss here only the considerations which we believe to be Space Shuttle-independent. Primarily, we discuss the construction of our hybrid distributed computing model by presenting the attractive features of the two underlying models.

3.1 Client-Server Model

The client-server model is the most influential part of the ISP model. Using this model, we separate the telemetry data source from the client programs. We create server programs to manage the data source and distribute preprocessed data, and we create client programs to do something useful with this information. Usually the client programs perform analysis of the information and display the results to the flight controllers. This separation of tasks is crucial to the hybrid model: it provides the cability to support a variety of data sources and various computing platforms with the minimum number of programs; and it allows user to run various combinations of clients in servers in different contexts, such as playbacks or offline simulations and training cases.

The ISP servers provide information to their clients on a change-only basis. This means that the client programs may operate in an event-driven fashion, instead of having to poll the data stream continuously.[1] When new data is made available to the client, that data is considered an interrupt "event" that the program must handle. The client handles this event by reading the data, processing and perhaps displaying it, then returning to its wait state to receive subsequent events. This provides an efficient allocation of computing resources and reduces the effort necessary to build real-time computer programs. Furthermore, to make better use of the distributed computing platform, we can divide this effort among several functionally-equivalent servers processing a different subset of the full telemetry stream. By not extraneously reprocessing the unchanged values every second, each client realizes a significant computational resource savings and, perhaps more importantly, appears more responsive to user interaction.

We assume that there will normally be many servers running. Each server will process a portion of the full data stream. At initialization time, the server assumes an ownership which identifies those *symbols* from the

[1] Historical Space Shuttle mission data shows that during on-orbit operations only 10-15% of the 25,000 telemetry values change during any second.

symbol dictionary that it becomes responsible for. The server will acquire and process all of the symbols from the symbol dictionary which have the same ownership as the server. For example, the Propulsion (PROP) console server started with ownership `prop` will acquire from the data source all of the symbols in its dictionary with ownership `prop`. All PROP client programs need connect only to a `prop` server; similarly all Guidance, Navigation and Control (GNC) console clients only need connect to a `gnc` server, and so on.

Although the server's primary purpose is to process telemetry data, the server is also capable of processing and redistributing data generated by publisher clients. This feature is vital to distributed client applications, particularly fault detection and diagnosis applications, which intend to share their results with other clients. Under the ISP, this distribution occurs when the publisher client writes its results back to the server, which enqueues the information for any subscriber clients. Since the distribution mechanism is the same for any kind of data, the ultimate subscriber client will not be able to discern where the data originated, *i.e.* from telemetry or from a publisher client.[2]

Finally, since the servers are meant to be background tasks, they do not have user interfaces. To provide server status information and control features, we simply employ a collection of client programs which act only as user interfaces to their server. These user interface clients don't subscribe for any data from the server; instead, they read the server log files for status information, and they send the server control commands through packet messages. Furthermore, this design embodies an additional feature that the separated user interface can converse with *any* of the servers without modification.

3.2 Peer-to-Peer Model

Commonly, clients from one flight control discipline may need to process symbols "owned" by another discipline. We perform this exchange with a server-to-server, or *peer-to-peer*, interconnection. Fundamentally, this means that one server defers requests for symbols that it doesn't own to another server. For example, if a PROP client program needs access to some GNC telemetry symbols, the `prop` server will defer requests for these symbols to the `gnc` server. In this situation, the `prop` server effectively becomes a client of the `gnc` server. The `gnc` server will send the requested data to the `prop` server, which in turn will forward the data to the `prop` client program.[3] This situation is depicted in figure 1. The ownership field in the symbol dictionary provides some of the information needed to establish this connection. Basically, it provides the name of the server which publishes that symbol. Therefore, when a client subscribes with its server for a symbol published by a different server, that client's server looks for the publishing server running somewhere on the network. The ISP network registration service (NRS) is employed to perform this search. Once connected, the deferring server forwards the symbol subscription to the peer server, while the peer server registers the deferring server as its client. Symbol values will flow from the peer server to the deferring server then on to the client. We use this process to perform data security: by requiring configuration management of the symbol dictionaries, we can prevent clients from subscribing to symbols that they aren't authorized to receive.

[2]In fact, we have built a server which does not process a data source at all: it simply redistributes information generated by its publisher clients. We refer to this as a *null data source* server.

[3]Note that although they are independently processing different information, these two server processes are instances of the same program that were given different ownerships at run time.

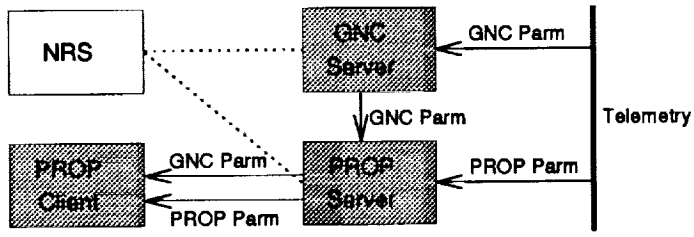


Figure 1: Inter-operator data distribution model. The PROP client can receive both PROP and GNC telemetry data, but it must acquire the GNC-owned data indirectly from the GNC server.

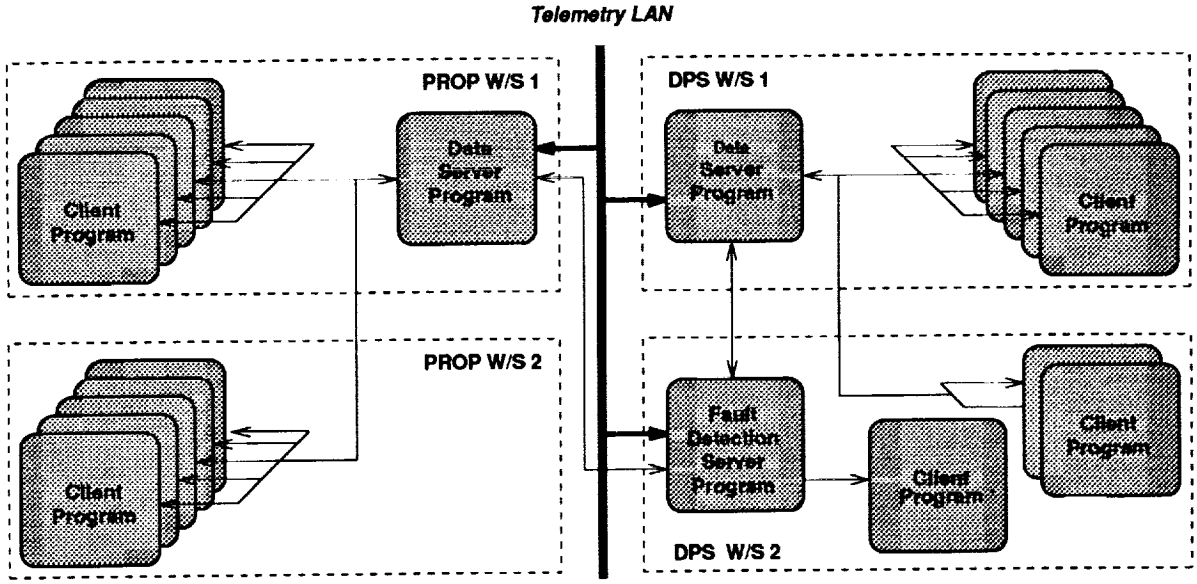


Figure 2: A distributed-resource situation. The DPS flight control discipline allocates a special server to the task of processing fault detection messages generated by the spacecraft software. This server distributes these messages to DPS clients directly, and to the PROP clients through a peer-to-peer connection.

These peer-to-peer connections also are important because they manifest how the model achieves global data processing. All of the processing necessary to make a particular data value "useful" can be performed by the server or client that "owns" that symbol. All other clients which may need the data, but don't own it, do not need to duplicate the processing; they simply request it from the owner. This feature essentially replaces the centralized computing feature which performs all of the data preprocessing before passing the data on to user applications.[4] This enables a wide variety of distributed resource allocation schemes in which each server provides a different data processing function. Figure 2 suggests one such distributed-resource situation.

3.3 Implementation

In realizing this protocol we have employed many industry standards. Primarily, we allow Unix do the things it was meant to do, such as managing resources. We use the TCP/IP protocols for network services, and we use point-to-point connection-oriented stream sockets for interprocess communications. Since the majority of our facilities employ the X-Windows system, we mimic certain facets of the X toolkit. Each ISP *event* has an associated enumerated type definition uniquely identifying that event. Many events also have associated data structures containing event-relevant information, such as a telemetered sensor value and time stamp. All of the events are distributed via message packets. Each message includes a header and a body, and the body can be of arbitrary length. Like the X toolkit, the ISP toolkit provides a *callback* mechanism whereby the programmer can register a series of functions to be called automatically upon arrival of a given event packet. Each registered instance of a callback function can have unique *callback data* to be passed to that function through the argument list. Essentially, this implementation allows programs built around the X protocol to use the same processing logic for "data" events that they use for "display" events.

When a client wishes to establish a session with a server, a variety of information must first be exchanged (figure 3 depicts the event exchanges). The client opens a socket connection with a server, then issues a connection-request event to it. If the server receives and accepts the connection request, it will reply with a connection-accept event. Upon receiving the connection-accept event, the client submits a series of subscription requests for all of the symbols it wants to process. The server validates all of these subscription requests and updates its internal symbol tables. When the client is done submitting subscription requests, it sends an event to enable the data stream. The server will then begin issuing data-change events to the client as its symbol values change. The client will receive these events and process them, possibly sending some synthesized data back to the server for publication. When the session has completed, the two programs exchange disconnection events to gracefully terminate communications and release resources.

4 Distributed Expert Systems

Some of the most exciting applications of the ISP technology lie in the support of distributed expert systems applications. Traditionally, these distributed expert systems have required that each component employ the same inference engine or expert system

[4]This is particularly important for data that is limit-sensed in some manner. The owner of the data applies the proper limits before forwarding the results to subscriber clients.

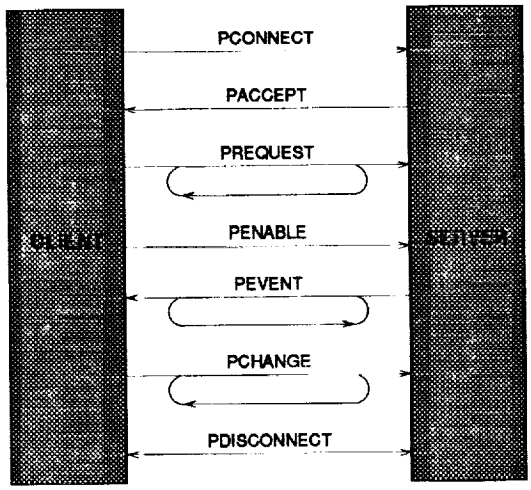


Figure 3: Client-server event exchanges.

shell. Using the ISP, however, we overcome this limitation by establishing a common interface enabling these expert system components to communicate with each other. Furthermore, since the ISP message packets can contain arbitrary data, a variety of information can be communicated. The information can be a sensor reading, a CLIPS fact, the results of a simulation or analysis, a fault message, and so on. These features promote a gradual development of heterogeneous distributed expert systems using the most appropriate inference engine or shell for the task.

Using the ISP, we have already begun to deploy some exciting distributed expert system applications for the CCC. We summarize a few of the highlights below:

PRS We have already deployed the ISP within prototype applications of the Procedural Reasoning System (PRS) [1, 2]. PRS is a real-time, multi-agent reasoning system capable of monitoring and controlling complex physical systems. The PRS development work performed for NASA has focused on the problem of handling spacecraft malfunctions and executing diagnostic procedures, assisting the operator in monitoring execution of these procedures, and adapting them to new situations. The PRS agents converse with each other in a peer-to-peer fashion, distributing the workload and knowledge bases according to the various tasks being performed by the application. These agents exchange information on a change-only basis, updating each other's data bases and contexts in order to monitor and control the behaviors of the physical system. Using the ISP, we empower sophisticated fault detection and analysis programs to provide status information to the PRS malfunction management systems, and provide an interface through which we merge PRS results into conventional telemetry displays.

VISTA Working with the Vista project team at the Rockwell Science Center Palo Alto Laboratory, we are developing an integrated decision-theoretic approach to the problem of managing the complexity of displays used in high-stakes, time-critical decision contexts [3, 4]. As a side effect of this effort, we have also employed the same belief network and utility models we use for display management for the problems of

fault detection, diagnosis, and ideal action selection. By using the ISP to distribute the results of these models among various applications, we are able to build integrated decision-making systems based on probability and utility. For example, we can distribute the results of decision-theoretic models to window manager clients to automatically select the optimum collection and configuration of displays for the human operator, thereby maximizing the presentation of relevant information while minimizing irrelevant clutter. We can also distribute the results of the various action models to the Flight Director, who integrates these actions within the current context into a prioritized list of actions for the astronauts or ground crews.

SELMON We are working with the JPL selective monitoring (SELMON) project team to integrate SELMON techniques within the distributive computing models [5, 6]. SELMON provides a collection of *sensor importance measures* to determine which of the observered sensor values contain the most interesting information. There currently are seven sensor importance measures that can be applied to each value. The four empirical measures, *surprise*, *alarm*, *alarm anticipation*, and *value change*, can be applied directly within the ISP servers as part of the preprocessing, providing a relative information content "score" to each changing sensor value. This score is passed along with the sensor value to the clients. The three model-based measures, *deviation*, *sensitivity*, and *cascading alarms*, are excellent candidates for client-based calculations which can be redistributed through the servers. Since it is then left to the consumer clients to deal with this relative information score, we can gradually deploy this additional information within the clients as they become more "intelligent." For example, we can use color changes to attract attention to sensor readings with high scores, directly influencing the user's decision-making efforts while monitoring component behaviors. We can also accumulate aggregate scores for entire displays, based on the accumulation of individual sensor scores, and prompt the user to concentrate his attention on that display because it contains more interesting information than others. These techniques fall nicely in line with the automated display management ideas of the Vista project. This work also is providing a substrate for the follow-on diagnostic reasoning embedded in monitoring (DREMON) project currently being pursued at JSC.

Beyond these few examples, there are also a variety of other distributed expert system applications, in various stages of development, which assuredly will benefit from a common communications protocol.

5 Information

We encourage questions and comments about the ISP from both inside and outside the spacecraft operations community, as we hope to diversify into other application areas as we continue our refinement of this distributed-computing protocol. For more information, or to obtain the latest ISP source and program distribution, contact the authors at NASA/Johnson Space Center, Mail Code DF6, Houston, TX, 77058, or send electronic mail to `barry@rpal.rockwell.com`, `nunikls@jscprofs.nasa.gov` or `nunispw@jscprofs.nasa.gov`.

References

[1] Georgeff, Kinny, Hodgson, and Lee. *PRS Operational Evaluation.* SRI International, Project ECD 333, Menlo Park, CA, 1993.

[2] Georgeff and Ingrand. *Real-Time Reasoning: The Monitoring and Control of Spacecraft Systems.* Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications, Santa Barbara, CA, 1990.

[3] Horvitz, Ruokangas, Srinivas, and Barry. *A Decision-Theoretic Approach to the Display of Information for Time-Critical Decisions: The Vista Project.* Proceedings of Fourth Annual Workshop on Space Operations Applications and Research (SOAR '92), NASA/Johnson Space Center, Houston, TX, 1992.

[4] Barry, Horvitz, Ruokangas, and Srinivas. *Vista Goes Online: Decision-Analytic Systems for Real-time Decision Making in the Mission Control Center.* Submitted to the Ninth Annual Goddard Conference on Space Applications of Artificial Intelligence, NASA/Goddard Space Flight Center, Greenbelt, MD, 1994.

[5] Doyle, Chien, Fayyad, and Wyatt. *Focused Real-Time Systems Monitoring Based on Multiple Anomaly Models.* Seventh International Workshop on Qualitative Reasoning, Eastsound, WA, 1993.

[6] Doyle, Charest, Rouquette, Wyatt and Robertson. *Causal Modeling and Event-driven Simulation for Monitoring of Continuous Systems.* AIAA Computers in Aerospace 9, San Diego, CA, 1993.

N94- 35070

ISTAR: Intelligent System for Telemetry Analysis in Real-time

Charles Simmons
Computer Systems Division
The Aerospace Corporation
El Segundo, CA 90245
simmons@arecibo.aero.org

ABSTRACT

The Intelligent System for Telemetry Analysis in Real-time (ISTAR) is an advanced vehicle monitoring environment incorporating expert systems, analysis tools, and on-line hypermedia documentation. The system was developed for the Air Force Space and Missile Systems Center (SMC) in Los Angeles, California, in support of the Inertial Upper Stage (IUS) booster vehicle. Over a five year period the system progressed from rapid prototype to operational system. ISTAR has been used to support five IUS missions, and countless mission simulations. There were a significant number of lessons learned with respect to integrating an expert system capability into an existing ground system.

INTRODUCTION

There are several trends taking place in the world of defense spacecraft that are making the mission support task increasingly difficult. First, defense spacecraft are becoming increasingly complex. With increased complexity comes an increase in the amount of health and status information that is sent to the ground for monitoring. It is currently not uncommon for a booster or satellite to downlink many thousands of telemetry measurands in a fraction of a second. These massive amounts of data must be quickly and reliably interpreted on the ground by mission control. Teams of controllers must detect, diagnose, and respond to anomalies in the vehicle. This task may be subject to rigid time constraints, especially with short duration booster vehicle missions. Current methods used by ground controllers for performing detection and diagnosis of anomalies are primarily manual and therefore slow and unreliable.

Second, there are an increasing number of defense space vehicles and booster launches to support. Launch frequency will increase in the future to support new satellite systems and to increase constellations for existing systems. The workload placed upon mission support will dramatically increase to accommodate this trend.

Third, there is a diminishing supply of qualified mission controllers. Typically, controllers with the greatest amount of experience and expertise are also the ones closest to retirement age. As they retire, they are often replaced with inexperienced controllers who may lack the capability to deal with complex problems.

Finally, there is the trend towards reduced staffing, and less contractor support. Due to cutbacks in the defense budgets, government spacecraft programs are being forced to make do with static or decreased budgets. This usually translates into hiring freezes or layoffs. The result is that there are fewer personnel available for mission support. Many mission support duties currently performed by vehicle contractors will be assigned to military personnel with considerably less experience and training.

It is clear that with respect to mission support, defense related spacecraft programs cannot afford to operate as they have in the past. Several government space organizations have realized this and have begun to look to advanced technologies for solutions [1,2,3].

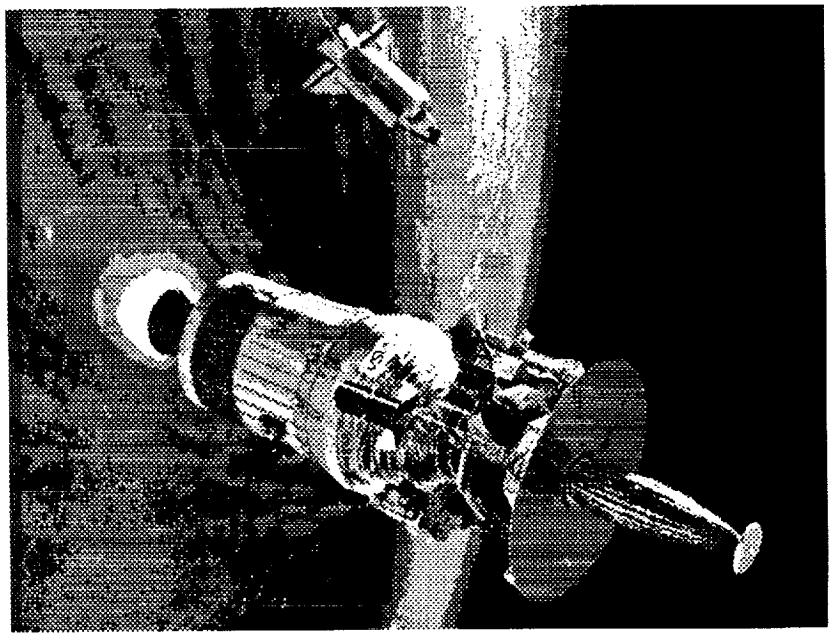


Figure 1: Inertial Upper Stage (IUS) booster deployed from Shuttle with Galileo Probe

IUS MISSION SUPPORT

The Inertial Upper Stage (IUS) vehicle presents a typical example of the problems outlined above. The IUS is a multi-stage booster (see figure 1) that supports delivery of civilian and defense satellites from low earth orbit to higher energy earth or interplanetary orbits. It is deployed from either the Shuttle or a Titan rocket. With many thousands of interdependent and redundant components, the IUS vehicle is highly complex. Ground controllers at the Air Force Consolidated Space Test Center in Sunnyvale, California are responsible for IUS mission control. They are required to monitor and reliably interpret over 1300 telemetry measurements to determine the health status of the vehicle. There are several significant problems that currently face the IUS mission control team in performing this task.

One problem relates to anomaly detection methods. Detection of anomalous conditions is primarily done by recognizing combinations and trends of telemetry measurements displayed on computer screens. Because this task is largely one of visual detection, the

possibility of missing or incorrectly interpreting a significant event is substantial. The potential for controller fatigue and overload makes this problem even worse. An IUS mission controller's shift can last anywhere from 8 to 18 hours. At the end of such lengthy shifts, fatigue and boredom levels are high, and the potential for operator mistakes is great. Also, there is the question of how many telemetry measurements a controller can adequately monitor. If the controller is asked to monitor too much data, information overload can significantly handicap his or her ability to recognize anomalous conditions. It is very clear that the IUS mission controllers needed a tool that could assist them in their vital task of vehicle monitoring.

Another problem relates to the time critical nature of IUS missions. The typical IUS mission is quite short relative to most spacecraft missions. The time from deployment to spacecraft separation is less than seven hours. If problems develop after deployment, there is a limited time window in which to respond. There is significant pressure placed upon mission support to quickly assess anomalies and recommend some action. In this "heat of battle" situation, the potential for mistakes by the controller is at its greatest. IUS mission controllers needed a tool that would assist them in making correct decisions in time critical situations.

Loss of expertise is a major problem currently facing the IUS program. To perform the tasks of a mission controller, the person needs to posses a significant level of monitoring "expertise". The controller must know what measurements to view at a given phase of the mission, and how to interpret their values. This expertise is based largely on experience and detailed knowledge of the internal operation of the IUS. Many of the current controllers have been with the program from its inception and have developed a tremendous amount of experience and knowledge of the vehicle. However, many of these controllers are at or near retirement age, or are transferring to other jobs. Loss of just a few valuable controllers could have a substantial impact on the ability of the mission control team to adequately support a mission. IUS mission control needed some way to retain portions of the expertise of these controllers for training new controllers, and for utilization during future missions.

ISTAR SOLUTION

In response to the problems facing IUS mission control, the ISTAR automated telemetry monitoring prototype system based on expert system, graphical user interface, and hypermedia technologies was commissioned [4,5,6,7]. The prototype was developed by the Expert Systems Section of the Aerospace Corporation, a federally funded research and development contractor. It was determined that, in addition to automated fault detection, the system should provide a variety of tools for assisting the mission controller in performing normal duties. In essence, the system was not to attempt to "replace" the controller, but rather to "assist" the controller in the task of mission monitoring. It was determined that the system should be able to receive telemetry in real-time or playback, detect anomalous IUS conditions, and interact with the user to isolate and diagnose the conditions. The system should provide a graphical user interface, allow replay and graphic presentation of past telemetry, provide diagrams representing the current state of the vehicle, provide on-line documentation pertinent to mission support, and operate on existing ground workstations.

The ISTAR system was designed to be operational with the Spacecraft Monitoring And Real-time Telemetry (SMART) system within MCC-8 of the Air Force Consolidated Space Test Center (CSTC). SMART is a real-time data acquisition and analysis system based on the System-90 Product by Loral Data Systems. It distributes its processing load among three types of equipment: telemetry front end equipment (TFE), a Host processor (DEC VAX 4000), and a network of workstations. Figure 2 shows the basic architecture for the SMART system. Telemetry is processed by the TFE and Host processor, stored on the Host disk, and broadcast to the workstations over an ethernet network, in the form of a current value table.

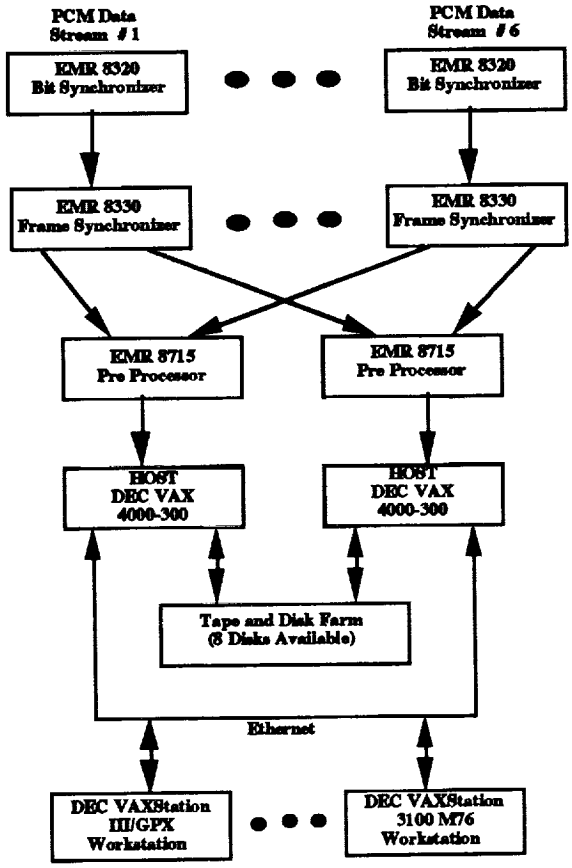


Figure 2: SMART System Architecture

The ISTAR system was implemented to run on existing workstations within the SMART system, which are Digital Equipment Corporation VAXstation IIIs and 3100s, running the VMS operating system. The ISTAR system utilizes a distributed architecture consisting of components for performing display, analysis, retrieval, and archive of IUS telemetry data. These *processes* communicate data and information by way of dedicated communication links (VMS mailboxes). Processes for display, analysis, and on-line documentation use Commercial Off The Shelf (COTS) products. The system can be operated in either real-time or playback mode. Figure 3 shows a functional diagram of the ISTAR architecture.

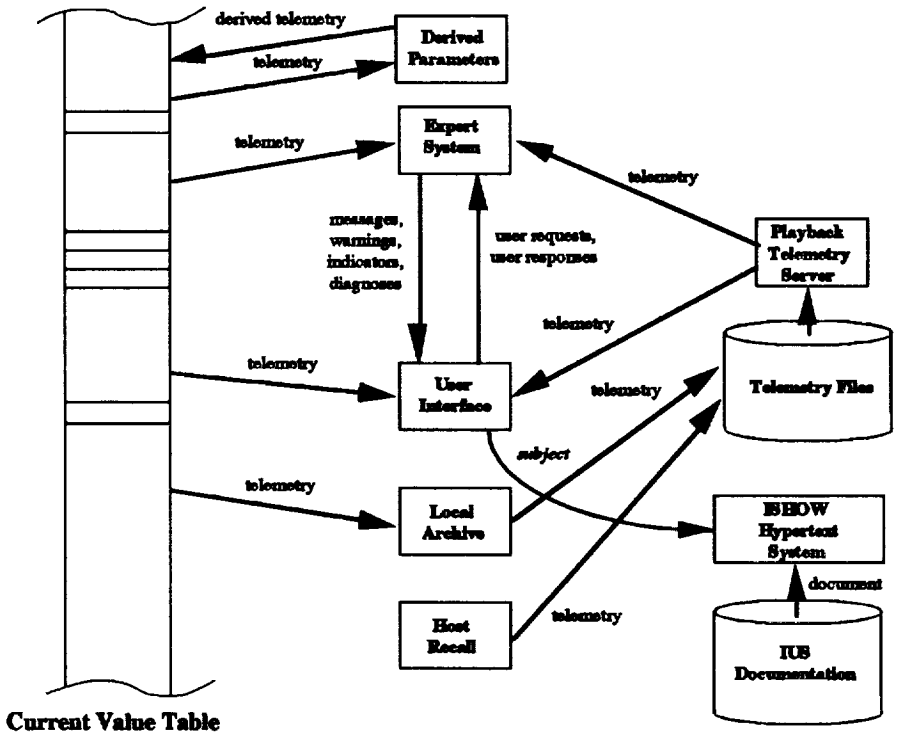


Figure 3: ISTAR Architecture & Data Flow

In the *real-time* mode, the Expert System Process performs analysis of dynamic telemetry data received from current value table (the current value table (CVT) contains one sample of every telemetry measurand) broadcasts from the SMART Host computer. It presents status information, and detects and diagnoses anomalous conditions. It utilizes the NEXPERT expert system shell developed by Neuron Data Corporation, and custom software. The User Interface Process is used to provide the user with status messages, warnings, diagnoses, and advice from the expert system, as well as animated schematics, telemetry graphs, and telemetry retrieval and manipulation tools. It consists of the Dataviews graphics development system developed by V.I. Corporation, and custom software. The Derived Parameter Process reads values from the CVT, performs calculations on the values, and inserts the resulting data back into the CVT. This derived data is then available to other processes through the CVT. The Local Archive Process samples specified telemetry data from the CVT and stores the data to telemetry data files. The Host Recall Process acquires specified telemetry data from the SMART Host computer and stores the data to telemetry data files. The resulting telemetry data files are available for playback into the ISTAR system. The IUS System for Hypertext on Workstations (ISHOW) documentation system displays IUS Orbital Operations Handbooks and other mission support documents. It consists of the Farview Hypertext System developed by Farsight Technologies Inc., and custom software. The User Interface process can request display of specific topics or sections from the ISHOW documents.

In the *playback* mode, instead of acquiring telemetry from the CVT, the expert and user interface processes receive telemetry from the Playback Telemetry Server Process, which reads a specified telemetry data file.

The ISTAR system was designed to operate concurrently on the same workstation with existing SMART software. The SMART workstation software's primary role is to provide graphical and alphanumeric displays of telemetry to the end user. It provides a large number of pre-configured displays for viewing various aspects of the IUS mission. During normal operation, the ISTAR system functions in the background. If an anomalous condition is detected, the operator is alerted and ISTAR is brought to the foreground, allowing further investigation of the problem.

In addition to an expert system, the ISTAR system provides an integrated set of tools for analysis of the vehicle. These tools include the following:

Graphing Tools - Graphs can be created on-the-fly showing either real-time or history graphs of telemetry. Support is provided for both analog and discrete telemetry measurements, and multiple graphs per screen.

Local Telemetry Archive - As the telemetry CVT is broadcast to the workstations, specific measurands can be stored to a file at the local workstation in real-time, for later replay into the ISTAR system. The CVT broadcasts, and thus the local archive, are limited to a once per second update rate.

Host Telemetry Recall - Specific telemetry data can be retrieved from the SMART System Host computer and stored to a file at the local workstation. The SMART disk farm typically contains telemetry from current and past IUS missions. Telemetry from the Host recall contains every sample of the requested measurand over the specified period.

Data File Manipulation Tools - Once telemetry is written to the local disk by the Host Telemetry Recall or Local Archive, a set of file manipulation tools can be used to search for events or combination of events, merge and overlay telemetry from different files, and edit parameter names and start/stop times. Using these tools allows, for example, construction of a file combining measurements from the current mission with those from a previous mission.

Expert Message Log - The message log provides a database of all messages received from the expert system. Associated with selected anomaly messages is a *diagnostic help option* that instantly configures ISTAR tools to investigate and justify the identified anomaly. By clicking on an icon next to the message, the user can rapidly view telemetry and information related to the specific anomaly.

On-line Hypermedia Vehicle Documentation - Several important mission support manuals have been placed on-line. These manuals are *hyper-linked* to allow rapid navigation to access needed information. The expert system may also bring up pertinent sections of these manuals in response to a detected anomaly.

KNOWLEDGE BASES

Knowledge bases are invoked by the user as needed from a menu of knowledge bases. Knowledge bases can be driven with real-time telemetry, or with past telemetry using the ISTAR playback utility. Knowledge bases may generate status, warning, anomaly

detection, or anomaly diagnosis messages. Anomaly detection and diagnosis messages may contain an anomaly explanation, justification, recommended action, hints on further investigation, and automatic configuration and branching to ISTAR tools such as graphs, schematic diagrams, and relevant sections of on-line manuals, that are pertinent to the anomaly investigation. The knowledge bases in ISTAR are of several types:

Status Knowledge Bases - These knowledge bases provide general vehicle status information to the user. For example, knowledge bases were developed that generate status messages based on vehicle events, and significant changes or limit violations of voltage, current, temperature, and pressure telemetry.

Monitoring Of Specific Mission Phases - Specific knowledge bases were developed for monitoring various phases of the IUS mission. For example, knowledge bases for monitoring deployment, stage separation, solid rocket motor burns, and separation were developed. These knowledge bases monitor specific events and attempt to detect and diagnose based on anomalous conditions that are most likely to occur, or most critical to detect. They are based largely on the heuristic knowledge acquired from vehicle experts.

Monitoring Of Specific Subsystems - Specific knowledge bases were developed to monitor subsystems or components of IUS. For example, knowledge bases to monitor the Power Distribution Unit subsystem, and the Signal Conditioner Unit Multiplexer component were developed. These knowledge bases typically run the entire duration of a mission, and are configured to detect predetermined anomalous conditions that are most likely to occur, or most critical to detect. They are based largely on the heuristic knowledge acquired from vehicle experts.

Automaton Of Handbook Procedures - Knowledge bases were developed which implement procedures from the IUS orbital operations handbooks. These procedures include uplink command sequences to accomplish various vehicle configurations, and redundancy restoration. For example, a procedure is available for restoring redundancy to an avionics channel after it has been lost due to an anomaly. These procedures typically monitor uplink commands to verify correct execution.

ISTAR DEPLOYMENT

The ISTAR project was initiated in May of 1988, by the Air Force Space and Missile Systems Center IUS Office, to be performed by The Aerospace Corporation. A rapid prototype was developed on an IBM PC in four months as proof of concept. The prototype utilized a simple COTS expert system development tool and was driven with simulated telemetry. A rapid prototype was then developed on the target machine, a DEC VAXstation III workstation. This prototype was completed in six months, and used a more sophisticated expert system development tool. In May of 1990, an operational prototype was completed and installed at the Air Force Consolidated Space Test Center (CSTC) for evaluation during simulations.

Prior to evaluation, ISTAR underwent extensive testing at CSTC to insure that it would reliably operate on current SMART workstations, yield results consistent with SMART, and function without adversely affecting the operation or performance of the workstation, the SMART Host Computer, or the SMART network. The groundrule for use of the

system was that no expert system results would be used without verification from an official Air Force telemetry data system. After completion of several simulations, it became apparent that the ISTAR system provided valuable, timely information to the support team. The excellent performance of ISTAR generated support from management at Air Force SMC, Aerospace, and the IUS prime contractor (Boeing Aerospace & Engineering) for its continued development and use.

Over the next three years, the system was used at CSTC to support five IUS missions, and countless mission simulations. During this time, the system often played a key role in detecting or verifying the presence or absence of anomalies. It was even used as a primary tool in the investigation of an inertial measurement unit anomaly, which lead to a launch abort. Enough confidence was placed in the system to allow it to assist in determining a go/no-go launch decision. Due to the fact that it was a prototype, ISTAR was allowed to continually evolve based on lessons learned during operation. This resulted in a much more effective system than would have otherwise been possible.

LESSONS LEARNED

The ISTAR system represented the first significant attempt to acquire an expert system capability for the Air Force SMC Space Launch Operations Program. Over the five years of development and operation, many lessons were learned. The following outlines the most significant lessons learned (in no particular order).

Need champions at the top.

A significant lesson learned was that it is very important to have the support of upper management. The new technology had to be sold to management both initially and continually. Through presentations and demonstrations, expert systems combined with advanced graphics were shown as a way to address the current problems of reduced staffing, and more reliable anomaly detection and diagnosis. Frequent demonstrations were given to management over the development period showing progress; this kept them interested in the project. Unwavering management support allowed continuous funding over the five year development period, and made insertion of the prototype into the operational environment possible.

Need a user who is a champion of the new technology.

In addition to champions at the top, a champion in the field is also needed; that is, a member of the user community who understands and can sell the new technology to other users. Such a person makes technology infusion a lot easier. A vehicle expert was available who possessed a global knowledge of the vehicle, anomaly detection and diagnosis, and who had enthusiasm for developing a support system. The expert served as the primary source for initial knowledge base development. Perhaps of most value was his ability to generate enthusiasm for the new capability among IUS management, and other mission control team members, and to provide direct use and evaluation of the system during mission support.

System must be of maximum benefit to users.

To gain acceptance by IUS mission controllers, the ISTAR system had to provide capabilities that addressed their needs. To build just an expert system would have addressed one particular need, but perhaps not their strongest need. Therefore, the ISTAR system was designed with the philosophy of providing an integrated set of tools, expert systems being one of those tools. Other important tools included graphical subsystem browsing, real-time and historical graphing, local data archiving, Host data recall, data file search & manipulation, and on-line vehicle documentation. This proved to be a wise decision. After the users decided they liked the system, it was easy to add more expert system capability.

There are benefits to making the system transparent to the operational environment.

There was much resistance to allowing ISTAR into the operational environment. There was fear that system errors might negatively impact the existing operations. Because of these fears, strict configuration management, and political considerations, it was decided that no dedicated hardware would be allowed into the operational environment for the ISTAR effort. In addition, no operational code could be modified to accommodate ISTAR. Thus, the only way to gain access was to run transparently on existing workstations. Fortunately, ISTAR was designed to do just that. No software or hardware changes to the SMART workstation or Host were required, just a simple change to the system startup file. Once installed, the user was given the option to run the ISTAR system alone, the SMART system alone, or both concurrently. The fact that the ISTAR system could simply be switched off if not desired, without any impact to SMART capability, was key to gaining acceptance. Moral: If you can insert the new technology in a transparent, non-threatening way, it is best to do so.

Start with simple, rapid prototypes, and proof of concepts.

A rapid proof of concept prototype proved to be a valuable exercise. Within four months, a prototype was built showing a solution of a very limited subset of the entire problem. These four months proved to be a great learning experience for the customer as well as the developers. The developers learned that there were going to be difficult problems to overcome, such as data and knowledge acquisition. The customer learned that what they thought they wanted in the beginning was not exactly what they wanted or needed. The result of the prototyping effort was a new perspective on program needs, and problems to be solved. It helped immeasurably in defining requirements for the next phase.

Make sure all users are involved from the start.

It was learned that getting all potential users involved in the beginning is key to gaining user acceptance. In the initial development phases, only Aerospace Corporation mission controllers were involved in the system definition. During later development phases, the IUS prime contractor Boeing Aerospace & Engineering was funded to participate in the project. After only a short period of time, Boeing engineers made important recommendations on additional system requirements, which resulted in greater user acceptance.

Most effort spent in developing and integrating core system.

About two-thirds of the entire ISTAR project effort was spent in developing the core system. Less than one-third of the effort was spent on knowledge base development. ISTAR development tasks included designing the basic architecture (10%), interfacing the core architecture with the existing environment (25%), interfacing the core system with COTS products (25%), and development of special functions and tools to augment the capabilities of the various COTS products (40%). After the core system was built, the knowledge base development effort was increased. It should be noted that COTS products are now available with many of the features that had to be custom developed for ISTAR.

Need strong software developers.

Even though COTS products were used, custom software had to be developed to interface the COTS products to data sources and to each other. In addition, software had to be written to augment the built-in capabilities of the COTS products. The system ended up with over 30,000 lines of source code, excluding COTS products. The software was developed by programmers with many years of experience in the target language, operating system, and target machine architecture. In general, timely, reliable software products were created. Due to the many subtle coding and integration challenges, schedule impacts would have undoubtedly occurred without experienced programmers. It should be noted that COTS products are now available with many of the features that had to be custom developed for ISTAR. If a similar effort were started today, much less custom software development would be necessary.

Use COTS products whenever possible.

One significant lesson learned was the advantages of using COTS products over custom development. First, support costs are lower for COTS products than for custom software because the maintenance costs are shared by all users. Second, COTS products are more adaptable to changing requirements over a project's life span due to support of multiple platforms, multiple applications, and multiple users. Third, the features of a COTS product are often thoroughly tested by a large community of users. The advantages of COTS over custom could be clearly seen in experiences with the SMART system, which was a totally custom software development. After DEC announced that they would no longer support the VWS user interface, it was desired to move SMART to the new DECWindows interface, based on the X-Windows standard. However, due to the large amounts of custom code developed around VWS, and low-level calls into the graphics hardware, such a transition would have required virtually a total rewrite of the SMART System-90 display software. However, such a transition would have been virtually automatic for ISTAR, which relies heavily on COTS, since the COTS vendors had already worked out the port to DECWindows.

Choose COTS products carefully.

It is important to choose the right COTS products. Since the time that COTS products were chosen for ISTAR, many similar products have come and gone. Had the system been built around a failed product, it would have been a disaster for the project. It was also learned that the COTS product must be able to handle current as well as unanticipated future requirements. It often makes sense to choose a product that is more

robust than needed. As requirements change, the system may eventually grow into those capabilities. It is very important that the COTS product provide an open design to allow custom extensions if needed. Also, financial stability of the vendor should be a consideration. New and better COTS products are being offered at an ever increasing rate. In fact, if a COTS decision were made today for this effort, totally different packages would be selected. Therefore, it is important to keep the overall system architecture generic enough to allow upgrade to a different COTS product when necessary, without extensive redevelopment.

Start with simple, well-defined knowledge bases first.

The first knowledge bases developed for ISTAR were for vehicle redundancy management anomaly detection, diagnosis, and resolution. These knowledge bases were rather complicated, and usually required user input of certain information before diagnoses and restoration could be made. While these knowledge bases worked well with simulated cases, they represented situations that were very unlikely to occur during missions. A decision was then made to develop broader, more simplistic knowledge bases with emphasis on vehicle anomaly state determination and simple anomaly detection. These knowledge bases proved to be valuable assets for mission controllers in detecting vehicle developments that might have otherwise been missed -- one of the major goals of the system. After early successes, these simple knowledge bases were enhanced, and more sophisticated knowledge bases were attempted. It was also learned that anomaly isolation and diagnosis are much more difficult than detection. Automated diagnosis would be nearly impossible in many failure situations. In these situations, it was best to give the users pertinent information that would assist them in performing their own anomaly isolation and diagnosis (such as relevant graphs, diagrams, and vehicle documentation).

Automate diagnostic information in spacecraft operations manuals.

Knowledge acquisition from a human expert is clearly a difficult task, and the major bottleneck in developing an expert system. However, much of the knowledge necessary to develop simple knowledge bases may be contained in vehicle operations handbooks. Several IUS handbooks were identified that provided valuable information for ISTAR knowledge base development.

Make sure the expert can read and understand the knowledge base.

For ISTAR, the knowledge acquisition process was typically an iterative task performed by an expert and a knowledge engineer. The knowledge engineer interviewed and/or obtained written requirements from the expert, generated an initial knowledge base, reviewed the knowledge base content and performance with the expert, and made agreed upon changes. It was very helpful to have an expert who could understand the basic concepts of a knowledge base, and the syntax of the rules. After only a short period of time, knowledge bases were given to the expert for visual verification. Eventually, the expert was able to make simple changes to knowledge bases. This accelerated the knowledge base development effort. It also helps to have an expert system product that presents rules in a readable, English-like syntax.

Tackle problems that are best suited for expert systems.

One of the lessons learned was that not every problem is suitable for an expert system. It worked out best to tackle problems that were declarative in nature, rather than procedural. Detecting combinations of vehicle events that indicate a problem was a very appropriate use of the expert system tool. However, trying to implement procedures in a rule based expert system proved to be difficult. If procedures are to be implemented, it is best to use an expert system shell with procedural capability. In general, the approach taken here was to try to mimic the declarative techniques that the human expert used in monitoring and detecting anomalies.

Develop a reasonable verification and validation standard.

Validation of expert systems is always difficult, and sometimes impossible. The approach taken by the NASA RTDS project was adopted here [1]. Their approach was to validate expert systems based on use. In order to qualify for mission use, a knowledge base had to work flawlessly in a specified number of mission simulations. Our adoption of this rule was that no new knowledge bases, or any other ISTAR capability, could be used during a mission without correct performance in at least two mission simulations and one mission dress rehearsal.

Common user interfaces needed.

The screen layout and functionality of ISTAR interface are much different than that of the SMART system. Therefore, users of SMART had to be trained on how to use the ISTAR interface. A common screen layout and functionality for ISTAR and SMART would have resulted in reduced training requirements and increased user acceptance. Future adherence to emerging standards for spacecraft command and control user interface design will address this problem [8].

Choose a powerful workstation.

One lesson that was quickly learned was not to underestimate the amount of computing power needed for the application. As more COTS products, knowledge bases, and system features were added to the ISTAR prototype, the computing capacity of the workstation was quickly reached. Midway through the development it was necessary and permitted to upgrade to a more powerful hardware platform that maintained compatibility with the SMART system.

Deal with real-time data and control issues early.

One of the technical challenges was determining how to connect to and process the real-time data stream. The problem was more difficult than was originally anticipated. However, it was important to establish this connectivity in order to demonstrate the capabilities of the system in the operational environment. Demonstrating successful real-time processing of live vehicle data greatly increased the credibility of the system.

It is important to have support for derived measurands.

Capabilities for derived telemetry measurements proved to be important for a number of reasons. There were many situations where it was desired to have telemetry parameters

combined in some predetermined fashion. For example, taking a voltage and a current to produce a wattage. The result of such calculations was often needed by the expert system, as well as the user interface. Rather than burden the expert system with the overhead of such calculations, a separate derived process was created. This process allowed derived parameters values to be written to the SMART current value table. The derived parameters could then be displayed by the interface process, incorporated by the expert system process, and even archived by the local data archive process.

It is important to have local telemetry archiving and playback capability.

Local telemetry archiving at the user workstation proved to be a very valuable capability. The prime advantage was that once the telemetry was archived to the workstation disk, it could be accessed freely and rapidly. There were several situations where the network had gone down during mission operations. Those workstations relying on SMART network distribution of playback data were unable to perform analysis during the down time. However, users of ISTAR were able to analyze locally archived telemetry during this period. Also, this analysis could be done at a speed superior than SMART network access would have allowed.

It is desirable to have an integrated on-line documentation system.

ISTAR provides on-line hypermedia access to important IUS mission support documentation. The hypermedia links allow rapid access to needed information in text and graphical form, by mouse-clicking. A valuable capability was the ability to perform inter-document referencing. In many cases, while browsing a document, the user can request and branch to information on a subject that is contained in a separate document. Another valuable capability is the ability of the expert system to recommend and allow branching to relevant sections of the documentation, based on the anomaly detected. The hypermedia capability helped the mission controller decrease reliance on paper documents, and improved access to critical information during IUS missions and simulations.

SUMMARY & PLANS

This paper has presented an overview and lessons learned from the ISTAR expert system prototype developed by The Aerospace Corporation for the Air Force Space and Missiles Systems Center, for support of IUS booster missions. In 1993, the ISTAR system was turned over to Boeing Aerospace & Engineering, the IUS prime contractor, for continued development and maintenance. The existing ISTAR system is currently being ported to the UNIX operating system and Sun workstations, to allow easier integration into the future IUS ground system, which will be based on the Loral Open System 90 telemetry system.

REFERENCES

1. Heindel, T.A., "Shells for Mission Control," Aerospace America, October, 1991

2. Atkinson, D.J., et. al., "SHARP: Automated Monitoring of Spacecraft Health and Status," Space Operations Applications and Research Conference Proceedings, June, 1990

3. Schwuttke, U.M., et. al., "MARVEL: A Knowledge-Based Productivity Enhancement Tool for Real-Time Multi-mission and Multi-Subsystem Spacecraft Operations Status," Jet Propulsion Laboratory Technical Report, June, 1991

4. Simmons, C.B., "Intelligent System for Telemetry Analysis in Real-time (ISTAR) Part I: Overview & Project Legacy," The Aerospace Corporation Technical Operating Report (3464)-4, September 1993

5. Simmons, C.B., "Intelligent System for Telemetry Analysis in Real-time (ISTAR) Part II: User's Guide," The Aerospace Corporation Technical Operating Report (3464)-4, September 1993

6. Simmons, C.B., "Intelligent System for Telemetry Analysis in Real-time (ISTAR) Part III: Programmer's Guide," The Aerospace Corporation Technical Operating Report (3464)-4, September 1993

7. Lee, J.S., Simmons, C.B., "Intelligent System for Telemetry Analysis in Real-time (ISTAR) Part IV: ISHOW Hypermedia Reference Manual," The Aerospace Corporation Technical Operating Report (3464)-4, September 1993

8. Air Force Space Command, *Integrated Satellite Control Human Computer Interface Standard Draft Version 1.0,* ISC HCI Working Group, August 1993

N94- 35071

Engineering Large-Scale Agent-Based Systems with Consensus

A. Bokma, A. Slade,
S. Kerridge & K. Johnson

Artificial Intelligence Systems Research Group,
SECS,
University of Durham
DURHAM DH1 3LE
e-mail: Albert.Bokma@durham.ac.uk

Abstract

The paper presents the Consensus method for the development of large-scale Agent-Based Systems. Systems can be developed as networks of Knowledge Based Agents (KBA) which engage in a collaborative problem solving effort. The method provides a comprehensive and integrated approach to the development of this type of system. This includes a systematic analysis of user requirements, as well as a structured approach to generating a system design which exhibits the desired functionality. There is a direct correspondence between system requirements and design components. The benefits of this approach are that requirements are traceable into design components and code thus facilitating verification. The use of the Consensus method with two major test applications showed it to be successful and also provided valuable insight into problems typically associated with the development of large systems.

1. Introduction

In recent years there has been a noticeable shift from large mainframes towards networked hardware architectures, requiring a different approach to the construction of large systems to run on these new platforms. In addition, there is an increased need for faster and more intelligent systems, which, however, need to be engineered to the same rigorous standards expected of systems developed for more traditional implementation paradigms. Consequently, there is a need for methods which cover the following points:

- addressing the special needs of knowledge based systems
- developing distributed solutions
- reducing response times to meet real-time requirements
- meeting validation, verification as well as quality assurance constraints

The Consensus method focusses on the design concerns of:

- distributed knowledge based applications
- systems of arbitrary size (typically large scale)
- real-time systems

A comprehensive method for the development of such systems requires a systematic approach to help the user to specify a solution for a given problem. In the design of concurrent Knowledge Based Systems (KBS) it is important to handle both the technical aspects of specifying systems, as well as the organisational aspect of managing projects, especially where the development of large systems are concerned. The Consensus method includes both a specification technique and a life-cycle model, although the latterwill be covered in less detail.

The method specialises in the development of large real-time applications, that build concurrent knowledge based components into an agent-based architecture, covering the complete life cycle from concept through to operation, and includes:

- an analysis of the need and functioning of the system,
- the selection of design approaches appropriate for the task,
- detailed design and implementation,
- verification, validation and testing,

The Consensus method, is the result of research into software development methods and approaches, and has been refined through practical experience gained from applying the method to the development of a large-scale application in Air Traffic Control, as well as a second application concerned with Dynamic Tactical Planning.

2. Current Developments

There are a number of approaches which have been influential in the formulation of the Consensus method, which can be divided into three distinct areas:

- life-cycle models
- system specification methods
- KBS approaches

There are various life-cycle models which have been proposed over the years to help in a ordered and systematic development and maintenance of systems. Many of the traditional sequential models *[ROY70]* and their variants have been found wanting as they prescribe a one-way development process which does not promote iteration and feedback. This is a serious shortcoming as there are a number of factors that have to be weighed against each other in the process of developing systems which both fulfil the needs and are feasible and well engineered. The Spiral Life Cycle model *[BOE88]* has embraced what is actual development practice, which the other life-cycle models have tended to deny. As this approach is much closer to actual development practices and particularly suited for the development of distributed systems Consensus has opted for this model.

The Spiral Life Cycle Model is iterative by nature and therefore an iterative method for system analysis and design is required. One such method which has increasingly been used in the last two decades is structured analysis and design. Consensus has been influenced particularly by Hatley-Pirbhai *[HAT87]*, a complete specification method for real-time systems which has been used successfully in industrial and commercial applications. It uses both structured analysis and design and proposes the joint development of a system requirements model and a system architecture model. The motivation shared by Consensus is that for successful system specification one needs to balance what the user wants with what can be done, given the available implementation technology. As Hatley-Pirbhai has been successful in actual system development this proved to be a good starting point for a system specification method.

Hatley-Pirbhai also specialises in real-time system development which Consensus is also interested in although it does not specialise in it. At the same time real-time systems are becoming incerasingly widespread and provisions that can be made for them will prove useful in the future. Amongst real-time extensions to the basic structured analysis and design approach, apart from Hatley-Pirbhai there are also Ward-Mellor *[WAR85]* and DARTS *[GOM84]*. There are no fundamental differences between the notations of Ward-Mellor and Hatley-Pirbhai, which would have been significant for Consensus (which adopts the Hatley-Pirbhai conventions). By contrast, DARTS has proven to be interesting as it uses the concept of dividing the problem solving activities into tasks. The guidelines for the identification of tasks were useful for another goal of Consensus, namely the identification of components which could engage in collaborative problem solving and operate in a distributed fashion.

One problem with structured specification methods including Hatley-Pirbhai is that they are geared towards procedural implementation paradigms. Consensus, however, addresses the specification of parallel and cooperating KBS. A different approach was therefore needed to deal with the specific needs of distributed systems. There are a number of approaches which have been developed in recent years and may be thought of as being relevant to Consensus. A number of approaches, methods and tools were examined. Amongst the different approaches the most significant for Consensus was Cassandra *[CRA89]* as it provides an architecture paradigm specific for distributed KBS, which fills the gap left in the architecture model of Hatley-Pirbai.

Cassandra does provide a paradigm for developing KBS as networks of smaller knowledge based components embedded in a collaborative system architecture. This satisfies the goal of Consensus to develop parallel and cooperating KBS. At the same time little help is given as to how to go about specifying large systems and how to arrive at a suitable system architecture for a given problem. By combining the benefits of these approaches and by adding the necessary methodological support to steer the specification process, the systematic specification of distributed KBS can be achieved. Adaptation of existing analysis methods and experience from developing two major test applications helped to fill this gap and to generate guidelines to direct the specification process towards suitable system acrchitectures.

In recent years a number of KBS technologies were developed. One of the most widely known is KADS *[SCH89] [DEH92]*, which allows the modelling of the application domain from different perspectives with the help of a number of interconnected models. Other projects like Reakt *[FJE92]* and AC-Knowledge *[ACK92]* have sought to enhance KADS and to deal with issues such as real-time specification and knowledge acquisistion. Knowledge acquisition tools will be useful for a number of applications but most tend to guide the specification process towards emulating the application domain. Although this may be desirable in some cases the specification of large and complex systems requires to balance both the need of the user and constraints imposed by software engineering principles and the available implementation technology. Structured analysis also uses a modelling technique, but it is more flexible and allows the developing of the requirements and architecture in a way that suits both the need of the user and the constraints of the system. The use of knowledge elicitation tools should therefore be confined to the specification of individual components which have a strong expert knowledge element that cannot be addressed with structured analysis alone. Alternatively, they can be useful as a prototying tool to develop an operational model of components to determine its behaviour and to ensure completeness of requirements.

3. The Consensus Method

The purpose of the Consensus project is the generation of a software engineering method for the development of large parallel Knowledge Based Systems, this involves the development process, from conception through to operation, including the requirements definition, the design of the solution and the implementation culminating in a working system.

Consensus believes to have produced a comprehensive, yet compact method, which is intuitive and easy to use while being effective enough to deal with large complex applications.

It's structured analysis approach facilitates the development of large complex systems by subdividing them into manageable parts which can in turn be further analysed and specified. It allows for systematic exploration of the requirements and forces the analyst to focus on the specific requirements of sub-components and their inter-relationships.

Given the current shift towards networked architectures, distributed systems are becoming more common-place. Distributed, collaborative problem solving is in step with these developments and allows exploitation of the benefits of networked architectures; the response of Consensus is to develop systems as networks of medium grain-sized knowledge based agents, which engage in a collaborative problem solving activity.

Paired with this development is the increased need for intelligent processing in industrial and commercial applications satisfying stringent software quality constraints. Thus the successful advance of Knowledge Based Systems especially for large applications requires a distributed and modular architecture and a method which can deal with the specification of systems of that type.

From a software engineering perspective, one consideration that is often overlooked is that in order to develop successful solutions,

the analysis of desired functionality has to be weighed against constraints imposed on the solution by the implementation platform. Therefore, the analysis of requirements cannot be divided from the specification of the system architecture. Consensus adopts an integrated approach where the requirements are developed in conjunction with the design of the solution, thus ensuring that at every stage of development the resulting system fulfils the requirements while taking account of the constraints.

The structured analysis and design approaches are used to provide a distributed architecture of independent tasks which communicate with other tasks in the overall system process. These can in turn be translated into a network of distributed KBA. Consensus combines a distributed agent architecture with structured analysis and design providing a software engineering method for developing systems as networks of KBA.

The key concepts that the method makes use of, can be summed up as follows and are depicted in figure 1:

- the System Specification should comprise not only the System Requirements but also the System Architecture - these should be developed together;
- the System Requirements specify what the system is to do and is independent of the implementation technology;
- the System Architecture specifies how the system is to be structured and is dependent on the implementation technology.

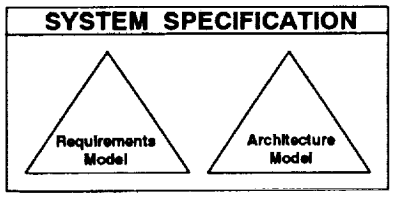


Figure 1: System Specification

The System Requirements and System Architecture are considered together, starting with a high level model of the system, and proceeding by refinement and iteration until a detailed, complete and comprehensive specification of the system is produced.

The primary benefit of this approach is that early partitioning and allocation of functions in the system helps to identify critical functions, which can be prototyped or reappraised, thereby leading to a clearer understanding of the need and the best way to structure the design. This results in a comprehensive and integrated system development process and favours traceability and consistency which are vital in the development of large, complex systems.

3.1 The Requirements Model

Problems are frequently too large and complex for solutions to be developed in one step. The overall problem can be considered as a complex task, and can be more readily tackled by partitioning it into a number of sub-tasks together with an indication of how these sub-tasks interact with each other in order to solve the problem. This process of partitioning, when applied repeatedly, reduces the complexity of an individual section to a level where the problem can be more easily understood and specified.

The purpose of requirements analysis is to specify the requirements as opposed to designing the software components, the focus is on *what* the system must do and not on *how* this is achieved. The result of applying this technique is the generation of a model of the problem to be addressed.

As information flows through a system it is transformed: the system can accept a variety of different forms of input and applies hardware, software and human interaction in order to transform the input into output. Structured analysis is a technique for modelling the flow and content of information by subdividing the overall task performed by the system into a series of individual processes. In the modelling process three different models are used:

- *Data Flow Diagrams* - These specify how data flows through the system and is being processed .

- *Control Flow Diagrams* - Many applications are time-dependent and process control information rather than data. Optional additional control flow diagrams specify flows of control information and control signals
- *State Transition Diagrams* - These describe the different states of a process and the transitions between states. They can be used during validation to ensure that control specifications are complete.

A process is the encapsulation of some requirements which perform a specific task. A software system can be represented as an information process and the overall function of the system can be graphically represented as a single process, with a number of inputs to the system from external entities and a number of outputs of the system. As depicted in the figure below, the process can be graphically represented as a bubble and the external entities as rectangular boxes. The process is connected to the external entities via arrows which represent data-flow as well as flow direction, thus denoting inputs and outputs.

The single process can be broken down into a number of smaller processes together with data-flows between them. Each of the processes thus identified can then be further broken down and so on. *Data Flow Diagrams* (DFD) are a graphical representation depicting data flows and processes which are applied to the data in the process of transformation from input to output, and may be used to represent the system at any level of abstraction. This means that it is possible to generate a hierarchy of data flow diagrams to depict the system at arbitrary levels of abstraction. The aim of the Requirements Model is to generate such a hierarchy.

The development of the Requirements Model proceeds top-down, from the most general abstraction to the most specific in a series of levels. By convention the diagrams are labelled, starting from level 0 to level n. Level 0 represents the system at the most general level. It is also known as the context diagram, representing the system as a single process showing its connections to external entities, as depicted in the figure below:

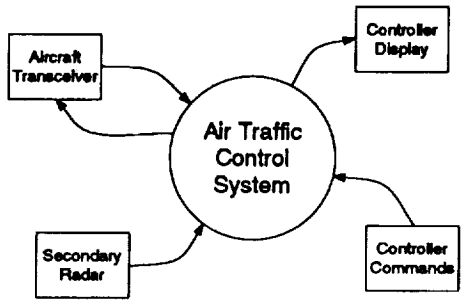


Figure 2: A Sample Context Diagram

In the process of breaking down individual processes it is essential to maintain the existing flows which connect to the parent process. The fundamental principle is that the connections are inherited from the parent process. Thus a check is carried out to ensure that the flows of the parent process are equivalent to those of the child processes, and is called balancing. In addition to inherited flows, new flows need be introduced to interconnect the child processes. All components depicted on the diagram need to be labelled (for reasons of clarity these labels have been omitted from the sample diagrams) and explained in separate process and flow descriptions which are entered in the *requirements dictionnary*. The figure below shows a sample level 1 diagram:

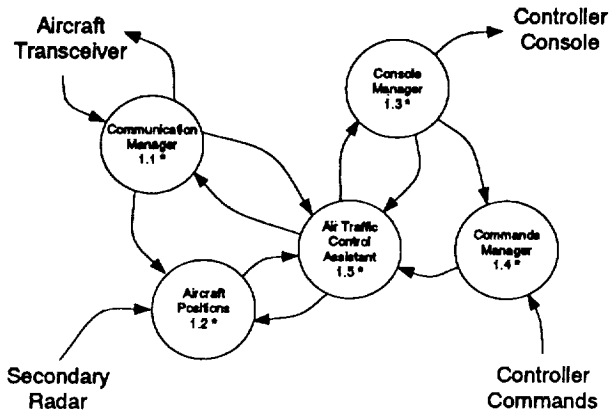


Figure 3: A Sample Level 1 DFD

Processes are labelled with a name and a unique nested numbering system indicating the level and parentage of the each process. There may be a need for data-stores to temporarily store data required for further

processing. These are denoted by two horizontal parallel lines with the datastore name in between, as well as arrows to and from them to indicate which processes modify or read the information contained in them.

Once a point is reached in the analysis where a process cannot be usefully decomposed further, it is specified by a *process specification* (PSPEC) which is readily implementable and describes how the inputs of the process are transformed into the outputs. Processes at this level are known as *primitive processes*. It may take the form of pseudo-code or a function to describe the algorithm used to carry out the transformation. At this point the decomposition stops for that process. The decomposition however continues for all other processes until a point is reached where all processes are primitive and specified by their respective PSPECs. Once this point is reached the hierarchical set of DFDs that have been produced for the system are complete. The PSPECS are then gathered in a separate section for process definitions.

There are a number of applications which are time-dependent and may process more control information than data. Real-time systems in particular interact with external entities on a time frame that is dictated externally and this places a number of important constraints on the demands of the system specification technique. In particular such a technique has to allow the analyst to represent control flow and control processing as well as the usual data flow and processing, which is achieved bt Consensus.

In order to distinguish normal data flows and processes from control processing, some additional notation is required. Thus control flows are used to describe the flow of control information and continuing the conventions established for data flow and processing, control flows are denoted by a dashed line as opposed to a solid line. A process that handles only control flows is called a *control process*, again denoted by a bubble with a dashed line, and associated with each will be a *control specification* (CSPEC). The appropriate descriptions of control processes and flows needs to be entered in the *requirements dictionnary* (and the associated CSPECs entered).

Control flows and processes are crucial to the behaviour of a system, and there are two ways in which the appropriate functional behaviour can be defined. A *Program Activation Table* shows the different permutations of states and the actions taken in each case to ensure that all possible states have been covered and receive the appropriate action. Alternatively, *State Transition Diagrams* (STD) can be used to give a behavioural model of the control process which shows the different states the system may be in and the connection between states. The latter is of particular significance for the purpose of verification. The use of STD (otherwise known as finite state machines) can be extremely powerful, but this observation is not widely appreciated, especially in current approaches for the development of KBS.

In some applications system inputs must be received at a certain rate and sytem outputs generated within a given time. These requirements are termed *timing requirements* and are associated with specific flows and processes. There are a variety of potential timing requirements from less strict response times to user input for interactive interfaces, to strict output rates of real-time and safety critical systems. An indication should therefore be given as to whether or not particular timing requirements are critical. Timing specifications connect specific input and output events. As a result there needs to be a list of input events and the respective output events and the timing relations between them.

3.2 The Architecture Model

The Architecture Model is developed in conjunction with the Requirements Model, describing and defining the system in terms of the implementation platform. The process of functional decomposition should produce collections of functions, which can be grouped together into separate processes which can work concurrently, and collaborate by communication towards the overall goal of the system.

The Consensus method specialises in the development of parallel KBS architectures, where a collection of interconnected KBA

collaborate to meet the requirements of the application. These independent KBA communicate via pre-defined data channels and co-operate towards the system's goals.

The purpose of the Architecture Model is both to translate and map the requirements into actual system components, as well as to guide the requirements analysis to take into account potential constraints imposed by the target platform. It may be possible to identify groups of potentially concurrent functions at different degrees of resolution, and as the decision on this matter is more dependent on implementation constraints it needs to be based on the system Architecture Model. The aim is to achieve a specification which fulfils the requirements and is implementable. There are three important aspects provided by the Architecture Model:

- an architecture paradigm for large, parallel KBS
- a mapping of requirements into design components
- feedback to help reject unfeasible requirements

The construction of large KBS as a single, monolithic system presents problems for a number of reasons. The execution of KBS is not strictly procedural by nature, and an order of processing may have to be enforced to meet requirements. Conflicts may arise when different knowledge sources or rules want to execute on the same data, thus interfering with each other. The risk of these situations occurring becomes more acute as systems increase in size. In addition, large monolithic systems may be too slow to meet real-time performance requirements. An approach which leads to the development of modular systems, which can execute concurrently on a distributed platform, and which can deal with interference problems is therefore preferable. The Consensus method builds systems as collections of connected knowledge based agents (KBA), each with the same basic structure containing:

- a local blackboard
- local knowledge sources
- a local controller
- communication channels

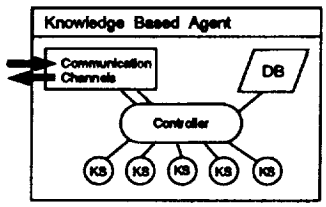


Figure 4: A Knowledge Based Agent

The local blackboard contains data and results of processing. Access to it is restricted to the local knowledge sources and controlled by the local controller. Each KBA has a set of local knowledge sources which perform the tasks assigned to that KBA. They operate solely on the information contained in the local blackboard and may request the local controller to communicate with other KBA to transfer data. The local controller is in complete control of processing at a local level within the agent. It schedules the execution of local knowledge sources which operate on the local blackboard and initiates communication with other KBA on behalf of its knowledge sources as well as receiving messages and posting them to the local blackboard.

The KBA which make up a particular system are connected by pre-defined communication channels. A number of important constraints are imposed on the communication mechanism in order to preserve the benefits of modularity and reduce interference between individual KBA. Thus communication with other KBA is managed by the local controller on behalf of its knowledge sources, and no other contact between KBA is allowed. These constraints are designed to help ensure consistency between KBA and in avoiding deadlock.

There are a number of important properties of this architecture that are relevant to the aims of Consensus:

- **Modularity**: Designing systems as collections of autonomous agents encourages a partitioning of the system into clearly defined system modules.
- **Information Hiding**: hiding the details and data of each KBA from the view of other KBA avoids uninten-

tional interference by one KBA with another.

- **Deadlock Avoidance:** insisting that the communication between KBA occurs indirectly, avoids direct interference between KBA, their data and control and is designed to reduce the risk of dead-lock. Although it is difficult to completely rule out deadlock, tests can be carried out to spot circular dependencies.
- **Traceability:** A direct mapping from general to specific requirements, and a direct mapping from specific requirements to system design components and code ensure traceability of requirements.
- **Concurrency:** Developing systems with an agent based architecture is an effective approach for building systems as sets of independent cooperating modules which are inherently concurrent and parallel.

Developing the Architecture Model

In accordance with the framework provided by the system specification, there is a close correlation between architecture and requirements, where processes in the requirement model will have corresponding entities in the Architecture Model and vice-versa. It is the purpose of the system Architecture Model to:

- identify the set of agents which form the system
- define the information flow between the agents in the system
- specify the channels on which the information flows

In the process of developing the system architecture the decomposition and specifications produced by the system requirements are used as the basis to develop the design of the system. The requirements which specify what the system is to do need to be transformed into a viable design. As KBA are more coarse grained than the primitive processes identified and specified in the Requirements Model, one needs to group components identified there into KBA which can operate relatively independently and in parallel. This process includes the identification of KBA, mapping requirements to individual KBA and specifying the interfaces.

As the Requirements Model is developed so the development of the Architecture Model follows one step behind to try identify a suitable system design which fulfills the requirements while taking into account external constraints. The second role of the Architecture Model is to force the develoment of the Requirements to take account of limitations in the specification process.

Starting from a global view of the system in a "first-cut" approach, the design needs to be refined and finalised to a point where ultimately it is very close to source code. The process of design of the system architecture takes the decomposition of DFDs from the Requirements Model and performs a transform analysis.

One has to determine properties of the DFDs which are important to a candidate system structure with a view at arriving at a suitable architecture that meets both functional requirements and additional constraints imposed by the implementation platform (where the platform includes both the target hardware and software). The following steps help drive a design:

1) **Review of the DFDs:** The review looks at the DFDs with the purpose of exploring the fundamental structure and characteristics of the system. It is important to familiarise oneself with the Requirements Model (especially if that is developed by another team) and to understand the structure of the requirements which may influence the structure of the design, and secondly, to consider the properties of the target platform in order to try to identify requirements or their decomposition which could create problems. If there are problems, the affected components need to be re-developed either at the present level or at the parent level to produce a new decomposition. The earlier problems are identified the easier it is to redevelop the affected components

2) **Analysis of Flow Characteristics:** In the next step the DFDs are examined to determine whether they display *trans-*

form or *transaction* characteristics by the following definition. If a DFD shows a chain of transformations where incoming information is gradually modified and transformed into an outgoing flow, the diagram is said to display a transform characteristic. Alternatively, transaction flows are characterised by processes which have only few inputs but a range of action paths to different processes.

3) **Preliminary Grouping:** If the DFD displays transform characteristics, the transformation centre needs to be isolated from the incoming and outgoing flows. Three main parts are thus identified: *inflow transforms*, the *transaction centre* and the *outflow transforms* are candidates for implementation as separate KBA, producing an incoming flow controller which co-ordinates the receipt of incoming data, a transform controller supervising the main transformation of data, and an outgoing controller co-ordinating the generation of the output information.

   Alternatively, if the DFD displays transaction characteristics, the transaction centre needs to be identified and is characterised by a number of action paths that flow radially from it. The reception path leading to the transaction needs to be isolated and so do the separate action paths leading out from the transaction centre. Each of the reception and despatch or action paths need to be re-examined to determine whether they have, themselves, transform or transaction characteristics. Each of these components identified are candidates for implementation as separate KBA.

4) **Review of Grouping:** The preliminary architecture is analysed, to consider whether the proposed KBA are sufficiently large and complex to be viable as separate KBA. The scope of the proposed KBA should be reviewed to reduce coupling and increase cohesion:

   Grouping should try to minimise the number of communication channels; processes which are highly connected should be in the same KBA. High fan-out from the outgoing flows of a KBA to a large number of other KBA should be avoided.

   KBA should have a clear purpose and function and candidates lacking in this respect may need to be redefined, or may point to problems in the Requirements Model.

   The candidate KBA identified in the preliminary grouping step need to be analysed concerning their size and complexity, to determine whether they are rich enough to be implemented as separate KBA or whether they need to be subdivided. This decision can be taken once the next level decomposition is known and the next iteration of the design process is carried out.

5) **Overall Review of Design:** The emphasis in the initial architecture specification stages is on analysis of the requirements from an implementation perspective, to ensure that the requirements definition does not contravene constraints imposed by the target platform. In later stages candidate KBA are identified, and finalised in an architecture diagram once the decomposition of the requirements is complete. During the review, one needs to decide whether previously identified KBA are viable on their own or whether they need to be further divided. In addition, the mapping of the processes from the DFDs supported by the requirements dictionary, needs to be specified in the architecture dictionnary. Architecture considerations may also influence the next level of decomposition of the requirements once external constraints become apparent.

Once the transform and transaction analysis has successfully been completed and the KBA have all been identified, the architecture diagrams can be finalised. At this point the additional documentation known as the *architecture dictionary* needs to be completed and contains the following:

- a definition of each architecture component depicted on the diagrams
- a definition of each architecture flow connecting KBA

- a definition of each architecture flow to external entities
- a narrative description of each KBA

The purpose of the narrative is to give a brief description of the purpose and functionality of each KBA. In addition a precise statement is required to associate all the components of the Requirements Model inherited by the KBA. This is an important step to allow verification as well as helping during maintenance. The local data structures need to be defined. The precise interfaces and communication mechanisms also need to be specified for the communication channels between KBA to allow the separate development of KBA by different teams. Finally, attention needs to be given to critical functions to ensure that they will not cause "bottle-necks" in the implemented system, thus limiting the overall performance of the system especially if there are real-time constraints.

4. Decision Support for ATC

The Consensus project has developed a test application which implements a decision support system, designed to help air traffic controllers in their task of safely controlling air traffic in their sector (based on [BEL88]). The system, known as the ATC Workstation, specialises in en- route air traffic control, i.e. air traffic control for sectors which do not contain aerodromes and where aircraft will pass through on their way to their final destination. The system also includes a simulator to simulate air traffic for testing and training purposes.

The controller workstation provides an integrated set of tools to support the en-route controller. These include the following:

- A **Predictor 2.1** which, given the current aircraft position and the current flight plan, predicts the courses of aircraft and warns of potential conflicts.
- The **Wotifer 2.2** which enables the controller to plan new routes for aircraft which need to be re-routed, while assessing the consequences of candidate new plans in the light of the current air traffic situation.
- A **Communicator 2.4** to provide electronic communication link between the controller and aircraft under his control.
- A **Monitor 2.5** to check the progress of aircraft, to warn the controller about aircraft which deviate from their flight plans or behave abnormally.
- The **Man Machine Interface 2.6** which handles the user interface relating information from the separate tools to the controller and directs the controller input to the appropriate tools.

The system operates an ATC workstation and, apart from the decision support functions, emulates a typical working environment.

Figure 5 shows the top level decomposition of the Requirements Model for the ATC workstation, followed by the further decomposition of the Wotifer in Figure 6.

The basic functions of the Wotifer is to recommend possible routes to the controller, to check the feasibility of candidate routes and to allow the controller to modify them at leisure and finally to accept one of them as the new flight plan for a particular aircraft.

Supposing a wotif is requested for an aircraft, the Wotifer goes through a number of steps. This involves the collection of all routes from present position to the required destination and pruning all plans the aircraft is not capable of following. Checking the remaining plans for compatibility with current air traffic situation, one has to select the best routes and to give them to the controller to select and/or modify. It may then be useful to store routes which are amended by the controller for future reference and obviously to file the route selected by the controller as the new flight plan for the aircraft.

The decomposition of the Wotifer was based on the sequence of steps required to carry

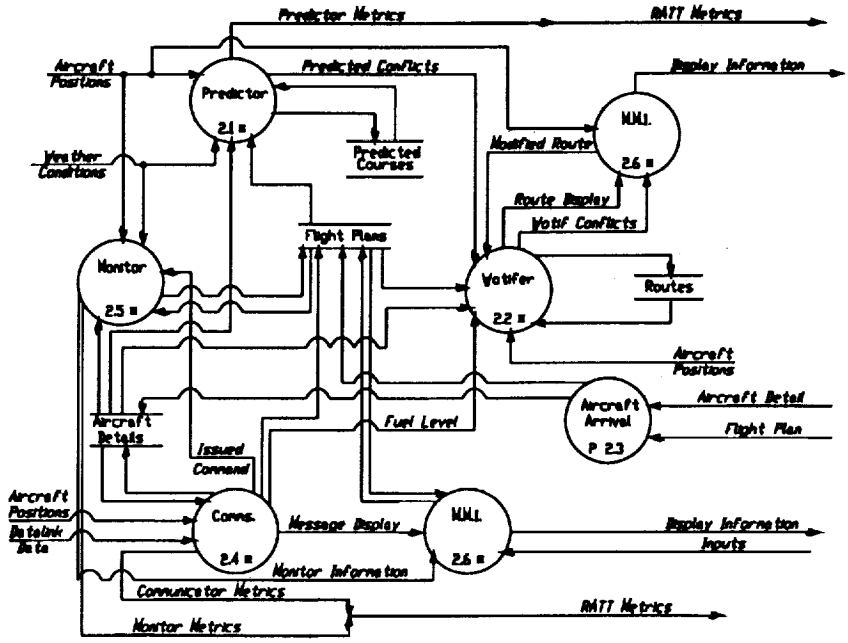


Figure 5: DFD Level 1 Decomposition of the ATC Workstation

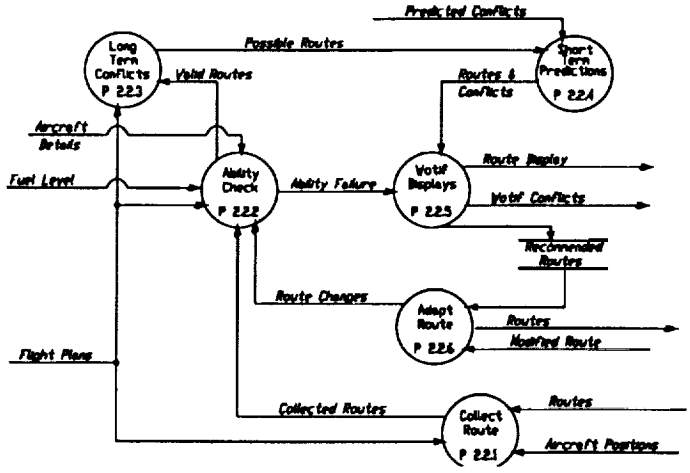


Figure 6: DFD Level 2 Decomposition for the Wotifer 2.2

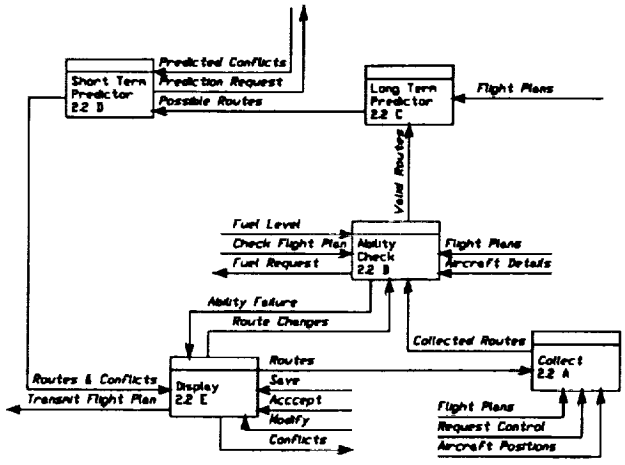


Figure 7: KBA Design for the Wotifer 2.2

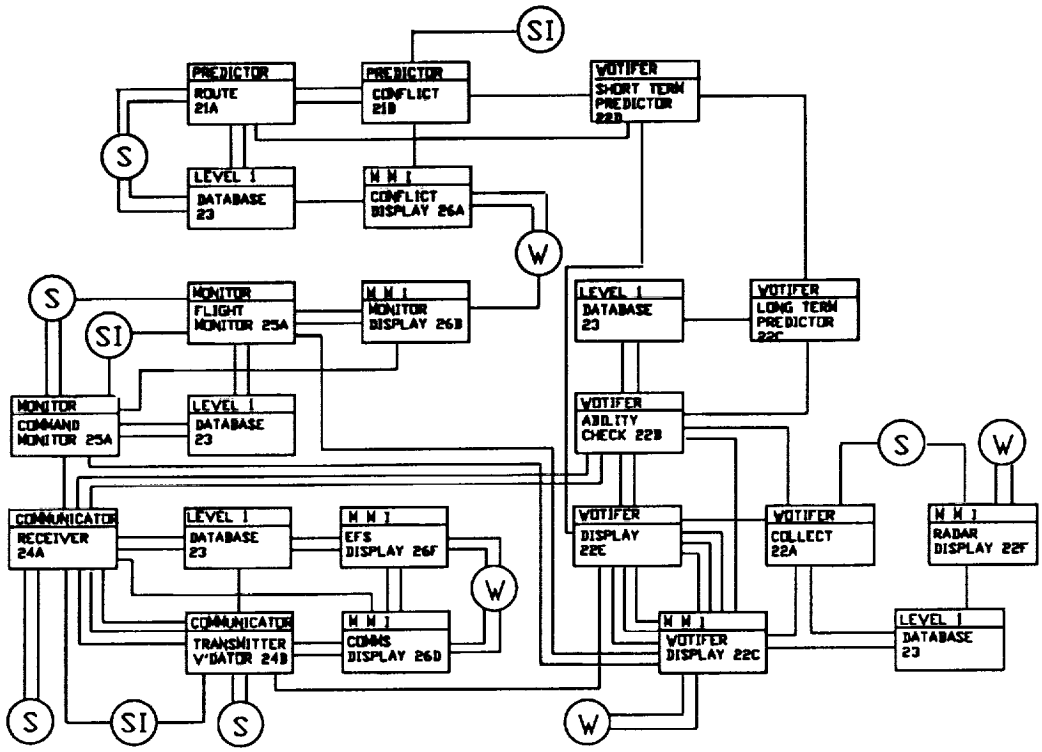


Figure 8: KBA Architecture of the ATC Workstation

out the task of planning and replanning of routes for given aircraft. The different processes depicted represent the major steps in that process.

This results in the following processes specified in the Level 2 decomposition shown in figure 6:

- **Collect Route 2.2.1** identifies all possible routes which would take a specified aircraft from its current position to the point where it wishes to leave the sector.
- **Ability Check 2.2.2** checks that the aircraft in question can perform the manoeuvres required.
- **Long Term Conflicts 2.2.3** identifies the more obvious potential conflicts that would be caused if the route in question was adopted as the aircraft's new flight plan.
- **Short Term Predictions 2.2.4**, in conjunction with the Predictor calculates the precise conflict points for the next 20 minutes.
- **Wotif Displays 2.2.5** calculates characteristics about a route to be presented to the controller and sends them to the MMI.
- **Adapt Route 2.2.6** deals with the response from the controller and deals with modifying, accepting and saving a route.

In the identification of KBA from the decomposition of the System Requirements, one aim is to exploit parallelism where ever possible. Thus, if a process can perform an action at the same time as another, this may be a reason for not joining them into a single KBA. In this case, the routes are processed sequentially, and it is possible for the Collect Route 2.2.1 process to collect a route and pass it onto the checking process Ability Check 2.2.2 which can then pass it onto the Long Term Conflicts 2.2.3 process which, in turn, can pass it on to the Short Term Prediction 2.2.4 process.

The processes can be pipelined to operate in parallel and in this case it meant that they were kept as separate KBA. The other two processes only process information when all routes have been processed. They work on the same information (Recommended Routes) but at different times, and because of the information sharing and asynchronous processing the processes were joined into a single KBA.

One problem that needed to be overcome was to determine what to do with the datastore of possible routes (Routes), which are used by the collecting process but updated by the modify process. One could have used a separate KBA, but the information is not used in many different places and does not change a great deal. It was the fact that the information does not change much which led to the decision to put the datastore together with the collect process. This led to five separate KBA which could work in parallel.

In Figure 7 the overall architecture of the ATC Workstation is depicted (note that S = Simulator, SI = Simulator Interface and W = Windowing System), thus combining the Wotifer Architecture with the other components of the system:

To conclude, the ATC System, including the simulator has a number of interesting statistics which demonstrate its size and complexity:

- 29 individual UNIX processes (including the Simulator and a C front end)
- networked on 3 to 13 workstations
- communicating via 174 socket connections
- 350 class definitions, 450 rules and 400 procedures
- about 3.5 man years of effort

This shows the ATC system to be a substantial application. The Consensus method gained considerably from this realistic test application by ensuring that the provisions capable of dealing with systems of this size and complexity, as well as giving valuable input for the generation of suitable development guidelines in the areas of the architecture of knowledge based components, hardware and software platform considerations, modularity and logical partitioning of the System Requirements and Architecture, considerations of concurrency, distribution and performance, as well as performance considerations and the handling of datastores.

5. Conclusion

The Consensus method fills a gap in the current field of software engineering, providing a comprehensive and integrated method for the development of large agent-based systems. As the Air Traffic Control application shows, agent-based systems can be successfully applied to systems which are relevant to current needs. The method covers the whole process from the statement of the user's need, to a detailed specification and through to implementation. The gradual progression from basic user requirements to a more refined specification with the help of a diagrammatic approach allows the customer to verify their requirements. The direct mapping from detailed requirements to system architecture and process specification also facilitates validation. There is a strong emphasis on sound software engineering principles in the system development process.

The application of Air Traffic Control is exemplary for a range of possible applications in the domain of aerospace and the experience gained by the project shows that Agent-Based Systems have considerable potential to provide solutions in this area.

There are a number of reasons for using distributed systems of this kind, one of which is the fact that it is difficult to conceive of using mainframe computere on board an air- or space-craft. Given high processing requirements and the need to keep weight to a minimum, the use of a number of networked processors would therefore apear to be much more appealing. Considering that systems on board, say, a shuttle have a number of distinct functions to fulfill, such as (amongst others) manoeuvering the craft, life-support systems, communication with ground-based systems and aiding in scien-

tific tasks, it seems sensible to divide a system to combine these functions into entirely separate sub-systems which could run on different processors and communicate with each other as and when necessary. This could be combined with a separate control system to supervise the proper functioning of these separate sub-systems, and which would take appropriate action when problems areise. Given the need to avoid malfunctioning at all cost, such a distributed system could also be made to be more resilient than a monolithic system as the malfunctioning of a sub-system would leave the other sub-systems still operative and enable the control system to take corrective action to re-establish full functionality. In addition, given stringent real-time requirements the lack of performance of one sub-system need not necessarily affect the performance of other sub-systems.

Bibliography

[ACK92] A Anjewierden, B Wielinga & N Shadbolt: Supporting Knowledge Acquisition: The ACKnowledge Project, ESPRIT-92 Knowledge Engineering, (Eds. L Steels & B Lepape)

[BOE88] Boehm B W, A Spiral Model of Software Development and Enhancement, IEEE, May 1988

[CRA89] Craig I D, The Cassandra Architecture, Distributed Control in a Blackboard System. Ellis Horwood, 1987, ISBN 0-7458-0579-5

COU91] Coulson I.C. et al., "Design Document D7", Consensus Report No. CNS-BAE-WIT-3.1-INT-IDC-056A.

[JOH91] Johnson K & Slade A, ATC Design Document, Wotifer 2.2, August 1991, Consensus Doc No. CNS-DUR-DCS-3.1-INT-KJ-052A

[JOH92] Johnson K. & Slade A, "Consensus Methodology Standards", Consensus Report No. CNS-DUR-CSC-6-INT-KJ-083B.

[HAT88] Hatley D & Pirbhai I, Strategies for Real-time System Specification, Dorset House, 1988

[KAD89] Taylor R M (Ed.), System Evolution - Principles and Methods, KBS Centre of Touche Ross Management Consultants, London, December 1989

[STA87] The STARTS Guide to methods and software tools for the construction of large real-time systems, NCC Publications, Hobbs Southampton, 1987

[FJE92] Fjellheim R, Pettersen T & Chriastoffersen B, REAKT Application Methodology Overview, ESPRIT-IIP5146 REAKT, Computas Expert Systems A.S, October 1992, CX Doc.# CXN92125

[ROY70] Royce W, Managing the Development of Large Software Systems: Concepts and Techniques, Proceedings of WestCon, August 1970, (Chapter 3)

SPACE/GROUND SYSTEMS AS COOPERATING AGENTS

T. J. Grant

BSO/Aerospace & Systems b.v.
P. O. Box 1444
3430 BK Nieuwegein
The Netherlands

Tel: +31-3402 88888
Fax: +31-3402 60577
Email: tigr@bsonwg.bso.nl

ABSTRACT

Within NASA and the European Space Agency (ESA) it is agreed that autonomy is an important goal for the design of future spacecraft, and that this requires on-board Artificial Intelligence. NASA emphasises deep space and planetary rover missions, while ESA considers on-board autonomy as an enabling technology for missions that must cope with imperfect communications. ESA's attention is on the space/ground system.

A major issue is the optimal distribution of intelligent functions within the space/ground system. This paper describes the Multi-Agent Architecture for Space/Ground Systems (MAASGS) which would enable this issue to be investigated. A MAASGS agent may model a complete spacecraft, a spacecraft subsystem or payload, a ground segment, a Spacecraft Control System, a human operator, or an environment. The MAASGS architecture has evolved through a series of prototypes. The paper recommends that the MAASGS architecture should be implemented in the operational Dutch Utilisation Centre.

INTRODUCTION

Within NASA and the European Space Agency (ESA) it is agreed that autonomy is an important goal for the design of future spacecraft, and that this requires on-board Artificial Intelligence. NASA's emphasis has been on deep space and planetary rover missions. ESA is considering greater on-board autonomy as a potential enabling technology for missions that must cope with communication delays or interruptions, as well as a way of reducing spacecraft operations costs. A series of ESA studies has resulted in the development of the Standard Generic Approach to Spacecraft Autonomy and Automation (SGASAA) concept.

Until recently, the emphasis has been on the space segment. ESA's attention is now turning to the complete system comprising both the space and ground segments: the *space/ground system*. A major issue is the optimal distribution of intelligent functions such that the space/ground system design results in a clearly quantifiable reduction in operational costs, without other adverse effects (e.g., on spacecraft reliability). Potential applications are foreseen in the ground-based Command and Control (C^2) of spacecraft which are subject to delays or interruptions in communication, e.g. deep space missions and missions partly visible from ground stations.

This paper describes the Multi-Agent Architecture for Space/Ground Systems (MAASGS) which enables the issue to be investigated. A MAASGS agent may model a complete spacecraft, a spacecraft subsystem or payload, a ground segment, a Spacecraft Control System, a human operator, or an environment. The architecture - developed for the Dutch Utilisation Centre (DUC) (Pronk, Koopman & de Hoop, 1992) - is based on Multi-Agent Systems (MAS) techniques. A MAASGS agent may model a complete spacecraft, a spacecraft subsystem or payload, a ground segment, a Spacecraft Control System, a human operator, or an environment. The MAASGS architecture has evolved under company and Dutch national investment through a series of prototypes. The paper concludes that the architecture is now mature, and recommends that it be implemented for use in the operational DUC.

There are five sections in this paper. Section 2 outlines the SGASAA concept. Section 3 motivates the use of MAS techniques. Section 4 describes the MAASGS architecture, including its evolution and associated development methodology. Finally, Section 5 draws conclusions and makes recommendations.

SGASAA CONCEPT

Defining Autonomy

Spacecraft autonomy can be loosely defined as the ability of a spacecraft to be largely or wholly independent of ground control (Pidgeon, Seaton, Howard and Peters, 1992). More precise definitions are mission-dependent. For scientific and communications satellites, the main drivers for autonomy are:

- Short and infrequent periods of ground station contact mean that there is little visibility of on-board events. Consequently, there is little opportunity for ground-based control to influence on-board events. The spacecraft must perform basic monitoring and control.

- Long transmission delays mean that the mission would not be practicable without some degree of autonomy.

- The need to maximise the mission product in the event of an internal or external event (e.g. on-board failure or change in its environment) means that the reaction time should be kept as short as possible. Autonomy reduces the need for the spacecraft to refer to the ground segment for a decision.

- Long duration missions where operations costs could be significantly reduced.

Autonomous Functionalities

A number of studies (Devita and Turner, 1984), (Doxiadis, 1988), (Drabble, 1991), (Elfving and Kirchhoff, 1991) have been conducted for various agencies to investigate approaches to spacecraft autonomy. ESA's studies, begun in the early 1980s, culminated in the SGASAA concept (Berger, Cornet, Cellier, Riou, Sotta and Thibaut, 1984). By the beginning of the 1990s, ESA had progressed to validating the SGASAA concept, in the Spacecraft Autonomy Concept Validation (SACV) study (Pidgeon, Seaton, Howard and Peters, 1992).

For scientific satellites, autonomy is viewed as replacing (or supplementing) ground-based operator functions with on-board functions. The SACV study listed the foreseen on-board functionality as:

- Execution, updating and rescheduling of a *Master Schedule*, which is a set of high-level, time-tagged, goal-oriented commands stored on-board. Rescheduling would take into account the commands' resource requirements, the availability of on-board resources, the dependencies between commands, and environmental and timing constraints.

- Fault diagnosis would be performed on-board. The autonomous spacecraft would attempt to recover from a failure, while ensuring the spacecraft's safety and minimising the loss of the mission product. Fault diagnosis could only cater for foreseen failure modes. Unforeseen failures would have to result in the spacecraft adopting a safe mode to await ground intervention.

For reasons which are unclear, the SACV study omitted a third possible on-board functionality: goal-oriented planning. Goal-oriented planning was always seen as having an equal priority with other functionalities (e.g. see (Berger, Cornet, Cellier, Riou, Sotta and Thibaut, 1984), Volume 1, Figure 5.2/4). Therefore, this paper assumes that the on-board functionality must include:

- Goal-oriented planning (and re-planning), which must:
  - Take the (re-)planning activity into account.
  - Be interruptible.
  - Be able to generate alternative plans for the same requirements.

The Concept

A common thread amongst the spacecraft autonomy studies has been the adoption of a hierarchically-based On-Board Management Systems (OBMS). The OBMS

consists of a high-level On-Board Mission Manager (OBMM) together with various subordinate subsystem and payload managers (generically termed Sub-System Managers (SSMs)). The OBMM monitors, coordinates and controls the SSMs, and each SSM monitors, coordinates and controls a subsystem or a payload. The OBMS is supported by a distributed on-board communications architecture with the managers communicating via a LAN or databus, and each subsystem and payload being connected to its SSM via a subassembly LAN. The SSM effectively acts as a bridge between the subassembly LAN and the spacecraft LAN. From outside the subsystem or payload, the SSM appears to 'wrap' the subsystem or payload with additional functionalities.

The SGASAA concept adopted a distributed hierarchy on the grounds that decision-making should be devolved to the lowest possible level in the hierarchy. Figure 1 depicts the conceptual SGASAA architecture. The spacecraft consists of a set of "intelligent" subsystems and payloads, each of which has the capability to interpret Telecommand (TC) packets and to generate Telemetry (TM) packets. Packetised TM/TC is a prerequisite for the SGASAA approach.

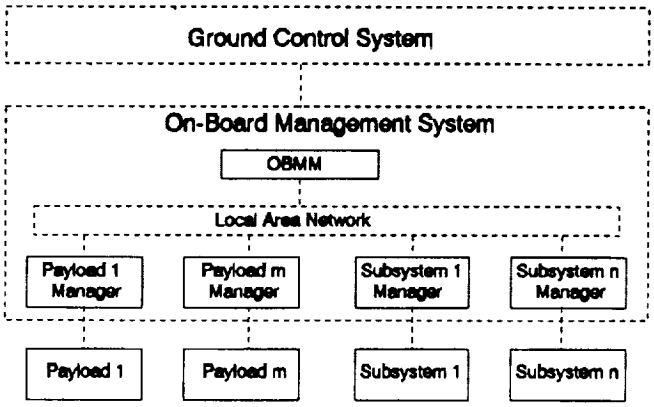


Figure 1: SGASAA Architecture.

The SGASAA approach relies on the concept of a series of layered mission plans. Low-level plans are generated from the plan above by adding detail. For example, the *Long Term Operations Plan* (LTOP), which defines the objectives for an entire mission or mission phase, can be broken down into a series of components (*Links*) which are uplinked to the spacecraft and executed on-board. Links describe actions which can be achieved at system level by a combination of activities at subsystem level. The *Short Term Operations Plan* (STOP), covering a period of several days, consists of a set of links with coarse parameters. These are rescheduled with precise parameters, reflecting the current on-board state, and broken down into a sequence of blocks. The *Executable Operations Plan* (EOP) contains blocks of actions for a single subsystem or payload, usually in the form of macro-commands which are expanded on-board. The *Elementary Commands* (EC) are time-tagged commands contained within an EOP, each normally affecting only a single element of the subsystem, e.g. switching a heater on.

Plans are validated and optimised at each level, refining the plan from a coarse LTOP to detailed ECs. The SGASAA concept also allows for direct commanding of subsystems, bypassing the OBMM. A limitation of the SGASAA plan hierarchy is that higher-level plans must contain information about the lower-level plans, such as resource usage, duration, dependencies, etc. This means that planning is necessarily an iterative process, with lower-level plans providing feedback to higher-level plans.

The OBMM and the Subsystem and Payload Managers have prescribed roles (see (Pidgeon, Seaton, Howard and Peters, 1992), section 2.3). Comparison of these roles shows that each SSM has the same functionality as the OBMM, albeit for a more detailed subset of the spacecraft (i.e. the subsystem or payload for which the SSM is responsible). The common functionalities are:

- Distribution and execution of TCs.
- Generation of TMs.
- Fault diagnosis.
- Failure recovery.
- Localised planning.
- Self-checking.

The SGASAA concept defines three modes of operations:

- Routine Mode, in which nominal and expected tasks are executed.

- Crisis mode, which is the handling of unexpected events on-board the spacecraft or due to external influences.

Check-out Mode, in which the spacecraft is placed in a configuration which allows hardware and/or software to be tested.

MOTIVATION FOR MULTI-AGENT SYSTEMS

Multi-Agent Systems

Distributed Artificial Intelligence is defined as "the subfield of AI concerned with concurrency in AI computations" (Bond and Gasser, 1988, p.3). Bond and Gasser divide the world of DAI into three arenas: Distributed Problem Solving, Multi-Agent Systems, and Parallel AI. In this paper, we are concerned with Multi-Agent Systems (MASs), i.e. "with coordinating intelligent behaviour among a collection of (possibly pre-existing) autonomous intelligent 'agents', which can coordinate their knowledge, goals, skills, and plans jointly to take action or to solve problems" (*ibid.*, p. 3). Typical intelligent behaviours are to generate plans and schedules, to react appropriately to situations (including diagnosing and recovering from failure), and to learn. The agents may be working towards a single, global goal, or towards separate, individual goals that can conflict. Crucially, "they must ... reason about the *processes of coordination among the agents*" (Bond and Gasser, 1988, p.3, italics in original). The task of coordination can be difficult, because there may be situations where there is no global control, no globally consistent knowledge, no globally shared goals, and/or no global success criteria. Reviews of MAS techniques and trends may be found in (Castillo-Hern and Wilk, 1988), (Grant, 1992), and (Chaib-Draa, Moulin, Mandiau and Millot, 1992).

There is no consensus definition of an agent. Bond and Gasser skirt around the issue; they rely on a simple and intuitive notion of an agent as a computation process with a single locus of control and/or "intention" (*ibid.*, p.3, footnote 1). For the purposes of this paper, an *agent* will be defined as a software entity with autonomous processing capabilities and a private database, which acts on its environment on the basis of information it receives from the environment.

Motivations for Using MAS Techniques

Huhns (1987) lists five primary reasons why one would want to use MAS techniques:

- MASs can provide insights and understanding about interactions among humans, who organise themselves into various groups, committees, and societies to solve problems.
- MAS techniques can provide the means for interconnecting multiple expert systems that have different, but possibly overlapping, areas of expertise. This permits the solution of problems whose domains lies outside the area of expertise of any one expert system.
- MASs can potentially solve problems that are too large for a centralised system because of resource limitations (eg bandwidths, computing speeds, and reliability) induced by technology.
- MASs can potentially provide a solution to a current limitation of knowledge engineering: the use of only one expert. If there are several experts or several non-experts whose ability can be combined to give expert-level behaviour, there is no established way to use them successfully.
- MAS techniques are the most appropriate solution when the problem itself is inherently distributed, as in distributed sensor networks and distributed information retrieval.

Clearly, the last reason is the prime motivation for applying MAS techniques to space/ground systems.

Huhns (1987) also listed the following advantages for system development:

- Partitioning the software system into agents reduces the complexity, resulting in a system that is easier to develop, test, and maintain.
- The software subsystems (i.e., agents) can operate in parallel.
- The software system can be designed - using the *functionally accurate* approach (Lesser and Corkill, 1981) - to continue to operate even if part of it fails.
- It is easier to find experts in narrow domains.

MAS Issues

The following issues are relevant to this paper:

- Structuring the functionalities internal to an agent. A wealth of differing agent structures exists in the MAS and C^3I literatures. The MAASGS architecture has evolved an agent structure from object-oriented systems by adding concepts from C^3I theory and then by specialising this for spacecraft operational control.

- Representing the agent's knowledge of its environment. Investigation of possible ways of representing the agent's knowledge of its environment is a major sub-field of MAS research. Representations vary from the agent-attribute-value model to logics of belief which are modelled on human psychology. We use the simple agent-attribute-value model, with the attributes being typed according to the functionality which operates on them. For example, the attributes might be rules, Horn clauses, planning operators, or constraints, as well as datatypes such as booleans, integers, reals, strings, etc.

- Enabling agents to communicate with one another. Chaib-Draa, Moulin, Mandiau and Millot (1992) identifies solutions to inter-agent communications ranging from *no communication*, through *primitive communication, plan and information passing*, information exchange via a *blackboard*, *message-passing*, to *high-level communication*. The MAASGS architecture adopts the message-passing model because this models closely the packetised TM/TC used in modern spacecraft, and can be readily implemented using the message-passing model employed in object-oriented programming languages such as Smalltalk, C++, CLOS and Eiffel.

  Enabling agents to coordinate their actions. Agents must coordinate their distributed resources, which may be physical or computational. The most appropriate coordination technique depends on the distribution of the shared resources and on the local autonomy of agents, which may have disparate goals, knowledge and reasoning processes. Generally, DAI researchers use the *negotiation* process to coordinate a group of agents. There are various definitions for negotiation. We adopt Bussmann and Müller's (1993) definition of negotiation as "the communication process of a group of agents in order to reach a mutually accepted agreement on some matter". A variant is *arbitration*, in which the group of agents appeal to an impartial agent to reach the agreement. In DAI, negotiation is often implemented as the Contract Net Protocol (Davis and Smith, 1983), in which an agent needing help decomposes the problem into subproblems, announces the opportunity to solve the subproblems to the group, collects bids for their solution from group members, and awards the subproblems to the most suitable bidders. The MAASGS architecture can accommodate a range of coordination protocols, including an arbitration protocol which supports inter-agent learning (Grant and Lenting, 1993).

- Modelling domains by means of agents. Borrowing from object-oriented simulation, we follow the fundamental principle of modelling each real-world object - whether or not it has any intelligent functionality - as an agent. There are two ways to model domains in this way: agents may be *specialists* or they may be *generalists*. Specialist agents have functionality that is specific to the role of the real-world object being modelled, e.g., transforming X-rays into data, calculating spacecraft orbits, and so on. By contrast, generalist agents all have the same generic functionalities, e.g., rule-based inference, goal-oriented planning, constraint-based scheduling, and so on. The MAASGS architecture employs generalist agents. Examples of the generic functionalities in the MAASGS architecture are receipt of TCs, generation of TMs, monitoring other agents' status, diagnosis, selection of procedures, goal-oriented planning, scheduling, etc. There is a small set of agent-classes, derived from Grant's (1992a) abstraction hierarchy. The agent-class which models the non-intelligent domain objects, such as payload components,

implements only receipt of TCs, generation of TMs, and internal computation. Another agent-class models intelligent domain objects, such as SSMs.

Organising agents to represent distributed systems. Elaborate schemes have been devised to represent organisations of agents. We have adopted the simple idea that an agent can be decomposed into more primitive agents. Despite its anthropomorphic title in the DAI literature - where it is known as Minsky's (1986) "Society of Minds" concept - the idea of decomposition is to be found in any industrial-strength software analysis or design method, e.g. SADT and dataflow diagramming. Domain decomposition hierarchies are usually easy to find. For example, the very first sentence in an ESA Bulletin article on the use of spacecraft simulators at ESOC (Gujer and Jabs, 1991) states (p. 41):

"A satellite mission can be considered in its simplest form to consist of a space segment, a ground segment and a user community ... The ground segment for an ESA mission ... includes: a set of ground stations, ... a communications network, ... the Operations Control Centre (OCC), ... payload data-processing facilities ..."

The same article later states (p. 46):

"Figure 6 shows the layout of a typical spacecraft model as implemented in most simulators. It closely reflects the standard decomposition of a spacecraft into subsystems."

In the MAASGS architecture, decomposition hierarchies are modelled by enabling any agent to have zero or one *superior* agents and zero or more *subordinate* agents. The superior represents the assembly of which the agent is a part, and the subordinates represent the component parts of the agent. This approach implies that each node in the decomposition hierarchy is modelled as an agent, and not just the leaf-nodes.

THE MAASGS ARCHITECTURE

Evolution

The MAASGS architecture has evolved by specialising the generic agent structure. The first step was to incorporate classic C^3I features, based on Wohl's Stimulus-Hypothesis-Option-Response (SHOR) model of decision-making (Wohl, 1981). This resulted in the Message-Based Architecture testbed (Grant, 1991), developed as a private venture. The testbed was designed primarily as a "test harness" for an inductive learning algorithm. The reactive and generative planning functionalities were deliberately designed to be the minimum necessary to close the loop from the inductive learning algorithm's output back to its input. The testbed successfully demonstrated *learning-by-doing* (Anzai and Simon, 1979).

The second step was a paper study of an agent structure suited to the Columbus User Support Organisation (USO), based on the lessons learned in developing and using the Message-Based Architecture testbed. This study was a part of BSO/Aerospace & Systems' "DUC Preparation" (DUCPREP) project. The DUCPREP project was funded by company and Dutch national investment and performed in informal cooperation with a number of other Dutch companies. The agent structure proposed for the DUC was documented in (Grant, 1992a).

In the third step, the internal functionalities of a Message-Based Architecture agent were extracted and enhanced. The resulting DUC Activity Scheduling System (DUC-ASS) is a single-agent software system capable of integrating the support of payload design, planning, scheduling, and control (Grant, 1992b). Prototyping of the DUC-ASS was performed under BSO/Aerospace & Systems' "MILDS" project, also funded by company and national investment. The MILDS project formed an element of the larger "DUC-Pilot" project performed by a Dutch consortium. Under the DUC-Pilot project, an interface was defined (Grant and Tusveld, 1992) for coupling the DUC-ASS to a diagnostic system which used model-based reasoning techniques. Current DUC-related developments (Pronk, Visser and Sijmonsma, 1993) centre on linking the pilot DUC to ESTEC's Crew Work Station testbed for Mission Simulation purposes.

The MAASGS architecture uses the DUC-ASS agent structure, enhanced to incorporate the scheduling and model-based diagnosis functionalities. Although the MAASGS architecture has not yet been implemented in full, the key functionalities have all been implemented. The interfaces between them have been defined to varying levels of detail. Two related issues have been addressed by exploratory prototyping: recognising objects during learning (Grant, van Meenen and Stroobach, 1992), and the modeller's graphical user interface (Grant, 1993a). In addition, the Message-Based Architecture testbed has been recently enhanced to enable agents to exchange learned knowledge, i.e. they can also learn-by-being-told (Grant, 1993b).

Agent Structure

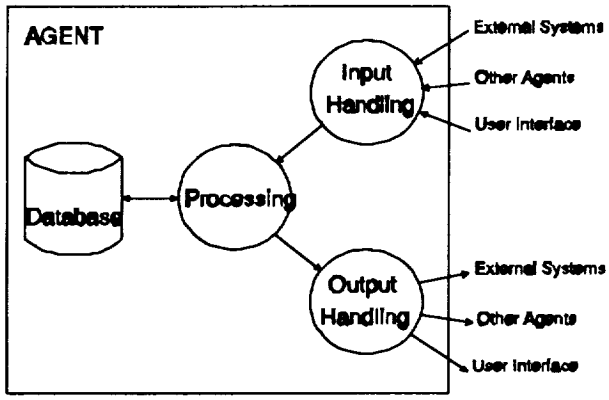


Figure 2: Generic Structure of an Agent.

There are four components in a generic agent (see Figure 2). *Input Handling* receives and filters the incoming messages from other agents, from a user interface, and from external systems. *Processing* operates on the filtered incoming messages, retrieving and storing information in the agent's private *Database*, and generating messages to send. *Output Handling* formats and despatches the outgoing messages to other agents, to the user interface, and to external systems.

Such an agent is a specialisation of an object in object-oriented systems. An agent has a unique name, autonomous processing capabilities (cf. methods), and a private database (cf. attributes and their values), and exchanges information with its environment (cf. message-passing). Some MASs also have agent classes and inheritance. MASs have functionality that extends beyond that of object-oriented systems. In particular, agents in MASs are *intelligent agents*. Typical intelligent behaviours are to react appropriately to situations, to generate plans, and to learn. These behaviours can be best modelled using AI techniques, such as expert systems, knowledge-based planning (Georgeff, 1987), and machine learning (Michalski, Carbonell and Mitchell, 1983). We distinguish agents from objects by requiring that an agent minimally includes the abilities to:

- Model its own state and behaviour, and
- Decide whether or not to accept a new state its environment attempts to impose on it.

Such an agent is termed a *non-intentional* agent (Grant, 1992). *Intentionality* means to have attitudes towards other agents (Searle, 1980), such as intentions, goals, desires, or beliefs. Any agent which generates instructions, forms plans, or learns about other agents is necessarily intentional.

An abstraction hierarchy of agents may be built on the minimal set of abilities. Grant (1992) proposes an abstraction hierarchy in which an *intentional* agent also has the abilities to:

- Model its own goals,
- Model other (non-intentional) agents, and
- Manage a negotiation or arbitration process between other agents.

At the very least, intentional agents are aware of the existence of other agents in their environment. Following the precedent set by the MACE testbed (Gasser, Braganza and Herman, 1987), the other agents are usually known as the agent's *acquaintances*. In many MASs, agents also know about their acquaintances' behaviours, i.e., their *capabilities*.

Agent-Based Simulation

Application domains may be modelled as collections of agents. As in object-oriented simulation, a set of entities must be provided which the modeller can instantiate to represent the domain. There are two fundamental set-elements in object-oriented simulations: *object-classes* (cf. Smalltalk's Class object-class) and *messages* (cf. Smalltalk's Message object-class). In agent-based simulation, the equivalent entities are *agent-classes* and *messages*. The distinction between

objects and agents implies that additional entities are needed to represent the agent structure. Precisely what additional entities are provided depends on the simulation development environment designer. Our experience shows that a suitable set of additional entities should include *agent structuring*, *inter-agent message-handling*, *inter-system interfacing*, *user interface*, and *agent organisation* entities. More details are in (Grant, 1993a).

In addition, there must be a domain-independent Simulation Development Environment (SDE), comprising a simulation executive, a set of tools, a user interface, and, optionally, interfaces to external systems. A database management system may also be provided where the agents in the simulation model are not persistent. The SDE may itself be implemented as a second collection of agents. Issues concerning the SDE are outside the scope of this paper. The wider issues concerning how AI and simulation techniques may complement one another are covered by Widman, Loparo and Nielsen (1989).

Adding C³I Features

The SHOR model describes a data-driven or reactive approach to problem-solving and decision-making. The model identifies four information-handling processes. *Stimulus* involves the processing of raw data received from the decision-maker's environment via sensors. Processing includes searching for data, scanning or sampling it, reducing, compressing, and aggregating the scanned/sampled data, and detecting, recognising and confirming events signalled by the data. The decision-maker interacts with the environment, eg by directing sensors. *Hypothesis* involves the generation and evaluation of hypotheses concerning the environment's state-of-affairs, based on the outputs of Stimulus. Processing includes data association and correlation, state and parameter estimation, hypothesis generation, situation assessment, and decision state estimation. The decision-maker is essentially passive to the external environment while he/she focuses on the analysis task. *Option* concerns the generation, planning and evaluation of alternative options for the decision-maker's response to the estimated decision state. *Response* concerns the execution of the selected response. Execution involves the issue of information to the decision-maker's environment, either by physical action or by communicative action.

Various authors view Wohl's Stimulus and Hypothesis in terms of the data processing techniques employed. Event detection, recognition, and confirmation through to decision state estimation are grouped together as *data fusion*, defined (Waltz & Llinas, 1990, p. 1) as:

> "A multi-level, multi-faceted process dealing with the detection, association, correlation, estimation and combination of data and information from multiple sources to achieve refined state and identity estimation, and complete assessments of situation ...".

Thus, data fusion maps onto the latter part of Stimulus, combined with Hypothesis. In Wohl's Hypothesis process, the decision-maker associates the events recognised during Stimulus processing with objects in the environment. The states and other parameters of these objects can then be estimated.

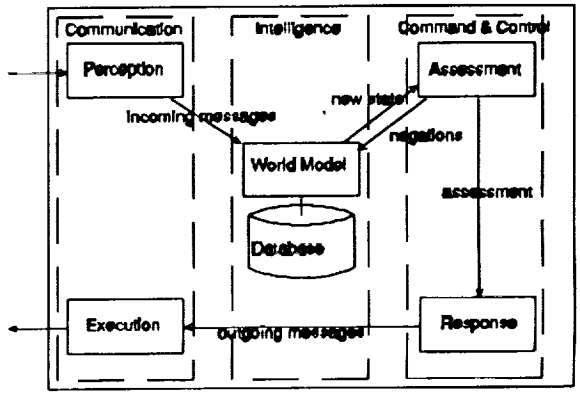


Figure 3: Agent Structure Enhanced with C³I Features.

Enhancing the generic agent structure results in the structure shown in Figure 3. Input Handling (cf. Stimulus) becomes the Perception module, and output processing becomes Execution. Processing is now divided into the Assessment and Response modules (cf. Hypothesis and Option), and World Model encompasses Database. Figure 3 shows the modules grouped by the military terms "Communication", "Intelligence", and "Command and Control".

The C³I-enhanced structure works according to the "do-as-little-work-as-possible" principle. For example, if the Perception module filters out an incoming message as having nothing to do with the agent, processing stops

at that point. Similarly, the Assessment module, which uses constraint-based techniques to check the consistency of the World Model after the incoming information has been added to it, can decide to terminate processing if the incoming information is consistent with what the agent already knows. Only if an inconsistency or conflict is found does the Assessment module trigger the Response module.

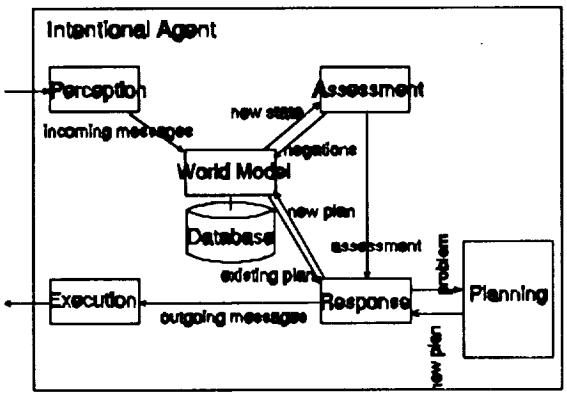


Figure 4: Structure of Intentional Agent.

The agent structure shown in Figure 3 has the capability of a non-intentional agent. It can react to events, but it cannot plan ahead. In AI terms, it is limited to forward-chaining (or data-driven) reasoning. The danger is that such an agent can get into a situation from where it is impossible to reach its goals: a "cul-de-sac" world-state. To obtain an intentional agent, the agent structure must be further enhanced with planning functionality (see Figure 4). The Planning module can be seen as providing a planning service to the Response module. Other intentional functionalities, such as scheduling and learning, can be added as further services to the Response module.

MAASGS Architecture

The MAASGS agent structure, shown in Figure 5, has clear correspondences to the structure of an intentional agent in the Message-Based Architecture test-bed. The Monitoring module, together with the Status Database, is equivalent to the Message-Based Architecture's Perception and World Model modules. The Detection module performs situation assessment. Response selection is performed by the Isolation, Diagnosis, and Recovery modules. The Real-Time Replanning and Scheduling module replaces the Planning Module. The Execution and Predictive Payload Simulator (PLS) modules perform the functions of the Message-Based Architecture's Execution module.

The Detection, Isolation, Diagnosis and Recovery modules are grouped together as the ADIR assembly, where the "A" stands for "Anomaly". Normally, such a grouping of functionalities would be termed "FDIR", where the "F" stands for "Fault". In the MAASGS architecture, the functionalities are generalised to encompass anomalous situations. An *anomaly* exists whenever the telemetry indicates that a parameter has a value which either falls outside its alarm or warning levels or is unplanned or unexpected. Unplanned values may still be beneficial, i.e., serendipitous. Therefore, this functional group must not be regarded as FDIR, until the presence of a fault has been confirmed.

Limitations in the available funding have meant that the MAASGS concept has not yet been implemented fully. A number of prototypes of MAASGS modules exist. For example, the ADIR group of modules has been developed fully. Prototype PLSs exist, but have not been tailored for prediction and anomaly detection. The DUC-ASS application can be seen as a prototype non-real-time Replanning and Scheduling module. User interface issues have been partly explored in the agent-based Application Data Source Simulation Tool (ADSST) (Grant, 1993a). Between them, these prototypes have covered the MAASGS functionalities. It now remains to integrate them, ideally using the emerging international standards for knowledge representation (Grant and Poulter, 1993).

Modelling Methodology

The MAASGS modelling methodology starts with domain decomposition. This can be done top-down, bottom-up, or middle-out. The second step is to define the internal database and processing functionality of each agent. Inter-agent connectivity is defined in the third step in terms of links and switches. The agents are initialised in the fourth step. In the fifth step, simulation scenarios are defined. In several of the MAASGS prototypes, scenarios can be defined by direct manipulation of the agents, with the system capturing the user's manipulations and compiling them as a scenario. A chosen simulation scenario is run in the sixth step, and the results are evaluated in step seven.

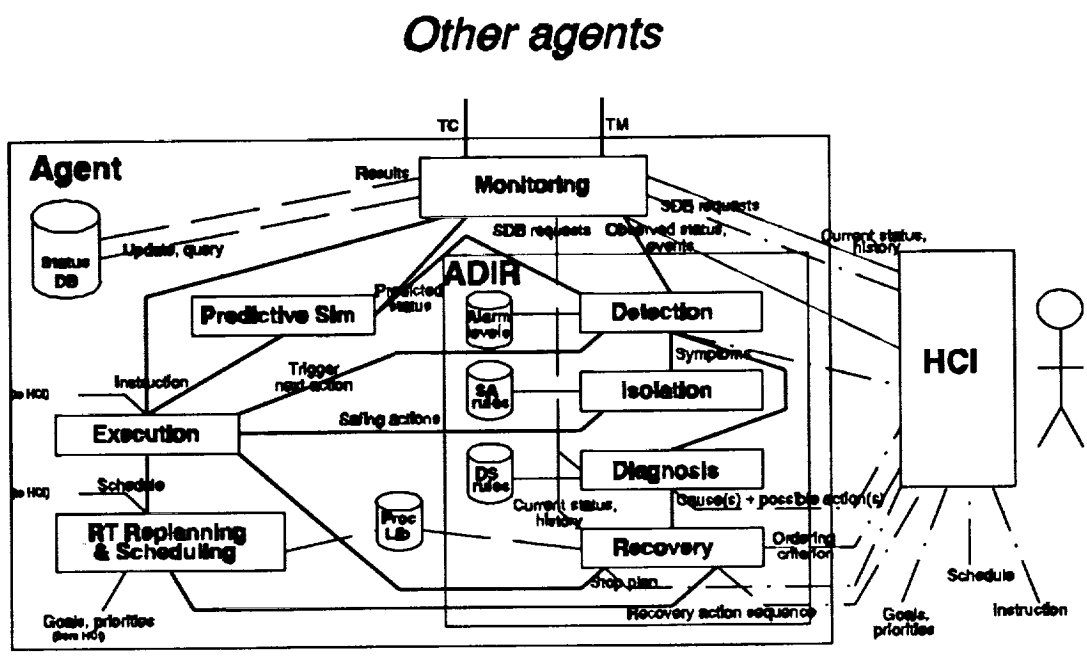


Figure 5: MAASGS Agent Structure.

Further Work

Conceptually, the MAASGS architecture is mature. It has evolved through iterative enhancement, with each step being tested by implemented systems. All the MAASGS components exist in developed or prototype form. The MAASGS architecture should now be implemented in full.

The first step would be to define all the inter-module interfaces, preferably using a knowledge communication standard. In parallel, the user interface prototyping begun in the ADSST should be extended. The MAASGS implementation would be built up step-by-step, starting with a payload simulator to represent the agent's environment. A suitable sequence for adding the modules would be: HCI, Monitoring, Status Database, Anomaly Detection, Isolation, Execution, Predictive PLS, Diagnosis, Recovery, Real-Time Replanning, and finally Scheduling.

Having built up to the full agent structure, a second phase could begin. From a single agent communicating with a (simulated) payload, a second agent could be introduced. This would be best done by inserting the second agent between the simulated payload and the existing agent, to model the situation in which a User Home Base (i.e., the first agent) is subordinate to a Payload Operations Control Centre (i.e., the second agent). This situation would model the Dutch Utilisation Centre (DUC), which combines payload control at the Dutch national level (i.e., a User Support Operations Centre, in Columbus/Space Station Freedom terminology) with a User Home Base. Additional agents could then be introduced to model further User Home Bases.

Having achieved a hierarchical agent model of the ground segment, attention could then turn to the space segment model. The simulated payload would be replaced by a further agent. The remainder of the space segment would then be introduced as additional agents. A final refinement would be to represent the communications chain linking the space and ground segments also as agents. Any part of the complete agent-based space/ground system model could be modelled to a higher fidelity at any time by

decomposing the part concerned into subordinate agents.

Such an agent-based model of the complete space/ground system could be used in an Operations Research (or Management Science) role. Functionalities of selected agents could be switched on or off to investigate the optimal distribution of functionality. New functionalities, such as agent learning, could be assessed by enhancing the generic agent structure. Different space/ground system architectures could be investigated by altering the connectivity between selected agents. For example, "deputy" agents could be introduced for key roles, such as the OBMM or the Payload Operations Control Centre. Instead of an OBMS hierarchy, a heterarchy of SSMs could be evaluated. Another application would be to model spacecraft constellations by instantiating multiple space segments.

An agent-based model of the space/ground system could also be used for operational purposes. It could be used to evaluate the introduction of additional payloads and Principle Investigators. It could be used to evaluate mission timelines. Finally, by replacing one or more software agents with real payloads, subsystems, control systems, and people, it could be used for verification and training purposes.

CONCLUSIONS AND RECOMMENDATIONS

This paper has described ESA's Standard Generic Approach to Spacecraft Autonomy and Automation (SGASAA). The on-board functionalities have been outlined. The SGASAA architecture has been depicted. Multi-Agent Systems (MAS) have been defined, the motivations for using MAS techniques have been listed, and relevant MAS issues have been discussed. The paper has shown that MAS techniques are the most appropriate solution to modelling space/ground systems, because such systems are inherently distributed.

BSO/Aerospace & Systems' Multi-Agent Architecture for Space/Ground Systems (MAASGS) has been documented. Its evolution has been sketched. The MAASGS agent structure has been detailed. The addition of agent-based simulation and C^3I features has been described. The methodology for using the MAASGS architecture has been outlined. Further work has been identified. The paper concludes that the MAASGS architecture is conceptually mature, and recommends that the architecture should now be implemented in full.

REFERENCES

Anzai, Y., and H. A. Simon. (1979). The Theory of Learning by Doing, Psychological Review, 86, 2, 124-140.

Berger, G., J. Cornet, M. Cellier, L. Riou, J. Sotta, and M. Thibaut. (1984). Standard Generic Approach for Spacecraft Intelligence and Automation: Final Report, ESA Contract number 4869/81/NL/PP, ESA Report CR(P) 1759 Volumes 1 and 2.

Bond, A. H., and L. Gasser, (eds). (1988). Readings in Distributed Artificial Intelligence, Morgan Kaufman, San Mateo, CA, USA.

Bussmann, S., and J. Müller. (1993). A Negotiation Framework for Cooperating Agents, In Deen, S. M. (ed). (1993). 1992 Proceedings of Special Interest Group on Cooperating Knowledge-Based Systems (CKBS-SIG'92), DAKE Centre, University of Keele, UK, 1-17.

Castillo-Hern, L. E., and P. F. Wilk. (1988). Describing DAI Models: A Framework and Examples, Proceedings, Alvey workshop on Multiple Agent Systems, Philips Research Laboratories, Redhill, UK, 14/15 Apr 88.

Chaib-Draa, B., B. Moulin, R. Mandiau, and P. Millot. (1992). Trends in Distributed Artificial Intelligence, Artificial Intelligence Review, 6, 1, 35-66.

Davis, R., and R. G. Smith. (1983). Negotiation as a Metaphor for Distributed Problem Solving, Artificial Intelligence Journal, 20, 63-109.

Devita, E. L., and P. R. Turner. (1984). Autonomous Spacecraft Design Methodology, NASA CR 849323, Jet Propulsion Laboratory, Pasadena, California, USA.

Doxiadis, A. S. (1988). Expert Systems for Spacecraft Autonomy, Ford Aerospace, International Federation of Automatic Control Workshop in Spacecraft Autonomy.

Drabble, B. (1991). Spacecraft Command and Control using Artificial Intelligence, JBIS, 44, 6, 251-254.

Elfving, A., and U. Kirchhoff. (1991). Design Methodology for Space Automation and Robotics Systems, ESA Journal, 15, 149.

Gasser, L., C. Braganza, and N. Herman. (1987). MACE: A Flexible Testbed for Distributed AI Research, chapter 4 in (ed) Huhns, M. (1987). Distributed Artificial Intelligence, Research Notes in Artificial Intelligence, Pitman, London, UK, 119-152.

Georgeff, M. P. (1987). Planning, American Reviews in Computer

Science, 2, 359-400. Also published in (eds) Allen, J., J. Hendler, and A. Tate. (1990). Readings in Planning, Morgan Kaufman, San Mateo, CA, USA, 5-25.

Grant, T. J. (1991). Integrating Reactive Planning, Plan Generation, and Planning Operator Induction in the Message-Based Architecture, Proceedings, 10th UK Planning SIG workshop, Logica Cambridge Ltd, Cambridge, UK, 25/26 April 1991.

Grant, T. J. (1992). A Review of Multi-Agent Techniques, with application to Columbus User Support Organisation, Proceedings, AI and KBS for Space Workshop, ESTEC, 22-24 May 1991, ESA Document WPP-025, Volume 1. Also published in Special Issue on AI in Space, Future Generation Computer Systems, 7, 413-437.

Grant, T. J. (1992). Integrating Payload Design, Planning and Control in the Dutch Utilisation Centre, Proceedings, 2nd International Symposium on Ground Data Systems for Space Mission Operations (SpaceOps'92), Pasadena, California, USA, 16-20 November 1992, JPL Publication 93-5, 237-242.

Grant, T. J. (1993). Beyond Objects: An Agent-Based Simulation Tool, Proceedings, 1993 European Simulation Symposium (ESS'93), Delft, The Netherlands, 25-28 October 1993, 665-670.

Grant, T. J. (1993). Why Two Heads are Better than One, 1993 Promovendi-dag, Rijksuniversiteit Limburg, Maastricht, The Netherlands, 9 December 1993.

Grant, T. J., and F. H. Tusveld. (1992). Interface Control Document for DUC-Pilot Activity Scheduling and Diagnosis System, BSO/Aerospace & Systems Technical Report 2205792, Issue D, 3 August 1992.

Grant, T. J., R. J. van Meenen, and A. J. Stroobach. (1992). Recognising Objects by Cooperating Prototypes, 1992 Workshop on Cooperating Knowledge-Based Systems (CKBS'92), University of Keele, England, 24-25 September 1992.

Grant, T. J., and K. J. Poulter. (1993). European Initiative for Knowledge Representation Standardisation, Proceedings, 4th workshop, Artificial Intelligence and Knowledge-Based Systems for Space, ESTEC, Noordwijk, The Netherlands, 17-19 May 1993, 303-314.

Grant, T. J., and J. H. J. Lenting. (1993). An Arbitration Protocol for Inter-Agent Learning, 1993 Cooperating Knowledge-Based Systems (CKBS'93) workshop, Keele University, UK, 8-10 September 1993.

Gujer, J. J., and E. Jabs. (1991). Use of Spacecraft Simulators at ESOC, ESA Bulletin, 59, 40-48.

Huhns, M. N. (ed). (1987). Distributed Artificial Intelligence, Morgan Kaufman Publishers, Los Altos, California, USA.

Lesser, V. R., and D. D. Corkill. (1981). Functionally Accurate, Cooperative Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics, SMC-11, 1, 81-96. Also published in (Bond and Gasser, 1988), 295-310.

Michalski, R. S., J. G. Carbonell, and T. M. Mitchell, (eds). (1983). Machine Learning: An Artificial Intelligence Approach, Volume 1, Morgan Kaufman, San Mateo, CA, USA.

Minsky, M. (1986). The Society of Mind, Simon and Schuster, New York, USA.

Pidgeon, A. N., B. Seaton, G. Howard, and K-U. Peters. (1992). Spacecraft Autonomy Concept Validation by Simulation: Phase 2 Final Report, ESA CR(P) 3604, Issue 1, 28 August 1992.

Pronk, C. N. A., N. Koopman, and D. de Hoop. (1992). Development Concept for Dutch User Support, Paper IAF-92-0711, Proceedings, 43rd Congress, International Astronautical Federation, 28 August to 5 September 1992, Washington D.C., USA. Also available as NLR TP 92272 L, provisional issue dated 25 June 1992, Nationaal Lucht- en Ruimtevaartlaboratorium, Amsterdam, The Netherlands.

Pronk, C. N. A., F. B. Visser, and R. M. M. Sijmonsma. (1993). Preparation and Demonstration of a Support Technology Concept for In-Orbit Payload Operations, Proceedings, 3rd European In-Orbit Operations Technology Symposium, ESTEC, Noordwijk, The Netherlands, 22-24 June 1993.

Searle, J. R. (1980). Minds, Brains and Programs, The Behavioural and Brain Sciences, 3, 417-24. Also published in Boden, M. A. (ed). (1992). The Philosophy of Artificial Intelligence, Oxford Readings in Philosophy, Oxford University Press, Oxford, UK, 67-88.

Waltz, A., and J. Llinas. (1990). Multisensor Data Fusion, Artech House Inc, Boston, USA.

Widman, L. E., and K. A. Loparo. (1989). Artificial Intelligence, Simulation, and Modeling: A Critical Survey. In Widman, L. E., K. A. Loparo, and N. R. Nielsen, (eds). (1989). Artificial Intelligence, Simulation and Modeling. John Wiley & Sons, New York, USA, 1-44.

Wohl, J. G. (1981). Force Management Requirements for Air Force Tactical Command and Control, IEEE Transactions in Systems, Man, and Cybernetics, SMC-11, 618-639.

Call for Papers

NASA 1995

Goddard Conference on Space Applications of Artificial Intelligence

May 1995
NASA Goddard Space Flight Center
Greenbelt, Maryland

The Tenth Annual Goddard Conference on Space Applications of Artificial Intelligence will focus on AI research and applications relevant to space systems, space operations, and space science. Topics will include, but are not limited to:

- Knowledge-based spacecraft command & control
- Expert system management & methodologies
- Distributed knowledge-based systems
- Intelligent database management
- Fault-tolerant rule-based systems
- High Performance Computing
- Fault isolation & diagnosis
- Planning & scheduling
- Knowledge acquisition
- Robotics & telerobotics
- Neural networks
- Image analysis

Original, unpublished papers are now being solicited for the conference. Abstracts should be 300–500 words in length, and must describe work with clear AI content and applicability to space-related problems. Two copies of the abstract should be submitted by September 1, 1994 along with the author's name, affiliation, address and telephone number. Notification of tentative acceptance will be given by September 16, 1994. Papers should be no longer than 15 pages and must be submitted in camera-ready form for final acceptance by November 16, 1994.

Accepted papers will be presented formally or as poster presentations, which may include demonstrations. All accepted papers will be published in the Conference Proceedings as an official NASA document, and select papers will appear in a special issue of the international journal *Telematics and Informatics*. There will be a Conference award for Best Paper.

Please e-mail or FAX submissions if possible.
No commercial presentations will be accepted

Sponsored by NASA/GSFC

Mission Operations and
Data Systems Directorate

1995 Goddard Conference on Space Applications
of Artificial Intelligence
May 1995 — NASA/GSFC, Greenbelt, MD

❑ ***Abstracts due:*** Sept. 1, 1994
❑ ***Papers due:*** Nov. 16, 1994
❑ ***Further info:*** (301) 286-3150
❑ ***FAX:*** (301) 286-1768

❑ ***Send abstracts to:***
Walt Truszkowski
NASA/GSFC Code 522.3
Greenbelt, MD 20771
truszkow@kong.gsfc.nasa.gov

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1994	Conference Publication - May 10-12, 1994

4. TITLE AND SUBTITLE

Carl F. Hostetter, Editor

5. FUNDING NUMBERS

510

6. AUTHOR(S)

1994 Goddard Conference of Space Applications of Artificial Intelligence

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS (ES)

Goddard Space Flight Center
Greenbelt, Maryland 20771

8. PEFORMING ORGANIZATION REPORT NUMBER

93B00066

9. SPONSORING / MONITORING ADGENCY NAME(S) AND ADDRESS (ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING / MONITORING ADGENCY REPORT NUMBER

NASA CP-3268

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATMENT

Unclassified - Unlimited
Subject Category 63

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This publication comprises the papers presented at the 1994 Goddard Conference on Space Applications of Artificial Intelligence held at the NASA/Goddard Space Flight Center, Greenbelt, Maryland, on May 10 - 12, 1994. The purpose of this annual conference is to provide a forum in which current research and development directed at space applications of artificial intelligence can be presented and discussed.

14. SUBJECT TERMS

Artificial Intelligence expert systems, planning, scheduling, fault diagnosis, control, knowledge representation, knowledge acquisition, neural networks, distributed systems, fuzzy logic

15. NUMBER OF PAGES

383

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL