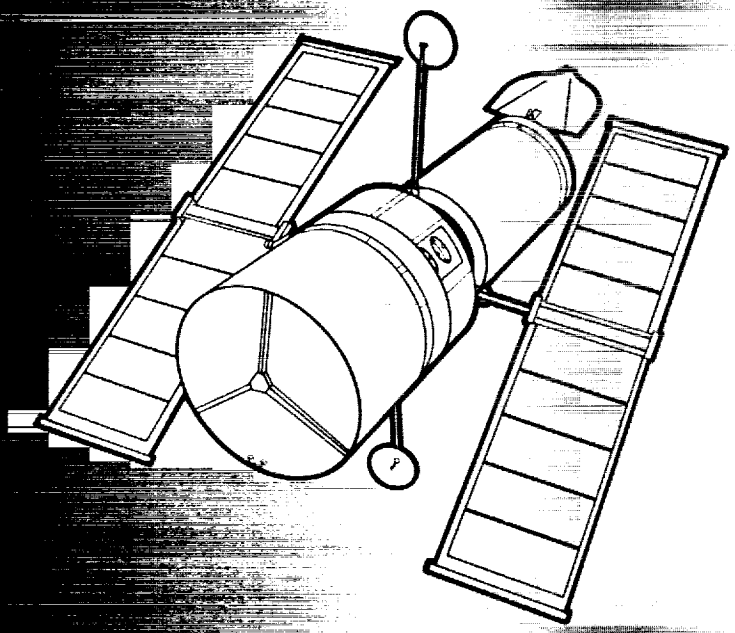
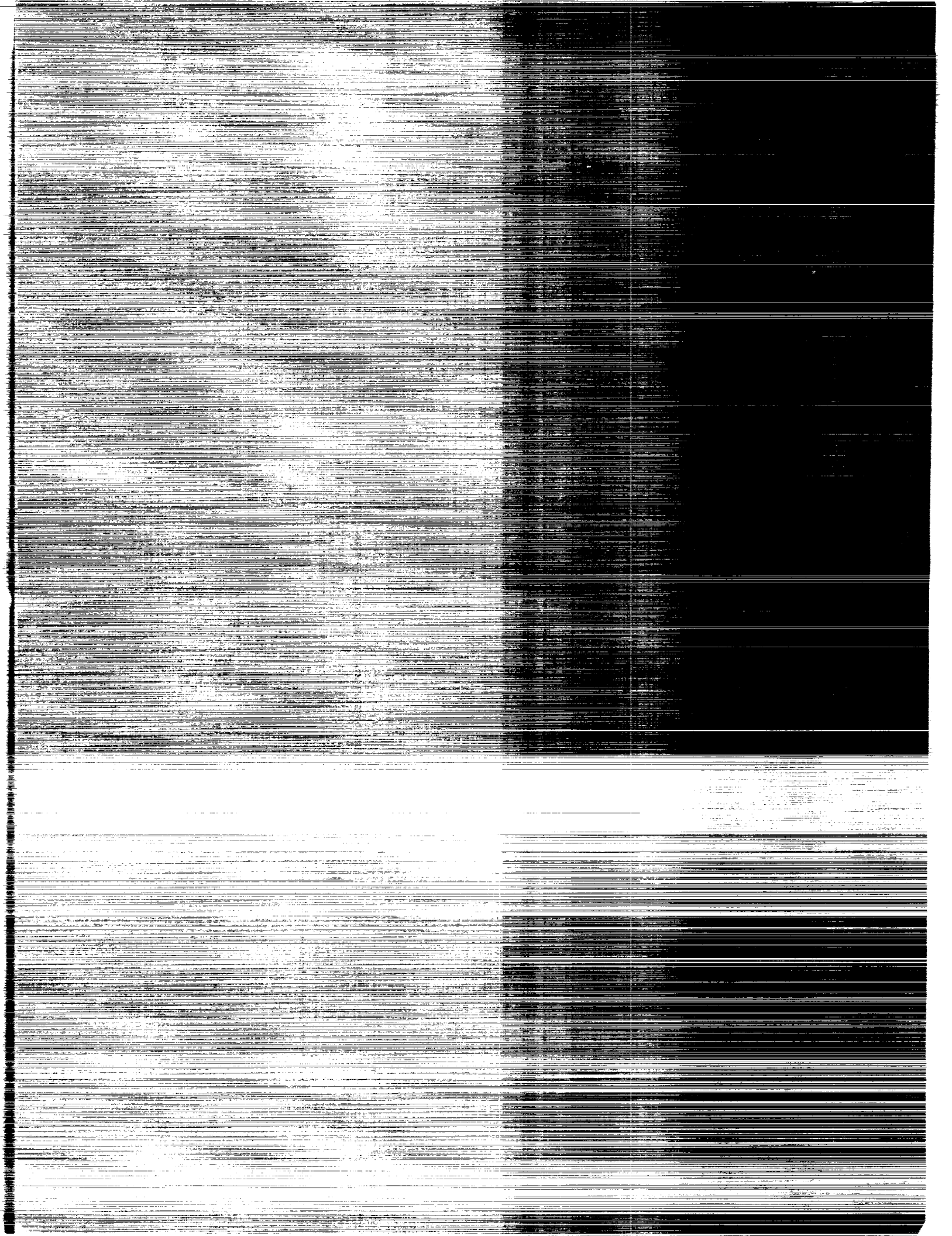
NASA Conference Publication 3200

1993 Goddard Conference on Space Applications of Artificial Intelligence



Proceedings of a conference held at
NASA Goddard Space Flight Center
Greenbelt, Maryland
May 10–13, 1993

NASA



NASA Conference Publication 3200

1993 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 10–13, 1993

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1993

1993 Goddard Conference on Space Applications of Artificial Intelligence

Conference Committee

J. Michael Moore, GSFC (Co-Chairman)
Robert Cromp, GSFC (Co-Chairman)
David Beyer, AlliedSignal Technical Services Corporation
Samir Chettri, GSFC
Carolyn Dent, GSFC
Jonathan Hartley, GSFC
Carl Hostetter, GSFC
Peter Hughes, GSFC
Allan Jaworski, Loral AeroSys
Catherine Jordan, Futron Corporation
James Jordan, Computer Sciences Corporation
Robb Lovell, GSFC
Joseph Mica, GSFC
Karen Moe, GSFC
Jidé Odubiyi, Loral AeroSys
James Rash, GSFC
Nicholas Short, GSFC
Walter Truszkowski, GSFC

Foreword

The state of computer technology can be measured by inspecting the contents of this year's proceedings for the 1993 Goddard Conference on Space Applications of Artificial Intelligence. Approaches are prevalent that only a few years ago would have been impossible due to constraints on memory and speed of computing. As a result of the increased rates of processing, an intensive research effort is now evident in interpreting and managing the data that are being produced. Parallel computing, distributed computing, object-oriented methodologies, constraint satisfaction, and wavelets show up to various extents in some of the papers, and it will not be surprising in the next few years to see an infusion of papers involving applications with these topics. Undoubtedly, future conferences will cover the role of multimedia in presenting information, improved methods for classifying and detecting changes in images, innovative uses of networked computers, and eventually applications of optical computing.

We call your attention to the Call for Papers for the 1994 Goddard Conference on Space Applications of Artificial Intelligence, which can be found in the back of these proceedings. We look forward to your participation.

We would like to thank the members of the conference planning committee, the reviewers, the authors, presenters, and invited speakers for investing their valuable time into making this a successful endeavor.

Bob Cromp
Mike Moore
Co-chairs
1993 Goddard Conference on
Space Applications of Artificial Intelligence.

Table of Contents

Image/Data Classification/Interpretation

Knowledge Engineering

Information Management

Call for Papers, 1994 Conference

Planning and Scheduling

USING AI/EXPERT SYSTEM TECHNOLOGY TO AUTOMATE PLANNING AND REPLANNING FOR THE HST SERVICING MISSIONS

L. Bogovich, J. Johnson, A. Tuchman, D. McLean, B. Page,
A. Kispert, C. Burkhardt and R. Littlefield
AlliedSignal Technical Services Corp. (formerly Bendix Field Engineering Corp.)
Seabrook, MD 20706

W. Potter
Goddard Space Flight Center
Greenbelt, MD 20779

ABSTRACT

This paper describes a knowledge-based system that has been developed to automate planning and scheduling for the Hubble Space Telescope (HST) Servicing Missions. This new system is the Servicing Mission Planning and Replanning Tool (SM/PART). SM/PART has been delivered to the HST Flight Operations Team (FOT) at Goddard Space Flight Center (GSFC) where it is being used to build integrated timelines and command plans to control the activities of the HST, Shuttle, Crew and ground systems for the next HST Servicing Mission. SM/PART reuses and extends AI/expert system technology from Interactive Experimenter Planning System (IEPS) systems to build or rebuild timelines and command plans more rapidly than was possible for previous missions where they were built manually. This capability provides an important safety factor for the HST, Shuttle and Crew in case unexpected events occur during the mission.

Keywords: HST Servicing Mission, AI, Expert System, Automation.

INTRODUCTION

The IEPS Group

The IEPS group at Bendix has been building spacecraft ground support systems with embedded AI/expert system capabilities since 1985. The IEPS group in conjunction with the Spacecraft Control Programs Branch (Code 514) has built several powerful planning and scheduling systems using the C language and conventional hardware (PCs and UNIX-based workstations) rather than traditional AI languages and specialized AI machines. It has been possible to quickly and efficiently change or enhance these knowledge-based systems to adjust to new scheduling conditions.

The IEPS Development Approach

The IEPS systems have been developed with an evolutionary prototyping approach. In contrast to the more traditional waterfall approach, the evolutionary prototyping approach starts with the assumption that a software application cannot be totally specified at the start of the development process. The evolutionary prototyping approach uses the basic cyclical paradigm: gather requirements, create/evolve a prototype, evaluate the prototype, and improve the prototype. In this approach, developing a system is considered to be a discovery process which results in continuously evolving specifications.

In contrast to rapid prototyping approaches, the evolutionary prototyping approach emphasizes the evolution and reuse of generic software tools. By more effectively reusing generic software tools developed in earlier systems and prototypes, the evolutionary prototyping approach reduces the overall system development time.

In the IEPS development approach, several prototypes are delivered to the customer for evaluation before the final system is delivered. This approach allows the customer to provide feedback about the prototypes and results in improved functionality for the final system with decreased risk for the customer. The final system is delivered only after the customer is satisfied with the system's performance.

The ERBS System

In 1987, the ERBS-TDRSS Contact Planning System (McLean *et al*, [1]) was delivered to the Earth Radiation Budget Satellite (ERBS) Flight Operations Team (FOT) at GSFC. This system is written in C and is implemented on an IBM PC/AT. The system automates the process of generating requests for communications support from the NASA Tracking and Data Relay Satellite System (TDRSS), and it was the first expert system at GSFC to provide ground-system support for an on-going mission.

The ERBS system uses scheduling environment data from the Flight Dynamics Facility at GSFC along with strategic planning knowledge from a Knowledge Base (KB) to build a 1-week schedule of TDRSS requests. The system uses alternative scheduling strategies and traditional conflict avoidance techniques to perform conflict resolution (McLean *et al*, [2]).

The ERBS system uses the Planning and Resource Reasoning (PARR) shell to build timelines in batch and interactive scheduling modes. Using PARR, a schedule of requests can be built in a few minutes, compared with several hours by the manual method. After a schedule of requests is built in the batch mode, a graphical timeline can be displayed. Users can edit the timeline in an interactive mode, while obtaining "expert" help from PARR.

The ERBS system has been used steadily since its delivery. In addition, the system has been modified or enhanced several times to meet changing mission requirements. These changes were easily made because of the knowledge-based features of the system (McLean [3]).

Explorer Platform Planning System

In 1991, the Explorer Platform Planning System (EPPS) was delivered to the Extreme Ultra-Violet Explorer (EUVE) FOT at GSFC (McLean *et al*, [4]). EPPS uses AI/expert system technology from the ERBS system.

In addition, EPPS provides several enhancements to the ERBS system. First, EPPS runs on a UNIX-based workstation with X-Windows/Open-Look. Second, EPPS schedules several types of EUVE mission support activities in addition to TDRSS service requests. Third, EPPS provides knowledge acquisition tools so that EUVE FOT can modify the strategies and constraints in the KB and try "what-if" scenarios to adapt EPPS to handle new scheduling situations. Finally, EPPS uses an Ethernet to electronically receive resource data from the Flight Dynamics Facility at GSFC, TDRSS schedule data from the Network Control Center at GSFC, and planning data from EUVE Investigators at the University of California at Berkeley. This Ethernet is also used to send TDRSS schedule data to the Network Control Center and sequences of EUVE command procedures to the Command Management Facility at GSFC.

The IEPS Software Toolkit

As IEPS systems were developed, many generic tools for building new IEPS systems were also developed. Eventually, these generic tools were formally organized into a software toolkit called the IEPS Software Toolkit (NASA-GSFC, [5]). This toolkit contains several types of system-building tools: data formatting and report generation tools, user interface tools, database tools, strategic planning tools and tactical planning tools.

To build a new system using the generic tools in the IEPS Software Toolkit, a software engineer first examines the basic requirements for a new system and identifies the IEPS tools that can be applied to the new system. Next, individual IEPS tools are configured to handle specific tasks, and script files are created to link the individual tools into a system that can be tested. Finally, the unified system is tested and iteratively refined until it meets all of the initial, plus discovered, requirements. Recently, IEPS tools have been used to build another planning and scheduling system, SM/PART.

SM/PART OVERVIEW

HST Servicing Missions are Shuttle missions that are expected to occur about every three years to upgrade or replace failed HST components and to help the HST function to its fullest extent over its 15-year mission lifetime. SM/PART is a planning and scheduling expert system that automates the complex process of building or rebuilding integrated timelines and command plans for the HST Servicing Missions (Johnson, *et al.* [6]). Integrated timelines and command plans are used to coordinate the activities of the HST, Orbiter, Crew and ground systems during the servicing missions.

SM/PART is currently being used to prepare for the first HST Servicing Mission that is scheduled for launch in 1993. It is expected that SM/PART will also be used to support all the other future HST Servicing Missions. For each servicing mission, HST Servicing Mission engineers must provide SM/PART with detailed planning and scheduling data. The planning and scheduling data that is required includes scheduling environment (resource) data, event definitions, sequence definitions and command procedures. SM/PART provides powerful data and knowledge acquisition tools for users to enter this planning and scheduling data.

Before a timeline or command plan is built, a defaults file and a Data Set Configuration (DSC) file must be created. The defaults file provides basic display information for a timeline and command plan such as the mission launch time, the timeline start time and stop time, timeline and command plan header information, and colors to be used on the displays. The DSC file provides the names of the defaults file, Merged Resources file, Event Definition KB, Sequence Definition KB and Procedure Definition KB that are to be used for a particular timeline and command plan.

After all of the required files and KBs have been constructed, SM/PART uses PARR to build a timeline in a batch (automatic) scheduling mode. In this process, PARR places each HST event on a timeline in accordance with pre-defined scheduling strategies and constraints in the Event Definition KB. The data and knowledge components that make up a timeline are shown schematically in Figure 1.

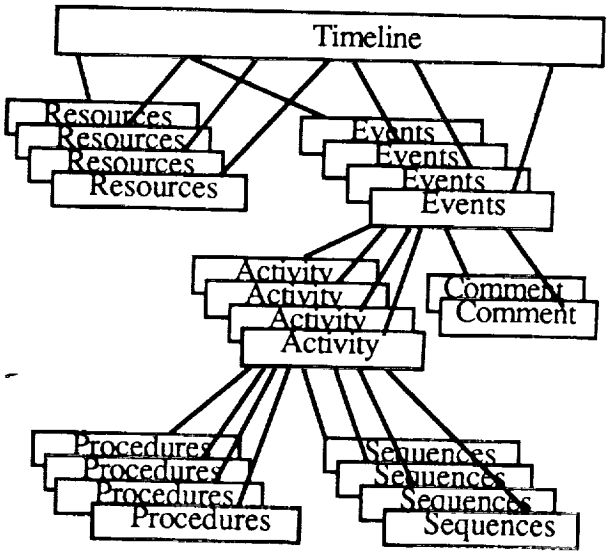


Figure 1. Timeline Components

The timeline that has been built in the batch mode can be graphically displayed with its scheduling environment data (resources) and scheduled HST events (activities and comments).

A section of an integrated timeline is shown in Figure 2.

Because the data objects displayed on an integrated timeline are actively connected to the Event Definition KB (via PARR), users are able to edit a timeline during an interactive scheduling session while they obtain "expert" scheduling help from PARR. For example, as an event is changed, the definition of the event in the Event Definition KB is automatically updated. If an event is dragged by mouse to a place where a scheduling constraint is violated, a prominent

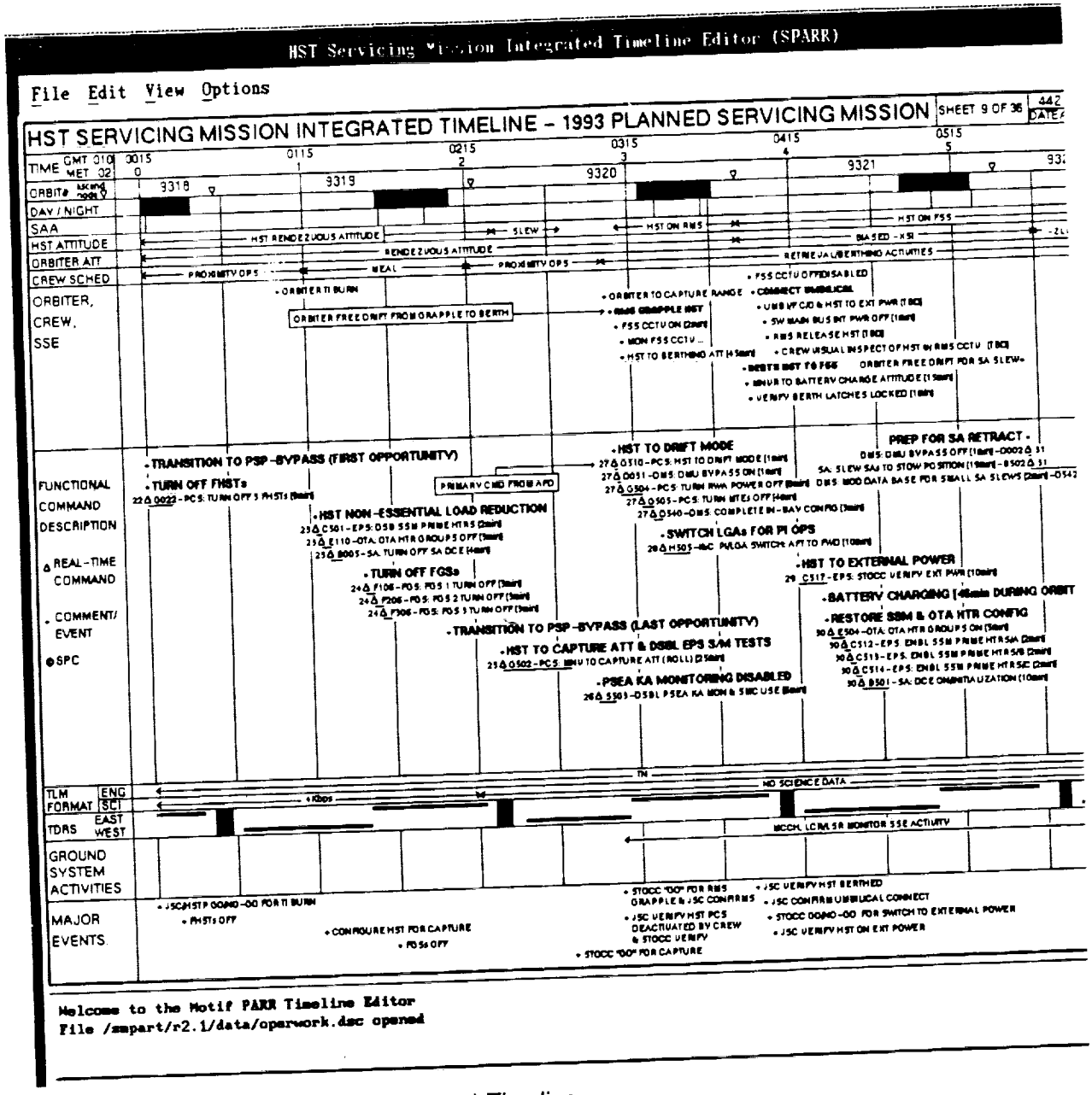


Figure 2. SM/PART Integrated Timeline

"VIOLATION" message is displayed in the Event Definition KB and on the timeline.

After an integrated timeline has been built, sequence definitions and command procedures can be combined with the scheduled timeline event data to automatically generate a command plan. Command plans are used by Servicing Mission personnel at their consoles during the mission.

SM/PART was built in an eight month period using an evolutionary prototyping approach that reused AI tools from earlier IEPS systems. Two prototypes were delivered to HST Servicing Mission engineers for their evaluation before the final system was delivered.

In addition to reusing IEPS tools, SM/PART provides several new and enhanced features. For example, SM/PART is the first IEPS system that uses Motif software for its multitude of windows and pop-up/pull-down menus. Also, SM/PART features enhanced data and knowledge acquisition tools, as described below.

ENVIRONMENT DATA ACQUISITION

One type of planning and scheduling information that must be acquired for SM/PART to function is scheduling environment, or resource, data. Scheduling environment data is part of the strategic planning information (data/knowledge) that is required by PARR to automatically build timelines and command plans during the tactical planning process.

Several types of scheduling environment data displayed on an integrated timeline are acquired electronically from external sources. For example, ORBIT#, DAY/NIGHT, SOUTH ATLANTIC ANOMALY, and TDRS data are received electronically from the Flight Dynamics Facility at GSFC via the HST Application Processor. This data is not generated or modified by HST Servicing Mission personnel, but just reformatted by SM/PART.

Other types of scheduling environment data on an integrated timeline are acquired directly from HST Servicing Mission personnel. Examples of this data include: HST ATTITUDE, ORBITER ATTITUDE, TELEMETRY FORMAT, CREW SCHEDULES, and GROUND SYSTEM ACTIVITIES. For acquiring this data, SM/PART provides several types of Motif-style data-entry forms.

Eventually, the various types of external and user-entered scheduling environment data must be merged into a single data file, the Merged Resources file. Later, data from this file is used by PARR, along with strategic planning knowledge, to automatically place HST events on a timeline.

KNOWLEDGE ACQUISITION

Another type of planning and scheduling information that must be acquired for SM/PART is strategic planning knowledge. For SM/PART, strategic planning knowledge includes activity event definitions, comment event definitions, sequence definitions and command procedures. This knowledge is acquired from HST personnel and stored in various KBs. HST activity event definitions and comment event definitions are stored in the Event Definition KB, sequence definitions are stored in a Sequence Definition KB, and command procedures are stored in a Procedure Definition KB.

For acquiring strategic planning knowledge, SM/PART provides new knowledge acquisition tools. For example, to acquire complex scheduling strategies and constraints, event definition forms with linked push-button or pop-up menus and various options are provided. An Activity Event Definition Form, for AD# B508, is shown in Figure 3. This event is also seen scheduled on the timeline shown by Figure 2.

For acquiring the "start event" attribute of an event, linked push-button and/or pop-up menus are provided to allow the user to specify that an event start when a second event or resource starts or stops. Also, the user may specify a plus or minus offset for the "event start" relative to the start or stop time of the second event or resource.

For acquiring "constraints" for events, linked push-button and/or pop-up menus are provided to allow the user to specify that an event occur only when a second specified event or resource occurs. Alternatively, an event can be specified so that it avoids a second event or resource. In addition, the user can enter plus or minus offset times for the various options selected.

SM/PART also allows users to specify alternative scheduling strategies that can be tried when there is a scheduling conflict. One type of alternative strategy has SM/PART

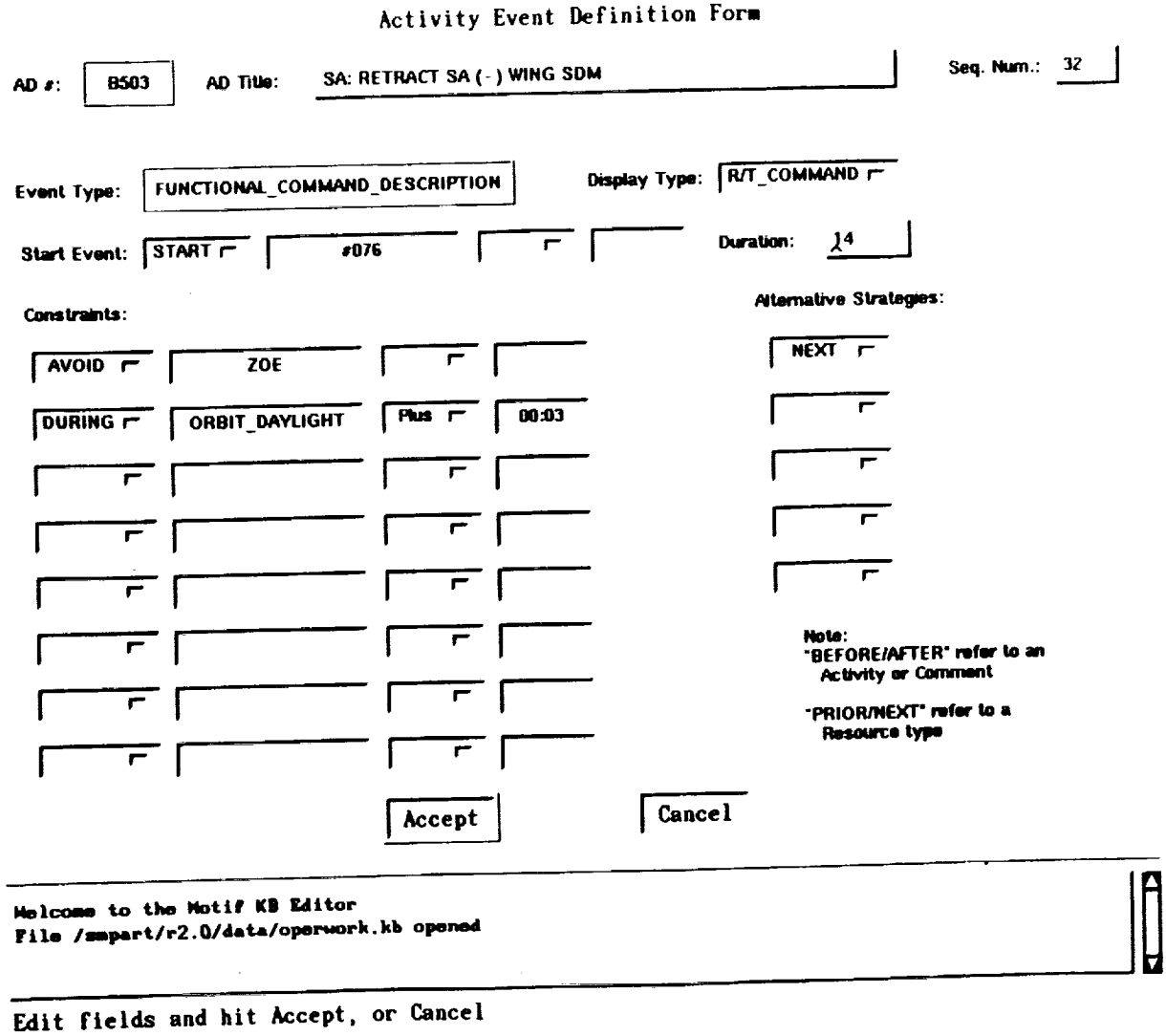


Figure 3 . SM/PART Activity Event Definition Form

schedule an event just before or just after a conflicting event. Another type of alternative strategy has SM/PART schedule an event during the resource window that occurs just prior to or just after the resource window where the conflict occurs.

Sequence Definition Forms are provided for acquiring sequence information such as sequence number, sequence title, the activity events included in each sequence, and special ordering instructions for the activity events within each sequence.

Procedure Definition Forms are provided for acquiring detailed command procedures associated with HST activity events. Procedure Definition Forms allow users to enter information describing the procedures to be performed by operations personnel for each activity event, the effects of each procedure, the duration of each step/substep, and the actions expected in space and throughout the ground system.

BUILDING A TIMELINE

Batch Scheduling

After scheduling environment data and strategic planning knowledge have been acquired, SM/PART is able to build a timeline in the batch (automatic) scheduling mode. This process is referred to as tactical planning. To build a timeline in the batch mode, PARR reads scheduling environment data from the Merged Resources file and strategic planning knowledge from the Event Definition KB, dynamically allocates an internal frame structure to represent each HST event, and uses the information to place events on the timeline. If resources are not available or if constraints are violated, then alternative scheduling strategies are used to try to resolve the scheduling conflicts.

If there are no scheduling conflicts, the event is put on the timeline and the Event Definition KB is updated. If there is a scheduling conflict that cannot be resolved then a prominent "VIOLATION" message is written in the Event Definition KB.

Interactive Scheduling

A timeline that is built in the batch scheduling mode can be displayed graphically on the terminal screen with its scheduled events. Alternatively, a new timeline with scheduling environment data, but with no scheduled events, can be displayed graphically on the terminal screen.

In the interactive scheduling mode, the user can browse the timeline that is displayed and interactively add or change timeline events while receiving expert scheduling assistance from PARR. This expert scheduling assistance is possible because the timeline data objects are actively linked via PARR to the Merged Resource file and Event Definition KB.

As an example of editing a timeline in the interactive scheduling mode, a user may click on an HST activity event with the mouse and "drag" it to a new, valid position. In this case, the Event Definition KB is automatically updated. However, if the activity is dragged to a place where a scheduling constraint is violated, then a pop-up window with a "VIOLATION" message that the user must respond to is displayed on the screen.

BUILDING A COMMAND PLAN

After a timeline has been built, a detailed command plan corresponding to the timeline can be automatically built and displayed on the terminal. Building a command plan involves retrieving and combining scheduled timeline event information with sequence definitions and command procedures. Sequence definitions specify groups of HST activities while command procedures specify the detailed steps required to complete each scheduled HST event.

A command plan that is displayed on the terminal can be converted to an identical graphical command plan print. Command plan prints are used by HST Servicing Mission engineers at their control consoles during the HST Servicing Missions.

SM/PART is also able to automatically synchronize a command plan with a timeline. Synchronizing a command plan with a timeline is required whenever changes are made to either the command plan or its corresponding timeline.

REPLANNING

An important capability of SM/PART is to quickly rebuild a timeline and command plan. This capability is particularly important if unexpected events or changes in the scheduling environment occur during a mission. In critical situations, this capability provides an important safety factor for the HST, Shuttle and Crew. Initial results from the HST Flight Operations Team indicate that SM/PART is able to reduce the time to rebuild a timeline and command by a factor of ten compared with the former manual method using a Macintosh (Potter *et al*, [7]).

CONCLUSIONS

SM/PART has successfully reused and extended IEPS AI/expert system technology to build SM/PART and automate the complex task of building timelines and command plans for HST Servicing Missions. To automate this task, SM/PART initially provides capabilities for HST Servicing Mission personnel to acquire scheduling environment data and strategic planning knowledge. Next, SM/PART is able to use the acquired scheduling environment data and strategic planning knowledge to automatically place HST events on a timeline. During interactive scheduling sessions, SM/PART is able to provide "expert" scheduling assistance to users. Finally, SM/PART is able to combine timeline event data with sequence definitions and detailed command procedures to automatically generate command plans.

An evolutionary prototyping approach which emphasizes reusing and enhancing AI tools was successfully used to build SM/PART in an eight month period.

ACKNOWLEDGEMENTS

The authors wish to thank Patricia Lightfoot and Dorothy Perkins at NASA-GSFC (Code-510) and Ellen Stolarik and David Warren at Bendix Field Engineering for their continual support and many contributions to the IEPS task. This work was supported by NASA Contract NAS5-27772.

REFERENCES

1. McLean, D., Littlefield, R. and Beyer, D. (1987). An Expert System for Scheduling Requests for Communications Links between TDRSS and ERBS, *Proceedings of the 1987 Goddard Conference on Space Applications of Artificial Intelligence (AI) and Robotics*, Goddard Space Flight Center, Greenbelt, MD.

2. McLean, D., Page, B., Tuchman, A., Kispert, A., Yen, W. and Potter, W. (1991). Emphasizing Conflict Resolution versus Conflict Avoidance during Schedule Generation, *Expert Systems World Congress Proceedings*, Vol. 1, pp. 385-394, Orlando, Florida.

3. McLean, D. R. (1990). Maintaining an Expert Planning System: A Software Tools Approach, Jay Liebowitz (Ed.). *Institutionalizing Expert Systems: A Handbook for Managers*, (pp. 91-98) Prentice Hall.

4. McLean, D., Page, B., and Potter, W. (1990). The Explorer Platform Planning System: an Application of a Resource Reasoning Shell, *Proceedings of First International Symposium on Ground Data Systems for Spacecraft Control*, pp. 195-200, Darmstadt, Germany.

5. NASA-GSFC/Code 514, (August 1992). IEPS Software Toolkit, Vols. 1-7.

6. Johnson, J., Tuchman, A., McLean, D., Kispert, A., Bogovich, L., Burkhardt, C., Page, B., Littlefield, R., Potter, W., and Ochs, W., (1992). HST Servicing Mission Planning and Replanning Tool, *World Space Congress*, Washington, DC.

7. Potter, W., Bogovich, L. and Johnson, J. (1992). Hubble Space Telescope Servicing Mission Planning and Replanning Tool (SM/PART), *1992 GSFC Research and Technology (R&T) Report*, (in press).

Monitoring/Control/Diagnosis

KNOWLEDGE-BASED CONTROL FOR ROBOT SELF-LOCALIZATION

Bonnie Kathleen Holte Bennett, PhD

Honeywell Systems and Research Center
Minneapolis, MN

University of St. Thomas
Department of Computer Science
St. Paul, MN

ABSTRACT

Autonomous robot systems are being proposed for a variety of missions including the Mars rover/sample return mission. Prior to any other mission objectives being met, an autonomous robot must be able to determine its own location. This will be especially challenging because location sensors like GPS, which are available on Earth, will not be useful, nor will INS sensors because their drift is too large. Another approach to self-localization is required.

In this paper, we describe a novel approach to localization by applying a problem-solving methodology. The term "problem-solving" implies a computational technique based on logical representational and control steps. In this research, these steps are derived from observing experts solving localization problems. The objective is not specifically to simulate human expertise but rather to apply its techniques where appropriate for computational systems. In doing this, we describe a model for solving the problem (Ref. 1) and a system built on that model, called localization control and logic expert (LOCALE), which is a demonstration of concept for the approach and the model. The results of this work represent the first successful solution to high-level control aspects of the localization problem.

Keywords: Knowledge-based control, robotics

INTRODUCTION

Interest has been growing in the development of autonomous mobile robot systems. For example, autonomous mobile robots have been proposed for the Mars rover/sample return mission. In addition, applications for such systems are being proposed for military, industrial, and scientific endeavors. Missions include advanced reconnaissance, battle damage/contamination assessment, and exploration for cartographic, geographic, and geologic concerns. In each of these missions, an autonomous mobile robotic agent would be used in place of a human agent for cost savings and safety reasons. In order for a robotic agent to perform the above missions, it must be able to perform navigation tasks. These tasks generally include locating oneself on a map, determining a route to a specified location, performing some operation at that location, and continuing on to other locations or returning. The first of these tasks, locating oneself on a map, is the most critical because all the other functions rely on the agent having and maintaining accurate knowledge of self-location. The environments for these tasks are usually large outdoor spaces where environmental features are much larger than the robot, and the entire environment cannot be observed all at one time from the robot's sensors. Unambiguous, human-made landmarks and other location tools are not available.

There are several systems used by aircraft and other navigational systems to perform localization. They include global positioning systems (GPS) and inertial navigation systems (INS). GPSs use radio signal returns from orbiting satellites to determine an agent's current position on the Earth. The resolution of these systems is quite good and would preclude the need to solve the

localization problem for Earth-based scenarios. However, localization is a major problem for space exploration. No GPS satellites exist for Mars. It will not be cost-efficient to put a GPS system in place for this relatively low usage, so in the near term, autonomous systems on Mars will need the capability to localize. While INSs also provide localization information, they unfortunately experience drift on the order of feet per hour over the long run and meters per second in the short run, making these systems inadequate for localization in ground-based robot systems.

THE LOCALIZATION PROBLEM

Problem Description

The objective of the localization problem is specifying the current viewpoint and viewing direction in the map coordinate system. Knowledge of self-location is essential to any agent that will interact with an external environment. If self-location is defined in terms of the map coordinate system, then knowledge of it makes all other map data accessible. Given the constraints of current technology (e.g., videocameras, digital maps), self-localization becomes a translation from one input domain into another. For our research, two data sources were explored: visual information and map information.

At an abstract level, localization can be modeled as three interacting processes (see Figure 1). Two of the processes are perceptual: they identify the pertinent information from the view of the image and from the map. The inputs from a videocamera are a series of pixels, each defining a grey level or color. These need to be preprocessed to determine meaningful symbolic labels like hill, valley, saddle, etc. The inputs from a digital map are elevation points in a grid pattern over the map area. These, too, need to be preprocessed into meaningful symbolic labels. Ideally, both of these processes are able to operate in both data-driven and hypothesis-driven modes. In the data-driven mode, they reason bottom-up from the input data, gleaning all they can from new data and integrating it with old data. In the hypothesis-driven mode, they reason top-down and search for specified data of a certain type or in a specific location. The third process determines the correspondence between the features in the map and the features in the view. Correspondence is determined by matching features from the map and the view. This matching should be able to occur in both directions: map to view and view to map. This capitalizes on the results of data-driven reasoning in each domain and uses those results to drive hypothesis-driven reasoning in the other. The search for matches should be guided by knowledge of the environment and heuristics that reduce the computational complexity of the search. The correspondence process mediates between the two perceptual processes. For example, it translates between the map's plan-view (down-looking) representation, where elements are north or west of each other, and the image's lateral (side-looking) view where elements are left and right or in front of each other.

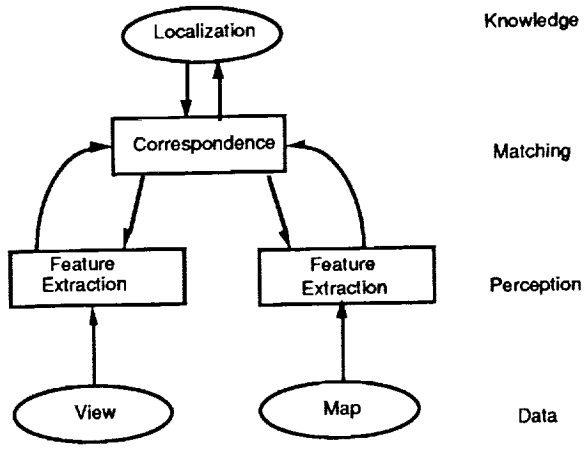


Figure 1. Top-level Model of the Localization Process (The perception process extracts features from the map and the view of the image. Matching determines the correspondence between the view and map features. Knowledge is used to determine the localization of the agent on the map.)

Problem Approach

Formally, the localization problem is matching image features to map features and using that information to hypothesize a current viewpoint. The goal of localization is to determine an estimate of the location where the image was shot and the direction from which it was shot (i.e., to derive a viewpoint hypothesis). In the case where one unambiguous estimate cannot be derived, a list of prioritized viewpoint hypotheses is generated. These viewpoint hypotheses constitute the best estimates derived along with rank-order preference for them.

Because the objective of this research was to develop a model to provide high-level control for localization, it determined strategies for effectively and efficiently generating and evaluating viewpoint hypotheses.

The rationale for using feature-matching techniques is that there is simply too much data to deal with individually. This is essentially an argument of granularity. Both raw map and image data are digitized for input to a computational system; however, the granularity of this digitization is extremely small in order to provide the computer with as much data as possible. The prospect of matching each picture element, or pixel, in the visual sensor input data to a point on the map is daunting. The approach of combining individual map and sensor data elements into features reduces the search required for matching. In this approach, many data elements are combined into geographic features and are dealt with on the level of hills, valleys, gaps, and so forth. Humans performing this task use data elements on the level of geographic features. It is therefore a natural representation level to communicate the computer system's abilities to its human builders and observers.

Demonstration Constraints

For this research, test cases with specific map and sensor data have been explored. In these test cases there are two available inputs: a topographic map and a single video sensor image. These inputs are assumed to be processed by a low-level processing system, which is not part of this research. Figure 2 shows an example view. Figure 3 shows the area of the topographic map used in this problem.

The rationale for limiting the inputs is that they are a minimal set of inputs. If a system can be built that works effectively with this constrained environment, it can likely be expanded to work in domains with richer inputs. The limit on the visual sensor to one input frame is quite severe. This means that no stereo or image-to-image information is available. The limits of a normal camera are also quite tight—the angle of view is limited. So, while a panoramic or preferably a full-circle view would give more data, we chose to explore what can be gained from the standard limited camera view. In addition to limiting the viewing angle from side to side, the standard camera also limits the viewing angle from top to bottom. So the data about the location on which the camera is standing, which could be quite useful, is unavailable. The main limitations on the map data are the resolution and the fact that it is limited to elevation data. Our goal was to focus on large outdoor environments, so we eliminated human-made features from our scenarios and picked areas where their effect was minimal. Thus, the elevation data in the digital map is essential and was readily available.

This work assumes that a low-level image and map processing system processes the raw image signal and map elevation data and sends processed information to LOCALE. The result of this processing is the location and classification of features in the map and image. Map features are peaks, valleys, ridges, etc. Image features are peaks, valleys, gaps, ridges, saddles, and inclines. Figure 4 shows the processed map information. The image and map processing system was simulated for this work because computational systems are only just being developed to this effect (Refs. 2 and 3). LOCALE can query the simulated image

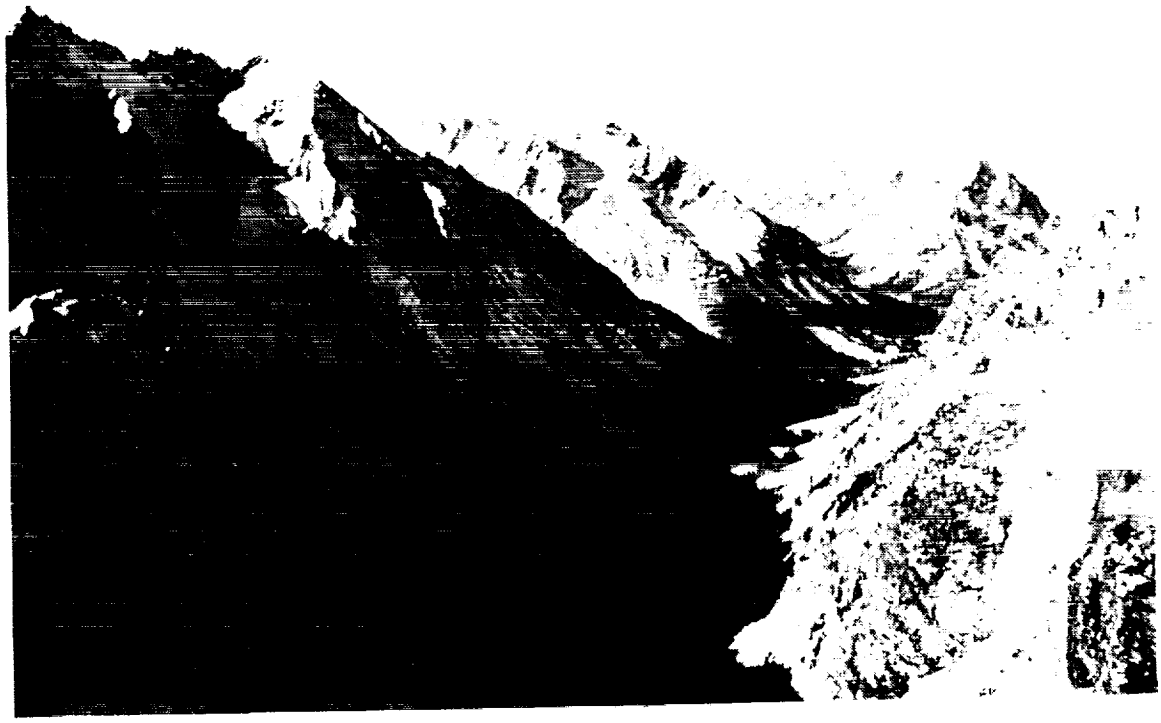


Figure 2. Example Videocamera View (In this example of a videocamera view, the most prominent features are the large valley in the middle and the two protrusions on either side of it in the front. Other valleys and peaks also appear in the view.)

and map processing system for specific data as required. The simulated image and map processing system replies by describing the map and image features (e.g., hills, valleys, etc.) at varying levels of detail.

Finally, the localization problem is actually a class of problems that fall on a spectrum determined by the amount of *a priori* information available to the system. Figure 5 shows the localization spectrum. Near one end of the spectrum are update problems where a lot of *a priori* information exists. In this region the typical problem is verifying one's location after a short move from a known location. Update problems are easier than dropoff problems because the agent has an indication of current location in an update problem. The agent needs to test actual sensor data against expected sensor data based on estimated location. In the dropoff scenario the agent must determine the estimated location in addition to testing its validity. In dropoff problems the agent has no *a priori* knowledge of where it is on the map. The research we have done addresses the dropoff problem and works with no *a priori* knowledge, not even a compass heading.

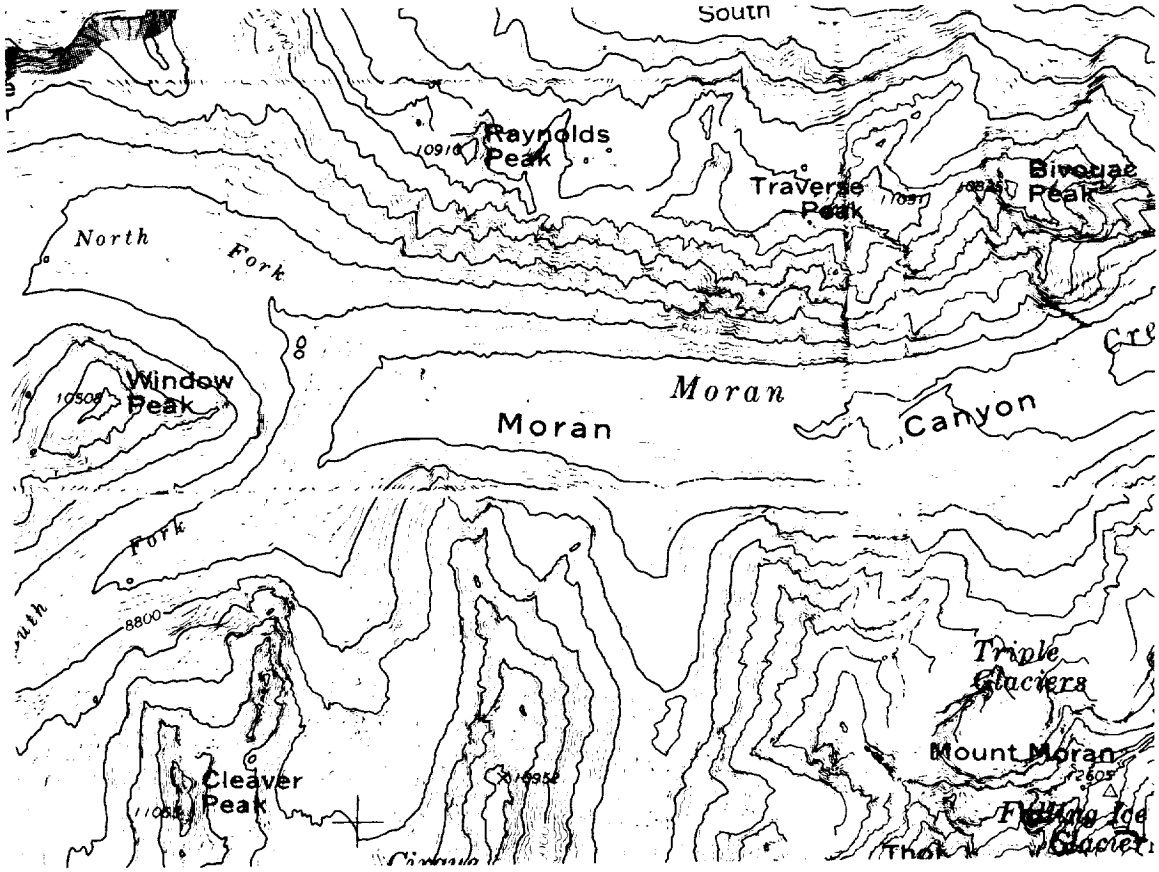


Figure 3. Example Topographic Map of Teton Region

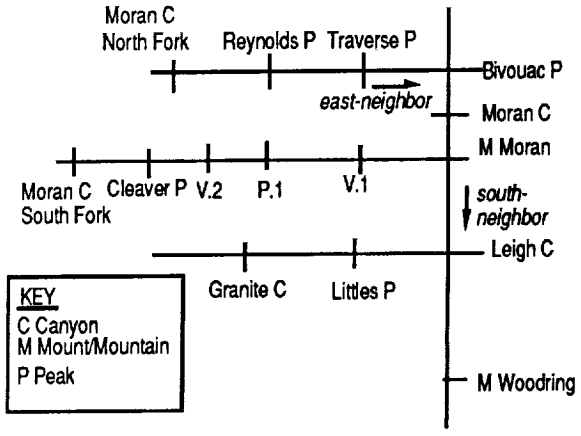


Figure 4. Processed Information from the Map (The processed map information is represented in a semantic network with proximity links between adjoining features.)

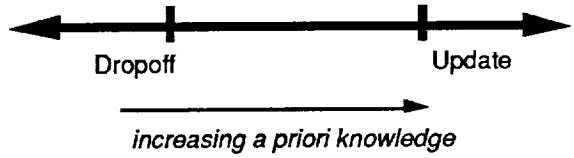


Figure 5. The Localization Spectrum (Problems with no *a priori* knowledge are dropoff problems. Problems with more *a priori* knowledge are update problems. This research focuses on dropoff problems.)

RELATED WORK

Traditional computational approaches to the localization problem and related problems have developed in several areas: pattern recognition, control and representation systems, and computer vision research.

Classic pattern recognition approaches to the localization problem have differed from this work in two aspects: their reliance on low-level matching and their reliance on *a priori* knowledge.

Past work has explored low-level signal matching techniques as opposed to frame-based approaches for correlating images with maps. There are two signal domains in which this work can be pursued: the image domain and the map domain. More work has been done in the image domain. Ernst and Flinchbaugh (Ref. 4) matched estimated features with sensed features and required a known sensor location within a small neighborhood. Stein and Medioni (Ref. 5) explored localization using panoramic horizons as the features. This approach requires extensive pre-computation of indexed synthetic horizon maps and then matches the actual horizon to these. This approach also requires a full 360° view. As for the map domain, Lavin's work (Ref. 6) centered around determining what depth map could cause a two-dimensional (2-D) projection. It requires multiframe moving images.

The HILARE project (Ref. 7) sought to develop an experimental testbed on which to study general robotics, and robot perception and planning. The position referencing subsystem on HILARE used infrared triangulation operating in areas where fixed beacons were installed. This allowed for position determination either relative to objects and specific environment patterns or in a constructed frame of reference.

Beyond the low-level matching, some attention has been paid to control for low-level image processing. Arkin et al.(Ref. 8) explored an integrated system for the interpretation of visual data in a mobile robot testbed. This work essentially explored the low-level processing tasks and relied heavily on *a priori* knowledge of expected location. In related work, Fennema, et al. (Refs. 9, 10, and 11) use a hierarchy of representation and control techniques to solve the planning concerns for control uncertainty but do not examine it in light of specific localization problems. In addition, some research has explored advanced representational structures. Binford (Ref. 12) and Kriegman, et al. (Ref. 13) explore a hierarchical representation model for robot navigation focusing on interior environments. Smith and Strat (Ref. 14) begin to explore a frame hierarchy and a community of independent processes for solving outdoor problems with human-made landmark recognition. Andress and Kak (Ref. 15) explore knowledge-based control for accumulating evidence and controlling reasoning in a hierarchical spatial reasoning system with a computer program called production system environment for integrating knowledge with images (PSEIKI) that reasons about interior environments.

Traditionally, vision system approaches have only examined the update problem. Update implies *a priori* knowledge, an accurate estimate of current location. Examples of such systems include the work by Davis, et al. (Ref. 16) on DARPA's Autonomous Land Vehicle (ALV) program, Carnegie-Mellon University's Navlab project [17], and Lawton, Levitt, et al. (Refs. 18, 19, 20, 21, 22 and 23).

Thompson, et al. (Refs. 24 and 25) define the aspects of the localization problem and specifically the dropoff problem in large-scale environments.

The research described here uses a different approach where abstract representations of both the map and image were generated by extracting high-level features from each domain. The correspondence between these features is then computed in this higher-level abstract domain.

The work of Thompson, Pick, et al. (Ref. 25, 26 and 27) is closely related to this research. Here, protocol analyses of experts

indicated that humans solving localization problems benefit from the following strategies:

1. Concentrate on the view first.
2. Landmark features should be organized into configurations.
3. Information about terrain at the viewpoint is important.
4. Multiple hypotheses need to be generated and examined.
5. Hypotheses should be compared using a disconfirmation strategy.
6. The ability to move to alternate viewpoints is important.

From work with experts, we made the following general observations:

- **Grouping things into configurations is important**—These configurations are linear and contain relationships among the constituent entities. This serves to constrain the search because the more complex a feature is the more specific the search can be. And, configurations are more complex than the features that compose them.
- **Working at various levels is important**—At times it is useful to take an overall view of the area or the map. At other times it is important to focus on increasingly minute details of an area. It is important to be able to swap back and forth between these levels, too.
- **Heuristic generation and testing of hypotheses is important**—For example, humans use the fact that a great deal of information is required to fully accept a hypothesis, while very little is required to reject one.
- **Data-driven and hypothesis-driven reasoning is used**—Early on, data about the viewpoint are gathered and interrogated—this is data-driven reasoning. Once enough data are present to construct sufficiently interesting hypotheses, they can drive the reasoning.

THE MODEL

From the discussion on human experts in the previous section, two principles stand out:

- Grouping objects into composite entities focuses attention and reduces search.
- Representing data and working at multiple levels allows opportunistic and agenda-driven reasoning to work cooperatively.

Grouping Objects

From a purely mathematical perspective, grouping objects into composites for matching has clear significance. If one is trying to match two sets of features (e.g., trying to match image features to map features) and there are five features in the first set and 40 in the second, then the number of possible matches is 90,536,361.

This calculation is

$$\sum_{j=0}^{\min(m,n)} \frac{n!}{j!\,(n-j)!}\,\frac{m!}{(m-j)!}$$

where m and n are the cardinality of the sets (in this case 5 and 40). If, however, the first set is actually grouped into two groups: one of three and one of two, and the second set is divided into eight groups of three and some singletons, then the number of possible matches between the groups of three in each set is only nine. The group of three from the first set could match any of the eight, or none at all. So, from a mathematical perspective, grouping clearly assists matching. In computational terms, grouping objects into composites and then working with the composites reduces the search space of the problem.

Grouping is observed in expert performance in the localization problem. Successful

experts group individual features into configurations. The configurations observed and used are linear and generally radial from the subject. The expert realizes that there are fewer groupings of hill-valley-hill in a straight line on the map than there are individual hills or valleys. So the expert chooses to reason at the configuration rather than the feature level.

As for the model, the goal is to capture the groupings that facilitate the heuristic solution to the localization problem. Practically, this means an enumeration of the terms experts use in problems of this type and a thorough understanding of the interrelationships of these terms. This understanding leads to illumination of constraints and other rules of thumb to focus matching and other reasoning processes for localization.

Multiple Levels of Representation and Reasoning

The second major principle of the model is that working at multiple levels provides the ability for opportunistic- and agenda-driven reasoning to work cooperatively. Data required for the model fall across a spectrum of levels of complexity. The levels of data required in the model reflect the derivative nature of the data. Low-level data are the raw inputs from the simulated image and map processing system. They consist of brief statements of fact, for example, that a certain hill is at a certain location. Higher-level data, including configurations, possible configuration matches, and viewpoint hypotheses derive from them.

Data at different levels are very different. Raw data are immutable facts. Derived data are less strong. It is useful to distinguish permanent and persistent data in this context. As the system approaches a given localization problem in a given geographic area, that is one problem-solving episode; there are some data that will be permanent to this problem-solving episode, and some that will not. The permanent data are facts like, "There is a hill at coordinate 335,432." Less permanent data (we use the term *persistent* data) may fall in and out of favor. Persistent data is a specialized example of a requirement for nonmonotonic reasoning. Hypotheses are examples of persistent data. At one point in the episode a hypothesis may look very promising, it may lose credibility, then gain it again as more data are gathered, but it is not truly temporary because even when it appears unlikely, the mere fact that a hypotheses has been explored to a certain degree of detail is important and should be preserved and not discarded as one would be tempted to do with false information. Like systems requiring full nonmonotonic reasoning, persistent data requires that the logical dependencies of conclusions are maintained; however, this is not a case where data will later be retracted, per se, as in a full non-monotonic system. In contrast, persistent data will not decrease the amount of knowledge held by a system (it will always grow), but this knowledge will simply have preference values that may change (increasing and decreasing) over time; however, all of the information used to solve a given problem is temporary in the sense that it holds for only one localization episode. In the next episode, when another given problem in another given geographic area is undertaken, all of these data will be gone, unlike the domain-specific information retained from problem to problem within a given geographic area.

In addition, we observe that two approaches to reasoning are employed by successful human experts. First, they use a *data-driven* approach to the problem, where they are gathering all the information they can bring to bear on the problem at hand. In this approach the expert is building up complex representations of the world. This is bottom-up reasoning from raw data. Once these representations have been built, and the pertinent data have been gleaned from them (e.g., there is a big valley in the middle of the image with a hill on either side, therefore, the configuration hill-valley-hill is important), then *hypothesis-driven* reasoning can begin (e.g., go look for hill-valley-hill configurations in the map). This is top-down reasoning from derived information. It is important to use both data- and hypothesis-

driven approaches because a data-driven approach works well when little is known about the problem at hand, but a hypothesis-driven approach focuses the search when specific hypotheses exist. And, it is important to be able to alternate between them during the course of a problem-solving episode. A strategic reasoning superstructure provides the capability for the system to assess its current state, select among alternatives for the next step, and choose the appropriate one. This is the self-conscious control of the system because the break points provided in the strata of reasoning components are the opportunities for evaluation and selection of the next course of action.

THE APPROACH

The approach used for this research was to understand the features in the domain relevant to solving localization and then to construct the representational and control structures to work with this information.

The individual features are hills, valleys, walls, etc. Image features have properties like membership in a group of similar features (valleys, hills, gaps) and relations to other features in the image (being right or left of one another, occlusion) and height in the frame. Map features have properties like location, slope, relation to other features (north-of, south-of, etc.), and elevation. The current implementation limits features to points on an X, Y coordinate. This limitation is used for simplicity of processing. The most significant of these properties are the relations among features. These relations are used to define configurations of features. One type of configuration is a linear configuration where three or more objects are in a line. In this case the relation between the first two objects is the same as between the second and third objects.

Hypotheses are expressions of potential solutions (or partial potential solutions) to the localization problem at hand. Multiple, conflicting hypotheses may be under consideration at any one time. There are three types of hypotheses: feature-match hypotheses, configuration-match hypotheses, and viewpoint hypotheses. Feature-match hypotheses acknowledge the possibility that a particular map feature may be a particular image features. These are constrained by matching rules derived from the possible visual appearance of map features. For example, a saddle from the map may appear as either a valley, a saddle, or a gap in the image. Only possible matches need to be posited. Configuration-match hypotheses are statements of the potential correspondence between a configuration in the map and a configuration in the image. These are constrained by the feature matches. For a configuration-match hypothesis to be retained, not only must the configuration forms match (two linear and three component configurations may be matched, but a linear configuration with three components and a right-angle configuration with four components may not be matched), but the individual features must be compatible. That is, the appropriate feature-match hypotheses must exist. Finally, viewpoint hypotheses are the outgrowth of configuration-match hypotheses. If two configurations do indeed match, then there is a limited area from which they can be viewed to give the appearance in the image. The viewpoint hypotheses are the representation of this. In addition to the individual components that must match for it to be true, the viewpoint hypothesis includes a description of the area where the observer must be located. This area is constrained to be within certain map coordinates limited by the visibility and intervisibility of the features in the image as related to their potential match partners from the map.

Representation Issues

The representation components of the model use a hierarchical semantic network. Figure 6 shows the data categories of the representation components. The lowest level data is the raw data input from the simulated

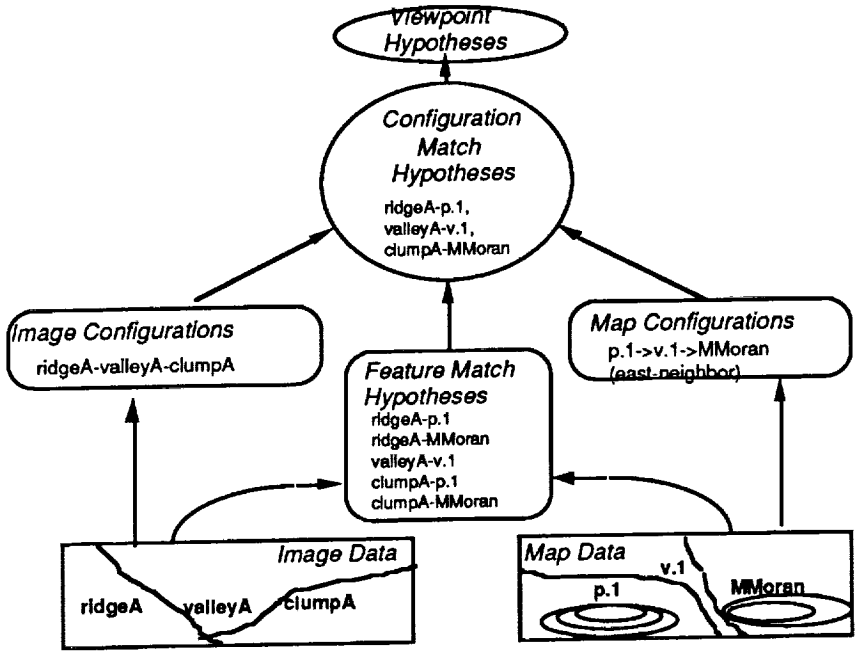


Figure 6. Data Categories in the Computational Model of Localization

image and map processing system. Successively higher levels of data represent abstracted, interpolated, or otherwise derived data that the system has concluded from the input data. The components of the semantic network are the objects and the relations between them. The components are represented in *frames* and the relations are represented in *slots* in the frames.

There are actually several hierarchies that are appropriate to this problem. The main data representation hierarchies are the configuration hierarchy and the feature taxonomy. Hierarchies are also used for rules and relations.

Individual map and image features are represented as instances of the classes defined in a domain-specific feature taxonomy that divides features into image features and map features. Image features are GAPS, IMAGE-RIDGES (so called to distinguish them from ridges that appear in the map), IMAGE-SADDLES, IMAGE-VALLEYS, INCLINES, and PEAKS. These are all of the elements that can be uniquely distinguished in an image. Map features are divided into BENCHES, DEPRESSIONS, PROTRUSIONS, and WALLS. DEPRESSIONS are divided into RE-ENTRANTS and VALLEYS. VALLEYS are divided into BASINS, DRAWS, GULLIES, HANGING-VALLEYS, and MAP-SADDLES. BASINS are divided into BOWLS and CIRQUES. MAP-SADDLES are divided into COLS and PASSES. PROTRUSIONS are divided into BUTTES, PEAK-PRIMITIVES, RIDGES, and SPIRES. RIDGES are divided into BUTTRESSES, SHOULDERS, and SPURS. WALLS can be distinguished into HEADWALLS.

Control Issues

There are many types of expertise brought to bear on localization problems. High-level reasoning expertise can select from among several high-level alternatives:

- Understand the viewpoint,
- Understand the map,
- Generate and test hypotheses.

In addition, these high-level reasoning processes can call on a number of lower level subroutines to perform their functions:

- Gather map data,
- Gather image data,
- Scrutinize the incoming data and connect them to known data,
- Match features,
- Locate configuration,
- Match configurations,
- Establish viewpoint hypotheses,
- Evaluate and refine viewpoint hypotheses.

Each of these reasoning steps (both high-level and low-level) is a specialized subroutine. These subroutines can encapsulate just enough information to perform one specific function. The implementation represents them independently and weaves them together as appropriate (e.g., where a high-level function calls one or more low-level functions). And, it coordinates the actions of the multiple experts.

THE SYSTEM

Figure 7 shows the system diagram of the computer implementation running on a Sun workstation using KEE® (by Intellicorp) and lisp. Data flow in from the simulated image and map processing system and are posted on either the map or the image knowledge bases (KBs). These KBs are built on top of the taxonomy KB, which contains the problem-specific data about the localization problem and the geographic region in general. The taxonomy is the hierarchy of geographic features that occur in this area. The control structures are the reasoners and rule bases that scrutinize the map and image information, taking into account their relationships within the taxonomy. The results of this scrutiny form the basis for the hypotheses that are posted in the hypothesis KB. Further scrutiny of the hypotheses may lead the control structures to send queries back to the simulated image and map processing system for more data.

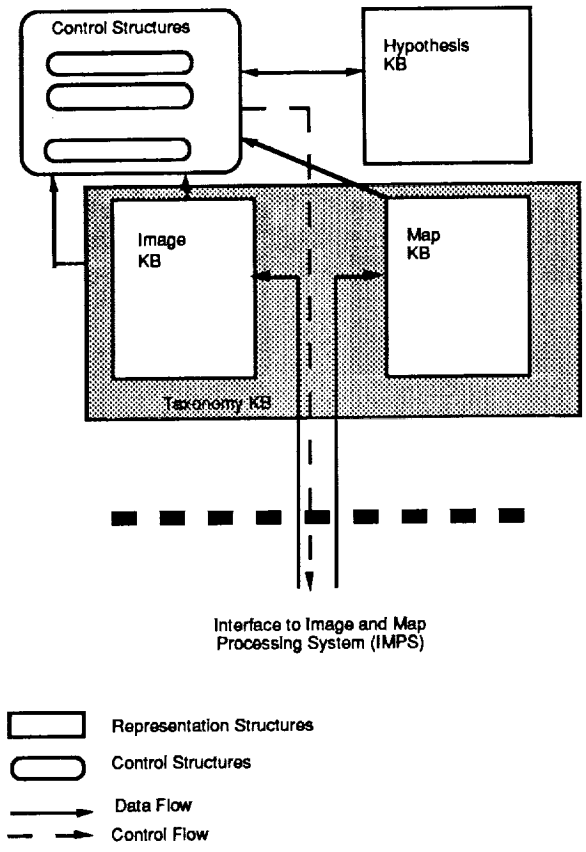


Figure 7. LOCALE System Diagram

These data will arrive as new postings to the image and map KBs.

The levels of representation of problem-specific information from lowest to highest are:

- Input data (map and image),
- Feature-match hypotheses,
- Configurations (map and image),
- Configuration-match hypotheses,
- Viewpoint hypotheses.

As an input datum arrives it is plugged in as an instance of one of the classes in the hierarchy. This allows it to inherit certain properties from its super classes and to be reasoned about as a member of the class.

The feature taxonomy provides the basis for feature matching. Table 1 shows a feature-

Table 1. Feature-Match Matrix(Potential features from the map and the image are compared for match quality.)

		Image-Features				
	Gaps	Image-Ridges	Image-Saddles	Image-Valleys	Inclines	Peaks
Map-Features						
Benches	0	3	0	0	1	1
Depressions	5	0	5	5	0	0
Re-entrants	5	0	1	3	0	0
Valleys	5	0	3	5	0	0
Basins	5	0	5	5	0	0
Bowls	3	0	3	5	0	0
Cirques	3	0	3	5	0	0
Draws	5	0	1	5	0	0
Gullies	5	0	1	5	0	0
Hanging-valleys	3	0	3	5	0	0
Map-Saddles	3	0	5	3	0	0
Cols	3	0	5	5	0	0
Passes	3	0	5	5	0	0
Protrusions	0	3	0	0	3	3
Buttes	0	3	0	0	3	3
Peak-primitives	0	3	0	0	3	5
Ridges	0	5	0	0	3	1
Buttresses	0	5	0	0	3	3
Shoulders	0	5	0	0	3	3
Spurs	0	5	0	0	3	3
Spires	0	3	0	0	3	5
Walls	0	3	0	0	5	0
Headwalls	0	3	0	0	3	0

match matrix between image and map features. Feature matches are ranked on a scale from 0 to 5, bad to good, where 0 indicates that a map feature can never appear as an image feature (for example, a gully in the map will never appear as a peak in the image), and 5 indicates a preferred match (for example, a peak in the image matches well with a peak in the map).

Reasoning is divided into task-specific subroutines and proceeds in the manner described in the approach section above. Components are high-level (strategic), and low-level (specific tasks). High-level components are the conscious reasoners of the system. They pick the strategic direction in which the system should proceed, initiate that work, evaluate its performance, and then choose the next strategic direction.

RESULTS

In LOCALE two types of heuristics were used. The first type of heuristic was the use of configurations. By considering features in groups instead of as individuals, search was limited to those features that were parts of appropriate groups. The second type of heuristic was the use of category limitations. Only map features of the appropriate type were considered for matching with the image features. In addition, matches were prioritized based on proximity in the feature hierarchy, so that stronger matches could be considered first. Each heuristic is useful, but the real power of this approach came from the combination of both heuristics. The result was to constrain the search space to

only those map features that were parts of appropriate configurations *and* were of the correct type to match with the image features. The effect of this is to determine the subset of features that meets the configuration constraints and to determine the subset of features that meets the category constraints, and then to take the intersection of those two subsets as the search space. We can quantify the benefits of this approach for an example problem. After three levels of map data detail and two levels of image data detail have been loaded into the system, there are thirty-seven map features and eight image features. The number of possible matches between these two sets is 6.48914×10^{16}. The power of this approach is that very few possible matches are actually considered and explored. Using the configuration heuristic, there are only 98 map configurations that match the current image configuration. Using the category heuristic, there are only 52 possible matches between the image features and map features that are constrained by the compatibility of their categories. Combining the results of those two heuristics, there are only twelve configuration-match hypotheses that can be developed into viewpoint hypotheses. This reduction of the search space is dramatic. Because this is a heuristic approach, its performance cannot be guaranteed in the same way an algorithm's performance can. The reduction in search depends on the uniqueness and identifiability of the feature categories and the availability of configurations; however, this magnitude of search reduction was consistently observed among all the test cases. Table 2 summarizes the state space reductions observed in both this and other test cases. The prospect of exploring and evaluating 10 to 20 test cases is reasonable. And, even if the correct solution is not always selected as the best alternative at any one time, the fact that it exists among the small, select set of alternatives is significant.

CONCLUSIONS

This work has analyzed the components of the localization problem. The solution of this problem is a critical component to future work on autonomous mobile robot systems like those proposed for missions such as the Mars rover/sample collector. Localization has the potential to become a computationally insurmountable problem. However, heuristic strategies for high-level control can be employed to combat this challenge. Two such strategies are the use of configurations of features to control feature matching and the use of category limitations. The LOCALE system has been implemented to demonstrate these strategies.

ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation grant IRI-9196146, with partial funding from the Defense Advanced Research Projects Agency.

Table 2. Comparison of Test Cases

Test Case	Number of Map Features	Number of Image Features	State Space	Viewpoint Hypotheses Explored
Moran	37	8	2.0×10^{12}	12
Teewinot	37	5	6.1×10^{7}	12
Bivouac	37	6	2.0×10^{9}	20

REFERENCES

1. B.H. Bennett. (1992). *A problem-solving approach to the localization problem.* PhD Thesis, University of Minnesota, Department of Computer Science.

2. S.L. Savitt, T.C. Henderson, and T.L. Colvin. (1992). Feature extraction for localization. *Proceedings of the DARPA Image Understanding Workshop.*

3. S.L. Savitt. (1991). *A context sensitive segmentation approach for outdoor terrain feature extraction.* PhD Thesis, University of Minnesota.

4. M.D. Ernst and B.E. Flinchbaugh. (1989). Image/map correspondence using curve matching. *Proceedings of the AAAI Robotic Navigation Symposium*, 23-30.

5. F. Stein and G. Medioni. (1991). Map-based localization using the panoramic horizon. Proceedings of the *8th Israeli Symposium on Artificial Intelligence and Computer Vision.*

6. M.A. Lavin. (1979). Analysis of scenes from a moving viewpoint. P. H. Winston and R. H. Brown (Eds.) *Artificial Intelligence: An MIT Perspective.* Cambridge: MIT Press.

7. G. Giralt, R. Chatila, and M. Vaisset. (1984). An integrated navigation and motion control system for autonomous multisensory mobile robots. *Proceedings of the First International Symposium on Robotics Research.*

8. R.C. Arkin, E.M. Riseman, and A.R. Hanson. (1987). AuRA—An architecture for vision-based robot navigation. *Proceedings of the DARPA Image Understanding Workshop*, 417-430.

9. C.L. Fennema, Jr., E.M. Riseman, and A.R. Hanson. (1988). Planning with perceptual milestones to control uncertainty in robot navigation. *Proceedings of SPIE - International Society for Photographic and Industrial Engineering.*

10. C. Fennema, D. Hanson, E. Riseman, J.R. Beveridge, and R. Kumar. (1990). Model-directed mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics,* 20(6), 1352-1369.

11. C. Fennema, and A.R. Hanson. (1990). Experiments in autonomous navigation. *Proceedings of the 10th International Conference on Pattern Recognition,* 24-31.

12. T.O. Binford. (1988). Image understanding: Intelligent systems. *Proceedings of the DARPA Image Understanding Workshop*, 17-28.

13. D.J. Kriegman, T.O. Binford, and T. Sumanaweera. (1998). Generic models for robot navigation. *Proceedings of the DARPA Image Understanding Workshop*, 453-460.

14. G.B. Smith, and T.M. Strat. (February 1987). Information management in a sensor-based autonomous system. *Proceedings of the DARPA Image Understanding Workshop*, 170-176.

15. K. M. Andress and A.C. Kak. (Summer 1988). Evidence accumulation and flow of control in a hierarchical spatial reasoning system. *AI Magazine*, 75-94.

16. L.S. Davis, D. Dementhon, R. Gajulapalli, T.R. Kushner, J. Le Moigne, and P. Vetach. (1987). Vision-based navigation: A status report. *DARPA Image Understanding Workshop*, 153-169.

17. C. Thorpe, M. Hebert, T. Kanade, S. Shafer, and the members of the Strategic Computing Vision Lab. (1987). Vision and navigation for the Carnegie-Mellon Navlab. *Annual*

Review of Computing Science (2), J.F. Traub, B.J. Grosz, B.W. Lampson, and N.J. Nilsson (Eds.). Palo Alto: Annual Reviews, Inc.

18. D.T. Lawton, T.S. Levitt, and P. Gelband. (May 1989). Knowledge-based vision for terrestrial robots. *Proceedings of the DARPA Image Understanding Workshop*, 128-133.

19. D.T. Lawton, T.S. Levitt, C. McConnell, and J. Glicksman. (April 1986). Terrain models for an autonomous land vehicle. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2043-2051.

20. D.T. Lawton, T.S. Levitt, C.C. McConnell, P.C.Nelson, and J. Glicksman. (1987). Environmental modeling and recognition for an autonomous land vehicle. *Proceedings of the DARPA Image Understanding Workshop*, 107-121.

21. T.S. Levitt, D.T. Lawton, D.M. Chelberg, and P.C. Nelson. (July 1987). Qualitative landmark-based path planning and following. *Proceedings of the National Conference on Artificial Intelligence AAAI-87.*

22. T. S. Levitt, D.T. Lawton, D.M. Chelberg, and P.C. Nelson. (1987). Qualitative navigation. *Proceedings of the DARPA Image Understanding Workshop*, 447-465.

23. T. S. Levitt, D.T. Lawton, D.M. Chelberg, K.V. Koitzsch, and J.W. Dye. (1988). Qualitative navigation II. *Proceedings of the DARPA Image Understanding Workshop*, 319-326.

24. W.B. Thompson. (September 1990). Vision-based navigation. *Proceedings of the DARPA Image Understanding Workshop*, 141-143.

25. W.B. Thompson, H.L. Pick, Jr., B.H. Bennett, M.R. Heinrichs, S.L. Savitt, and K. Smith. (September 1990). Map-based localization: The 'drop-off' problem. *Proceedings of the DARPA Image Understanding Workshop*, 706-719.

26. K. Smith, M.R. Heinrichs, and H.L. Pick, Jr. (August 1991). Similarity judgement and expert localization. *Proceedings of the Thirteenth Annual Conference of the Cognitive Society.*

27. M.R. Heinrichs, K. Smith, H.L. Pick, Jr., B.H. Bennett, and W.B. Thompson. (January 1992). Strategies for localization. *Proceedings of the DARPA Image Understanding Workshop.*

Parametric Motion Control of Robotic Arms: A Biologically Based Approach Using Neural Networks

O. Bock
Institute for Space and Terrestrial Science, and York University

G.M.T. D'Eleuterio
Institute for Aerospace Studies, University of Toronto

J. Lipitkas
Institute for Space and Terrestrial Science
and, Institute for Aerospace Studies, University of Toronto

J.J. Grodski
Defence and Civil Institute of Environmental Medicine

January 19, 1993

1 Abstract

A neural network based system is presented which is able to generate point-to-point movements of robotic manipulators. The foundation of this approach is the use of prototypical control torque signals which are defined by a set of parameters. The parameter set is used for scaling and shaping of these prototypical torque signals to effect a desired outcome of the system. This approach is based on neurophysiological findings that the central nervous system stores generalized cognitive representations of movements called *synergies, schemas*, or *motor programs*. It has been proposed that these *motor programs* may be stored as torque-time functions in central pattern generators which can be scaled with appropriate time and magnitude parameters. The central pattern generators use these parameters to generate stereotypical torque-time profiles, which are then sent to the joint actuators. Hence, only a small number of parameters need to be determined for each point-to-point movement instead of the entire torque-time trajectory. This same principle is implemented for controlling the joint torques of robotic manipulators where a neural network is used to identify the relationship between the task requirements and the torque parameters. Movements are specified by the initial robot position in joint co-

ordinates and the desired final end-effector position in Cartesian coordinates. This information is provided to the neural network which calculates six torque parameters for a two-link system. The prototypical torque profiles (one per joint) are then scaled by those parameters. After appropriate training of the network, our parametric control design allowed the reproduction of a trained set of movements with relatively high accuracy, and the production of previously untrained movements with comparable accuracy. We conclude that our approach was successful in discriminating between trained movements and in generalizing to untrained movements.

2 Introduction

An important problem in space robotics is point-to-point control of the robotic arm end-effector in an unstructured environment. Many attempts have been made to solve this problem: the usual methods are tedious and computationally intensive to solve in real-time, even with the most advanced computational methods ([4], [11], [13]). This paper introduces a different strategy based on motor control principles used by humans.

In many studies on human movements, consistent and stereotypical hand and joint trajectories have been observed across movement speeds, extents, directions, and external loads. Such findings support the notion that movements are controlled by prototypical *motor programs* which are stored in the central nervous system and scaled to fit the requirements of each particular movement task before playback [1], [2], [5], [7], [12], [15], [16]. In particular, it has been proposed that these motor programs may be stored as muscle force-time functions and that different movements along the same path, but with varying speed or paylod, can be executed by playing back those functions with appropriate time and magnitude scaling. Therefore, the human motor system replaces the explicit calculation of the entire muscle-force profile by the calculation of just a few scaling parameters which are used to control central pattern generators (CPG) where the motor programs are stored.

A problem emerging from the motor program concept is that, since an infinite number of possible movements exist, the nervous system must have some way to calculate or to look up an infinite number of possible scaling parameters. Recently, engineering solutions for similar problems have been introduced in the form of artificial neural networks (ANN's [14]). Essentially, an ANN consists of processing elements, interconnection topologies, and a learning algorithm governing the modification of connection strengths depending on mapping performance. Generally, an ANN allows the mapping of input values into output values based on previously established mapping rules. These rules are determined via a repetitive trial-and-error *learning* procedure rather than by explicit calculations. An important characteristic of ANN's is that once a correct mapping has been learned for a number of input values, the network can *generalize* and provide correct output values even for untrained input values. Thus the above problem of representing an infinite number of parameters can be overcome by using neural networks to find suitable solutions.

To summarize, control by motor programs appears to be potentially useful for manipulator control because the controller would only have to calculate a limited number of scaling parameters before movement onset rather than calculating the entire joint torque-time

profiles in real-time. This results in a robotic manipulator control system that can be referred to as a *Parametric Control System*, and is presented here as a means of controlling the joint torques of a two degree-of-freedom planar robotic manipulator. Furthermore, this approach is used in conjunction with a neural network which identifies the relationship between the task requirements and the torque parameters. Therefore, the approach presented here combines the motor program concept with neural networks to determine the joint torque-time functions necessary to drive a robotic manipulator end-effector from an initial to a desired final configuration.

3 Control Strategy

The control problem is to move a two-link planar robotic arm, as shown in Figure 1, from an initial position to a desired final position within the workspace shown. The robot manipulator control system, which was used, is designed to utilize the benefits of the motor program concept, and is illustrated in Figure 2. The adaptive controller is an ANN, trained to map inputs x_d, θ_a into outputs p. The parameters p are applied to a function generator which generates a prototypical time-function. This time-function is scaled by p to yield the force-time functions $T_d(t)$, one per joint, to be applied by the plant. In the work reported here, the plant is the *Radius* robotic manipulator at the University of Toronto Institute for Aerospace Studies [3]. This manipulator is a two degree-of-freedom planar manipulator with rigid links, where the links are supported by airjets in the horizontal plane. The airjets allow the *Radius* robotic manipulator to move in the horizontal plane without friction. The two joint actuators are harmonic-drive servomotors with the joint position θ_a being measured by precision potentiometers.

The ANN was implemented using the structure shown in Figure 3. Each neuron is a logistic unit having a working range of - 1.0 to + 1.0 with all of the neurons being fully forward-connected. Inputs to the ANN structure are x_d, the two Cartesian coordinates of the desired final gripper position, and θ_a, the actual initial angles of both joints, with θ_a and x_d being sampled once before a movement.

The input signals x_d and θ_a pass through a layer of 127 coarse-coding neurons [6] (each neuron being tuned to a range of input values with overlapping ranges for neighboring neurons), then through two hidden layers of 20 units per layer (the first layer containing 20 neurons and the second layer containing 20 neurons) and finally through a layer of six output neurons. The output signals provided by the last layer represent the values of the six parameters p which were previously described. Three of these parameters are used for each joint, p_1 to p_3 for the *shoulder* joint and p_4 to p_6 for the *elbow* joint.

The parameters p serve as inputs to the Function Generator (Figure 2), which in turn provides two output signals $T_d(t)$, one for each joint, which are applied to the plant. Both output signals are triggered synchronously when p changes after a new x_d has been entered, and each output signal consists of two successive sinusoidal half-waves having an overall duration of 4 sec. Figure 4 illustrates that p_1 and p_4 represent the percentage of movement time taken by the first lobe of the two torque profiles, one for each joint, and p_2, p_3, p_5, and p_6 represent the maximum torque amplitudes for each lobe of the two torque profiles.

The function of the ANN is, essentially, to map four discrete input signals x_d, θ_a for the two joints into six discrete output signals p_1, p_2, and p_3 (the torque parameters for the *shoulder* joint) and p_4, p_5, and p_6 (the torque parameters for the *elbow* joint). Only a single mapping action per movement is needed. The modifiable ANN weights are adjusted in order to achieve satisfactory mapping by a modified version of Direct Inverse Modeling [8], a known training procedure.

In this modified Direct Inverse Modeling training procedure, the initial *Radius* joint positions θ_a are first noted and an operator then moves the gripper into a selected final position x_s along an approximately straight path with an approximately bell-shaped, single-peaked velocity profile of 4 second duration. The joint trajectories $\theta(t)$ during that movement are recorded on a disk and subsequently transformed into predicted joint torque profiles $T_p(t)$ using *Radius's* Inverse Dynamics equations. Next, predicted joint torque profiles $T_p(t)$ are approximated by two sinusoidal half-waves of variable relative duration and amplitude and the corresponding parameters p are noted. Then, *Radius* having been reset to θ_a, p is provided as inputs to the function generator which supplies outputs $T_d(t)$ to the actuators in order to drive *Radius* to a final position noted as x_d. Since the parameterization is only an approximation, $T_d(t)$ and $T_p(t)$ will be somewhat different and x_d will be somewhat different from x_s.

The noted values of x_d, θ_a and p characterize one movement of a training set. The above steps are repeated for 225 different movements of various amplitudes and directions within the workspace shown in Figure 1 to yield a set of 225 training movements characterized by their respective values of x_d, θ_a and p.

Training of the ANN commences by initializing the modifiable weights with random values. Then x_d and θ_a of the training set are used as the ANN inputs and the corresponding outputs p are recorded. The difference between p as calculated by the ANN and the corresponding p as noted for the training set is the ANN performance error and is used for incremental weight changes according to the backpropagation rule. ANN performance is considered satisfactory when the output values p_1 to p_6, which are applied to the function generator, result in a gripper movement to the desired final position x_d such that $x_a \approx x_d$.

4 Results

An illustrative representation of network performance is given by Figure 5, where the final position error of the end-effector is plotted. The errors are coded as lines from the actual final position to the desired final position. Performance before training is shown in Figure 5A, and after training (10,000 iterations) in Figure 5B. As can be seen, the errors between the desired and actual final end effector positions are greatly reduced. In fact, the average error drops from 0.75 m before training, to 0.03 m after training: in comparison, the robotic arm is 2.12 m long. Therefore, the error after learning was almost an order of magnitude smaller than the inter-target distances which ranged from 0.1 m to 0.85 m. Thus, the system was able to discriminate between targets. Figure 5C shows the final position errors of the trained neural-network controller using a set of movements that were not previously trained. As can be seen, the average final position error of 0.07

m was slightly higher than the trained data set, but was again less than the shortest intertarget distance. Therefore, generalization within the workspace was successful.

5 Conclusions

We have described a solution to the control of point-to-point movements of a two joint planar robotic arm. This parametric control concept is qualitatively different from traditional approaches described earlier. Instead of explicitly calculating the torques for the entire trajectory, the new concept specifies only a limited number of characteristic parameters. In addition, the control system presented in this paper provides the following advantages over most other known types of systems:

1. No explicit knowledge of the manipulator's dynamics is required.

2. The nonlinear (ANN) stage is not in a control loop which will avoid any problems due to computational delays of the type generally caused by nonlinear stages.

3. The ANN can be easily retrained for different robotic manipulators and/or changing robot dynamics.

4. The design of a controller for a multilink robotic manipulator with $n > 2$ is not qualitatively different than that described in this paper since the nonlinear stage is designed by trial-and-error rather that by an analytical solution.

In addition, our control concept discriminates between targets, and generalizes to unlearned target positions. This parametric control should be particularly useful for real-time robot control in unstructured environments since only a limited number of variables need to be updated, therefore placing less of a computational burden on the controller. Moreover, our control concept may be improved to achieve a more accurate terminal approach to the targets by the addition of sensory feedback, as found in humans. Also, this concept could be easily expanded to allow for velocity control by direct scaling of the torque profiles, and better control of movement paths could be achieved by adding more parameters (p_i).

6 Acknowledgments

We gratefully acknowledge the financial support provided by the Defence and Civil Institute of Environmental Medicine under contract W7711-0-7118. Further support was provided by the Institute for Space and Terrestrial Science, and the University of Toronto Institute for Aerospace Studies. In addition, we would like to thank the many employees and students in the Space Robotics Laboratory at the University of Toronto Institute for Aerospace Studies for their help.

References

[1] Atkeson C.G., Hollerbach J.M. (1985). *Kinematic features of unrestrained vertical arm movements.* J. Neurosci 5, 2318-2330.

[2] Bock O. (1990). *Load Compensation in human goal-directed arm movements.* Behav Brain Res 41, 167-177.

[3] Carusone J., D'Eleuterio G.M.T. (1991). *Experiments in the control of structurally flexible manipulators with the Radius facility.* Proc. of the Second Joint Japan/U.S. Conference on Adaptive Structures, Nagoya, Japan, 589-605.

[4] Freund E. (1982). *Fast Nonlinear Control with Arbitrary Pole-Placement for Industrial Robots and Manipulators.* Int J Robotics Res 1,65-78.

[5] Gordon J., Ghez C. (1987). *Trajectory control in targeted force impulse. II. Pulse height control.* Exp Brain Res 67, 241-252.

[6] Hinton G.E., McClelland J.L., Rumelhart D.E. (1986). Distributed Representations. In Rumelhart D.E., McClelland J.L. (Eds). *Parallel Distributed Processing: Vol 1* (pp 77-109). Cambridge:MIT Press.

[7] Hollerbach J.M., Flash T. (1982). *Dynamic interactions between limb segments during planar arm movements.* Biol Cybernetics 44, 67-77.

[8] Jordan M.I., Rumelhart D.E. (1991). *Forward models: Supervised learning with a distal teacher.* Submitted to Cognitive Science.

[9] Lacquaniti F., Soechting J.F., Terzuolo C.A. (1982). *Some factors pertinent to the organization and control of arm movements.* Brain Res 252, 394-397.

[10] Lipitkas J. (1992). *Organizational Principles Underlying Movements of the Upper-Limb.* unpublished Ph. D. thesis, Univ. of Toronto.

[11] Luh J.Y.S., Walker M.W., Paul R.P.C. (1980). *Resolved-Acceleration Control of Mechanical Manipulators.* IEEE Trans on Automatic Control 25, No. 3, 468-474.

[12] Meyer D.E., Smith J.E.K., Wright C.E. (1982). *Models for the speed and accuracy of aimed movements.* Pyschol Rev 89, 449-482.

[13] Paul R.P. (1981). *Robot Manipulators: Mathematics, Programming, and Control.* MIT Press, Cambridge, Mass., USA.

[14] Rumelhart D.E., McClelland J.L. (1986). *Parallel Distributed Processing.* Cambridge:MIT Press.

[15] Schmidt R.A., Zelaznik H.N., Hawkins B., Frank J.S., Quinn Jr. J.T., (1979). *Motor output variability: a theory for the accuracy of rapid motor acts.* Psychol Rev 86, 415-451.

[16] Sherwood D.E., Schmidt R.A., Walter C.B. (1988). *Rapid movements with reversals in direction II: Control of movement amplitude and inertial load.* Exp Brain Res 69, 355-367.

[17] Stein R.B., Cody F.W.J., Capaday C. (1988). *The trajectory of human wrist movements.* J Neurophysiol 59, 1814-1830.

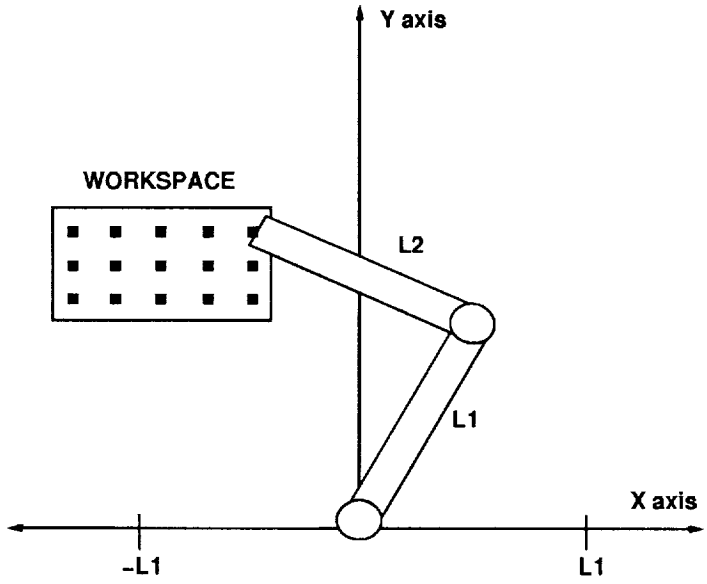


Figure 1: Two-link planar manipulator and workspace (L1 and L2 are the link lengths of the first and second links, where L1 = L2 = 1.06 m).

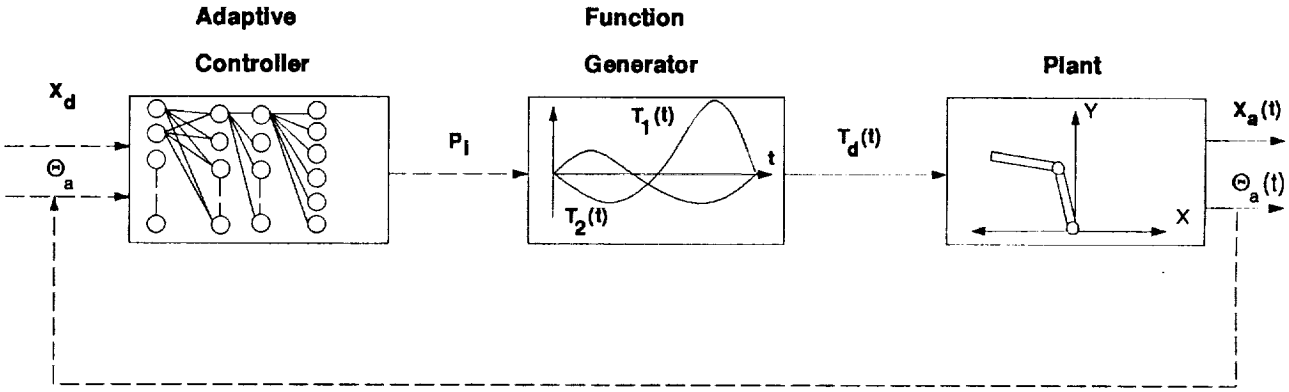


Figure 2: Block diagram of the control system utilized, where solid lines represent time varying actions and hatched lines represent a single mapping actions per movement (X_d and X_a are the desired and actual end-effector positions, θ_a the initial robot configuration in joint coordinates, $P_i's$ are the torque scaling parameters, $T_1(t)$ and $T_2(t)$ are the joint torque-time profiles for the shoulder and elbow joints, and $T_d(t)$ represents the input torques to the plant).

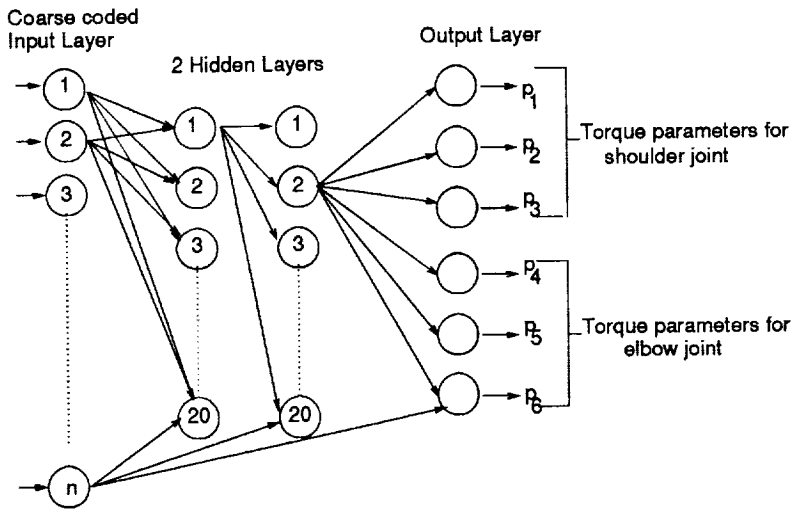


Figure 3: Artificial neural network architecture used in the simulations reported here (n = 127). All neurons are fully forward- connected to the neurons in the layers in front.

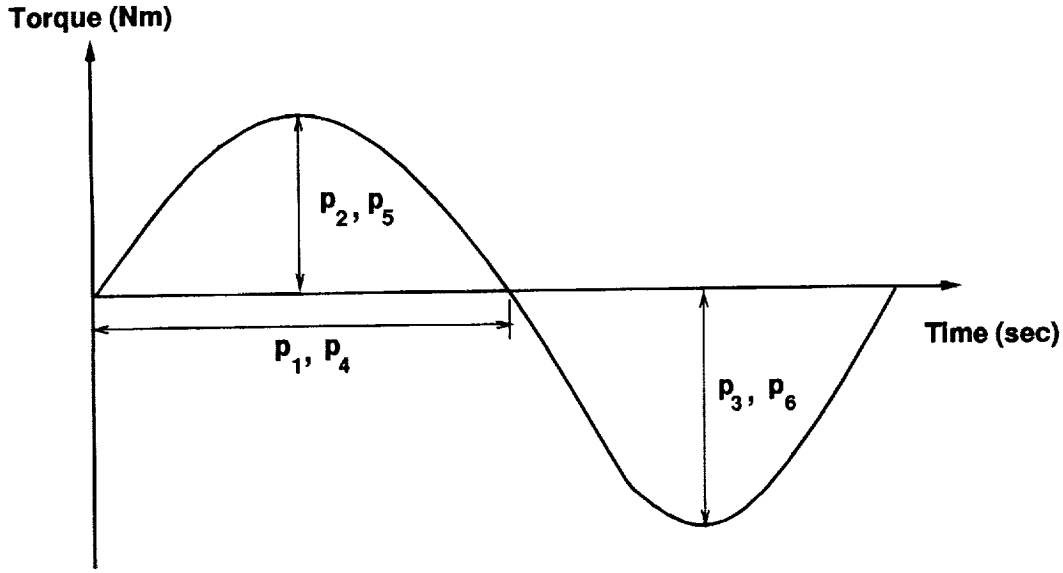


Figure 4: Torque parameterization scheme employed. Where p_1, and p_4 are the time of switching from the first lobe to the second lobe for torque profiles T_1 and T_2 respectively, p_2 and p_5 are the amplitudes of the first lobe, and p_3 and p_6 are the amplitudes of the second lobe for torque profiles T_1 and T_2.

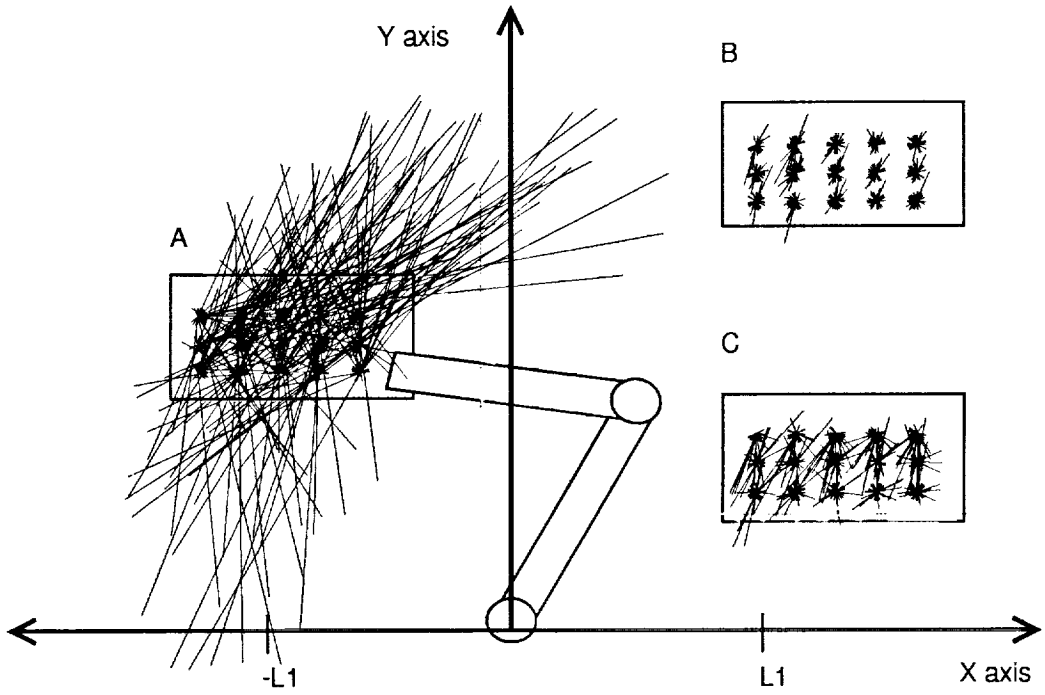


Figure 5: Graphical illustration of the final position errors from the actual to the desired final end-effector positions (A - final position errors before training, B - final position errors after training, for same workspace as A, C - final position errors for an untrained data set, for same workspace as A also).

Detection of Bearing Failure in Mechanical Devices Using Neural Networks

Richard A. Burne, Paul F. Payer, R. Paul Gorman, Dan T. Horak
Microelectronics and Technology Center
AlliedSignal Aerospace Company
Columbia, MD 21045

Abstract

We present a novel time-domain method for the detection of faulty bearings that has direct applicability to monitoring the health of the turbopumps on the Space Shuttle Main Engine. A feed-forward neural network was trained to detect modelled roller bearing faults on the basis of the periodicity of impact pulse trains. The network's performance was dependent upon the number of pulses in the network's input window and the signal-to-noise ratio of the input signal. To test the model's validity, we fit the model's parameters to an actual vibration signal generated by a faulty roller element bearing and applied the network trained on this model to detect faults in actual vibration data. When this network was tested on the actual vibration data, it correctly identified the vibration signal as a fault condition 76% of the time.

1.0 Introduction

A critical aspect of the Space Shuttle Main Engine (SSME) as a reusable space vehicle is the durability of its components. One major inadequacy has been the insufficient life of the bearings in the SSME's turbopumps. The life expectancy of the turbopump was designed to be 55 missions, but actually the pumps require an overhaul every one to three missions As a result, a significant ground test program has been required to provide "flight-qualified" turbopumps. One means of reducing the cost associated with ground testing is to provide a preflight, non-invasive monitoring procedure that can detect subtle bearing failures without requiring the firing of the SSME. This paper describes a novel bearing failure detection technique that is suitable for preflight inspection of SSME components.

The most common failure modes of rolling element bearings are local defects in the outer race, the inner race, or a rolling element. As the bearing rotates, whenever the defect passes through the element-to-race contact area, a short duration impact is generated that can be detected by accelerometers or acoustic emission sensors mounted near the bearing. A typical accelerometer signal generated by a faulty bearing is shown in Figure 1. This signal is characterized by transient events caused by bearing imperfections. These transients occur against a background of minute transients whose sum is approximated well by a Gaussian distribution. The fault transients typically exhibit a quasi-periodicity governed by the rotational speed and the bearing geometry[1]. The interval between such transients is typically much longer than the duration of the transient itself. Such impact transients have been recorded from SSME turbopumps using acoustic emission sensors[2]. Because the structure of each fault transient is generally random, the challenge associated with the early detection of bearing faults is to detect the fault transients' periodicity.

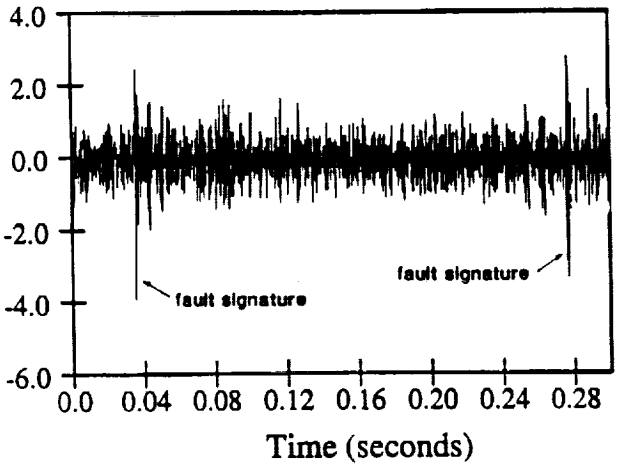


Figure 1 A typical faulty vibration signal. Impact transients are indicated.

In the past, spectral analysis had been used to analyze the acceleration signals from faulty bearings[1]. The basis for this analysis is that the frequency corresponding to the spacing of the fault transients shows up as a peak in the frequency domain. The problem with this method is that the noise in the interval between fault transients tends to dominate the power spectrum due to the temporally local nature of the fault transients. Consequently, only large bearing faults can be detected using this method. This has been shown to be the case for detecting turbopump bearing cage failures. A severely damaged turbopump bearing exhibits peaks in the accelerometer signal's power spectrum at 214 Hz and 428 Hz. These are the primary and secondary harmonics of the bearing cage at 104% of the turbopump's rated power level. Even under severe fault condidions, these spectral peaks cannot be reliably detected.

More recently, time-frequency methods, such as wavelet transforms, have been applied to transient signals in an attempt to address the averaging problems associated with Fourier techniques[3]. Such techniques have proven quite useful in characterizing temporally local events. However, due to the local nature of their basis functions the time-frequency techniques are inappropriate for detecting periodicity in the signal.

As an alternative to spectral and time-frequency techniques, we propose to use time-domain analysis as a means of detecting the inter-pulse interval associated with bearing fault transients.

The simplest time-domain algorithm for detection of pulse trains in the presence of noise is averaging of points in the vibration signal which are one period of the pulse train apart[4]. This averaging enhances the pulses by reducing the effect of random background noise. While this method is effective for enhancing deterministic signals in rotating machinery, such as cylinder pressure in internal combustion engines, its use is limited in the bearing fault detection application. The main problem is that the pulses are random, being a sum of stress waves that reach the sensor through multiple paths between the bearing contact points and the sensor mounting point. The rolling element-to-race impact that produces the pulse is itself random, being dependent on the exact orientation and vibrations of all the bearing components at that instant. Consequently, the pulse spacing is not exactly constant and no two pulses have the same shape. As the result of the randomness, the averaging process attenuates the pulses just as it attenuates the background noise, and does not improve significantly the detectability of the fault.

Another practical limitation of the averaging method is the need for accurate data alignment during the averaging process. In the engine cylinder pressure example, a

one-per-revolution signal is sufficient for perfect alignment of signals of any length. In the bearing case, the frequency of the pulses is a product of the rotational speed and a geometric constant. Therefore, a one-per-period signal cannot be generated, making it impossible to align long records accurately.

The contribution of this paper is the use of a feed-forward neural network as an alternative time-domain detection method for pulse trains generated by faulty bearings. The three main features of our method overcome the limitations of the averaging method. First, the fault transients are not required to possess a specfic structure. Second, there is no need for data alignment. Finally, our algorithm can tolerate moderate variations in pulse spacing. In summary, our method can detect pulse trains in noise without excessive sensitivity to the features and repeatability of the pulses.

The next section details the signal model used to train the neural network. Section 3.0 describes the neural network experiments conducted on the modelled data. Section 4.0 provides the results of those experiments and Section 5.0 presents the results of an experiment applying the network detector to actual vibration data generated by a faulty roller bearing. This data was collected from a simple test device. Finally, Section 6.0 discusses the conclusions drawn from these results.

2.0 Vibration Signal Model

To develop the neural network-based fault detector, we modelled the vibration signal generated by a faulty bearing as a pulse train embedded in Gaussian noise. The pulse train possessed a specific periodicity. These pulse train signals generated using various signal-to-noise ratios (SNRs) were used to train the neural network. Once trained, the neural network was tested using actual vibration data collected from a faulty ball bearing in our laboratory.

In order to train a neural network to serve as a generic fault detector for rolling element bearings, a general signal model was developed. Faulty vibration signals are characterized by quasi-periodic impact transients. Figure 1 shows a faulty vibration signal with two impact transients indicated. We were interested in a signal model that provided only quasi-periodicity information as a classification cue. Therefore, we used the same Gaussian statistics to generate both the pulses and the background noise. The only difference between a pulse and noise was the mean amplitude of their respective distributions.

Two classes of vibration signal were generated. The first class of signals possessed pulses whose inter-pulse interval was random (uniform distribution between zero and twice the mean interval). The second class was designed to represent a vibration signal generated by a faulty bearing. This signal possessed a pulse train that exhibited a quasi-periodicity (a Gaussian distribution with a variance equal to 20% of the mean inter-pulse interval). The pulse width to inter-pulse interval ratio was 0.22 and the position of the initial pulse was chosen randomly (a uniform distribution between zero and the mean inter-pulse interval).

The signal-to-noise ratio of a model signal was computed as follows

$$SNR = 10\log\frac{A_S^2}{A_N^2} \qquad (1)$$

where A_S and A_N are the means of the Gaussian distributions used to generate the

pulse and the noise, respectively. The final signal was generated by adding the Gaussian noise to the pulse train signal.

3.0 Neural Network Training

Feed-forward neural networks with two layers of modifiable weights were used throughout the study. Two sets of experiments were conducted. In the first experiment, we trained the network on input signals containing an average of ten pulses. In the second experiment, the network was trained on signals containing only three pulses on average. The network trained on ten pulses contained 192 input units, 20 hidden units, and 2 output units. The network trained on three pulses contained 63 input units, 10 hidden units, and 2 output units. The size of the input layer was determined by the number of signal sample points required to provide the appropriate number of pulses to the network. The number of hidden units was chosen to provide a sufficient number of degrees of freedom to solve the classification problem. In both cases, the desired output was [1 -1] for the good bearing and [-1 1] for the faulty bearing.

The network was trained using the back-propagation learning algorithm[5]. The learning rate and the momentum were set at 0.01 and 0.9, respectively. These values provided the best convergence and training rates for the two networks.

During training, network weights were adjusted so as to minimize the difference between desired and actual output values. Each pattern presented to the network was generated at the time of presentation and, therefore, the network never saw the same pattern twice. Training continued until the average improvement in performance over weight updates fell below a fixed threshold. This is termed asymptotic performance.

4.0 Network Performance

For the ten-pulse and the three-pulse networks, training was conducted on signals with a given SNR. Performance figures for each network was obtained for various SNR values. Table 1 provides asymptotic performance levels for the ten-pulse network at the SNRs indicated. Although the SNR for the last two experiments was less than zero, the average amplitude of the pulse was larger than the background noise as a result of adding the pulse vector to the noise vector to produce the final signal.

Ten-Pulse Neural Network	
SNR [dB]	Performance [%]
20.0	87.7
14.0	86.4
6.0	81.9
0.0	80.2
-3.5	79.6
-6.0	54.6

Table 1 Detection performance for ten-pulse network (% correct classification).

As can be seen from the performance values, the network's ability to detect the quasi-periodic pulse train degrades less than 10% as the SNR is decreased from 20.0 dB to -3.5 dB. However, at an SNR of -6.0 dB the performance falls to near chance. This represents a precipitous drop in performance below an SNR of -3.5 dB.

Table 2 presents performance values for the three-pulse network. This network

consistently performed 10% to 15% below the ten-pulse network. This indicates that

Three-Pulse Neural Network	
SNR [dB]	Performance [%]
20.0	75.4
14.0	76.5
6.0	74.0
0.0	69.7
-3.5	65.0
-6.0	60.0
-12.0	53.0

Table 2 Detection performance for three pulse network (% correct classification).

the network performs better as a function of the number of pulses within its input window, as expected.

5.0 Vibration Data

To test the model against actual vibration signals, we obtained vibration data from a faulty ball bearing. The data was acquired from an accelerometer mounted on the bearing housing which held the outer race. The inner race of the bearing was mounted to a rotating shaft which was driven by an electric motor. The bearing was disassembled and the outer race was damaged with a grinding tool. During data collection, the shaft was rotated at a constant RPM and the vibration signal was digitized and recorded by a personal computer equipped with an A/D converter.

The faulty vibration signal exhibited quasi-periodic pulses similar to the model signal. The frequency of the pulse train was proportional to the RPM of the rotating shaft. We estimated the signal-to-noise ratio of the vibration signal to be 12.5 dB. We computed this value by measuring the mean amplitude of the signal within the inter-pulse interval which we used as the mean of the noise. We then measured the mean amplitude of the pulses and subtracted the mean amplitude of the noise to obtain the mean amplitude of the signal. We then used Equation 1 to compute the signal-to-noise ratio.

A model of the vibration signal was developed by fitting the parameters governing the periodicity of the modelled fault signal to the actual statistics of the vibration signal. In this case the pulse width to inter-pulse interval ratio was 0.054 which was a factor of 4 times smaller than the original model. The variance in the interval between each pair of pulses was a 20% of the mean inter-pulse interval. This value was used previously to generate the modelled signal for the simulation experiments described above.

A three-pulse neural network was trained on model signals as described above. This network achieved an asymptotic performance of 89% correct classification on the modelled data. The network weights were then fixed and tested by presenting the network actual vibration data obtained from the faulty roller element bearing. The network classified the fault signal as a fault 76% of the time.

6.0 Discussion

A feed-forward neural network was trained to detect modelled roller bearing faults on the basis of the quasi-periodicity of impact

pulse trains. The network's performance was dependent upon the number of pulses in the network's input window and the signal-to-noise of the input signal. To test the model's validity, we fit the model's parameters to an actual vibration signal generated by a faulty ball bearing. We then applied the network trained on this experimental model to the detection of faults in an actual vibration signal.

The performance of the three-pulse network trained on the modelled signal whose parameters were fit to actual vibration signal statistics performed much better than the three-pulse network trained during the original set of experiments on signals with the same SNR. This is accounted for by the difference in the pulse width to inter-pulse interval ratio between the two cases. In the simulation, model the ratio of the pulse width to the interval between the pulses was four times as large as the same ratio derived from the actual vibration signal. Therefore, the percentage of confusable patterns generated using random inter-pulse intervals was significantly larger for the simulation model.

However, the performance of the same network applied to the actual vibration signal was much closer to the performance of the network trained on the simulation model. This suggests that perhaps the actual variance in the inter-pulse interval exhibited by the actual vibration data should have been measured and used as a model parameter in the experimental model. In any case, the differential in classification performance on the modelled and the actual signal data suggests that a more accurate signal model is required.

It should be pointed out that the performance figures presented in this study were obtained by requiring the neural network to make a decision based on a very small portion of the signal. In the case of the actual vibration signal, the network's decision was based on a signal segment only 3.75 ms in length. We could improve the performance of a neural network-based fault detector significantly by using a time-delay neural network which would allow us to scale the amount of information available to the network a couple orders of magnitude.

The current model completely ignores any characteristic structure of the impact pulses. This was done to ensure that the network detector would be applicable for a variety of bearing faults and systems being monitored under various environmental conditions. However, if the application were restricted sufficiently to allow the use of characteristic impact pulse features, a second neural network could be used to extract such features allowing the detection of faults at much lower SNRs. The capability of neural networks to detect transients in noise was demonstrated in a previous paper[6]. This work showed that a neural network trained to detect a transient with specific structural characteristics consistently out-performed a matched filter designed for the same purpose.

In future work, we plan to apply this technique to the monitoring of bearing failure for the Space Shuttle Main Engine turbopumps. This application would allow us to monitor the system under controlled conditions ensuring that the RPM of the pump was held to a fixed value. However, in some practical applications, where the RPM of the rotating shaft could vary widely, it would be necessary to either restrict the range of RPMs monitored by a neural network fault detector or use a bank of network detectors each tuned to detect faults in a specific RPM range. This is due to the fact that the network cues on periodicity

information which must be restricted to a finite range in order to distinguish a periodic pulse train from random pulses.

References

1. Taylor, J. I., "Identification of Bearing Defects by Spectral Analysis," ASME Journal of Mechanical Design,Vol. 102, April 1980, pp. 199-204.

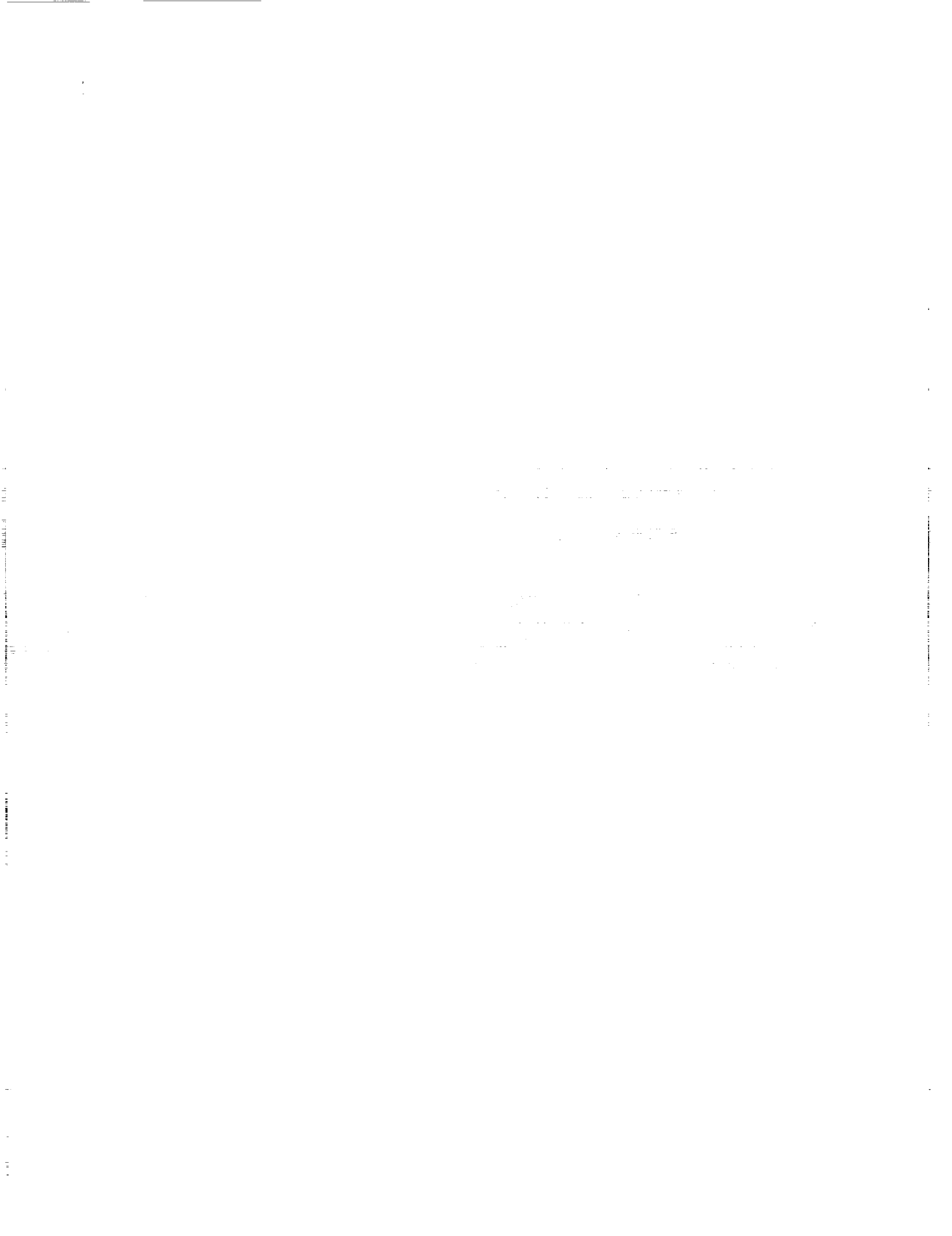
2. Hawman, M. and Galinaitis, W., "Acoustic Emission Monitoring of SSME-ATD Roller Bearings", AIAA 89-2849, 25th Joint Propulsion Conference, Monterey, CA., July 10-12, 1989.

3. Friedlander, B., and Porat, B., "Performance Analysis of Transient Detectors Based on a Class of Linear Data Transforms," IEEE Trans. on Information Theory, Vol. 38:2, pp 665-673, 1992.

4. Braun, S. G., and Seth, B. B., "On the Extraction and Filtering of Signals Acquired from Rotating Machines," Journal of Sound and Vibration, (1979), 65(1), pp. 37-50.

5. Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," in Parallel Distributed processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press Cambridge, MA, 1986.

6. Burne, R. A., Baindur, S., Gorman, R. P., "Detection of Transients in Gaussian Noise a Using Neural Network," to be submitted to IEEE Trans. Aerospace and Electronic Systems.



Machine Learning Techniques for Fault Isolation and Sensor Placement

James R. Carnes
Advanced Computing Laboratory
Boeing Defense and Space Group
Huntsville, AL 35824
ray@hsvaic.boeing.com

Douglas H. Fisher
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
dfisher@vuse.vanderbilt.edu

November 16, 1992

Abstract

Fault isolation and sensor placement are vital for monitoring and diagnosis. A sensor conveys information about a system's state that guides troubleshooting if problems arise. We are using machine learning methods to uncover behavioral patterns over snapshots of system simulations that will aid fault isolation and sensor placement, with an eye towards minimality, fault coverage, and noise tolerance.

1 Introduction

Accurate and timely fault diagnosis is critical in the life cycle of many physical systems. Seemingly minor faults can, if unremedied, lead to catastrophic faults that disable a system permanently. To identify faults, (human or machine) diagnosticians observe the system's behavior primarily through sensor readings. Sensors should generally be selected to be maximally informative about the state of the system. In the best of all possible worlds, we might expect that sensors should be placed on all measurable quantities of a system; anomalous values on one or more sensors could then readily identify the presence of and help isolate system faults. However, costs are associated with sensors. These costs correspond to actual monetary cost as well as costs due to the physical design constraints of the system such as power, mass, and volume which are at a high premium in systems such as Space Station Freedom. In addition, increased numbers of sensors introduce more information that an operator must attend to; too many sensors can lead to information overload, thus actually contributing to a degradation in (human) diagnostic performance.

In many cases it is neither feasible nor desirable to measure all quantities of a system. Thus, the diagnostician must interact with the system in two other ways: *probing* and *testing*. One can think of probing as sensing a quantity dynamically to determine its value at a particular point in time. In testing we examine component output quantities while systematically varying its inputs.

Probing and testing increase the cost (e.g., time) of diagnosis and may even be impossible on remote systems such as unmanned spacecraft. Moreover, probing and testing are only initiated when there is some indication of a fault. Thus, we would like to judiciously place sensors so that they indicate the existence of faults and focus attention on their plausible causes.

Sensor placement is the task of determining a set of sensors which allows the most accurate determination of the overall state of a monitored system while minimizing costs relating to the number of sensors, power consumption, cost, and weight. Reducing these quantities is particularly important in space platforms due to power and space restrictions. In response, we are using two machine learning methods to identify categories of system behavior that are similar in terms of measurable quantities. In this paper we describe the specific methods used and analyze their results. As we will illustrate, these results can be exploited for purposes of diagnosis and design for diagnosability, notably sensor placement.

We describe a methodology for applying inductive learning systems to the discovery of 'rule bases' for diagnosis. Our primary reason for doing so is to facilitate system design. In particular, rules suggest measurable quantities that are most diagnostic. Given a suitable tradeoff between coverage, accuracy and sensor cost, we envision a tool that aids system designers in sensor selection. We are currently in the process of systematically exploring the interaction between these factors in the context of two learning systems, Quinlan's C4.5 [13] and Fisher's COBWEB [6], with a longer-term goal of developing objective function(s) that reflect such a tradeoff.

2 Supervised Learning Approach

Supervised learning systems discover rules that characterize preclassified observations. For example, supervised machine learning systems are used in medical diagnosis; given patient case histories that record features such as gender, age, aspects of medical history, and a variety of test results, as well as a diagnosis provided by a physician, a supervised system discovers rules that are consistent with the physician-supplied diagnoses. We can also use this technology for purposes of fault diagnosis. In particular, consider the model of a thermal subsystem given in Figure 1.

We have used the following strategy to learn rules that distinguish a variety of conditions that can cause anomalous behavior in this system.

[1] Specify a simulator that represents each major system component as a function that maps component inputs to outputs. Simulation using a model-based methodology similar to Kuipers' [10] begins with an initial state of system parameter settings and propagates parameter changes through component functions until the simulator converges on a steady state.

[2] Associated with each system component are permissible parameter (continuous and discrete) ranges, within which the component is assumed to operate satisfactorily. Initial simulator parameters are systematically perturbed beyond extreme ends of these ranges for each component, thus yielding conditions under which the system is liable to malfunction.

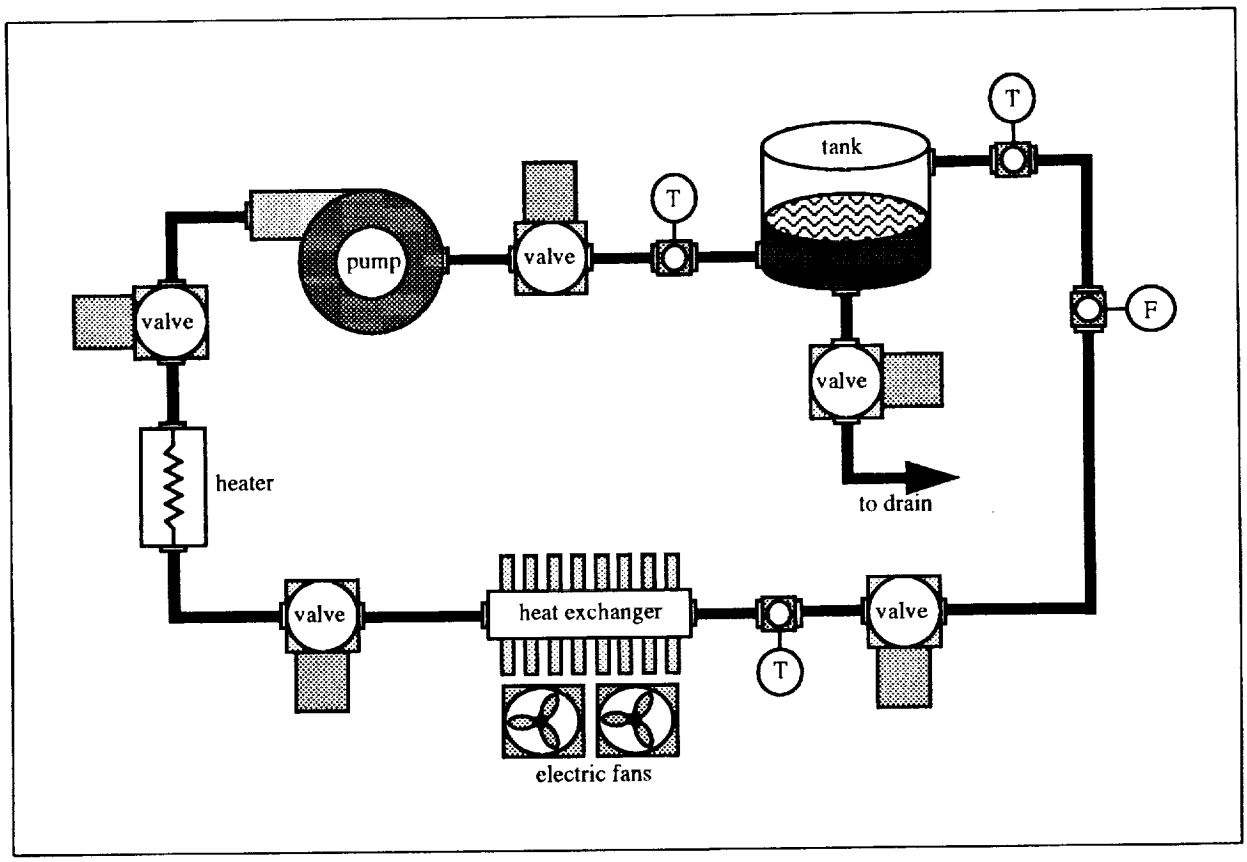


Figure 1: A thermal model.

[3] Each condition set generated in step [2] is propagated through the system until a steady state (or some error condition) is reached. A database record (which consists of measurements from each observable parameter in the system, labeled by the initial perturbed condition) is generated.

[4] The system state descriptions of all simulations are collected together and passed to a supervised learning system.

[5] The learning system forms a decision tree, then extracts rules that distinguish anomalous behaviors that were caused by different parameter perturbations.

We have used a supervised learning system known as C4.5 to form a diagnostic rule base. C4.5 has separate programs that (1) construct a decision tree and (2) form a rule base. In particular, C4.5 was used to discriminate the system perturbations ('faults') generated in step [2] of the simulation/learning procedure outlined above. Our thermal model contained a total of 87 fault types. In addition, three versions of each perturbation type were generated, corresponding to cases where the selected parameter value was perturbed just above (or below) acceptable ranges, moderately out of range, and far out of range. Intuitively, these corresponded to conditions of high (low), very high (low), and extremely high (low) values, but each case was labeled by a single fault (e.g., the parameter was 'above acceptable range'). Thus, the decision tree

had to distinguish 87 'faults', derived from over 261 observation sets (snapshots). Each snapshot was represented by 23 system parameter values. Using C4.5, we constructed decision trees much like the one partially shown in Figure 2.

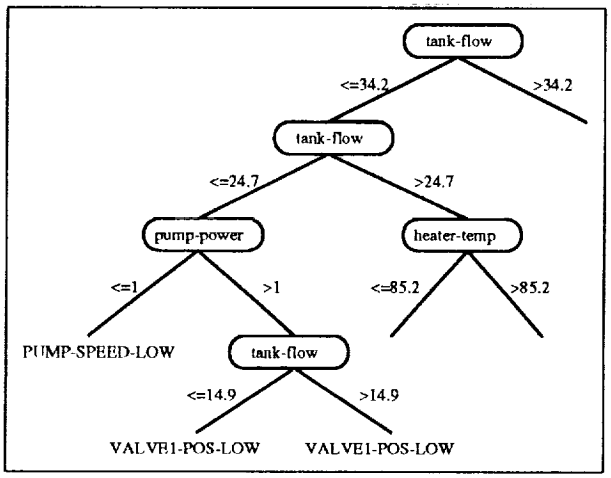


Figure 2: A partial decision tree over anomalous behaviors.

Initially, we are interested in two items: (1) the diagnostic accuracy of this tree, if we insist that faults must be perfectly isolated, and (2) how much the tree 'compresses' the parameters needed to attain a desired accuracy. We call this second factor the *parameter compression ratio.*

In this example, the decision tree correctly and uniquely classified 73% of the snapshots over which it was constructed. Note that the failure to perfectly classify all known behaviors is the result of C4.5's information-theoretic measure which could not reliably distinguish certain behaviors with the existing observable parameter values. These points of ambiguity are precisely where system designers should focus sensor placement efforts in order to better distinguish faults. It required that approximately 18 of the 23 parameters be consulted in order to achieve this accuracy – a parameter compression ratio of $(23 - 18)/23$ or 0.22.

The statistics above reflect a bias that the decision tree (or any rule-based system for that matter) should not attempt to perfectly isolate a fault. However, we can relax the diagnostic task, and allow categorization to identify an observation's fault as one of a small number of possibilities. The tree above will correctly identify each observation as exhibiting 1 of at most 3 fault possibilities (pump-speed-low, valve1-pos-low, valve1-pos-high) in 100% of the cases. Thus, we are are interested in the degree to which the tree isolates a fault. In this case, our minimal *fault compression ratio* is $(87 - 3)/87$ or 0.97.

Three aspects of this inductive analysis are of interest. Each of these speaks to the success of the diagnostic task, and provides guidelines for fault isolation and sensor placement. Our particular concern in this latter regard is with sensor placement.

- The *fault compression ratio* tells us the degree to which a behavior's fault can be isolated using the rule base. Inversely, it is a measure of the extent that we will have to rely on other sources of knowledge and diagnostic procedures, such as an expert or system simulation in conjunction with model-based diagnosis, to discriminate the fault from the reduced set of possibilities.
- The *parameter compression ratio* indicates the proportion of system parameters that need to be accessed for diagnosis over a population of behaviors. This is a guide to the number of sensors that will be required if diagnosis relies simply on sensor values.
- The *diagnostic accuracy* in a system is the percentage of behaviors that are

correctly categorized as one of several possibilities. It measures the *reliability* of diagnosis *within* the rule base, whereas fault compression measures the granularity.

These factors are, of course, interdependent. For example, decreasing allowable fault compression (undesirable) will tend to increase the required parameter compression (desirable), and increase diagnostic accuracy (desirable). In general, we cannot hope to optimize each of these parameters. Rather, design and sensor placement must optimize some tradeoff between them. For example, if accuracy is at a premium, then we may have to accept an decrease in fault compression. This implies a corresponding (but desirable) increase in parameter compression, and an expected decrease in sensor 'cost' as well. However, the undesirable decrease in fault compression implies that diagnostic cost will increase from having to employ secondary diagnostic procedures such as probing, testing, and simulation to a larger extent.

We are initiating systematic experiments across the range of diagnostic factors, with the eventual goal of defining an objective function that characterizes an appropriate tradeoff between them. Such a function will allow us to bound certain factors (e.g. accuracy, parameter compression or sensor 'cost') and to optimize for the remaining factors (e.g., fault compression). Our current version of C4.5 builds a decision tree based on the diagnosticity of system parameter values. Other variations that take into account the cost of sensing certain values have also been developed by Tan & Schlimmer [15].

A decision tree representation of a rule base is conceptually simple, and it has the desirable aspect of encoding the 'minimal' number of system measurements needed to isolate faults to a certain granularity. However, it also has some well-known disadvantages. Notably, a decision tree is very sensitive to noise in sensed system values (or faulty sensors, which we regard as another type of noise): a single misleading value can lead diagnosis considerably astray. One implication is that the minimality characteristic of decision trees may not be wholly desirable; uncertainty in a domain may insist on some redundancy in the sensed values, in order to better protect against the possibility of noise. Thus, in addition to our studies with C4.5, we are also investigating a second inductive approach known as *clustering*.

3 Cluster-Analytic Approach

A data analyst must often identify similarities and differences between observations. For example, a biologist will categorize a newly discovered organism into a known *genera* based on its similarities with known species of the class and differences with members of competing *genera*. An economist may recognize a trend in the market as having occurred previously, and forecast a particular outcome based on these historical similarities. The need to 'cluster' observations is critical in many fields, including the biological and social sciences, where it has spawned data analysis tools of *numerical taxonomy* or *cluster analysis* (e.g., Jain & Dubes [8]). Clustering methods have also evolved in artificial intelligence (AI) and machine learning (e.g., Michalski & Stepp[11]).

Clustering systems automatically discover categories of observations (events or objects) that are similar along some dimension(s). Once uncovered, these categories may sug-

gest features that characterize the observed data and/or facilitate predictions about the nature of future data. As in scientific endeavors, engineering disciplines can profit from clustering. For example, in diagnosis an observation may be a set of symptoms that collectively indicate a class of events that share a common diagnosis. We believe that discovered clusters can be used dynamically for automated diagnosis, and that like a data analyst, a system designer can use clusters over simulated behavior to facilitate design – in this case sensor placement.

3.1 COBWEB: A sample clustering system

A clustering system constructs a classification scheme over a set of observations. Figure 3 illustrates a classification tree constructed over five observations by a clustering system called COBWEB. Each node (class) in this tree represents a cluster of observations. Each cluster is represented by the distribution of attribute values over members of that node; this illustrative example assumes that observations are represented by attributes of `Size` (`small`, `medium`, `large`), `Shape` (`square`, `sphere`, `pyramid`), and `Color` (`blue`, `green`, `red`). Each leaf of the tree represents a category covering a single observation; the probability of each value in a leaf, $P(A_i = V_{ij}|\texttt{leaf}_k)$, is 1.0 (i.e., present in the corresponding observation) or 0.0 (i.e., absent, in which case it is not explicitly stored at the node). The root of the tree covers all observations, with base rate probabilities $P(A_i = V_{ij}|\texttt{root})$ that reflect global value distributions. In general, each node, C_k, contains probabilities, $P(A_i = V_{ij}|C_k)$, for each attribute value observed in a member of the node. In addition, the proportion of observations stored under each node relative to the node's parent is stored with the node. For example, forty percent of the observations stored under the root are stored under node C_1: $P(C_1|\text{root}) = 0.4$.

We will not describe the strategy used to build this categorization hierarchy over observations since it is of limited relevance in future discussion, and any of several strategies can be used. However, it is important to note that every clustering system relies on a measure of cluster quality. In COBWEB's case this is a measure of *category utility* derived from Gluck & Corter [3]:

$$CU(C_k) = P(C_k)\times$$
$$[\textstyle\sum_i \sum_j P(A_i = V_{ij}|C_k) \log_2 P(A_i = V_{ij}|C_k)$$
$$- P(A_i = V_{ij}) \log_2 P(A_i = V_{ij})],$$

which rewards clusters that *increase* the certainty inherent in the attribute value distributions. The expression above is appropriate for nominally-valued (i.e., discrete, unordered, finite) attributes, but several variations on this basic scheme (Gennari, Langley, & Fisher [7]; Reich & Fenves[14]) have been adapted to handle observations described over ordinal and continuously-valued attributes as well. The certainty-maximizing measure is used recursively, first to build a partition over the entire population of observations, and then to subpartition each of these initially-constructed clusters, thus yielding a categorization hierarchy. Our particular interest in this process is its ability to discover clusters over snapshots or instantaneous descriptions of system simulations.

3.2 Discovering Fault Modes

We use COBWEB to discover categories of fault conditions over system simulations. This proceeds in much the same way as

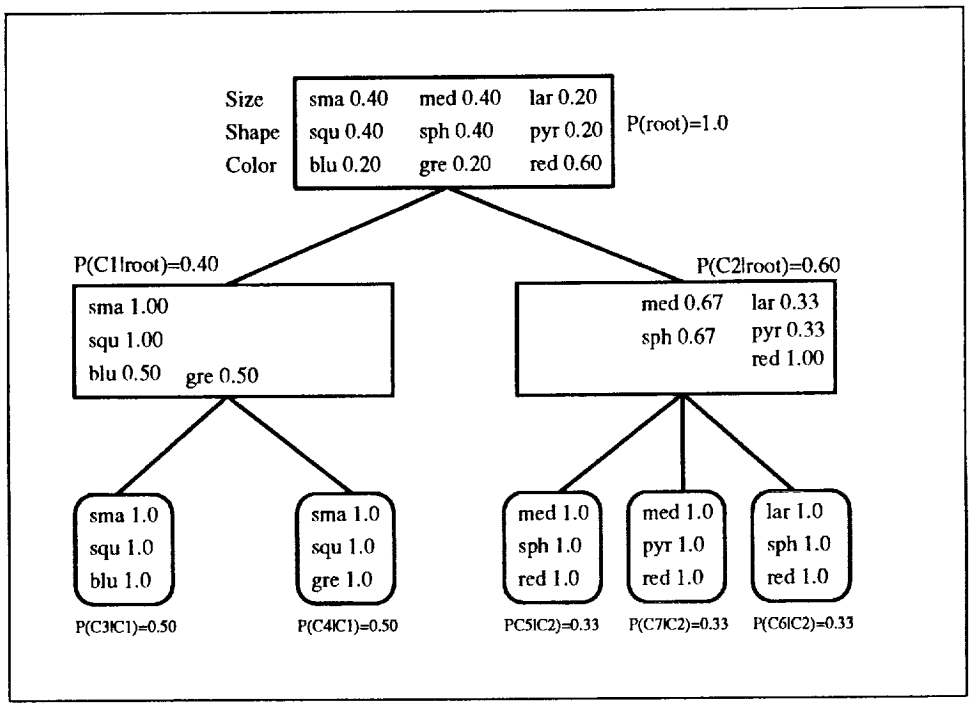


Figure 3: A classification tree constructed by COBWEB.

the simulation/induction procedure of Section 2, except that in Step [4], the snapshots are passed to our clustering system rather than a supervised one. An example of a categorization tree of discovered fault modes for the thermal system is partially shown in Figure 4. Each datum consists of inputs and outputs, for all components, including the single perturbed value (as described in step [2]); that is, each datum is a snapshot of the system. We do not show the probability distributions over all attribute values for clusters, but simply label each low-level node by a descriptor that conveys the fault-mode meaning. Thus, low flow through the radiator and a malfunction to the heater itself both result in high water temperatures (Example 1), despite the fact that this behavior emerges for very different reasons. Similarly, high flow through the pump appears somewhat similar to a second heater malfunction: both result in low water temperatures (Example 2).

As with C4.5, the benefits of clustering are at least two-fold. First, it is difficult for engineers to completely design against system faults in advance. Collectively, simulation and clustering identify fault models that benefit design decision making. For example, a faulty heater may overheat water in the thermal system, but this behavior may appear to be similar to, and thus be clustered with, a radiator (heat exchanger) that does not sufficiently cool water. Second, as with C4.5, these ambiguities can alert analysts to place sensors that better distinguish these conditions.

Again like C4.5, a COBWEB classification tree can also facilitate fault diagnosis. In particular, categories discovered through clustering associate observable/sensor/test features with component faults that lead to the observed anomalies. We wish to classify an observable set of sensor readings to a

level of the classification tree where a reasonably certain prediction of the underlying fault can be made. However, a categorization and diagnosis procedure is less clear with a COBWEB generated tree, since it does not specify a single value that should be sensed at any particular point as a decision tree does. Rather, we can exploit *characteristic* attribute values of discovered categories to direct sensor testing. There are a number of ways for identifying characteristic (or normative) values, as described in Fisher[6] and Reich & Fenves[14], but suffice it to say that they are values that are typically true of category members, and typically discriminate the category's members from other, contrasting categories. Characteristic values suggest tests that are likely to discriminate the most promising paths of the tree during classification: verification of a characteristic value(s) suggests that the associated path be followed, thus narrowing the plausible faults that are consistent with the known observables; failure to observe the expected value reduces the likelihood that the associated path will lead to a correct diagnosis.

The primary advantage of this strategy over C4.5 is that the categorization tree formed through clustering specifies a number of values at each node of the tree that can be sensed in order to guide further categorization or diagnosis. The decision tree structure is not generally as robust when certain values cannot be reliably sensed because of noise. In contrast, the increased information redundancy of the COBWEB tree is more robust in the face of noise, but redundancy also comes with the corresponding disadvantage that parameter compression is correspondingly lower.

4 Attention Focusing

Consider the space between the decision tree approach and the conceptual clustering approach as a continuum on feature structure. In decision trees the structure is fixed during training so that the order for feature testing during prediction is rigid. There is one feature test at each node with leads to a node at a deeper level (and another test).

In conceptual clustering there is no feature structure. To determine how to branch into the concept hierarchy, one must test every feature in the current node. In some cases this could lead to a significant number of tests (e.g., in our domain example from Section 2).

Optimally, we would like to classify an object or event in as few tests as possible with as few branches as possible. The decision tree approach would seem to have a tremendous advantage in classification of problems with highly independent feature spaces. However, when in a feature space with specific dependencies, it would be nice to cluster tests over these dependencies and branch deeper into the tree with fewer tests. One way in which we accomplish this is to examine the salience of each feature within each node, calculating what amounts to a category utility for each feature within the scope of its parent node.

The order of inspection for features in each node is then relative to its salience. The salience for a feature can be computed in any number of ways. In the equation below we show a general method for calculating salience based on standard deviation.

$$salience_i = \frac{\sum_k P(C_k)\frac{1}{\alpha_{ij}} - \frac{1}{\alpha_i}}{K}$$

where K is the number of classes, $P(C_k)$ is the probability of a particular class, and

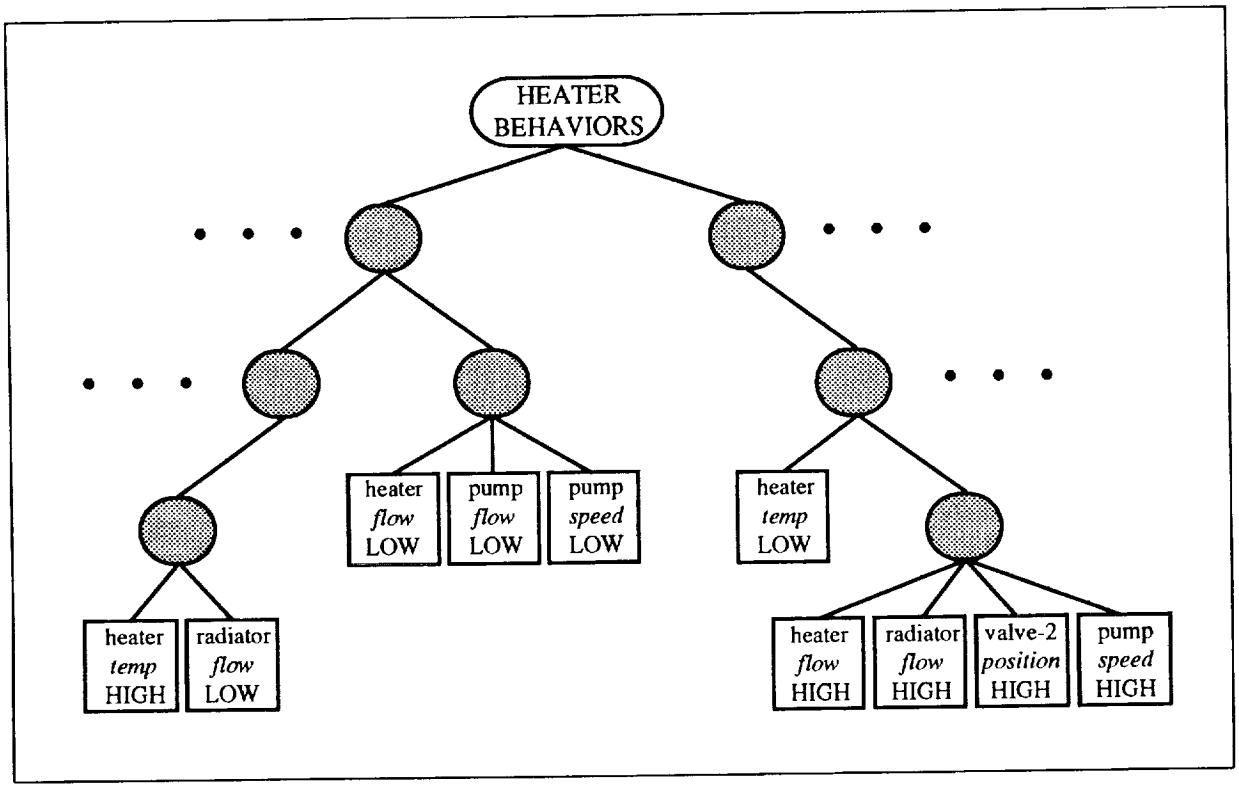


Figure 4: A partial classification tree of fault modes for the thermal model.

α_{ij} is the standard deviation of the feature within class k.

Using the notion of salience, an algorithm can be derived that focuses attention on the most informative features to test before branching into a behavior hierarchy. The following describes our algorithm for attention:

1. Select an unseen feature with probability based on salience scores stored at the parent.

2. Compute the salience of the selected feature; store this new score at the parent.

3. Compare the category utility score for the best classification, x, based *only* on features inspected so far.

4. Consider all remaining unseen features; if these were to match the second best classification, would the score be better than x?

5. If yes, goto step [1], otherwise ignore remaining attributes and branch to new node.

A problem closely associated with the calculation of feature salience is the selection of parametric measurements to ensure *complete* and *cost-effective* diagnosis. In analyzing a design for fault isolation we examine several additional factors, or properties, that belong to the device used for sensing a particular feature. A partial list of factors governing sensor selection follows:

So, when looking at which salient features to actually measure, an objective equation to minimize cost and maximize feature coverage must be designed. Below we offer a

· response time	· maintainability
· launch weight	· I/O performance
· criticality	· power consumption
· reliability	· procurement price
· repeatability	· number of sensors
· accuracy	· operating temperature
· resolution	· operating pressure

Table 1: Factors for sensor selection.

general form for such an objective equation:

$$min \sum w_i f_i'$$

where $\sum_i w_i = 1$, $f_i \in \{f_1, \cdots, f_i, \cdots, f_n\}$ are n sensor factors, and $f_i' = \| f_i \|$ is a normalized value representing the sensor factor within some range.

The following algorithm can be used for selecting which salient features to measure in a system under design.

1. Set threshold for objective equation.
2. Apply objective equation.
3. Collect sensor recommendations.
4. If *parameter compression* and *fault compression* (from decision tree analysis) are exceeded, then adjust threshold; goto root-node and restart. Otherwise branch and goto step [2].

5 Related Work

Work currently underway at JPL complements our research. JPL's AI Group has identified numerous factors that influence optimal sensor placement in Chien, Doyle, & de Mello[1], Chien, Doyle, & Rouqette[2], and Doyle & Fayyad[5]. Among these are factors that relate to the diagnosticity of sensors – i.e., the ability of sensed system quantities to predict the presence and location of faults. Roughly, diagnosticity is measured by simulating a fault on a system model, and then observing the changes to various model quantities. Quantities that differ most relative to their normal state (and possibly their value during other, competing fault conditions), are judged good predictors of that particular fault. In general, the approach makes pairwise comparisons between the same quantities under two different fault modes, and two different quantities under identical fault conditions. The approach appears to be generally helpful, but the utility of pairwise comparisons is limited. In contrast, our two learning approaches seek patterns or rules across multiple dimensions (i.e., multiple fault modes, and multiple sensed quantities) of system behavioral snapshots simultaneously. This approach can provide a more global perspective on system behavior, and makes certain multidimensional patterns explicit to the designer.

Furthermore, our approach to sensor placement is guided by an explicit model of the diagnostic process. This top-down approach contrasts with JPL's bottom-up approach, which is primarily responsible for enumerating a wider variety of factors that play a role in sensor placement. Our primary focus on a single aspect (i.e., information-content) of system parameter values that might act as good sensors is a disadvantage of our approach relative to JPL's. However, we view the two approaches as complementary, and are pursuing links between them.

6 Concluding Remarks

Our approach to sensor selection is distinguished from others in that it is guided by an explicit model of diagnosis; this top-down methodology promises principled criteria for sensor placement. Although our models of diagnosis are primarily useful for design, the rule bases developed through clustering and supervised methods could be used directly for diagnosis as well – either autonomously or by a human user. In this, we recognize the importance of both rule-based and model-based approaches as contrasted in Keller[9] and Davis[4]. Our bias is that inductive approaches can never replace model-based approaches in any but the most trivial of applications. As Keller points out, 'compiled' knowledge is most helpful in diagnosing relatively routine faults. To attempt a rule-based approach that covers idiosyncratic faults as well (i.e., achieves very high fault compression) invites 'overfitting' (i.e., unacceptably low accuracy and/or unacceptably low parameter compression). The overfitting phenomenon is well-known in machine learning, but inductive approaches to compilation for diagnosis have not traditionally addressed the issue, as shown in Pearce[12]. Rather, an ideal tradeoff between coverage, cost, and accuracy must only assume that a certain diagnostic burden is taken on by the compiled rule base. Our primary goal is to limit, but not eliminate, the space of faults that need be explored by probing, testing, and simulation.

References

[1] S. Chien, R. Doyle, and L. Homem de Mello. Model-based reasoning approach to sensor placement for monitorability. In *Proceedings of the Space Operations, Applications, and Research Symposium*, Houston, TX, 1991.

[2] S. Chien, R. Doyle, and N. Rouquette. Sensor placement for diagnosability in space-bourne systems: A model-based reasoning approach. In *Second International Workshop on the Principles of Diagnosis*, Milano, Italy, October 1991.

[3] J. Corter and M. Gluck. Explaining basic categories: feature predictability and information. *Psychological Bulletin*, 1992.

[4] Randall Davis. Form and content in model-based reasoning. In *Proceedings of the AAAI Workshop on Model-Based Reasoning*, Detroit, MI, August 1989.

[5] Richard J. Doyle and Usama M. Fayyad. Sensor selection techniques in device monitoring. In *Proceedings of the 2nd Conference on AI Simulation and Planning*, Cocoa Beach, CA, April 1991.

[6] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[7] John H. Gennari, Pat Langley, and Douglas H. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40, 1989.

[8] A.K. Jain and R.C. Dubes. *Algorithms for Cluster Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1991.

[9] Richard Keller. In defense of compilation. In *Proceedings of the Second AAAI Workshop on Model-Based Reasoning*, Boston, MA, August 1990.

[10] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, December 1986.

[11] R.S. Michalski and R.E. Stepp. Learning from observation: Conceptual clustering. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–403, San Mateo, CA, 1983. Morgan-Kaufmann.

[12] D.A. Pearce. The induction of fault diagnosis systems from qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 353–357, Saint Paul, MN, 1988. Morgan-Kaufmann.

[13] J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[14] Y. Reich and S. Fenves. The formation and use of abstract concepts in design. In D. Fisher, M. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, San Mateo, CA, 1991. Morgan-Kaufmann.

[15] M. Tan and J. Schlimmer. Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eigth National Conference on Artificial Intelligence*, pages 854–860, Boston, MA, July 1990.

Adaptive Laser Link Reconfiguration Using Constraint Propagation

M. S. Crone, P. M. Julich, L. M. Cook

HARRIS Corporation
GASD - 102/4844
P.O. Box 94000
Melbourne, Florida 32902
(407)729-3163

ABSTRACT

This paper describes Harris AI research performed on the Adaptive Link Reconfiguration (**ALR**) study for Rome Lab, and focuses on the application of *constraint propagation* to the problem of link reconfiguration for the proposed space based Strategic Defense System (SDS) Brilliant Pebbles (BP) communications system. According to the concept of operations at the time of the study, Laser communications will exist between BP's and to ground entry points. Long-term links typical of RF transmission will not exist. This study addressed an initial implementation of BP's based on the Global Protection Against Limited Strikes (GPALS) SDI mission. The number of satellites and rings studied was representative of this problem.

An orbital dynamics program was used to generate line-of-site data for the modeled architecture. This was input into a discrete event simulation implemented in the Harris developed **CO**nstraint **P**ropagation **E**xpert **S**ystem (**COPES**) Shell, developed initially on the Rome Lab BM/C^3 study. Using a model of the network and several heuristics, the **COPES** shell was used to develop the **H**euristic **A**daptive **L**ink **O**rdering(**HALO**) Algorithm to rank and order potential laser links according to probability of communication. A reduced set of links based on this ranking would then be used by a routing algorithm to select the next hop.

This paper includes an overview of Constraint Propagation as an Artificial Intelligence technique and its embodiment in the **COPES** shell. It describes the design and implementation of both the simulation of the GPALS **BP** network and the **HALO** algorithm in **COPES**. This is described using a Data Flow Diagram, State Transition Diagrams, and Structured English PDL. It describes a laser communications model and the heuristics involved in rank-ordering the potential communication links. The generation of simulation data is described along with its interface via **COPES** to the Harris developed **ViewNet** graphical tool for visual analysis of communications networks. Conclusions are presented, including a graphical analysis of results depicting the ordered set of links versus the set of all possible links based on the computed Bit Error Rate(BER).

Finally, future research is discussed which includes enhancements to the **HALO** algorithm, network simulation, and the addition of an intelligent routing algorithm for BP.

1. SDI BRILLIANT PEBBLES COMMUNICATIONS

During the course of the **ALR** program the space-based architecture was changed to be based on the concept of "Brilliant Pebbles (**BP**)". Each **BP** consists of a weapon system, sensor system, and a communications system. The focus of the **ALR** was on the communications system, as is that of this paper.

1.1. LINK RECONFIGURATION FOR BRILLIANT PEBBLES

The **BP** network consists of many platforms, each of which can receive simultaneously from a large number of neighbors, but which can only transmit via a laser to one other platform at a time.

This work was funded by the U.S. Air Force Rome Laboratory under contract number F30602-89-D-0096

It also runs in an open-loop fashion using simplex links, where a platform calculates the position of other platforms based on orbital predictions and periodic position updates. It then points at the selected platform only for the duration of a message. A link is never established in the manner of a typical RF architecture. There is, however, the possibility that pebbles will not recalculate links on a per message basis. Although the study does not address this possibility, the **COPES** implementation of the **HALO** Algorithm could be modified in a straightforward manner to accommodate such a change.

Given the large number of potential links from any node, a routing algorithm should not have to consider all potential nodes every time a message is sent. To work effectively it should only have to consider a subset of the links. This approach requires a database to be maintained which can effectively rate links according to constraints such as, longevity of LOS, range, probability of accurate position data for other nodes, beam-width limitations, probability of jamming, etc. Maintaining such an intelligent data base can significantly speed up the routing algorithm, and make it more robust in the face of enemy actions. The concept of link reconfiguration was redefined under the **ALR** program to mean the process of creating and maintaining such a database of rated links.

1.2. GLOBAL PROTECTION AGAINST LIMITED STRIKES (GPALS)

During the **ALR** study we simulated an initial implementation of the Brilliant Pebbles SDI architecture based on the Midcourse and Terminal Tier (MATTR) and Global Protection Against Limited Strikes (GPALS) studies. The SDI architecture has been radically altered since the BM/C^3 study, (Crone, Julich, 1990) with the incorporation of Brilliant Eyes(**BE**), Brilliant Pebbles(**BP**), and the Endo / Exoatmospheric Interceptors (E^2I). In addition, the concept of Battle Management has evolved, including both the location of Battle Managers and modes of operation. The MATTR study defined a BP-based SDI architecture which includes the midcourse and terminal phases. The GPALS study defined requirements for an SDI system which addresses a more limited size strike which may originate from any location. The GPALS architecture represents an initial but scaled-down version of an eventual phase 1 architecture, with less **BPs** and **BEs**, and without the GSTS system.

A significant difference in the mission of GPALS (as contrasted with the full scale SDS) is indicated by the name. First, the system provides *Global Protection*. The space elements of the system are intended to support defense both within CONUS and in overseas theaters. Advantages can be obtained by using a common communications architecture for the overseas theater and the CONUS implementation. Second, the term *Protection* suggests a different mission from the earlier SDIO Phase 1 architecture. The Phase 1 architecture had a primary mission of attack deterrence. Providing protection (zero leakage of attacking missiles) indicates a requirement for increased reliability and less probabilistic focus. This affects the communication requirements by placing a higher emphasis upon guaranteed delivery of messages. Third, the term *Limited Strike* indicates a smaller threat than the massive strikes considered in the Phase 1 (full scale) architecture.

The GPALS architecture is more distributed than previous architectures, with Battle Management being distributed along both regional and element lines. Weapon Target Assignments (WTA's) are generated much closer to the weapons. Control of the battle is hierarchical, however, through the use of Preplanned Response Options (PROs), Defense Employment Opportunities (DEOs), and Weapons Release Authority (WRA).

Figure 1 provides a view of the MATTR/GPALS architecture and connectivity. The legend describes the various elements involved in the battle. Control of the system is hierarchical beginning at the Command Center (CC) and proceeding through Regional Operations Centers (ROCs) and Element Operations Centers (EOCs).

In GPALS, battle management is distributed and co-located with sensor systems such as Brilliant Eyes(**BE**), Brilliant Pebbles(**BP**) and Ground Based Radars(GBRs). The **ALR** study was primarily concerned with track reports originating with **BPs** that are filtered by merge nodes as they are passed toward a GEP.

2. CONSTRAINT-BASED ALGORITHM DEVELOPMENT

Given the requirements of the GPALS simulation and the need to develop a heuristic algorithm to maintain a reduced set of potential links, constraint propagation as embodied in the **COPES** shell was chosen to accomplish both tasks. The application of constraint propagation to intelligent

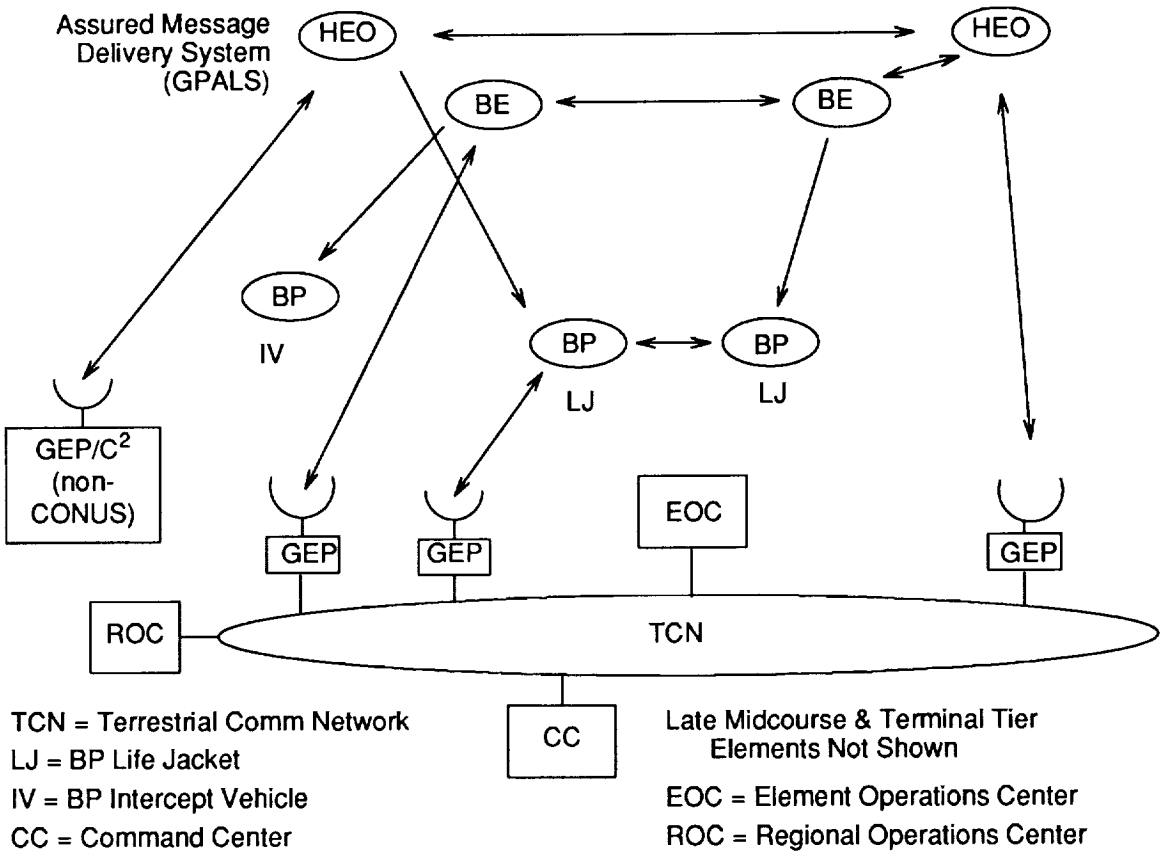


Figure 1 - Space-based GPALS Communications Architecture

problem solving was begun during the BM/C^3 program with Rome Lab.That effort resulted in the initial development of the **COPES** shell.(Crone, Julich, 1990) **COPES** was subsequently used in developing a distributed intelligent network manager in cooperation with the C Language Integrated Production System (CLIPS) rule-based language under the Distributed Intelligent Network Control (DINC) program for the U.S Army Strategic Defense Command. (Crone, Julich, 1991) This research was performed in cooperation with the Professor Ramamoorthy and students at the University of California Berkeley. The subject of this paper is the work done in intelligent link assignment for the SDI laser communications space network.

Since this work, we have used **COPES** to implement a version of the new Arpanet Shortest Path First(SPF) algorithm (McQuillan, 1980), a version of simulated annealing using **COPES** for the traveling salesman problem as a precursor to the problem of Weapon Target Assignment under Harris research, and have developed a neural network development tool. The simulated annealing technique utilized the discrete event scheduling capability of **COPES** to produce solutions to a 95 city problem which were consistently within 94% of the optimum solution. The **COPES** developed neural network tool is based on the Parallel Distributed Processing Project. (McClelland, 1988). In each case, as in all COPES development, the problem is represented in a distributed manner without a central executive process. In addition, the solution is distributed throughout the object database. For instance, in the case of the SPF algorithm, the shortest "next hop" to a particular node is maintained in the object representing the that node, as opposed to being maintained in a centralized routing table.

The remainder of this section gives the principles of constraint propagation in the context of Artificial Intelligence; its implementation in the **COPES** shell; and the current status of the shell as a basis of implementation for the **HALO** algorithm.

2.1. BACKGROUND

2.1.1. Traditional Search Techniques

A variety of search techniques exist, in the area of *combinatorial minimization* with an objective function to be minimized. (Press 1986) We have investigated some of these in the similarly complex domain of planning and have found them to be inadequate for knowledge-based problems. In particular the *Simplex Method* is a linear programming technique which can maximize a function subject to a set of constraints. This was found to be too slow and unable to provide partial schedules if all constraints could not be met. The *Constrained Minimization* technique in which a cost function is to be minimized subject to a set of constraints depends on the function being continuously differentiable. In this case standard techniques such as Penalty Function methods, and Conjugate Gradient methods could be used. Because of the discrete nature of planning and link assignment, such a function cannot be found. The concept of a heuristic cost function can be useful using a AI approach, however. Used in this way the function is used to evaluate potential link assignments and guide further refinement.

The method of *simulated annealing* is a technique which for practical purposes has solved the "traveling salesman" problem. It has been used successfully for designing complex integrated circuits to minimize interference among connecting wires in the arrangement of several hundred thousand circuit elements on a silicon substrate. These are both applications of *combinatorial minimization*. There is an objective function to be minimized over a discrete but very large configuration space. The method of simulated annealing based on the Metropolis algorithm always takes a downhill step while sometimes taking an uphill step thus avoiding being trapped in a local minima. (Press, 1986) It is applicable in cases where a simple measure of an objective function (analog of energy) can be defined. For most complex problems this cannot be defined by one function.

2.1.2. Artificial Intelligence

State space search is used in AI to move from an initial state representation of the problem(such as all potential links for each CV platform) to a goal state by the application of knowledge-based operators. (Rich, 1983) Given the nature of the initial and goal states, this search can be combinatorial. Algorithmic techniques exist such as *branch-and-bound*, and A* to reduce the search for some problems, but are inadequate for knowledge-rich problems. In this class of problem the cognitive activity of an intelligent agent involves two types of search: (1) *knowledge search*, that is, which operator to apply next, and (2) *problem-space search*, that is, search within the problem space for a goal state. (Gupta, 1983) Pruning of both spaces is crucial to reducing the search. In order to solve many hard problems efficiently, it is often necessary to to construct a control structure that is no longer guaranteed to find the best answer, but that will almost always find a very good answer. This is called *heuristic search* because knowledge is used to guide the search process. Heuristic search can be applied implicitly via the pattern matching of the rules against the problem-space data which takes place on each cycle in a production system, or explicitly via the weighting of constraints as in *constraint-directed* search in the ISIS scheduling system. (Fox, 1983) Blackboard Architectures address many of the issues of state space search and have been suggested as a control mechanism for problem solving. (Hayes-Roth, 1983)

Systems which take advantage of a great deal of knowledge are referred to as "Expert Systems", and have been shown to provide problem-solving computer programs that can reach a level of performance comparable to that of a human expert in specialized problem domains. (Barr, 1982) (Gevarter, 1983) They are in fact a form of qualitative model of both the problem space and the human problem solver. (Clancey, 1986) Expert Systems are characterized by the separation of data, rules(knowledge), and control. (Crone, 1985) They are usually rule-based, and due to the often enormous amount of pattern matching in rule-based systems, have not fared well in real-time applications. Some speed-up is predicted via parallel processing. (Gupta, 1983) Given the often autonomous intelligent activity which would be required in the link assignment problem, Expert Systems will be required, so research must uncover faster inferencing mechanisms. Rather than considering all data and knowledge in every inferencing cycle a more "object-oriented" (Stefik, 1986) knowledge representation is suggested. This is especially appropriate in cases where the problem space is in the form of a network with each node being connected to surrounding neighbors. The approach taken by this research is to use *constraint propagation* as the method of inference.

2.1.3. Constraint Propagation

An architecture which has been used with varying degrees of success, in physical reasoning, temporal reasoning, and spatial reasoning is to represent the knowledge base as a *constraint network* which performs inference by *propagating* labels. (Davis, 1987) These labels represent potential candidate values for nodes in the network.

2.1.3.1. Constraint Networks

A constraint network is a declarative structure which expresses relations among parameters. It consists of a number of *nodes* connected by "constraints". (Davis, 1987) A node represents an object which contains state and which is represented by the *value* of the instance variables of the object. A constraint represents a relation among the instance variables of the node and those of other objects it connects. As such, it is usually local in scope, but can connect all nodes in the case of a global constraint such as a heuristic weighting function. Examples of different applications of constraint propagation are numerous. (Crone, Julich, 1987,1990,1991) (Davis, 1983) (Fox, 1983) Forward inference on constraint networks, called *assimilation*, is usually done using *constraint propagation*, shown in algorithm 1. In constraint propagation, information is deduced from a local group of constraints and nodes, and is recorded as a change in the network. Further deductions will make use of these changes to make further changes. Thus, the consequences of each datum gradually spread throughout the network.

Algorithm 1. - Constraint propagation
repeat
-take some small group of constraints and nodes in some connected section of the network,
-update the information in this section of the network, given the information in the constraints and the nodes;
until no more updating occurs (the network is quiescent) or some other termination condition is reached.

In its most basic form, a set of potential labels for each node's instance variables are given, and then reduced based on constraint propagation to a unique solution or an inconsistent state.

A greater depth of knowledge concerning a particular system can be expressed in terms of constraints than is possible in a rule-based system alone. Model-based reasoning is a common application of constraint propagation where expected performance of a system is described through a set of constraints, which may contain mathematical models. Deviation from this behavior or observed similarities to expected failure modes can trigger corrective action or alter resource planning. This type of diagnostics is known as specification based as opposed to the symptom based approach used in rule-based diagnostic expert systems.

Unlike commercial AI shells such as ART, constraint propagation takes advantage of locality of information. Some of its valuable properties are:

- Forms a close analogy for systems in which physical effects propagate across connections between components.
- Constraint propagation consists of a simple control structure similar to a rule-based inference engine.
- Degrades well under time limitations; interrupting the process in the middle gives useful information already deduced.
- With assimilation, it is easily implemented in parallel, since updating can be performed all over the network simultaneously.
- Is easily expanded by adding constraints incrementally to the network.

2.1.3.2. Inferencing in COPES

In assimilation, the instance variable values for each node are represented by a set of labels which must be consistent with constraints relating the instance variable to those of other nodes. The general form of refinement is given by the following definition: **Definition 1.** Let C be a constraint on nodes $X_1, \ldots X_k$. Let S_i be the label set for X_i. Then

$$\text{REFINE}(C,X_j) = \{a_j \in S_j \mid \exists\, (a_i \in S_i, i=1,\ldots k, i \neq j) \; C(a_1,\ldots,a_j,\ldots a_k)\}.$$

That is, REFINE(C, X_j) is the set of values for X_j which is consistent with the constraint C and with all the labels S_i. A value a_j is in REFINE(C, X_j) if a_j is in S_j and it is part of some k-tuple $a_1, \ldots, a_k$ which satisfies C and all the S_i.

Applying the updating function REFINE within the constraint propagation control structure given in Algorithm 1, gives the Waltz algorithm. (Waltz, 1975) The Waltz algorithm applies constraints to nodes until no more changes occur (the network has reached *quiescence).* Algorithm 2 is an efficient implementation of the Waltz algorithm which served as the original basis for this research with many additions being added over time.

Algorithm 2. - Waltz Algorithm

```
/* The set
S_i is the current label set of quantity X_i */

REVISE refines all the parameters X_1 . . .
,X_k of a given
constraint C, and returns the set of all parameters
whose set was changed.

procedure REVISE(C(X_1 , . . . X_k ))
 begin CHANGED ← ø
  for each argument X_i do
   begin S ← REFINE(C, X_i)
    if S = ø then halt
    else if S ≠ S_i then
     begin S_i ← S
      add Xi to CHANGED
     end
   end
  return CHANGED
 end


procedure WALTZ
 begin Q ← a queue of all constraints
  while Q ≠ ø do
   begin remove constraint C from Q
    CHANGED ← REVISE(C)
    for each X_i in CHANGED do
     for each constraint C' ≠ C which has X_i in its
        domain do
        add C' to Q
   end
 end
```

2.2. APPROACH

2.2.1. Development of COPES Shell

In order to effectively apply Waltz's algorithm to a network type of problem we designed and implemented the **COPES** Shell at Harris to merge the concepts of *constraint propagation* as a method of inference, with *object-oriented programming* as a method of representation. Unlike most applications of constraint-based reasoning, the use of **COPES** provides a solution which is easily created and updated. The representation scheme allows the hierarchical definition of complex objects called classes which contain state information in the form of instance variables, and links to constraints which are applied to them. For some problems, constraints are inadequate to produce a unique state. We added a searching mechanism to the shell which allows back-tracking with or without selective pruning. An instance variable defined for a class is really represented by a complex data structure which includes its type, name, parent, etc. This allows generic functions to be developed where variable type is dynamically bound. This is similar to a Lisp/Flavors approach. Among the objectives of this work was to build the shell using the language "C" on a Unix environment such as the Harris HCX-9, and to emphasize run-time speed. Since a large part of the knowledge base is programmed directly in "C", and locality of information is considered; **COPES** offers an execution time advantage over rule-based systems.

The building of a tool to support Expert System development is compounded by the need to experiment with the tool during development to expose limitations and problems. The flexibility required for AI tool development leads one to follow the principles of object-oriented design, where possible. At Harris we have built object-oriented versions of both C and Ada to make implementation possible in a conventional environment. (Crone, Julich, 1987) (Simonian, Crone, 1989) For the sake of run-time speed, we did not use either for the development of **COPES**, but did follow many of the principles of object-oriented programming.

Object-oriented design methodologies typically start with an emphasis on the data representation. In order to support AI design, we added extensions to Entity-Relationship (E-R) diagrams which are typically used for database design. (Chen, 1976) To describe the software design, we modified the Jackson Structured Design(JSD) process model. (Cameron, 1986) The AI and software designs of **COPES** are described in detail elsewhere. (Crone, Julich, 1990)

2.2.2. Knowledge Representation in COPES

Knowledge takes two forms in **COPES**: (1) the constraint network, and (2) the constraint functions. The application of the constraints to the network is the function of the shell.

The creation of such a constraint network database is done either interactively for small problems or is read from a Unix file created by a C program. In most problems amenable to solution via constraint propagation, a network is often homogeneous with identical constraint relationships between neighboring nodes. We are currently developing a "class" language and "instantiation" commands to make the creation of such a network easier and more dynamic.

2.2.3. State of the COPES Shell

The following is a description of the current state of development of the **COPES** shell and some of the problems to which it has been applied.

- Hierarchical knowledge representation scheme using object oriented approach
  - — Metaclass, classes, subclasses
  - — Instance variables to define object state
  - — Class constraints (definition and instances)
- Waltz constraint propagation based on representation scheme.
- State saving and Restoring callable by constraint routines
- Abstract variable types with generic access methods
- Container variable types which support queues, stacks, sets, etc.
- Container variables also support distributed problem solving via TCP/IP sockets.
- Scheduled variable modification for discrete event simulation including cancellation of events.
- Variable access methods such as PUT, GET, WRITE, etc. either direct or "BY_NAME", where the class object and variable name is given.
- Database features to list class structure, constraint bindings, and error messages during creation
- A variety of tracing features for debugging
- Problems to which COPES has been Applied
  - — Distributed Intelligent Network Management
  - — Distributed Network Emulation
  - — Distributed Heuristic Algorithms
  - — Simulated Annealing WTA Research
  - — model-based diagnostics
  - — Modeling Neural Networks
  - — Distributed algorithms: A*, N-queens, SPF, TSP
  - — Dataflow-based discrete event simulation for SDI

3. HALO ALGORITHM USING COPES

This section describes the **HALO** Algorithm and its implementation using the **COPES** shell. We first introduce the Adaptive Link problem and develop a model to analyze it. Then we discuss the heuristics of the algorithm and consider their relevance to the actual Brilliant Pebbles scenario. Next, we discuss how the model is implemented using the **COPES** shell as a simulator. Finally, we discuss some results obtained from running the simulator on a typical scenario for the Brilliant Pebbles architecture.

3.1. HALO ALGORITHM DEFINITION

This section develops a model describing the **HALO** Algorithm in terms of objects and defines how these objects interact with one another. This model is defined with an Object Oriented structure which lends itself well to implementation in **COPES**. The section concludes with a definition of the heuristics of the algorithm.

3.1.1. Adaptive Link Problem

The **HALO** Algorithm considers a Brilliant Pebbles architecture consisting of a constellation (or constellations) of satellites in low earth orbit communicating with each other using laser links. The algorithm attempts to reduce the work of a routing algorithm by generating a ranked list of links ordered by the best to worst probability of successful transmission. The algorithm generates this optimally ordered list by applying a set of heuristics to the list of links such that in most cases the router would only have to consider a small set of these optimal links to make its routing decision. This is important in a laser based communication network where a large number of highly dynamic potential links exist.

Figure 2 shows a model of the data flow and objects used to implement both the **HALO** Algorithm and a simulation of a **BP** scenario. This model represents a single instance of the **HALO** algorithm running on one **BP**(referred to in this discussion as the *reference node*). In this phase we do not consider the router, consequently we are only concerned with the point of view of one pebble and how it orders its optimal set of links. Thus, Figure 2 does not consider any routing issues. In the rest of this section we develop the model shown in Figure 2 and describe the algorithm as a set of heuristics (or constraints), a set of objects, and the relation between them.

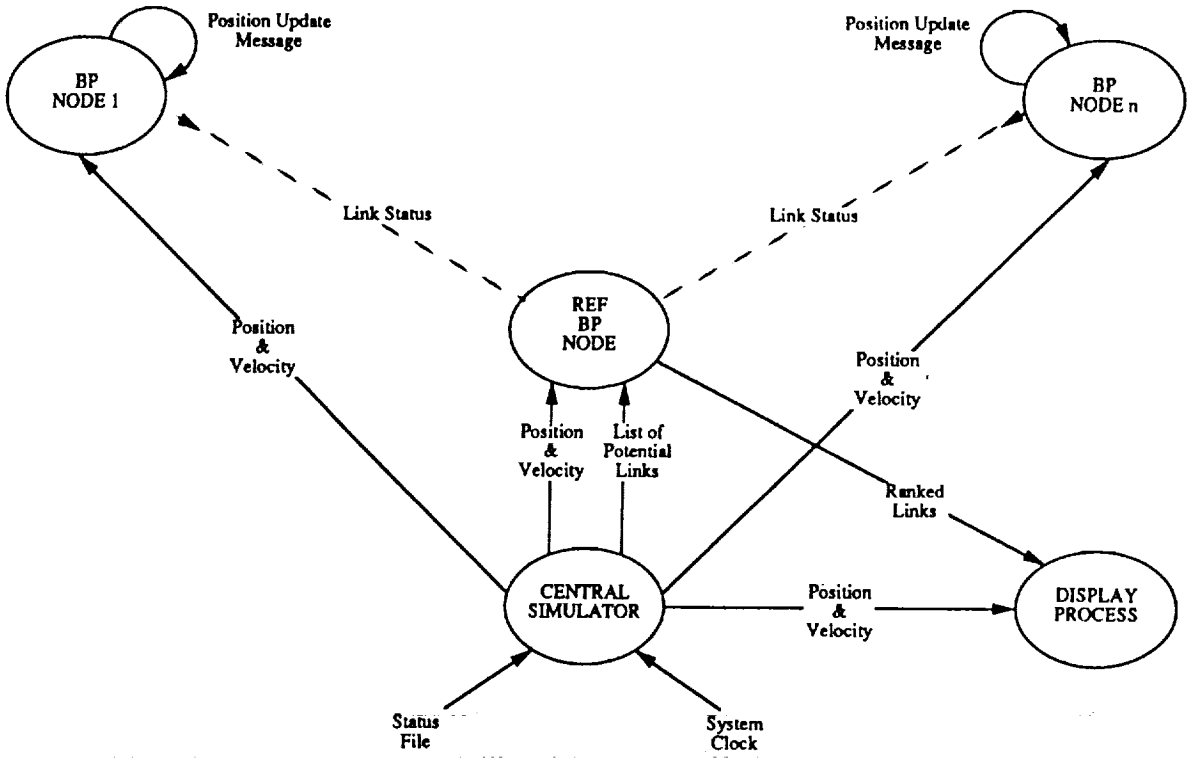


Figure 2 - HALO Data Flow

The data flow diagram in Figure 2 shows the main objects in the **HALO** Algorithm and the communications between these objects. The object **BP Node** describes parameters common to each **BP** in the constellation. It contains information such as the position of each satellite, its error term, and other instance information. This object is replicated many times within each **BP** which runs this algorithm (it is the reference node's view of other **BPs** in the network). The object **Ref Node BP** describes information unique to the reference node. It contains the ranked list of "links", some state information, and some information about the constellation. The object **Central Simulator** is not a physical object in the **BP** scenario, but contains data associated with the overall **HALO** algorithm and helps control the interactions between objects. It governs the operations of the simulation such as reading the orbital dynamics file, and starting and stopping the simulator. The object **Display Process** also is not a physical object in the **BP** scenario but is an entity which monitors the algorithm and simulator, and reports its progress for display and analysis. The lines with arrows show the communication between objects, the solid lines indicate actual messages being sent to the appropriate object(algorithmic communication, not to be confused with actual communication between physical **BPs**), and the dashed lines indicate an object viewing the data in other objects, e.g. **Ref Node BP** accesses link status from **BP** node 1.

The **HALO** Algorithm uses a set of heuristics which govern how the algorithm sorts the list of possible links. These heuristics are described below.

- LOS - This heuristic checks whether a **BP** is in LOS of the reference **BP**. For the purposes of the **COPES** implementation, this data is precomputed using an orbital dynamics package to model the satellite motion and visibility.
- Velocity - This heuristic checks whether the relative velocity between two **BPs** communicating with one another would cause a Doppler shift impairing the laser communications. An analysis of the need for this heuristic is described later in this section.
- Lasercomm Probability - This heuristic determines the probability of a successful communication between two **BPs**. It models the effects of pointing error, position uncertainty between pebbles, and other laser parameters.
- Position Error - This heuristic allows the position error to be corrected at a predetermined rate. Currently, the position error updates (which would normally be received from other **BPs**) are randomly scheduled with a period of

one satellite orbit.

These heuristics are imposed by the **HALO** algorithm and do not necessarily map into unique **COPES** constraints. Each heuristic defined above is developed further in the remainder of this section. In the next section we develop the **COPES** constraints which implement these heuristics.

3.1.2. Line Of Sight Heuristic

The Line Of Sight (LOS) heuristic determines whether a reference node can communicate with a given **BP**. This simply tests whether the specified **BP** is physically in LOS with the reference node. If it is not, then no further consideration is given to the **BP** as a potential link.

3.1.3. Velocity Heuristic

We now consider the requirements for a velocity constraint. The relative motion of two **BPs** may affect the communications between themselves due to the Doppler shift of the laserbeam. The maximum tolerable frequency shift of the laserbeam is on the order of 1 *nm* for the laser comm systems considered in the brilliant pebbles architecture. The Doppler shift of light between two bodies is defined by the following equation:

$$v' = (1 - \frac{u}{c})$$

u = relative velocity of BPs.

v = frequency of light source at rest.

v' = frequency of light due to u.

This equation is valid if $\frac{u}{c} \ll 1$.

In the brilliant pebbles simulations at typical satellite constellation altitudes the maximum relative velocity of **BPs** is about 7.5 *km/s*. Substituting the appropriate values into the above equation, the Doppler shift that the **BPs** see is approximately 20 *pm*. This is two orders of magnitude less than the laser design constraint of 1 *nm*. Thus, this constraint appears to be of limited concern in the initial analysis. There has been some discussion in the SDI community concerning a potential problem with link acquisition based on relative velocity. This remains an area of research.

3.1.4. Laser Probability Heuristic

The laser probability heuristic is a computation of the probability of successful reception of a laser transmission from the reference **BP** to a designated source **BP**. The probability is computed from a model of the laser communication link. This model is for a direct-detection receiver using multimode pulse position modulation (PPM) signaling and is currently being studied for the Brilliant Pebbles architecture.

The laser model defined for the **HALO** Algorithm has only one degree of freedom, the position error (described below). The model assumes fixed values for other parameters of the laser model. In addition controlling the transmitter beamwidth yields a better probability of successful communication over a wider range of distance between source and destination. The model assumes the beamwidth can vary from 2.5 *mRAD* to 25 *mRAD*. The model simulates this variance by keeping the following relationship:

$$\theta_b R = K$$

where

θ_b = receiver beamwidth.

R = receiver range.

K = constant.

The range of distances between the reference node and a designated node varies over the interval (200,4500) *km*. Thus, the laser model optimally sets its θ_b over this interval and then computes the probability of successful communication using the position uncertainty.

3.1.5. Position Error Heuristic

Each **BP** maintains a database of the current positions of other **BPs** in the network used for routing and link selection. As time passes, each **BP** calculates predicted position of the other **BP's**. Due to the relative infrequency of position updates, there is error associated with these predictions. This position update message has an inherent error associated with it as well.

In the adaptive link reconfiguration simulations, we model a position error as a growing sphere around the **BP** over time. Thus, we need a rate term to grow this sphere as the simulation progresses. The worst rate would result from the **BP** being at a less or greater orbit altitude than it is

supposed to be. This would cause the orbit period to be faster or slower, respectively, than a **BP** would predict it to be. Thus we will assume that the position update message has an error of a certain amount d which is in a direction greater or less than the actual orbit radius.

Assuming a spherical earth, the error will grow at a linear rate as computed below:

$$R' = R - d$$

$$v_e(t) = v(R-d) - v(R)$$

$$v(r) = \sqrt{\frac{\mu}{r}}$$

$$E_{orbit} = t\sqrt{\mu}\left(\frac{1}{\sqrt{R-d}} - \frac{1}{\sqrt{R}}\right)$$

where:

d is the error of the global positioning system.
R is the perceived radius of the BP orbit.
R' is the actual radius of the BP orbit.
$v_e(t)$ is the error velocity.
E_{orbit} is the error rate in m/s.
μ is the gravitation parameter in m^3/s^2

The adaptive link simulations assume the positioning system used in the **BP** architecture is accurate to 100 m. This causes the worst possible position error rate of 0.055 m/s at an altitude of 550 km. This value is used in the simulations run. The simulator also assumes that the position updates occur at a rate of once per orbit.

3.2. COPES IMPLEMENTATION

The model of Figure 2 presents a set of objects which describe the **HALO** Algorithm. These objects are described as classes in **COPES**. Each class defines the state information of a particular object in a model. There may be multiple instances of a class such as the **node** object in the model which is duplicated for each physical node in the system. A constraint function describes the interactions between the defined objects. A constraint is bound to variables in a given class instance. A constraint "propagates" or "fires" when a variable in a class instance that the constraint is bound to changes. When the constraint "fires" it observes the state of the objects it is bound to and changes the states of these objects appropriately. A constraint may be bound to a variable in two ways. The first way is for the constraint to "fire" when the variable changes. The second way is for the constraint to ignore changes to a variable but access this variable when the constraint does "fire" from some other binding.

The **HALO** Algorithm is defined as a set of classes and constraints. The classes defined below represent the objects of Figure 2 and some additional classes required for the **COPES** shell and to support the simulator for the **HALO** algorithm. The classes are:

Ref Node : This class contains information unique to the reference node above what is necessary to describe a general node.

Node : This class contains information unique to each node.

Link : This class contains information about the laser link between the reference node and the node with which this link is associated.

File In : This class contains file status and descriptor information used by the central simulator.

Display : This class contains file descriptors and flags used by the display object.

GLOBAL : This class contains simulation parameters and flags to which every constraint has access.

In developing the algorithm, we define state transition diagrams which describe the threads of the overall algorithm flow. A sample of one of the state transition diagrams is shown for the central simulator (Figure 3). The Central Simulator is responsible for managing the simulation and reading the new position and velocity (p&v) parameters from the orbital dynamic file. It loads the p&v information into each node, waits for the current cycle to complete and then starts the next cycle. When the simulation is complete, the central simulator causes the **COPES** shell to terminate. Referring back to Figure 2, a Node performs three tasks. During the initialization cycle, it schedules a position update message to correct the position error term. During a normal cycle it receives and interprets position update messages and it responds to new position and velocity parameters. The reference node shown in Figure 2 is the node on which the software is considered to be running in the simulation. In a real system each node would be a reference node. The reference node manages the list of ranked links. As each node updates its link status parameters, the reference node updates the list of ranked links. When all links have been updated in the current cycle, the reference node

indicates it has an updated links list which starts the display process. The Display Process monitors the simulation then prints out simulation statistics and formats the ranked list for viewing with the ViewNet program at the end of each cycle.
A sample of this output is given later.

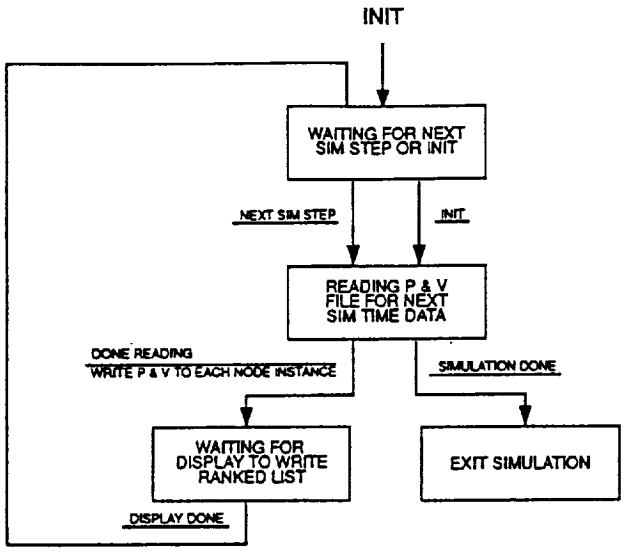


Figure 3 - Central Simulator STD

The constraints are derived from the heuristics (described in the previous section) coupled with the state transition diagrams. User defined constraints are detailed in the remaining part of this section. User constraints are defined using structured english PDL.

The first constraint is the **read_pos** constraint. This constraint performs much of the function of the Central Simulator object. It is responsible for controlling the simulations aspects of the **COPES** algorithm implementation, and reading the orbital dynamics file for the current computed satellite positions and LOS data.

This constraint has one instance and schedules itself to fire once each cycle. This constraint is bound to the instance of the class **file_in.** This class has a state variable, run_file_in, which the read_pos constraint schedules to change in the future. This allows the constraint to fire itself to run at the beginning of each simulation time cycle to read the current satellite position data. In addition to acting as the Central Simulator, the read_pos constraint represents a **BP** reference node computing the position, velocity, and LOS of each node in the constellation. It takes advantage of the discrete event scheduling capabilities of **COPES** to move the simulation, link ranking, and display through discrete phases. In this manner objects like the Display do not have to signal the Central Simulator when they are done. Their inactivity (lack of constraint propagation changes in the network) causes the removal of the next change from the schedule queue and constraint propagation continues in the next phase.

The next constraint is the **pos_update** constraint. This constraint is concerned with scheduling and receiving position update messages. It performs two functions. It initially schedules the random position update for each node. It then responds to the position update messages which cause the node to reset its position uncertainty. This constraint represents a reference node receiving a position update message from another node in the constellation. This message provides the actual position of the satellite which the reference node uses to reset its notion of that satellites position. A separate pos_update constraint is bound to each **node** instance in the **BP** scenario.

The next constraint is the **Node1** constraint. This constraint models the cumulative effect of the position error. It gets fired when a p&v recomputation event occurs (triggered by the Central Simulator) which causes the node to increase its position uncertainty. A separate instance of this constraint is bound to each class **node** instance.

The next constraint is the **comm_ber** constraint. This constraint fires when the position uncertainty parameter of a given node is modified. It then (if in LOS, meets the Doppler heuristic, and is within range) computes the bit error rate (BER) of successful laser communication. A short Structured English PDL is shown for this constraint is shown below as a design example.

```
comm_ber ()
{
  when new position uncertainty parameters for this node
   for each potential link
    if node in LOS and (Doppler and range thresholds valid) then
      compute new ber for successful tx (src to dest).
      store new ber in link.
      set link flag indicating to add/update link in ranked list.
    else if link currently in ranked list then
      set link flag indicating to remove link from ranked list.
    endif
   endfor
  endwhen
}
```

It sets this BER in the link instance for this node. A separate instance of this constraint is bound to each class **node1** instance.

The next constraint is the **rank_links** constraint. This constraint fires each time a node modifies its link quality parameters and then ranks all the links according to these parameters. When all links have been determined, the constraint sends the ranked list out to the display constraint. A separate instance of this constraint is bound to the each class **node.**

The last constraint is the **display** constraint. This constraint implements the display object. Its purpose is to take the ranked list of links from the reference node object and format it for display in ViewNet. Additionally, it outputs some statistics of the simulation for post analysis. A single instance of this constraint is bound to the class **Display Process.**

Finally, Harris developed constraint binding diagrams are produced for each constraint to show the dynamic bindings which link constraints to class variables. Figure 4 shows an example of a constraint binding diagram for the comm_ber constraint. This diagram is useful to understand how the constraints interact with the object instances. An instance of the comm_ber constraint is created for each **node** and **link** object as it is viewed from the reference node. An instance of the constraint fires when the position error of the node instance to which it is bound is modified. This causes the given **node** object to reset the concept of bit error rate for the link from the node to the reference node. The remaining variables are bound as access only and do not cause the comm_ber constraint to fire. An access only variable is indicated in the constraint binding diagram by placing an "A" on the end of the line linking the constraint to the variable.

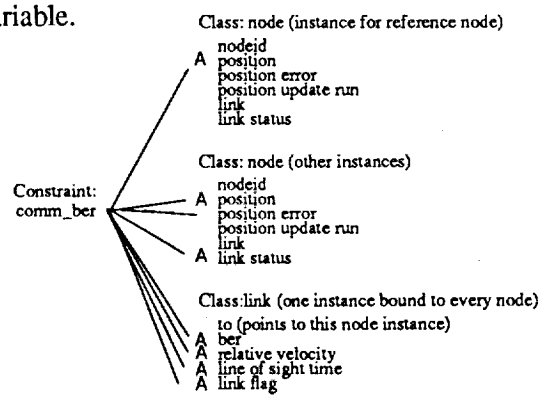


Figure 4 - Comm_ber Binding Diagram

The classes and constraints discussed in this section define the **COPES** model for the **HALO** Algorithm and simulator. In the next section we discuss the actual **BP** scenario used to test this Algorithm and the results of those simulations.

3.3. SIMULATION DATA FOR THE ADAPTIVE LINK ALGORITHM

This section presents the simulations run for the **HALO** Algorithm. The Adaptive Link simulator provides two types of output for analysis. The first is a visual display using the ViewNet tool developed at Harris. The second is a plot of the average ber rate of the top portion of the ranked list versus the ber of all possible links.

The Brilliant Pebble scenario used to test the algorithm is an unclassified network which is closer in size to a GPALS architecture. It consists of a single constellation of satellites at an altitude of 550 *km* and an inclination of 60°. The constellation contains 21 rings of 20 pebbles per ring. Only one constellation is used since the problem is not changed by multiple constellations and the implementation of the simulator is simplified. The simulation described above runs for a period of one earth day. This provides time for approximately 15 satellite orbits. The random position update messages are issued once per orbit.

The ViewNet graphical tool provides the capability to visualize the Adaptive Link algorithm in operation to gain an intuitive understanding of how it works. Figure 5 shows a snapshot in time of the ViewNet display. This figure shows the satellites in orbit around the earth, the reference node with the eight "best" links connected to the appropriate node. The actual ViewNet display is in color. Each node has a special color indicating its status as a potential link. In addition, the links are color coded from red to grey indicating their relative position in the list of ranked links. The laser probability model tends to select the closer links as opposed to the more distant links. However, it ranks extremely close links as less probable due to the fact that the position error becomes more significant at closer ranges. Observing the ViewNet display, as the links are reordered and displayed, this trend is apparent.

The second visualization of the simulation is a graph depicting the enhanced set of links available to a router versus the set of all possible. The set of all possible links is those links within LOS and within range. The ber is computed for each of these links and averaged. This plot is provided as a function of time. The **HALO** algorithm enhancement is shown by averaging the top 8 links in the ranked list of links. This average is plotted as a function of time also. The plot indicates that

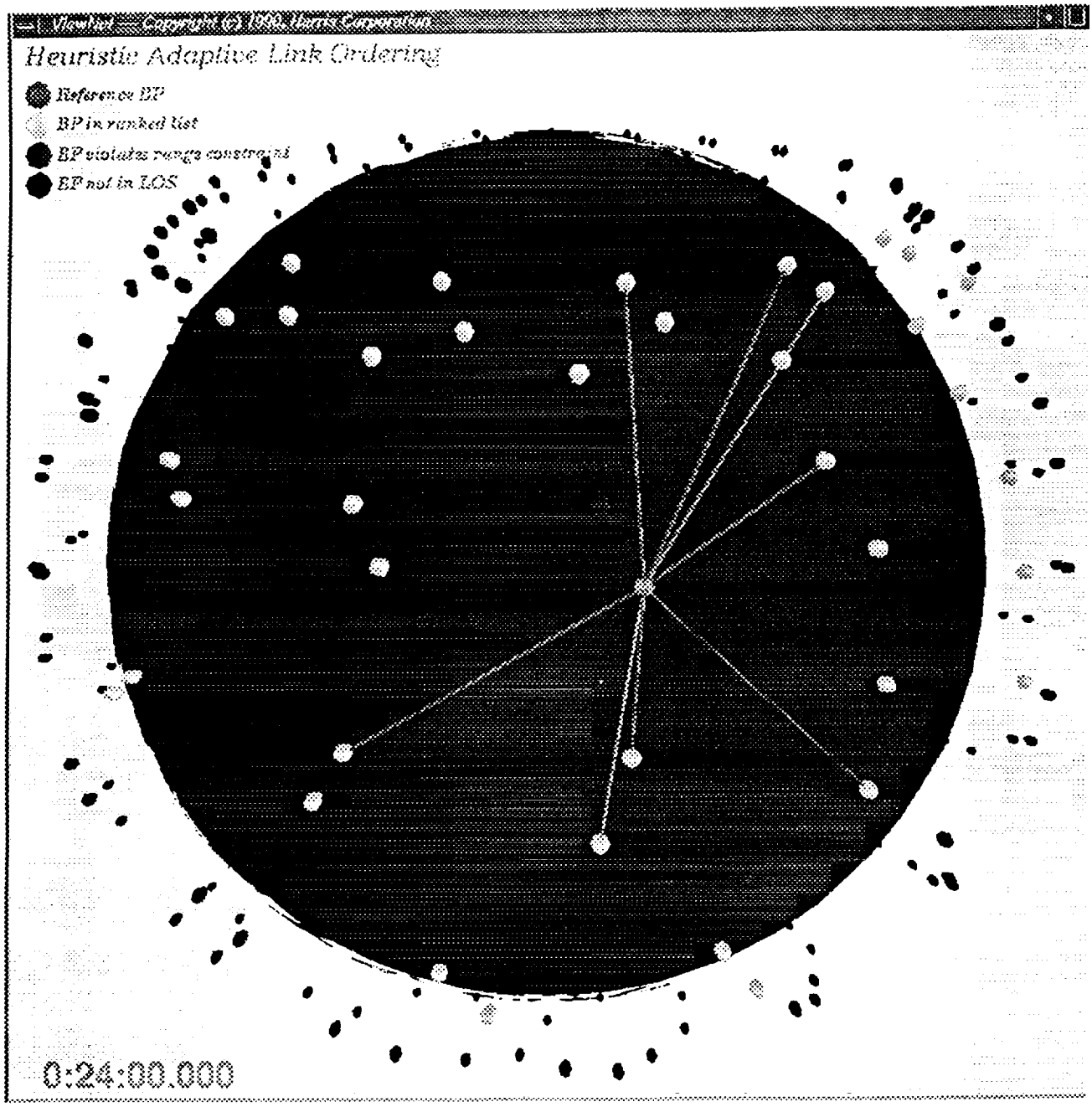


Figure 5 - Time Snapshot of Adaptive Link Viewnet Display

the top eight ranked links are an order of magnitude better probability of successful transmission over just any possible link. The plot is shown in Figure 6.

4. CONCLUSIONS

The previous section presented a successful simulation of a GPALS-based **BP** architecture using the Discrete Event Simulation capabilities of the **COPES** Shell. This simulation in **COPES** is very flexible and easily modifiable to address the **BP** network in its entirety, including any future architectural or procedural changes. The simulation of the network and results of the **HALO** algorithm were also interfaced with the Harris developed ViewNet graphics tool for network analysis on the Silicon Graphics Workstation. Also described was the **COPES** implementation of the **HALO** algorithm. A graphical analysis showed that the algorithm generates a reduced, improved, and ordered set of links for further use by a routing algorithm. The benefit of a lower bit error rate on the selected link is a reduction in the power requirement for the communications.

Given the flexibility of a constraint approach to the **HALO** algorithm written in **COPES**, changes in constraints can easily be made to, for instance, emphasize links which are more distant. This is an area for future research.

5. FUTURE RESEARCH

For the **ALR** program, the size of **BP** constellation which must be considered by a routing algorithm has been reduced to one closer to the GPALS architecture. We use an orbital dynamics program to remove all nodes which are never seen by the reference pebble we are studying. Finally, using a set of constraints defined above, the set of potential links is reduced and ordered by how well each meets the constraints. The next step is to develop an intelligent routing algorithm which would use this ordered list of potential "next hops" to choose a link or links for a particular message. The major advantages to this approach are that the set of potential links has been reduced significantly prior to the running of the routing algorithm, and the probability of successful transmission is higher. An intelligent routing algorithm might also contain heuristics to allow it to consider the first n potential links based on the situation, since the links are already sorted by how well they satisfy a set of link constraints.

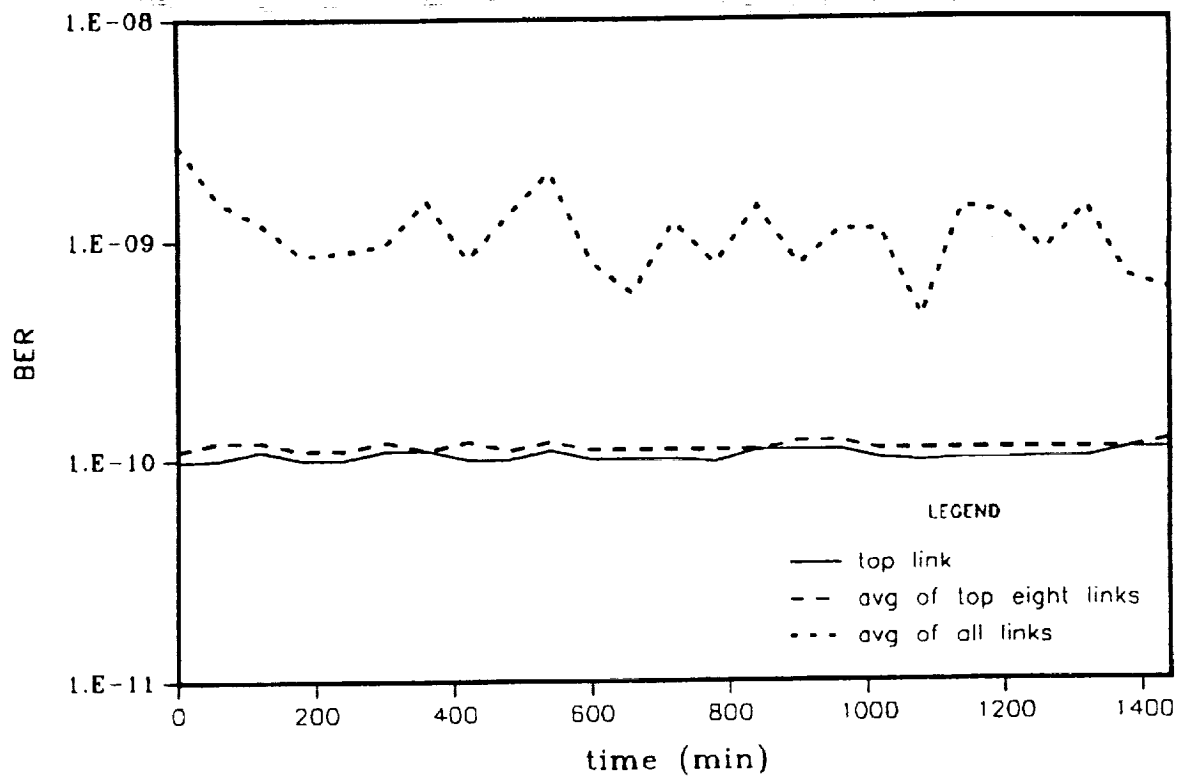


Figure 6 - BER of All Links vs Top Eight Ranked Links

6. REFERENCES

Barr, A. and Feignbaum, E. (eds.). (1982). *The Handbook of Artificial Intelligence,* HeuristicTech Press, Stanford, Cal.

Cameron, J. (1986). "An overview of JSD", *IEEE Transactions on S/W Engineering,* Vol SE-12, No. 2 (Feb.).

Chen, P. 1976. "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transaction on Database Systems,* Vol 1, No. 1 (Mar.), pp. 9-36.

Clancey, W. (1986). The Science and Engineering of Qualitative Models, *Proceedings of AAAI-86, Science Track.*

Crone, M. S., Hall, D. (1985). "Comments on the Procurement and Development of Expert Systems", *Proc. Expert Systems in Government Symposium, IEEE Computer Society.*

Crone, M. S., Julich P., Dash E., Wavering W. (1987). "Expert System Technology for the Space Station Communications and Tracking System, *Proceedings of the SPIE, Space Station Automation III, vol 851.*

Crone, M. S., Julich P. (1990). "Dynamic Network Reconfiguration Using Constraint Propagation" *Proceedings of the Society for Computer Simulation, Multiconference on Simulation and Artificial Intelligence, vol 21.*

Crone, M. S., Julich P. (1991). "Distributed Intelligent Network Management for the SDI Ground Network" *Proceedings of MILCOM*

Davis, R. (1983). "Diagnosis Based on Structure and Function", *Proceedings AAAI Conference.*

Davis, E. (1987). "Constraint Propagation with Interval Labels", Artificial Intelligence, vol 32., pp. 281-331.

Fox, M. (1983). *Constraint-Directed Search: Case Study of Job-Shop Scheduling,* Ph.D Thesis, Computer Science Department, Carnegie-Mellon University.

Gevarter, W.B. (1983). "Expert Systems: Limited but Powerful", *IEEE Spectrum* (Aug.), pp 39-45.

Gupta, Anoop. (1983). "Parallelism in Production Systems", Ph.D Thesis, Computer Science Department, Carnegie-Mellon University.

Hayes-Roth F., Waterman D., Lenat D. (eds.) 1983. *Building Expert Systems,* Addison-Wesley.

McClelland, J., Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing,* MIT Press

McQuillan, J., Richer, I., Rosen, E. May (1980). "The New Routing Algorithm for the Arpanet," *IEEE Transactions on Communications, vol.com-28, No. 5*

Press, W., et. al. (1986). *Numerical Recipes,* Cambridge University Press.

Rich, E. (1983). *Artificial Intelligence,* McGraw-Hill, Inc.

Simonian, R., Crone, M. May (1989). "True Object-Oriented Programming in Ada" *Signal Magazine.*

Stefik, M., Bobrow, D. (1986). "Object-Oriented Programming: Themes and Variations", *The AI Magazine.*

Waltz, D. (1975). "Understanding Line Drawings of Scenes with Shadows," in *The Psychology of Computer Vision,* edited by Patrick Winston, McGraw-Hill Book Co, N.Y.

Acknowledgements

- Sun is a trademark of Sun Microsystems, Inc.
- InnovAda is copyrighted by Harris Corporation, 1987
- HCX is a trademark of the Harris Corporation
- Unix is a trademark of the AT&T Bell Laboratories

An Architecture for Object-Oriented Intelligent Control of Power Systems in Space

Sven G. Holmquist, Prakash Jayaram, and Ben H. Jansen

Department of Electrical Engineering
University of Houston, Houston, TX

A control system for autonomous distribution and control of electrical power during space missions is being developed. This system should free the astronauts from localizing faults and reconfiguring loads if problems with the power distribution and generation components occur.

The control system uses an object-oriented simulation model of the power system and first-principle knowledge to detect, identify, and isolate faults. Each power system component is represented as a separate object with knowledge of its normal behavior. The reasoning process takes place at three different levels of abstraction: the Physical Component Model (PCM) level, the Electrical Equivalent Model (EEM) level, and the Functional System Model (FSM) level, with the PCM the lowest level of abstraction and the FSM the highest. At the EEM level the power system components are reasoned about as their electrical equivalents, e.g, a resistive load is thought of as a resistor. However, at the PCM level detailed knowledge about the component's specific characteristics is taken into account. The FSM level models the system at the subsystem level, a level appropriate for reconfiguration and scheduling.

The control system operates in two modes, a reactive and a proactive mode, simultaneously. In the reactive mode the control system receives measurement data from the power system and compares these values with values determined through simulation to detect the existence of a fault. The nature of the fault is then identified through a model-based reasoning process using mainly the EEM. Compound component models are constructed at the EEM level and used in the fault identification process. In the proactive mode the reasoning takes place at the PCM level. Individual components determine their future health status using a physical model and measured historical data. In case changes in the health status seem imminent the component warns the control system about its impending failure. The fault isolation process uses the FSM level for its reasoning base.

1 Introduction

Failure to provide a reliable, uninterrupted supply of electrical power under all circumstances may doom space missions. In case of impending or actual failures, decisions will have to be made about rescheduling load demand and/or reconfiguring the power generation and distribution system. These decisions will have to be made fast, often without the help of experienced control room operators, and often relying on incomplete information.

Knowledge-based (or intelligent) control systems have the ability to make decisions, and the capability to learn, and therefore seem ideally suited for the operation of complex systems such as electric power plants and distribution systems. However, practical applications of in-

telligent controllers are rare, and appear to be based on control strategies that use prewired solutions to a collection of potential problems, and/or use a supervisory planning approach to failure recovery. As a consequence, these systems have no way to deal with unanticipated, or multiple simultaneously occurring faults, and they have little or no capability to adapt to changing environments or to learn from past experiences.

We are working on overcoming these aforementioned limitations by developing an intelligent control system that uses quantitative and qualitative system models based on an object-oriented representation of the components of the physical system to be controlled. The object-oriented representation decentralizes intelligence by equipping each component with knowledge about how to detect its impending failure, and how to act in case of failure. This reduces the time required to detect faults when compared to an approach relying on a single central fault detector. Furthermore, the object-oriented representation can be implemented in a parallel computer, leading to even shorter response times. The intelligent controller will use these models to explore the "optimal" control actions to modify the system performance or operation. Also, by equipping the model components with knowledge about their behavior (e.g., a resistor will "know" how its temperature will rise in response to the voltage and current applied to it), and with memory (e.g., a record of its temperature for the last hour or so), *proactive* autonomous control can be achieved, even with incomplete sensor data.

Expert systems have been applied to the power engineering area before (see [10] for a review), but few such system are beyond the demonstration phase, and all were developed for large-scale, interconnected systems. The most promising approaches involve the use of object-oriented techniques because an object-oriented approach models the causal and functional relationships by inheritance and message passing mechanisms, and the part-of or component hierarchy [7]. Furthermore, objects are complete functional units that lend themselves to parallel implementations more easily than rule-based approaches, which is important for real-time applications.

A fairly small number of applications of object-oriented programming techniques for the intelligent control of power systems have been published [1, 2, 6, 9], with the prototypical system for event diagnosis and operation planning described in [3] being most closely related to our own work. However, it is unclear how much this system relies on reasoning from first principles (if it uses that concept at all), nor does it seem to have progressed beyond its first prototype state. Notwithstanding this criticism, [3] clearly shows that object-oriented, model-based methods are indeed advantageous for problems in control. The theory of model-based reasoning is explained by Kuipers [5]. Model-based systems are especially useful in the diagnosis of multiple faults as shown in [4]. Also, it is argued in [4] that diagnosing faults at multiple levels of abstraction, starting with the most abstract level, and examining the less abstract levels only when there is reason to suspect it, makes the generation of candidate solutions more efficient.

2 Architecture of the power system simulator and controller.

Our work is based on a multi-level model of the system, with intelligence built in at each level in the sense that each component can reason about its real-world state, as opposed to a higher level intelligence that reasons about

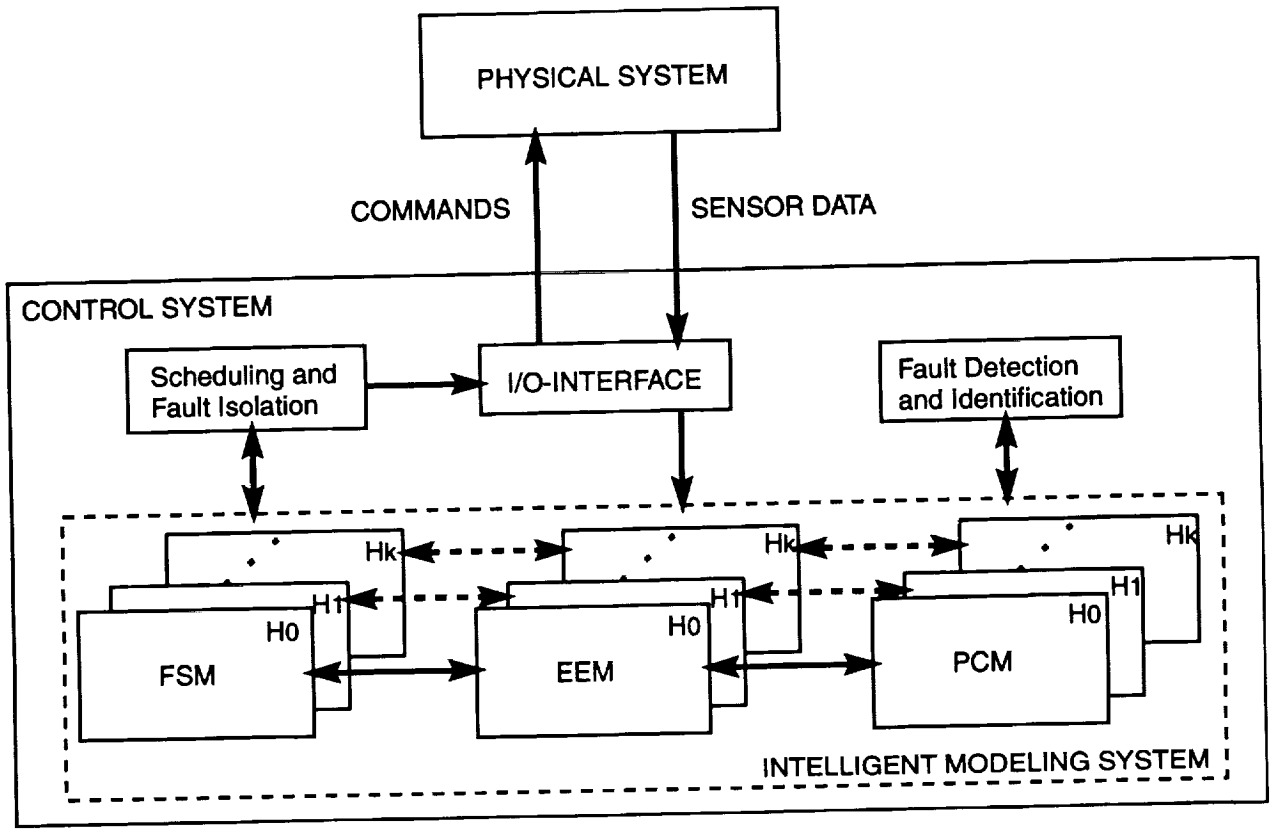


Figure 1: *Overview of the architecture of the model-based, object-oriented control system.*

all the "dumb" lower level objects. Also, the object-oriented design we follow is intended to support concurrency with only a minimal amount of knowledge being exchanged.

2.1 General System Description.

A diagram of the control system is presented in Figure 1. At its core is a model of the system to be controlled. This model represents the physical system under normal operating conditions, and is referred to as the H_0 simulator. At least three versions of H_0 exist, representing the physical system at various level of abstraction. First, there is the Physical Components Model (PCM), containing physically realistic models of the components of the system to be controlled. At the next level of abstraction, one finds the Electrical Equivalent Model (EEM). The latter is a representation of the physical system in terms of power sources, impedances, and switches. The Functional Subsystems Model (FSM) is the most abstract of all, and represents the system in the form of a reduced network in which sub-nets are represented by single functional blocks. An example of the PCM, EEM, and FSM of a simple physical system, consisting of a generator, switches, resistive loads (a light bulb and an electric heater) is shown in Figure 2. The electric heater consists of a fan, i.e., a motor (M1) and a resistive heating element (L2); and the light bulb is denoted by L3.

Each of the three models is an object-oriented representation of the actual system. That is, components are represented as data structures referred to as objects. The latter consist of attributes relating to properties of the component

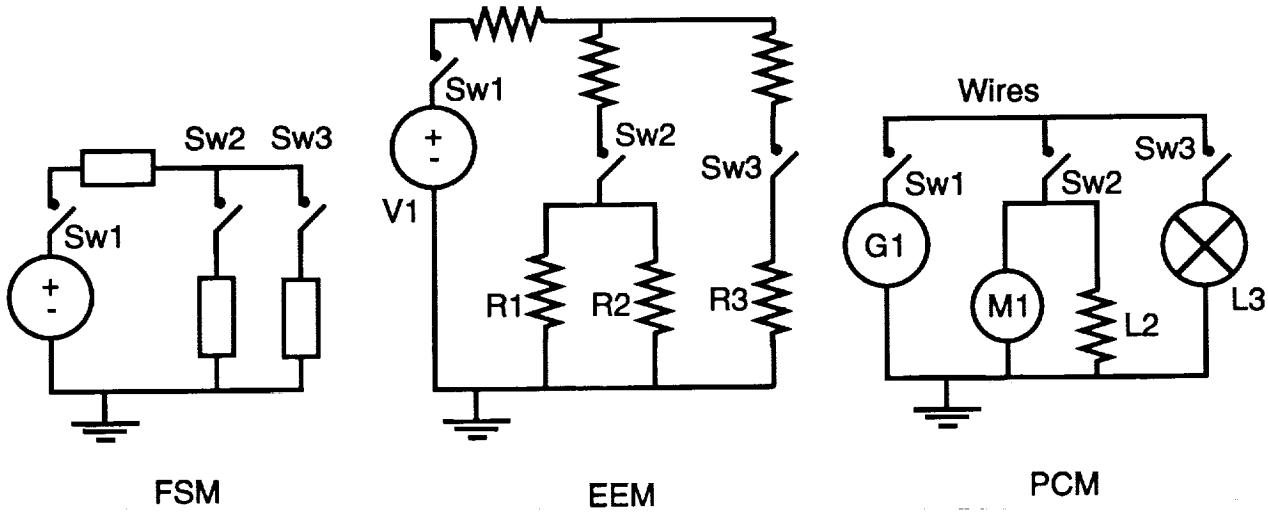


Figure 2: *An example of a Physical Component Model (right), its Electrical Equivalent Model (middle), and its Functional Subsystems Model (left).*

being represented, and attribute-values specifying the values of these properties and/or procedures that can be used to compute these values.

The topological relationships between components in the PCM, EEM, and FSM are specified by attributes describing the connections between the present component and others in the network. Expected voltages at nodes and currents through branches in the EEM are computed using the VIsolver. The VIsolver is an object that solves for the currents and voltages of the power system using the modified nodal formulation [8]. The solution is based on Kirchhoff's current and voltage laws in a matrix form with special considerations taken to reduce the size of the matrices but at the same time keeping it general. This method can be used on networks containing voltage and current sources, impedances, conductances, ideal two-ports, and switches. Historical data, intended for use in the proactive mode, for each component is stored in history attributes. Sensors placed at strategic positions in the physical system (in our case, the physical system is a software simulation as well) provide measurements of voltages and currents in the power system. The PCM and the EEM work in tandem, using the knowledge embedded in them, to detect potential faults. Once faults have been detected (see Section 2.2 below for an explanation of the fault detection process), additional versions of the PCM, EEM and FSM are automatically generated, representing models of the physical system modified in such a way as to account for the hypothesized cause of the fault. For example, H_1 and H_2 may be generated in case two explanations for the fault are possible. Competing hypotheses are eliminated on the basis of comparing future sensor data with predicted values, and/or heuristic reasoning. Once the fault has been determined (identified) remedial action is taken to return the system to a non-faulty state through reconfiguration of loads and sources.

2.2 Fault Detection

Faults may be present if discrepancies between sensor values and expected values are found in the EEM, or if a component in the PCM anticipates impending failure (on the basis of knowledge about the behavior of its physical equiva-

lent, and the historical data available). In other words, the system works in both a *reactive* and a *proactive* mode simultaneously.

As an example of reactive operation, consider the system shown in Figure 2. Assume that voltages and current measurements are available at the output of the generator (V1), the input to the heater (R1 and R2), and the input of the light bulb (R3). Assume that the measured voltage and current at the heater suddenly drops. The voltage at the light bulb will also change slightly, and the current at the source will decrease. Therefore there is a discrepancy between measured sensor values and simulated sensor values and a fault is detected. It is not obvious from the measurements which component is faulty. However, by reasoning using knowledge of the fault models for each component and their health status it is possible to narrow down the number of possibilities and, eventually, the fault can be identified and isolated through simulation (see Section 2.3 for details).

An example of proactive fault detection is the following: Assume that M1 in the PCM finds that its real-world counterpart is about to overheat due to a continuous overload beyond its rating. The M1 object then immediately signals its impending fault state to its equivalent counterpart (R1) in the EEM and tells R1 that the current needs to be reduced. The control system formulates strategies to reduce the current through R1, using the knowledge encapsulated in it (in this case the only possibility is switching off the motor). It is clear that hypothesis selection needs to be based taking into account the importance of the various subsystems in accomplishing the mission objectives. The components in the FSM have knowledge about these aspects, and this knowledge is used to determine which of the reconfigured systems best meets future objectives, and the H_j, that accomplishes this, becomes the new H_0 after the appropriate commands have been issued to the power system.

2.3 Fault Identification

Once the existence of a fault has been detected the location of the fault must be determined. A small change in a single component value can cause many sensors to indicate the existence of a fault. To determine which component has caused the fault (in the reactive mode), branch currents and node voltages are computed using the measured data, and each component's impedance value is computed based on the current running through it and the voltage across it. The EEM component compares its calculated impedance with its "known" impedance and if there is a difference, then the component is suspected of having caused the fault. All components have a health status attribute which is determined by the PCM and verified by the EEM. The PCM determines the health status using heuristics, historical data, and physical knowledge of the component. Hypotheses regarding possible faults are generated, based on the component's health status and impedance discrepancy using the component's fault-model, supplied by the PCM.

The aforementioned approach will work if the environment is sensor-rich, i.e., there are enough sensors in the network to calculate the impedance of *all* components. However, if the environment is sensor-sparse, i.e., there are relatively few sensors in the network, then a strategy will be followed that converts the sensor-sparse environment into a virtual sensor-rich environment. This approach is based on the concept of compound component models. The latter are formed by combining components connected in series, parallel, or in a bridge configuration to a single compound component. Compound components can be part of other compound components. The location of the

available volt-meters and current-meters guides the formation of compound models so that in the (reduced) environment the impedance of each compound component can be determined. In other words, the reduced network becomes virtually sensor-rich with respect to the compound components. The impedance and health status of the compound components is calculated based on the impedance, the health status, and the interconnection of the individual components that make up the compound component. The fault identification process can then function in a similar fashion in both a sensor-rich and a sensor-sparse environment. Of course, fault localization can then only pinpoint a compound component as the source of the trouble. However, using the fault models, heuristics, and historical data about the components making up the compound component can be used in a reasoning process to more precisely identify the fault location.

To illustrate the reasoning process, consider the case where a fault has been localized to a compound component consisting of two parallel resistive loads. Suppose that one of the loads is a motor, and the other a heater. Faults occurring in these components will reflect themselves as changes in the component's impedance (e.g., a short will cause a virtually zero impedance). Further, assume that only the voltage across the loads and the total current, but not the currents through each load, are known. In such a case, it will be impossible to determine which load is faulty based on the available measurements alone. However, using fault-models supplied by the PCM, coupled with the assumption that a single fault is considerably more likely to occur than a multiple fault, one or more hypotheses can be generated. For example, the PCM "knows" that a heater's most common failure mode is breakage of the heater element, causing the impedance to go to infinity. Therefore the H_1 hypothesis would replace the EEM of the heater by an infinite impedance, while leaving the EEM of the motor unchanged. In a similar manner H_2 would replace the motor EEM by an impedance reflecting its most prevalent fault state, i.e, a short in the motor coil. The voltages and currents predicted by each of the competing components are compared to the measured data, which will lead eventually to the elimination of all but one hypothesis. This process can be refined by utilizing the concept of the component's "health status". The latter can be used to determine the order in which components should be hypothesized as faulty. For example, the fact that a component has been in service for close to its expected life span, gives it a poor health status and thus it will be hypothesized as faulty prior to components with a good health status. The system will keep track of which components fail, and under what circumstances. This "failure log" is fundamental to the learning capabilities of the system, which will come to "recognize" previously encountered failure modes.

3 Design and implementation of the power system simulator and controller.

We are currently in the process of implementing the previously outlined architecture. The NeXT computer has been chosen as the implementation platform. The NeXT supports Objective-C and has extensive graphical interface capabilities.

The power system simulator has been designed and implemented. A graphics-based tool has been developed to interactively configure the power system to be simulated. A panel with icons, representing components typically encountered in a power system, is presented, and the user can "click-and-drag" these icons in the desired position in the power system win-

dow. The specifications for each component are entered by changing the attribute values, in an inspector window, for the component. The resulting power system can simulate voltage sources, switches, and resistive loads. We are only considering direct currents at the present, but a generalization to alternating currents is kept in mind.

A schematic diagram of the power system is shown on the screen in a power system simulator window with the component values and currents and voltages displayed next to each component. The power system's voltages and currents are calculated by the simulator's VIsolver. The VIsolver is an object that solves for branch currents and node voltages for any electric network including power systems using the nodal admittance matrix. The solution is based on Kirchhoff's current and voltage laws in a matrix form with special considerations taken to reduce the size of the matrices but at the same time keeping it general.

Changes in switch settings, load resistance, and source voltages can be made through an event queue or by clicking on the component in the schematic drawing of the power system. The event queue is editable and is used to insert faults into the power system. The power system's voltages and currents are automatically recalculated when the power system simulator receives an event or a switch position is changed by clicking on the switch with the mouse. The events are sent to the power system one after the other in order of occurrence in time.

A control system that reads data from the power system simulator has been implemented. It is possible to set which voltages and currents the control system can receive from the power system by inserting volt-meters and current-meters at the desired positions in the network. The data is displayed in a separate control system window containing the same diagram as shown in the power system simulator window. The control system is capable of issuing commands regarding switch settings to the power system. The control system is capable of forming compound models of components in series, parallel, and bridge configurations.

4 Future developments.

At present, a component library is being built for commonly used electric power components, including DC-motors, generators, circuit breakers. These components, with their embedded knowledge, form an important part of the fault detection system.

The current speeds of execution of the system suggest that parallel implementation is necessitated in order to achieve real-time implementation. Though we lack the hardware for such an implementation, a successful attempt has already been made at executing the various tasks in the program concurrently on the same processor using separate threads. We expect to implement the final system with a fair amount of distributed processing over a network of NeXT computers, so that each task will have its own processor, with the goal of achieving significant speed-ups.

Acknowledgements

This worked has been sponsored by grants from NASA/Johnson Space Center, and the Energy Laboratory of the University of Houston.

References

[1] Gholdston, E. W., Janik, D. F., and Newton, K. A. (1989). Hybrid approach to space power control utilizing expert sys-

tems and numerical techniques. *Proceedings of the Intersociety Energy Conversion Engineering Conference,* Piscataway, NJ: IEEE-Press, 177-182.

[2] Kao, C.Y., and Morris, W.S. (1989). Spacelab life sciences-1 electrical diagnostics expert system. *Telematics and Informatics, 6,* 201-220.

[3] Keronen, J.J. (1989). An expert system prototype for event diagnosis and real-time operation planning in power system control. *IEEE Transactions on Power Systems, 4,* 544-550.

[4] de Kleer, J., and Williams, B.C. (1987). Diagnosing multiple faults. *Artificial Intelligence, 32,* 97-130.

[5] Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence, 29,* 289-338.

[6] Lee, S.J., Yoon, S.H., Yoon, M.C., and Jang, J.K. (1990). Expert system for protective relay setting of transmission systems. *IEEE Transactions on Power Delivery, 5,* 1202-1208.

[7] Levitt, R.E., and Dym, C.L. (1991). *Knowledge Based Systems in Engineering.* New York: McGraw-Hill.

[8] Vlach, J., and Singhol, K. (1983). *Computer Methods for Circuit Analysis and Design.* New York: Van Nostrand Reinhold Company.

[9] Walls, B. (1989). Starr: An expert system for failure diagnosis in a space based power system. *Proceedings of the Intersociety Energy Conversion Engineering Conference,* Piscataway, NJ: IEEE-Press, 303-306.

[10] Zhang, Z.Z. , Hope, G.S., and Malik, O.P. (1989). Expert systems in electrical power systems - A bibliographic survey. *IEEE Transactions on Power Systems, 4,* 1355-1361.

The Use of Multiple Models in Case-Based Diagnosis+

Stamos T. Karamouzis* Stefan Feyock**

Department of Computer Science
College of William & Mary
Williamsburg, VA 23185

Abstract

The work described in this paper has as its goal the integration of a number of reasoning techniques into a unified intelligent information system that will aid flight crews with malfunction diagnosis and prognostication. One of these approaches involves using the extensive archive of information contained in aircraft accident reports along with various models of the aircraft as the basis for case-based reasoning about malfunctions.

Case-based reasoning draws conclusions on the basis of similarities between the present situation and prior experience. We maintain that the ability of a CBR program to reason about physical systems is significantly enchanced by the addition to the CBR program of various models. This paper describes the diagnostic concepts implemented in a prototypical case-based reasoner that operates in the domain of in-flight fault diagnosis, the various models used in conjuction with the reasoner's CBR component, and results from a preliminary evaluation.

Introduction

Reasoning about physical systems is a difficult process, and any attempt to automate this process must overcome a number of challenges. Among these are the tasks of generating explanations of normal behavior, fault diagnoses, explanations of the various manifestations of faults, prediction of future behavior, etc. The reasoning process becomes even more difficult when physical systems must remain in operation. During operation, a physical system changes dynamically by modifying its set of components, the components' states and pattern of interconnections, and the system's behavior.

To address these concerns a prototypical case-based reasoner (CBR), called Epaion, has been developed by the Intelligent Cockpit Aids Team at NASA Langley Research Center, in connection with ongoing work on AI-based systems for in-flight fault management [Schutte et al.]. The reasoner operates in the domain of in-flight fault diagnosis and prognosis of aviation subsystems, particularly jet engines. Automation of in-flight fault diagnosis and prognosis can be used as an aid to the flight crew for early detection of a problem or failure. This provides the crew with more time to respond more effectively and reduce potential damage due to the failure.

Several aspects of the aircraft domain make automation of in-flight diagnosis challenging. In contrast with non-operative diagnosis (i.e., diagnosis of systems that can be shut down), symptoms in aircraft subsystems may change with time because of failure propagation. Information about the operational status of many aircraft components may be unavailable or incomplete due to limited instrumentation, and safety and

+ Work supported by NASA grant NCC-1-159
* stamos@cs.wm.edu
** feyock@cs.wm.edu

comfort considerations place further constraints on in-flight testing.

The approach we are taking employs a novel methodology for dealing with physical systems in operation, and involves the use of case-based techniques in conjunction with models that describe the physical system. Case-Based Reasoning systems solve new problems by finding solved problems similar to the current problem and adapting their solutions to the current problem, taking into consideration any differences between the current and previously solved situations. Because CBR systems associate features of a problem with a previously derived solution to that problem, they are classified as associational reasoning systems.

We maintain that the ability of a CBR program to reason about physical systems can be significantly enchanced by the addition of various models to the CBR program. This paper describes the diagnostic concepts implemented in Epaion[1], the various models used in conjuction with the CBR component, and results from Epaion's preliminary evaluation. Although the examples presented pertain to aircraft malfunctions, it is clear that these techniques are applicable to spacecraft as well.

Knowledge Sources

Epaion draws its power from several knowledge sources, including a library of aircraft accident/incidents; a functional dependency model with deep domain information about the functional dependencies between the components of the aircraft; and a model representing causal information concerning transitions between various states of the aircraft.

Case Library

Epaion maintains a library of actual aircraft accident/incident scenarios called *cases*. Each case consists of a set of features that identify the particular scenario, a list of the relevant context variables and their particular status, a set of observable symptoms, the fault, and a causal explanation that connects the observable symptoms to a justifying cause. The set of identifying features includes information such as aircraft type, airline, flight number, date of the accident, and similar data. The list of context variables includes information such as the phase of flight, the weather, etc. The set of symptoms includes information about abnormal observations from mechanical sensors such as the value of the exhaust gas temperature, the value of engine pressure ratio, or from "human sensors," such as the sound of an explosion, or the smell of smoke in the passenger cabin. Cases containing all of this information are called *library cases*, whereas cases where the fault and the causal explanation are not available are called *input cases*.

In contrast to most other CBR research efforts, each case in our methodology consists not only of a set of previously observed symptoms, but also represents sequences of events over certain time intervals. The time intervals may have unknown and unequal lengths; it is the event ordering that is of importance. Such temporal information is necessary when reasoning about operating physical systems, since the set of symptoms observed at a particular time may represent improvement or deterioration from a previous reading, or may reveal valuable fault propagation information. In a jet engine, for example, the fact that the fan rotational speed was observed to be abnormal prior to an abnormal observation of the compressor rotational speed is indicative that the faulty component is the fan and that the fault propagated to the compressor, rather than the reverse.

[1] Ancient Greek for "expert"

Causality Model

Epaion's causality model contains information such as "fan-blade separation causes the rotational speed of the fan to fluctuate" and "the rotational speed of the fan causes the engine pressure ratio to fluctuate." Along with the causal information between two states, e.g. "inefficient air flow" and "slowing down of the engine", the model maintains a frequency count of the number of times that the system witnessed that inefficient air flow caused the engine to slow down.

Functional Dependency Model

The functional dependency model is a digraph model of an aircraft system, with nodes representing primitive components, and arrows connecting nodes representing functional dependencies. Component B is said to be *functionally dependent* on component A if the proper functioning of B depends on the proper functioning of A. For example, the control surfaces of an aircraft are functionally dependent on the hydraulic system, since they will cease operating if the latter fails. The functional dependency model contains two kind of arrows, representing immediate and non-immediate links between components. Two components C_1 and C_2 are connected via an immediate link (I-link) when C_1's failure propagates immediately to C_2, i.e., abnormal function of C_1 at time t_1 results in abnormal function of C_2 at time t_2 and $t_1 = t_2$. If $t_2 > t_1$ then C_1 is said to be connected to C_2 via an non-immediate link (N-link). For example, if the fan belt in an automotive engine breaks, the fault propagates immediately to the electrical system, as indicated by the generator light, but it may take some time before the propagation to the cooling system becomes evident from the temperature sensor.

Physical Dependency Model

The physical dependency model is a digraph of an aircraft system, similar to the functional dependencies diagraph, in which the links in the graph represent potential paths of fault propagation due to physical proximity. This sort of propagation occurs when uncontrolled discharges of energy attendant on component malfunctions propagate to neighboring systems. The severing of nearby hydraulic lines by blade fragments from a disintegrating turbine provides a typical example.

The Abstraction Hierarchy

The Case-Based Reasoning component of Epaion consists of a self-organizing memory structured as a frame-based abstraction hierarchy, as defined by [Schank 1982]. This memory forms an upper bounded semi-lattice that contains domain specific information at different levels of abstraction. The information contained in the lattice includes:

a. The names of all components in an aircraft engine.

b. The components that are sensors. The exhaust gas temperature, the rotational speed of the fan, and the fuel flow indicator are some of the mechanical sensors in an aircraft's engine. Vision, sight, and smell are the "human sensors" used in the diagnostic process.

c. The possible values for each sensor. For a mechanical sensor the allowable values are: lower than expected; normal; higher than expected. If a sensor initially indicates values that are normal, then at the following time interval indicates values that are lower than expected, and at the third time interval still indicates values which are lower than expected, then the status of the sensor during these three time intervals is *normal, lower, lower* which is a kind (i.e., subcategory) of *overall lower* than expected status which in turn is a kind of *abnormal* status.

d. The various faults that may be observed in an engine subsystem. For example, it is represented that *seagull ingestion* is a kind of *bird ingestion* fault which is a kind of *foreign object ingestion* fault and so on.

e. Information on how faults manifest themselves. For example, fan vibration and abnormality in the rotational speed of the fan are manifestations of a problem in the fan.

f. The accident/incidents that the system already knows. For example the system knows that the incident of a China Airlines Boeing 747 that suffered a mishap over the Pacific Ocean on February 19, 1985 [NTSB-AAR-86-03] is an instance of an accident/incident since it is a kind of *rotor related scenario* which is a kind of *engine related scenario* which is a kind of *accident/incident scenario.*

Reasoning Cycle

Epaion's reasoning cycle consists of the following three phases: input a new problem; retrieve the most similar case; adapt the retrieved case to fit the current scenario.

Epaion's input constitutes a set of symptoms experienced by an airplane's crew during a flight. When the system experiences a new set of symptoms, i.e., when faced with an input (new) case, it searches its case library for the most similar case. This is done by placing the input case in self-organizing MOP[2] memory under the most appropriate parents, determined as described in [Riesbeck & Schank 1989]. The siblings may therefore be assumed to be closely related. The nearest sibling is retrieved as the case that is most similar to the input case.

When the system finds and retrieves a similar case, Epaion assumes that the current fault is the same as the fault in the retrieved case and adapts the causal explanation of the retrieved case to fit the current case. The fault and the causal explanation are both stored in the case library for future usage. The system is provided with a set of adaptation rules which, in addition to adapting the retrieved causal explanation to fit the current case, find possible gaps in the causal explanation and fill in the missing causalities by using the models. This causal explanation connects the symptoms to a justifying cause, and thus the system's multiple-model-based causal reasoning ability produces a causal analysis of the new case, rather than simply a reference to a previous solution. The new causal analysis is not only stored in the case library as part of the input case, but is used to augment and modify the knowledge of the causal model. The following section provides details of this process.

Adaptation and the Models

Epaion's adaptation algorithm is summarized in the following two steps:

The first step involves the transfer of the fault from the library case in the input case and consists of two possibilities.

Case 1: If the match between the input case and the library case exceeds a threshold value then the fault is transferred intact. For example, if in the library case the fault was a malfunctioning fuel controller, then it is assumed to be the same in the input case.

Case 2: If the match is below the threshold value then an abstraction of the library case fault is transferred to the input case. For example, if in the library case the fault was bird ingestion, then it is assumed that in the input case the fault is foreign object ingestion.

The second step involves the adaptation of the causal explanation of the library case so it can explain each, or as many as possible, of the

[2] Memory Organization Packet

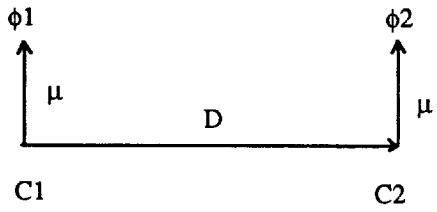
symptoms of the input scenario by connecting them to a justifying cause. This consists of the following possibilities:

Case 1: If the library case and the input case have identical symptoms then the causal explanation of the library case is transferred intact to the input case.

Case 2: If the input case contains symptoms that do not appear in the library case then the causal explanation of the library case is transferred in the input case and the following additional processing takes place. Let $\phi 2$ be an unexplained input case symptom.

Subcase 1: If the causal model contains the relation $\phi 1$ *causes* $\phi 2$, and $\phi 1$ is a symptom or manifestation in the input case, then the link $\phi 1$ *causes* $\phi 2$ is added in the causal explanation of the input case.

Subcase 2: The causal portion of the model does not contain the relation $\phi 1$ *causes* $\phi 2$, but the functional dependency model knows that component C_2 is functionally dependent on component C_1, and $\phi 1$ is a manifestation of abnormal behavior of component C_1, and similarly $\phi 2$ is a manifestation of C_2. This knowledge is depicted by the graph



where ϕ denotes a phenomenon that is a symptom or manifestation μ of abnormal behavior of a component. Additionally, if $\phi 1$ is a symptom in the input case and time($\phi 1$) $\leq$ time($\phi 2$), i.e., symptom $\phi 1$ appeared before or concurrent with $\phi 2$ then the link $\phi 1$ *causes* $\phi 2$ is added in the causal explanation of the input case.

At present, Epaion is implemented to diagnose faults in the engine subsystem of a generic twin engine transport. The programs currently run on various platforms using Common Lisp. Figure 1 displays the use of the various models during the adaptation process.

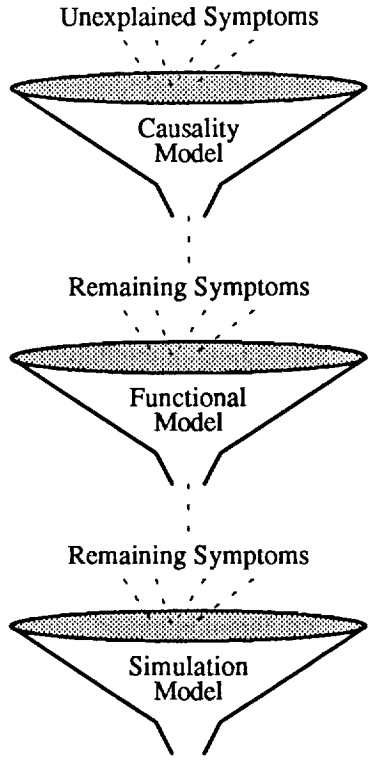


Figure 1: Use of models during adaptation

Simulation and the Physical Model

We have indicated that Epaoin uses a physical dependency digraph as one of its models. This is a makeshift measure, however, due to the fact that physical fault propagation, being the result of catastrophic component failures, is highly unpredictable. One expedient for dealing with this unpredicatability is to refer to previous cases, as Epaion does; another is to utilize spatial simulation models (SSMs) to determine the effect of uncontrolled energy releases. [Feyock & Li, 1990, 1992] describe the use of SSMs to

predict both fluidic and energy leaks[3]. These models, which are easily interfaced with host systems, require only the identity of the faulty component, which can be supplied by Epaion. The SSM then looks in the component database to determine the location and type of the component. If the component is of a type that can cause a fluid or energy leak, the system uses this information to set the initial conditions for the simulation. The simulation is then run, and the physical propagation paths predicted by the SSM are extracted from the run data.

In addition to addressing the chaotic nature of physical propagation, our use of simulation models in conjunction with more traditional reasoning systems is prompted by a belief that deriving answers to real-world questions by setting up the initial conditions of simulation models, running the simulations, and extracting information from the results of the run, constitutes a powerful but underutilized mode of operation for AI systems.

Results

We conducted an experimental evaluation of Epaion on actual aircraft accident/incident cases involving engine faults. Information provided in the individual accident/incident reports from the National Transportation Board (NTSB), the British Air Accidents Investigation Branch (AAIB), and data collected from test accidents staged at Boeing Inc. [Shontz et. al. 1992] was used to derive the appropriate information constituting each case, a process called *accident reconstruction*. We reconstructed a total of eighteen cases, of which sixteen were used as library cases, and six as input cases.

The evaluation process required that each input case be presented to Epaion separately, and that the system produce a diagnosis along with a causal explanation. The diagnosis produced by Epaion was then compared with the correct diagnosis for the particular scenario. In addition, the reasoner was evaluated based on the number of symptoms for which the reasoner was able to find a justification. A "correct diagnosis" is the diagnosis determined by NTSB, AAIB, or by [Shontz et. al. 1992]. Epaion is said to have produced a *complete explanation* if the system was able to explain each observed symptom by connecting the symptom to a justifying cause. The results achieved are very promising for the future success of the system. Based on the results we make the following observations.

- Classification

Five of the six cases in this evaluation were correctly classified. A case involving water ingestion [NTSB-AAR-78-3] was classified under the category of miscellaneous scenarios due to the lack of previously encountered water ingestion scenarios. An, expanded case library will enhance the systems classification capability and therefore offer better matches for each additional input case.

- Diagnosis

Epaion was able to correctly diagnose five of the six scenarios. A case representing the American Airlines Flight 566 scenario [NTSB-F-A067] was properly classified as rotor scenario but misdiagnosed as fan problem rather than turbine problem. This is a result of the fact that problems in the fan and problems in the turbine manifest themselves similarly, and therefore both kinds of faults are classified under the category of *rotor scenarios*. When the American Airlines scenario was used as input case the system retrieved as the most similar case a Dan Air incident [AAI-AAR-4/90], which is a fan blade scenario. With almost negligible difference in the degree of match between the input case and the

[3] We denote as "energy leaks" the catastrophic disintegration of components due to the uncontrolled release of kinetic or potential energy.

C-2

relevant library cases, the second best match was the accident of the United Airlines Flight 611 that took place on July 19, 1970 [NTSB-AAR-72-9]. This is a turbine fault scenario and would have achieved a higher degree of similarity with the input case if the time order of the symptoms in both cases had been represented more precisely. All symptoms used in reconstructing the case of the United Airlines Flight 611 were based on expert opinion, but none were explicitly stated in the NTSB report. With the exception of the behavior of the EGT, the same holds for the symptoms used to reconstruct the American Airlines Flight 566 scenario. This suggests that presenting the system with cases that are reconstructed based on an accurate set of symptoms is vital for correct matching and therefore correct diagnoses.

- Symptom explanation

In five of the cases presented as input Epaion was able to explain all of the symptoms experienced. When Epaion was presented with the symptoms of an icing scenario staged at Boeing [Shontz et. al. 1992] it failed to explain the presence of broad-band vibration. The failure is attributable to insufficient information in the abstraction hierarchy. If the fact that broad-band vibration is a manifestation of fan abnormality had been included in the abstraction hierarchy, the system's functional dependencies model would have explained the broad-band vibration symptom as a result of fan blade damage. The same result would have been achieved if the system had previously experienced other cases with broad-band vibration, thus enabling the causal model to explain the vibration. It is evident that the more knowledge the system contains in its abstraction hierarchy, the better its explanation capability will be. Current efforts are accordingly focused on expanding this knowledge to a substantial size.

Conclusion

Automation of inflight diagnosis and prognosis as an aid to the flight crew has great potential for improving the general safety of civil transport operations. The Epaion Case-Based Reasoning system we have developed for the purpose of performing fault diagnosis and prognosis of aircraft in operation uses a hybrid reasoning process based on a library of previous cases and several types of models of the aircraft as the basis for the reasoning process. This arrangement provides the methodology with the flexibility and power of first-principle reasoners, coupled with the speed of associational systems.

A major concern of this project has been to create a system capable of achieving a practically useful level of performance on a case base of significant size, thereby avoiding the "toy problem" trap besetting many AI systems. The extensive use of a classification hierarchy allows the system to achieve O(log n) search times, while the information abstraction attendant with accident reconstruction produces space-efficient representations. The system is currently hosted on a desktop personal computer, and is estimated to be capable of storing the full set of propulsion related aircraft accident for the last 20 years. These considerations, together with the encouraging level of success achieved by Epaion, support the expectation that this system will prove to be an effective contributor to aircraft safety.

References

AAI-AAR-4/90. (1990). *Report on the accident to Boeing 737-400 G-OBME near Kegworth, Leicestershire on 8 January 1989*, Air Accidents Investigation Branch, AAIB-AAR-4/90.

Feyock, S., and D. Li, Simulation-based Reasoning about the Physical Propagation of Fault Effects, 1990 Goddard Conference oh Space Applications of Artificial Intelligence, NASA/Goddard Space Flight Center, Greenbelt, Md, May 1990

Feyock, S., and D. Li, Qualitative Spatial Reasoning about Fault Effect Propagation, Computer Science Department Technical Report, College of William & Mary, Williamsburg, Va, 1992.

Feyock, S., and S. Karamouzis (1991). LIMAP . Technical Report WM-92-1, College of William & Mary, Computer Science Department, Williamsburg, Virginia.

Karamouzis, S., & S. Feyock (1992). "An Integration of Case-Based and Model-Based Reasoning and its Application to Physical System Faults", *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Lecture Notes in Artificial Intelligence, No. 604 (F. Belli and F. J. Radermacher, Eds), Springer-Verlag, Berlin, 100 - 108, 1992.

Lloyd, A. T. (1990). "Vulcan's Blast", *Airliner*, April-June 1990.

NTSB-AAR-72-9. *Aircraft Accident report: United Airlines, Inc., Boeing 737-222, N9005U, Philadelphia International Airport, July, 19, 1970,* National Transportation Safety Board, NTSB-AAR-72-9.

NTSB-AAR-76-19. *Aircraft Accident report: Overseas national Airways, Inc., DC-10-30, N1032F, John F. Kennedy International Airport, Jamaica, new York, November 12, 1975,* National Transportation Safety Board, NTSB-AAR-76-19.

NTSB-AAR-78-3. *Aircraft Accident report: Southern Airways Inc., DC-9-31, N1335U, New Hope, Georgia, April 4, 1977,* National Transportation Safety Board, NTSB-AAR-78-3.

NTSB-AAR-86-03. *Aircraft Accident report: China Airlines, Boeing 747-SP, N4522V, 300 Nautical Miles Northwest of San Francisco, California, February 19, 1985,* National Transportation Safety Board, NTSB-AAR-86-9.

NTSB-F-A067. *Aircraft Accident report: American Airlines, Inc., Boeing 727-023, Erlanger, Kentucky, May 21, 1978,* National Transportation Safety Board, File No. IAD-78-F-A067.

Riesbeck, C. K., and R. C. Schank (1989). Inside Case-Based Reasoning . Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schank, R. C., (1982). Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press.

Schutte, P. C., K. H. Abbott, and W. R. Ricks, W. R. *A Performance Assessment of a Real-time Diagnostic System for Aircraft Applications*, Unpublished Technical Report, NASA Langley Research Center, Hampton, Virginia.

Shontz, W. D., R. M. Records, and D. R. Antonelli (1992). ``Flight Deck Engine Advisor Final Report," NASA Contractor Report 189562, Langley Research Center, Hampton, Virginia.

An Autonomous Satellite Architecture Integrating Deliberative Reasoning and Behavioural Intelligence.

Craig A. Lindley

Department of Computer Science and Engineering
University of New South Wales, Kensington, NSW, Australia

ABSTRACT

This paper describes a method for the design of autonomous spacecraft, based upon behavioural approaches to intelligent robotics. First, a number of previous spacecraft automation projects are reviewed. A methodology for the design of autonomous spacecraft is then presented, drawing upon both the European Space Agency technological centre (ESTEC) automation and robotics methodology and the subsumption architecture for autonomous robots. A layered competency model for autonomous orbital spacecraft is proposed. A simple example of low level competencies and their interaction is presented in order to illustrate the methodology. Finally, the general principles adopted for the control hardware design of the AUSTRALIS-1 spacecraft are described. This system will provide an orbital experimental platform for spacecraft autonomy studies, supporting the exploration of different logical control models, different computational metaphors within the behavioural control framework, and different mappings from the logical control model to its physical implementation.

Keywords: Spacecraft Control, Space Robotics, Artificial Intelligence, Subsumption.

Introduction

AI applications in space systems are becoming more readily accepted, and constitute a key enabling technology for ambitious projects such as the Space Station Freedom and Space Exploration Initiative. Current or proposed constellations of unmanned spacecraft, particularly in low earth orbit, and multiple deep space missions with long telecommunication propagation delays, can also gain substantial benefits from the use of more autonomous spacecraft operation. Teleoperation of industrial space facilities and orbital experimental platforms using highly autonomous onboard systems may provide crucial competitive advantages in the commercial and industrial exploitation of space.

This paper presents an architecture for autonomous spacecraft control that supports the integration of behaviour-based approaches to emergent intelligence with numerical and computational simulation models, and symbolic reasoning systems such as expert and knowledge based systems. Firstly, a methodology is proposed for developing autonomous space systems. Using this methodology, system operational functions are hierarchically decomposed, but functional levels are not mapped directly onto computational models. The operational decomposition is used to refine specifications of layered competencies, based upon a generic layered competency model. Each competency level defines a virtual machine interface from the point of view of superordinate levels. Hence, the hierarchical decomposition of system functionality during operational analysis does not imply a strict corresponding hierarchical synthesis for design and implementation, but provides a framework for specifying system behaviours and resources, and for understanding their interactions.

The realisation or implementation of the functionality of successive virtual machines can be carried out using the most appropriate computational paradigm, or a rich combination of paradigms. From this point of view, a knowledge base or expert system can be regarded either as a convenient abstraction adopted during the design process to define the input/output behaviour of behavioural modules, or as a resource for use by behavioural modules much as human

operators would use expert systems for particular tasks.

A simplified example application of this approach is presented. The resulting satellite control architecture is significantly different from previous satellite designs, having improved robustness, decreased operating overheads, and more autonomous fault tolerance. Current plans are to validate and refine this approach in a rich simulation environment, and eventually to build and operate a satellite for ongoing in-orbit trials and experiments.

Precedents

Onboard autonomy is a matter of degree. Pidgeon et al (1992) describe how a number of current spacecraft have mechanisms for fault detection, reporting, and subsequent switching to component, subsystem, or system fail-safe modes. Human operators must then diagnose faults and initiate appropriate contingency recovery procedures. Increasing abstraction levels in spacecraft command languages have also been adopted. For example, in normal operations, the Hipparcos spacecraft is controlled by processed commands which are sent to the onboard computer for distribution to other systems, and for possible time tagging. Direct commands are also available, which bypass the onboard computer as a backup in the event of computer failure, and for more direct access to the controlled systems. Priority real time commands can also be issued, which allow direct switching of systems. 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 lowest levels of commands bypass the onboard computer and data handling system, again ensuring control if those systems fail. The EURECA system, comprising fifteen separate payloads, uses an onboard Master Schedule which contains a list of time tagged command macros for execution by the onboard data handling system. Those commands include rudimentary failure routines, backed up by safe modes to deal with command loop failure. Again, direct telecommands can also be used, to bypass the data handling system and onboard computer in the event of their failure.

The degree of onboard autonomy is continuously increasing, and can be expected to incorporate a wide variety of techniques from AI and intelligent robotics research. A number of prototype systems have been built to investigate onboard expert system applications, including DIPOLE, SAGES, APS, SACV, and SICON (see below). Operational systems may include the Cassini Titan Probe, and many Space Station Freedom applications. Most of these systems involve onboard architectures comprising a number of distinct modular functions. Autonomous systems of increasing size and complexity tend to have distributed functionality, with the various functions running on separate physical processors. Another recurrent theme is the devolution of autonomous functions to the lowest possible abstraction levels.

Tello (1986) describes DIPOLE, a system for satellite control which is intended to integrate "shallow" heuristic or rule based reasoning with "deep" model-based reasoning. The shallow system uses fault-tolerant mathematical and algorithmic subroutines, and has the form of real-time expert systems with data-driven switches for controlling their performance. The aim is for the deep reasoning system to take over when the shallow system gets into difficulty, and to allow the shallow system to resume when the deep reasoning system has resolved the problem. The DIPOLE architecture addresses two particular problems for real time deliberative controllers. *Circumspection* is the problem of enumerating all implicit conditions and assumptions associated with given knowledge, and the ability to handle situations when these are no longer valid. *Inference thrashing* is the situation when inferencing cannot produce solutions quickly enough to keep up with changing circumstances. DIPOLE seeks to address these problems by using shallow rule based expert systems as reflex processors in real time, with longer deliberative processes performed by the deep reasoning system.

Ciarlo et al (1987) describe a spacecraft expert system prototype study conducted for the

European Space Agency. Some of the conclusions of the initial study include:

- highly simplified interfaces typical of spacecraft modular units reduce integration and control problems. However, this severely limits the information available for monitoring each unit, and the choice of actions available to correct failure, to the point of making the advantages of expert systems questionable when compared to standard algorithmic or table-driven software.
- it is difficult, and not necessarily advantageous, to use an expert system in a satellite designed without this in mind.

Ciarlo and Schilling (1988) report upon work following on from this initial study to consider an expert system embedded within the Cassini Titan probe, for autonomously managing the descent of the probe into Titan's atmosphere. The authors note that to keep the complexity and susceptibility of the system to faults as low as possible, the autonomous system should be implemented at the lowest possible level, with capabilities such as component and sensor self testing and redundancy switching. Scientific management, which involves adaptation to the situation according to complex rules, is regarded as an appropriate function for implementation as a knowledge base. Engineering management, involving FDIR and subsystem control, is regarded as an appropriate function for conventional technology.

The Satellite Autonomy Generic Expert System (SAGES) architecture, developed by Rockwell, is based upon the definition of four intelligent agents, corresponding to phases of the mission operation cycle, including planning, scheduling, execution, and analysis (providing feedback into the planning phase; Raslavicius et al, 1989). A SAGES prototype has been developed for a "typical" surveillance satellite.

The Boeing Aerospace Autonomous Power System (APS) testbed has been assembled for use in developing improved control techniques for aerospace electrical power systems (Spier and Liffring, 1989). The main emphasis of APS is the development of a programming environment to properly control the concurrent execution of multiple autonomous algorithms coupled with continuous input and output data flow. Expert system functions include fault diagnosis and recovery, and battery charge control. The expert systems use event-driven processing within a blackboard environment.

The European Space Agency (ESA) Standard Generic Approach to Spacecraft Autonomy and Automation (SGASAA), is a hierarchical model which aims to devolve decision-making to the lowest possible level (Pidgeon et al, 1992). To this end, it is a distributed onboard architecture, with each payload and subsystem having a certain degree of "intelligence", in addition to an Onboard Mission Manager (OBMM) responsible for the control of the spacecraft as a whole. Separate subsystem managers are intended to handle their own failures and report the results of their diagnoses via LAN to the OBMM, along with a proposed recovery action. The OBMM can authorise the proposed recovery, or block it if the failure is caused by a failure elsewhere. SACV (*ibid*) is an investigation of the SGASAA concept involving the implementation of a fully autonomous spacecraft based upon EURECA.

SICON, built by LISP Machine Inc, is a simplified prototype system for satellite intelligent control, concentrating upon the electrical power system (Leinweber, 1987). The SICON prototype deals with load distribution and switching, solar array orientation, power system verification and checkout, fault diagnosis, trend analysis, contingency management, battery charge and reconditioning-cycle optimisation, and fuel cell monitoring and control. Leinweber notes that there are a number of SSF processes and subsystems that are amenable to real-time process control, including the electrical power system, attitude and orbital control system, environmental and life support system, propulsion system, monitoring of docked vehicles, manufacturing process control, and ground communications and network control. Such real-time onboard applications require particular expert system features, including high-speed context-sensitive rule activation, efficient memory recycling, acceptance of

interactive commands without suspending execution, and communication between multiple expert systems in order to provide redundancy.

The Space Station Freedom (SSF), will require an extensive data processing support environment. The communications and information processing backbone of the SSF is the Data Management System (DMS). The DMS has the dual role of providing hardware resources and software services which support data processing and communications needs of the system, its elements, and payloads (Erickson, 1987). It also functions as an integrating entity, providing a common operating environment and human-machine interface for the operation and control of orbiting SSF systems and payloads by both the crew and ground operators. The DMS provides signal conditioning, and timing synchronisation of data required for interpreting time-critical information and results between expert systems, knowledge-based systems, and robotics elements distributed throughout the SSF environment. Woods (1992) notes that the DMS may use artificial intelligence techniques for fault detection, isolation, and recovery (FDIR) on DMS components. An Integrated Systems Executive (ISE) will provide overall software scheduling and control for all other systems, experiments, and elements. Low level software modules will take care of time critical control loops and fault recognition. Development projects are currently underway in a number of SSF expert system applications.

A Methodology for Autonomous Spacecraft Development

In the ongoing development of autonomous spacecraft, devolution of decision-making to the lowest possible level, and the modularisation and distribution of functionality, are prominent trends. These trends, driven by the particular requirements of real time autonomous agency, have been most fully developed in the context of mobile robotics research, and are captured most strongly by behavioural approaches to autonomous systems design. Behavioural approaches to mobile robotics have also demonstrated more fundamental benefits in addressing the problems of circumspection and inference thrashing that have plagued deliberative robot control systems. There is therefore considerable potential for behavioural approaches to contribute to the increased automation of space systems. Toward this end, this paper proposes a methodology for autonomous spacecraft development which draws from both behavioural approaches to mobile robotics and an ESTEC (European Space Agency's technological centre) methodology for space automation and robotics. While behaviour-based robots have been suggested for planetary surface exploration (Brooks and Flynn, 1989), the behavioural paradigm has not previously been used to design orbital spacecraft. Hence, the proposed methodology is not fully articulated, but groundwork for a more complete approach is presented. Indeed, it is probably dangerous to suggest any comprehensive standardised approach until behaviour based spacecraft have been well demonstrated, and the paradigm and its benefits in this domain are well understood.

The main technical objectives of a design methodology include the achievement of full bidirectional traceability between user requirements and system solutions selected, the breakdown of complex problems into successively simpler ones, unity of system architecture, rigorous interfaces between subsystems, improved communications with end users, precise communication within a development team, efficient parallel development of subsystems, reduced control complexity, greater simplicity of design, and sound data analysis and administration (Elfving and Kirchoff, 1991). The resulting benefits include a high-quality product which is easy to maintain and upgrade, better project control during the design process, reduced time to completion, and lower cost for system development.

Elfving and Kirchoff (1991) present logical reference models which postulate the essential functions of an automation and robotics system, to be used for structuring user requirements and transforming them into distinct design solutions. This ESTEC methodology is derived from structuring

principles of hierarchical decomposition and hierarchical structuring, fundamental properties from disciplines such as control theory and mechanical engineering, and generic principles of structured analysis and structured design. It is characterised by a clear separation between operational analysis and system synthesis, and the application of the principle of abstraction in the form of reference model techniques. While the ESTEC methodology has many desirable features, it does *not* immediately lend itself to the synthesis of behaviour-based autonomous systems. Behavioural approaches are, however, in need of methodological guidelines to support a systematic composition of behaviours into competencies, to assist in ensuring that the resulting system design will meet its overall requirements for a given application (Brooks, 1990, 1991). It is therefore valuable to draw from both the behavioural approach and the ESTEC methodology, in order to achieve a methodology combining the benefits provided by both.

Essential Characteristics of the Subsumption Methodology

The subsumption architecture was specifically developed to address requirements for autonomous mobile robots (see Brooks, 1986, 1990, and 1991) including multiple, often conflicting, goals, multiple sensors, robustness, and extensibility. The approach is based upon a number of principles which can be drawn upon and adapted here as principles which bear upon the spacecraft autonomy problem:

1. complex or "intelligent" behaviour can be an emergent phenomenon, arising from the interactions of a spacecraft with its environment and users.

2. component interfaces should be simpler than the components that they interconnect.

3. if a module solves an unstable or ill-conditioned problem, then it is probably not a robust solution.

4. autonomous model-making is important, since idealised models may be inaccurate.

5. the spacecraft must operate in a three-dimensional world.

6. relational models can avoid the cumulative errors that characterise absolute coordinate systems.

7. there is no global internal model, or global planning activity with a hierarchical task structure.

8. for robustness, the spacecraft must be able to perform when one or more of its sensors fails or malfunctions. Recovery should be rapid, so built-in self-calibration is required at all times.

9. the spacecraft control problem is decomposed in terms of layers or levels of competency. Those levels are task-achieving behaviours, and as such are external manifestations of the control system. "Higher" levels correspond with more specific classes of behaviour.

10. each successive level subsumes as a subset each earlier level, and provides additional constraints on the class of valid behaviours defined by the earlier levels.

11. successive levels *extend* competency, but do not alter the structure of the implementation of lower levels. A level can receive data from lower levels in order to monitor their behaviour, and can output data to lower levels in order to modify behaviours by inhibiting or exciting them.

12. competencies are parallel processes. Hence, lower level competencies can ensure that spacecraft behaviour is sensible, while higher level competencies take time to produce more optimal control solutions.

13. there is no central locus of control, either for the system as a whole, or within any particular competency layer. This is essential for robustness.

14. each layer can run on its own processor, and individual layers can also be run over many loosely coupled processors.

15. within each particular layer, a traditional decomposition of functionality may be used "to some extent".

These principles conform with the paradigm for autonomous systems design which Maes (1990a) refers to as the *behavioural* approach. Inspired by biological models of autonomy, the behavioural approach has abandoned the older *sense-model-plan-act* (SMPA) paradigm (Brooks, 1990), and in so doing has achieved many successful demonstrations of greatly improved robot performance and robustness. Those demonstrations represent explorations within the new paradigm, but can by no means be regarded as definitive or fully matured.

It is important to distinguish the logical design of an autonomous control system from its implementation. Some explorations of the behavioural paradigm have concentrated upon software design, with the software being compiled to run on a single embedded processor (eg. the MIT Squirt robot, Brooks 1990). However, while such systems can demonstrate the effectiveness of a control architecture based upon situatedness and behavioural interaction, rather than model-based deliberative reasoning, the use of a single physical processing element creates a single physical locus of control, and therefore a single point failure mode in the resulting robot. The behavioural paradigm is a *systems* paradigm. As such it should lead to a distributed hardware functionality of the kind that typifies the more sophisticated behavioural robots. Research within the new paradigm continues at a vigorous pace, and the question of how competencies at a conceptual level can be achieved as emergent properties of increasingly parallel and distributed computational processes at the implementation level has only just begun to be investigated. Continuing advances in parallel and distributed hardware architectures reinforce the viability of the behavioural approach, and demand a radical rethinking of how autonomous systems can be structured.

A number of researchers have adopted the behavioural approach as a method for designing the lower level control functions within an autonomous system, and have then provided one or more "layers" above the behavioural layers which perform higher level deliberative and symbolic computations. For example, Steels (1991) proposes a frame based system for "high level" cognitive tasks (such as language use), in which frames are grounded by sensory inputs and influence the behaviour of the system in proportion to their "fit" to the current situation. Arkin (1990) describes the Autonomous Robot Architecture (AuRA), a framework for experimenting with the integration of behavioural approaches with model-based reasoning. AuRA allows the advantages of modularity, incremental design, adaptability, and robustness of the behavioural approach to be supplemented by the use of model-based knowledge to configure behavioural strategies in an efficient form. The AuRA architecture comprises five basic subsystems: Perception, Cartographic, Planning (both a hierarchical planner and a distributed reactive plan execution subsystem), Motor (the actuator set interface), and Homeostatic control (monitors internal conditions of the robot for both the higher level planning mechanisms and the motor schemas). Flexibility is incorporated into the AuRA system by drawing modularised behavioural patterns and sensory strategies from a library and configuring them to meet the needs of a particular mission and any known environmental constraints. World models play an important role in this configuration process. Bonasso (1991) also describes an architecture in which a declarative model is used for reasoning about plans and controlling the activation of behaviours implemented within an underlying subsumption layer. Malcolm and Smithers (1990) describe SOMASS, a robot assembly system that combines a PROLOG assembly planning subsystem with a plan execution subsystem that handles uncertainty by means of behavioural modules which accomplish useful motions of assembly parts. Malcolm and Smithers note that, although the "cognitive" planning function was implemented in a high-level symbolic language, and the "subcognitive" execution agent in a low level language, these decisions were motivated by convenience. Also, the cognitive/subcognitive distinction was itself found to be a useful construct for the SOMASS system, but is not a necessary or convenient construct for artificial mentality in

general. The interface between the cognitive and subcognitive systems presents a virtual machine model to the cognitive system, which heavily determines and permeates the design of the cognitive system. The need to present a virtual machine interface to the cognitive system results in a modularity of behaviours in the subcognitive layer, which is not needed in approaches (eg. subsumption) which do not include a cognitive component. Gat (1991) describes ATLANTIS, a system which integrates behaviours and deliberative processes in a three layered structure. The first layer comprises a behaviour-based system for robust motion control and low-level competencies. A deliberative layer provides high level reasoning, such as plan generation. These layers are joined together by a sequencing layer based upon Reactive Action Packages.

These different explorations of the relationship between deliberative reasoning and behavioural autonomy are not driven by any well-proven limitations of the behavioural approach. As Brooks (1990, 1991) notes, the question of how far a behavioural approach without the use of deliberative computations can go in achieving higher levels of competence in autonomous agents is one which must be addressed by ongoing empirical investigations. However, an issue arises within the behavioural paradigm regarding the range of computational metaphors that can usefully be adopted for designing behavioural units. In the subsumption architecture, Brooks uses augmented finite state machine (AFSM) models to define primitive behavioural elements, and AFSMs are further grouped into more complete behaviours. However, any number of computational metaphors can potentially be used to describe a system with a given transfer function. The effectiveness of AFSMs has been demonstrated for some behaviours, but metaphors of multiple interacting agents (see Grant, 1992), objects, processes, production systems, etc. may equally provide convenient metaphors. Similarly, knowledge-base systems, rule bases (AFSMs are defined by rule sets in Brooks' behaviour language), and expert systems metaphors may be convenient. The metaphor adopted should be that which most "naturally" describes the behaviour of a module from the perspective of the system designer. Adopting the behavioural paradigm for the overall control system architecture does not rule out the adoption of other metaphors for structuring and designing the computational processes within a given behaviour or module in order to achieve a desired set of input/output mappings. The danger in adopting heterogeneous metaphors within a behavioural control system is if any metaphor distorts the behavioural framework by encouraging the centralisation of behavioural coordination and control, or the centralisation of data flow. It can be argued that this should not occur in the case of spacecraft control if the control system is modelled upon manual spacecraft operation, since ground based, manual spacecraft control involves highly distributed, cooperative decision-making by numerous "agents" of mixed expertise, generally in a way that is highly redundant, robust, and adaptive. The behavioural paradigm is not violated by experimentation with alternate metaphors for developing the *internal* structure of behavioural modules *within* a behavioural control framework.

Towards Reference Models for the Behavioural Methodology

The reference models described by Elfving and Kirchoff are intended to provide clear traceability from user requirements to design solutions, which justifies technical decisions made and avoids excessively flexible, expensive, and complex systems with high technical risk. The reference models are intended to cover robots, mobile vehicles, and process control, and reflect the operational use of the system. Elfving and Kirchoff divide A&R capabilities in space into three major fields:

- external servicing of payloads
- servicing of scientific experiments within pressurised orbiting laboratories
- surface mobility and sample acquisition for planetary exploration

To this can be added the field of current concern:

- autonomous orbital spacecraft control

The starting point for this ESTEC automation and robotics (A&R) control design methodology is an established conceptual layout for the A&R system, typically available as a result of mission and system definition studies. The logical reference models capture the essential principles of this methodology.

The objective of *Operational Analysis* (or *Task Analysis*) is to derive the essential abilities of an A&R system such that mission objectives are sure to be fulfilled. This is done by a step-by-step decomposition into levels of equal importance, from mission objectives down to a level of elementary actions. This defines "what has to be done", ideally without anticipating any solution, but by using initial knowledge about system functional layout based upon initial mission and system definition studies. This analysis phase requires system operational expertise. *System Synthesis* involves the definition of solutions that satisfy the different operational features required at the various levels of decomposition, thereby addressing "how is it to be done?". System Synthesis involves an aggregation process, from elementary abilities to system capabilities. Synthesis requires A&R technology expertise.

Given these processes of analysis and synthesis, the key methodological question is that of how to achieve traceability between the two areas, assuming that the decomposition logic of analysis must be in agreement with the synthesis logic, so the inherent structure of the analysis process is a virtual system solution. This nexus is achieved by the use of the logical reference models. Three logical reference models are distinguished:

1. *Functional Reference Model (FRM)*: represents a decomposition of all functions and information flow and structures, and is valid for all A&R applications.

2. *Application Reference Model (ARM)*: represents an FRM derivative which focuses on individual classes of A&R applications.

3. *Operations Reference Model (ORM)* : represents an FRM derivative which focuses on models of operation and systematics for man/machine and preparation/utilisation allocation.

The objectives of applying reference model techniques are:

- guarantee a common 'thinking model' which is valid for all A&R applications, to enable and ease communication within and between development teams
- provides a generic system information structure and identifies essential functions for which application-specific design solutions should be found
- establishes "rules-of-thumb" and heuristic design strategies that allow the designer to systematically derive good and relevant solutions to common types of problems
- uses formal documentation techniques and graphic support tools that emphasise the hierarchical functions and information flow and structures of complex systems

A *logical model* represents the essentials of a system, assuming ideal internal technology of the system and excluding application-specific topics. A physical model represents the implementation of a specific application, and therefore needs to represent the actual constraints imposed by the chosen internal technology. Hence, only logical models can be reference models that meet the requirement of being valid for a multitude of applications and/or implementation technologies. However, a major goal of a design methodology is to maintain a strong and traceable link between the logical and physical models of a system in order to achieve design commonalities and open implementation architectures.

A Behaviour-Based Functional Reference Model

The FRM proposed by Elfving and Kirchoff involves a hierarchical ordering of functions and information with increasing precision and decreasing planning horizon from top to bottom, where the hierarchical information interface layers are:

1. A&R mission: the objectives of end users

2. task: clustered activities performed on a single subsystem defined as the element submitted to motion or actuation by the A&R system

3. action: elementary activity for a functional subsystem

This decomposition leads to hierarchical functional and information layers, with the controlled devices and processes at the bottom, and successive layers for action, task, and mission execution, planning and control. State information flows from each layer to the next high layer, while commands and activity attributes flow down through the structure.

This hierarchical decomposition, with "vertical" divisions between distinct objectives, tasks, and actions at each level, is not compatible with a behavioural methodology. To be compatible with a behavioural approach, the following modifications are required:

- instead of a decomposition of system functions in terms of a tripartite structure of mission, tasks, and actions, the system can be decomposed in terms of layered competencies. This overcomes the somewhat arbitrary parsing of activity into three levels, providing a more flexible layering and abstraction mechanism which subsumes and extends the tripartite structure. It also has the advantage of retaining the visibility of what the system is required to do, rather than decomposing system functions according to how those functions are implemented. System requirements can be mapped directly onto competencies, and competencies then onto their implementation.
- instead of placing all sensor and actuator interfaces at the lowest level of a hierarchical structure, each competency level is associated directly with a subset of the total set of sensors and actuators required, in addition to having interfaces to the competency levels above and below itself.

A&R Mission planning and control can be retained as the highest general level of system competency. The lowest level is also general, and comprises basic survival competencies. Intermediate levels are variable in number and form across different A&R application classes. The resulting vertical FRM structure is then as shown in figure 1.

Elfving and Kirchoff describe forward control, nominal feedback, and non-nominal feedback functions across all hierarchical levels. Forward control involves activity decomposition and execution planning, based on a priori knowledge, and plan execution. Nominal feedback deals with forseen refinements and updates to ensure that the system achieves the forward control execution goals. Non-nominal feedback covers the case of system performance diverging from the allowable region around the nominal execution plan and is realised by means of monitoring discrepancies between actual and allowable states, diagnosis of reasons for possible discrepancies, and generation of recovery strategies and constraints.

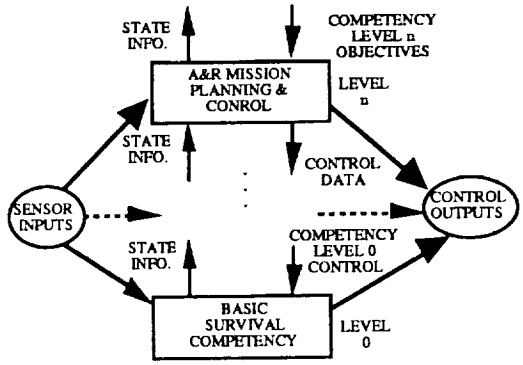


Figure 1. Behavioural FRM.

This model may be used to address individual competencies in the vertical decomposition of system functionality. However, nominal feedback, non-nominal feedback, and feedforward control functions should not be regarded with restricted computational paradigms in mind. It must be regarded as an open question how the respective functions can best be modelled for any particular application or level of problem abstraction. It is yet another question how the modelled functionality should then be implemented. For example, a logical model of planning may or may not be the best way of specifying a type of planning activity that may be required.

Moreover, if the planning activity is defined logically, the implementation of the planner may take quite a different form, such as that of a highly distributed and emergent function (eg. gradient field approaches, as described by Payton 1990, analogical planning, as described by Steels 1990, or emergent goal arbitration, as described by Maes 1990b). In a behavioural system, inputs and outputs for these functions include behaviour monitoring and control signals, in addition to sensor inputs and actuator outputs, depending upon their role within the competency in which they occur. In other words, nominal feedback, forward planning, and non-nominal feedback become *behaviours* within a competency layer.

These redefinitions result in a very different FRM, less detailed than that presented by Elfving and Kirchoff, and as such placing more importance upon Application Reference Models for the different classes of A&R applications.

Behavioural Application Reference Model

The Application Reference Model (ARM) tailors the functional blocks and the information structure of the FRM to the characteristics of each class of A&R application, in order to ease acceptance and understanding. Elfving and Kirchoff identify these classes as:

- motion systems with fixed linkages to the environment (eg. robot manipulators)
- mobile systems (eg. moving vehicles)
- continuous processes (eg. climate control)
- event/sequence control (eg. PLC equipment)

To this can be added the class of current concern (a subtype of mobile systems):

- autonomous orbital spacecraft

In physical realisations, a combination of these classes may be integrated to form the overall A&R control function.

The ARM leads to a more detailed decomposition than the FRM. From the behavioural viewpoint this amounts to identifying for each class of application:

- the competency layers required
- generic requirements for nominal feedback, non-nominal feedback, and forward control within each layer
- individual behaviours within each competency

Ongoing research is needed to identify alternate implementation strategies for various competencies, and to systematically analyse the tradeoffs between different strategies as a basis for rational design decisions.

Proposed ARM for Autonomous Orbital Spacecraft

Due to very significant domain differences, the competency layers proposed here for autonomous orbital spacecraft bear little resemblence (other than at the highest level) to those proposed by Brooks (1986) for biologically-inspired mobile robots operating in earth-gravity environments. Some of these differences arise due to the different operational environment and sensor and actuator sets of orbital spacecraft in comparison with mobile surface robots. Other differences arise due to a major aspect of spacecraft functionality which concerns the collection, distribution, and reception of data.

The definition of layered competencies is made in terms of decreasing criticality and increasing autonomy. In this sense, the approach represents an extension of automated safe mode transitions into a more complex set of behaviours, and a set of autonomous operations which function during normal spacecraft operation in addition to emergencies. Consideration of the layered structure shows the non-hierarchical nature of the control system. Mission critical survival decisions, often made at the lowest levels, must have priority over higher-level decision making during emergency situations. However, the higher levels must be able to modify resource allocations, timing relationships, and data flow within the lower levels in order to establish priorities between activities within a wide range of variation of nominal operating modes in order to achieve higher levels of competence in autonomously meeting and optimising mission objectives.

For example, critical docking operations cannot have their power supply interrupted. Ensuring that this does not occur can involve power conditioning schedules that come into operation well in advance of the docking activity, and therefore must be planned by higher competency levels, and those plans then conformed to or used by the behaviours of the lower levels.

A proposed competency model is shown in figure 2. The Maintain Thermal Balance competency is placed at level 0 as the most fundamental precondition of spacecraft survival. That is, any significant over- or under-heating can permanently damage or destroy the spacecraft. However, temperature variations tend to be gradual, and good thermal design can ensure that the normal range of temperature variations for the spacecraft is limited to a 20° range. The greatest danger is from more localised temperature fluctuations (especially accumulation of heat), possibly due to faults, which can be dealt with by temperature dependent inhibition or excitation of heat generating processes.

9	Plan and Execute Mission Based Upon Objectives
8	Rendezvous and Dock (| Land | Take Off)
7	Plan and Schedule Data Acquisition and Transmission
6	Plan and Execute AOCS Maneuvers
5	Acquire, Condition, Downlink Instrument Data
4	Maintain Attitude Stability and Orbit
3	Acquire, Condition, and Downlink Engineering Data
2	Maintain Telecommand Override
1	Maintain Battery Condition
0	Maintain Thermal Balance

Figure 2. Competency Layers for Orbital Spacecraft.

The Maintain Battery Condition competency maintains a battery charge/discharge cycle that will ensure long battery lifetime. This must be adaptive in the face of variations in battery and solar cell characteristics over the course of their lifetime, and must accommodate individual cell failures and variable solar irradiation cycles.

Levels 0 and 1 ensure spacecraft survivability, while successively higher layers are concerned with spacecraft accessibility, performance, and increasing independence from the ground. The Maintain Telecommand Override (level 2) competency is provided to ensure that, so long as the survival of the spacecraft is not threatened, the behavioural control system can be overidden by direct ground control. It is not, however, envisaged that this should sever the layered structure of the behavioural system. Rather, it provides an orthogonal access mechanism by which it is possible to inject data into the behavioural control system, and should have the capacity to override internal interconnections between spacecraft behaviours. Hence this level ensures command access to the spacecraft.

Acquistion, Conditioning and Downlink of Engineering Telemetry data (level 3) ensures access to data which is critical for diagnosing and understanding the status of the spacecraft from the ground. Maintainance of Attitude Stability and Orbit (level 4) are secondary priorities after maintaining the engineering telecommunications link, since the link is vital for understanding the state of the spacecraft, and initiating telecommand override if necessary.

The Acquisition, Conditioning, and Transmission of Instrument Data (level 5) can be ensured as a level above maintaining basic survival, two way telecommunications, and stability. Stability control (level 4) is an operational precedent for Planning and Execution of attitude and orbital control (AOCS) Maneuvers (level 6), and the acquisition and transmission of instrument data must also be accounted for in the AOCS maneuver planning process. AOCS Maneuvering competency is a precondition for autonomous Planning and Scheduling of Data Acquisition and Transmission (level 7).

Rendezvous and Dock (level 8) is a high level competency. Although this is already

automated at a low level in systems for docking Salyut and Progress vehicles with the MIR space station, in an integrated, behavioural control system it occupies a high level in order to automate all of the planning and resource allocation activities associated with the basic rendezvous and docking procedure. Landing and Take Off competencies are mentioned at this level as competencies of orbital spacecraft which also land on and take off from planetary bodies. Plan and Execute Mission Based Upon Objectives (level 9) allows the highest level of command abstraction in specifying behaviour desired of the spacecraft.

Operations Reference Model

The FRM and ARM represent general functions necessary to decompose a global task into process variables and the related information architecture. The Operations Reference Model (ORM) addresses a different question, that of how the overall A&R system is operated. The aim is to help to structure the definition of operating modes and achieve a clear and common understanding of what is meant. The ORM can also be used as a support tool for a systematic and consistent draft specification of man/machine task allocations, and allocations of which tasks are to be performed on the ground and which on the spacecraft.

Virtual machine models are useful as a basis for increasing command abstraction. Hence, on the basis of the ORM, separate virtual machine models can be used for telecommand generation and subsequent injection of data into each competency level. This provides a basis for telecommand language specification at several abstraction levels.

From Analysis to Design Synthesis

The primary sources for the operational analysis are the global spacecraft mission objectives, the characteristics of the process to be handled, and the layout of the spacecraft devices to handle this process. The application of the reference models means that the decomposed activity hierarchy of the operational analysis can be used to define functional and performance specifications for system competencies. Hence the hierarchical structures defined during operational analysis are used to develop specifications for the non-hierarchical layered competency model, and this competency model provides the basis for system design synthesis, as shown on Figure 3.

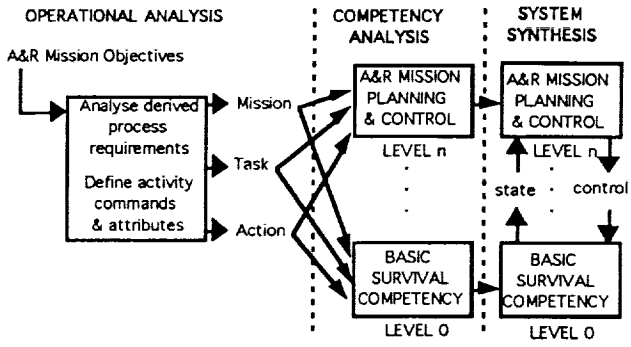


Figure 3. Sequence from analysis to synthesis.

System synthesis involves firstly finding the intermediate detailed specification of system competencies based upon a general Application Reference Model and the operational analysis, and then finding the appropriate design solution for the competency layers of the controller, as shown in the right hand side of Figure 3. This results in system solutions that are valid for classes of mission scenarios, in that activities to be performed become more generic towards the lower decomposition levels, supporting a 'standard set' of competencies which can accommodate evolving or changing mission task requirements without a critical impact upon the design.

Control System Synthesis

Elfving and Kirchoff (*ibid*) suggest that clear traceability of solution decisions made during design synthesis can be achieved by an inverse of the design process: use is made of a set of possible technical solutions which originates from existing fundamental engineering theories and which can be systematically ordered into solution trees by applying hierarchical structuring principles. While this is a desirable goal, it depends upon a well understood synthesis process. For a behavioural approach to orbital spacecraft design, this is premature. Since the spacecraft

control architecture presented here is novel, it represents at least one traversal of a more comprehensive solution tree, yet to be fully articulated. There are also major ongoing research questions regarding the relationship between the logical definition of system competencies and the implementation of the logical design (as discussed above).

Control System Example

The version of the subsumption architecture adopted for current purposes allows a module input to be suppressed by an output from another module, and the input signal is replaced by the suppressing signal (following Brooks, 1986). Limited aspects of control will be considered here to illustrate the principles involved. Behavioural interactions to achieve level 0 and level 2 competencies within a very simple model will be considered. Sensor inputs for this example comprise a set of temperature sensors. Actuators will comprise a telemetry transmitter and a receiver, each with controllable power levels (telemetry data flows will not be considered).

The level 0 competency maintains the thermal balance of the spacecraft. Temperature sensors distributed throughout the spacecraft and associated with each controllable component and subsystem are the primary inputs to this competency. The hypothetical spacecraft is a very simple design, and the only active temperature control available is by increasing or decreasing the power consumption, and hence heat dissipation, of controllable units. To achieve adaptivity and maximum robustness, the sensor inputs are mapped into an analogical representation of the temperature of the spacecraft (Steels, 1990, describes analogical maps). The analogical representation stores a model of each heat generating component of the system, conductive paths, uncontrollable heat sources and sinks (eg., the sun and "dark" space, respectively), and the spatial interrelationships between these elements. The resulting analogical representation resembles a low-resolution finite-element model. For each controllable temperature source in the model, there is a separate behaviour which computes a power level signal as a function of the current and previous temperatures of the particular element, its immediate spatial neighbours, the heat transfer characteristics of their interconnections, and an approximate measure of the specific heat of the component. This method deals very robustly with the failure of any given component to respond to direct temperature control, by calculating power (and therefore temperature) levels as a function of the average temperature of the component and its neighbours, thereby allowing indirect control of components which are not directly accessible by the use of temperature flow relationships. Each control behaviour could be implemented by a separate processor, and the method of obtaining temperature control for inaccessible elements also deals with the control of elements whose associated control processors have

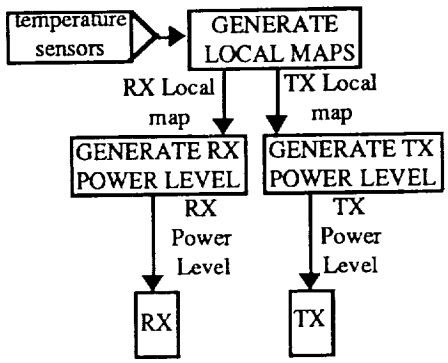


Figure 4. Example level 0 competency.

malfunctioned or failed. The resulting behavioural model is shown in Figure 4.
The analogical representation is an internal state model used by the Generate Local Maps behaviour. This behaviour may actually comprise a separate local map generator associated with each controllable unit, so the analogical model is a virtual construct. Each local map is a model of the immediate (ie. directly connected) thermal environment of the associated component, which is used by the power level generation behaviour of that component to calculate a power level which is then sent to the component.

The next level of competency considered is the level 2 Maintain Telecommand Override competency. For the current highly simplified

example, this is easy to create using a single behavioural module. The Maintain Telecommand Override module monitors the receiver power level, and suppresses the level 0 control signal if it falls below the minimum level required to operate the receiver for telecommand override. The minimum power level is then injected into the receiver input in the form of the suppressing signal from the Maintain Telecommand Override module. The structure giving this competency is illustrated in Figure 5.

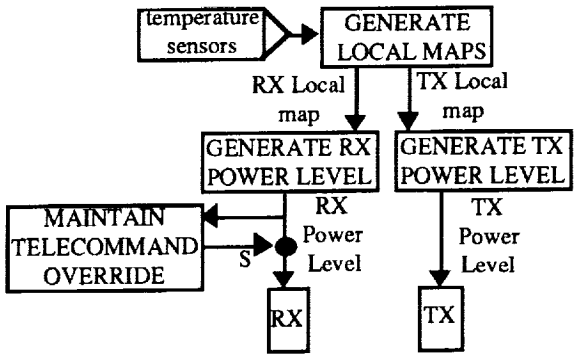


Figure 5. Example level 2 competency.

The level 0 competency can still deal with overtemperatures in the region of the receiver, not by reducing its power level below the minimum required for telecommand override, but if necessary by powering down neighbouring units. The structure and functionality of the level 0 competency ensures that this will happen automatically, without explicit control by the level 2 competency or by ground operators

In a more complete example, this minimum power level required to Maintain Telecommand Override is a function of spacecraft orientation, orbital height, position in relation to ground control stations, the states of a number of alternate transmitter system modules, etc., so the Maintain Telecommand Override competency emerges from a much more complex interaction between a number of modules.

Conclusion: The AUSTRALIS-1 Spacecraft Project and Ongoing Research

AUSTRALIS-1 is a microsatellite project currently under development by a number of Australian universities. Ongoing research with the spacecraft control architecture described here will be conducted in relation to the AUSTRALIS-1 project. A comprehensive control simulator for the satellite based upon the described architecture is currently under development. The outcome of simulation studies will be a detailed control architecture design, with well understood tradeoff studies of different control system configurations. Another major ongoing area of research will consider the relationship between the logical control structure and the physical computational architecture upon which it is implemented. It is particularly desirable to define a hardware architecture compatible with many possible mappings from the logical control model to its implementational structure. To this end, the hardware design approach will follow the following sequence:

1. a basic set of modules will be defined representing subsystems and/or components required to achieve desired spacecraft competencies in relation to the specific requirements of the AUSTRALIS-1 mission. Module definitions will include the definition of outputs from the modules (which are equivalent to sensor values or status signals), inputs to the modules (equivalent to actuator commands), the different operational modes of each module, the association of different sensor outputs and actuator inputs with different operational modes, and the definition of the conditions under which transitions between the different operational modes will occur (both in response to command inputs, to sensor values, and to internal states). This set of modules will be referred to as *substrate* modules, and constitute the lowest level description of the machine to be controlled.

2. a generic module interface unit will be defined at the physical and protocol levels. The connection between any two modules should be asynchronous, bidirectional, and

configurable between 1- and n-bit binary numbers.

3. each substrate module defined in step 1 will have a standard microprocessor-based interface unit associated with it.

4. in addition to the substrate modules, based upon an estimate of overall bandwidth requirements to achieve all levels of competency (or a specified subset thereof), a number of additional processing nodes will be specified, each with a standard interface, and a generic processor defined in order to implement each additional node. (The number and capacity of additional nodes required will be explored by simulation of the architecture.)

5. all behaviours will be integrated into a fully connected control signal network via their standard interfaces.

6. each standard processor will in addition have a dual-redundant connection to a behavioural override processor (BOP). The BOP will be able to download transfer functions to each processor in the control network, to allow downloading of software implementing different logical control architectures. The BOP will also be able to drive each behaviour directly, or bypass the behavioural control network altogether.

A significant difference between spacecraft and the mobile robots typically used to explore the behavioural paradigm is the explicit and substantial role of spacecraft as data acquisition, conditioning, and transmission systems. Options for accommodating this function include distributed routing of data via the behavioural control network, or use of a more conventional data communications topology (eg. a bus, ring, or star network). The approach adopted will be to provide for any of these mechanisms:

7. a high speed asynchronous communications link will be part of the standard processor interface, with multiplexed connections both to other processors in the network and to the BOP processor.

This control architecture will provide a hardware platform that conforms with the behavioural paradigm, but allows for considerable experimentation with different methods of mapping the logical control structure onto the hardware structure. In particular, the ability to download module transfer functions from the BOP will support experimentation with:

- dynamic control system reconfiguration
- adaptivity and learning of module transfer functions, network topology, and behaviours; ie. all of the control structure above the substrate level
- the distribution and form of multiple computational paradigms within the behavioural framework

In-orbit experiments will comprise a series of empirical evaluations aimed at refining and evaluating different detailed logical control models, the development of a software architecture and mappings onto the hardware architecture for each logical model, and methods of achieving adaptivity and robustness within these models, at increasing levels of behavioural competence. The results of the project will be valuable in defining and understanding techniques for increasing the autonomy of space systems in many diverse applications.

References

Arkin R. C., 1990, "Integrating Behavioural, Perceptual, and World Knowledge in Reactive Navigation", in Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 105-122.

Bonasso R. P., 1991, "Integrating Reaction Plans and Layered Competencies Through Synchronous Control", *12th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, 1225 - 1231.

Brooks R. A., March 1986, "A Robust Layered Control System For A Mobile

Robot", *IEEE Journal of Robotics and Automation*, 2(1), 14-23.

Brooks R. A., 1990, "Elephants Don't Play Chess", in Maes, P. (ed) *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 3-15.

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

Brooks R. A. and Flynn A. M., Oct 1989, "Fast, Cheap, and Out of Control: A Robot Invasion of the Solar System", *JBIS*, 42(10), 478-485.

Ciarlo A., Donzelli P., Katzenbelsser R. and Møller B. A., 1987, "Satellite On-Board Applications of Expert Systems", *ESA Journal*, 11, 31-44.

Ciarlo A., and Schilling K., 1988, "Application of Expert System Techniques to the Cassini Titan Probe", *ESA Journal*, 12, 337-351.

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

Erickson J. D., 1987, "Intelligent Systems and Robotics for an Evolutionary Space Station", *JBIS*, 40, 471-482.

Gat E., 1991, *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots*, PhD Thesis, Virginia Polytechnic Institute and State University, revised version.

Grant T. J., 1991/1992, "A review of Multi-Agent Systems techniques, with application to Columbus User Support Organisation", *FGCS7*, 413-437.

Leinweber D., Spring 1987, "Expert Systems in Space", *IEEE Expert*, 26-36.

Maes P., 1990a, "Guest Editorial", in Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 1-2.

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

Malcolm C. and Smithers T., 1990, "Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System", in Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 123-144.

Payton D. W., 1990, "Internalised Plans: A Representation for Action Resources", in Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 89-104.

Pidgeon A., Howard G. and Seaton B., 1992, "Operational Aspects of Spacecraft Autonomy", *JBIS*, 45, 87-92.

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

Spier R. J. and Liffring M. E., Nov, 1989, "Real-Time Expert Systems for Advanced Power Control", *IEEE AES Magazine*, 33-38.

Steels L., "Exploiting Analogical Representations", in Maes, P. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT/Elsevier, 71-88, 1990.

Steels L., 1991, "Emergent Frame Recognition and its Use in Artificial Creatures", *12th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, 1219 - 1224.

Tello E. R., 1986, "Dipole: An Integrated AI Architecture Suitable for Space PRogram Applications", *Expert Systems in Government Symposium*, IEEE, 168-180.

Woods D., April 1992, "Space Station Freedom: Embedding AI", *AI Expert*, 33-39.

ANOMALOUS EVENT DIAGNOSIS FOR ENVIRONMENTAL SATELLITE SYSTEMS

Bruce H. Ramsay
National Oceanic and Atmospheric Administration
National Environmental Satellite, Data, and Information Service
Camp Springs, Maryland

ABSTRACT

The National Oceanic and Atmospheric Administration's (NOAA) National Environmental Satellite, Data, and Information Service (NESDIS) is responsible for the operation of the NOAA geostationary and polar orbiting satellites. NESDIS provides a wide array of operational meteorological and oceanographic products and services and operates various computer and communication systems on a 24-hour, seven days per week schedule.

The Anomaly Reporting System contains a database of anomalous events regarding the operations of the Geostationary Operational Environmental Satellite (GOES), communication, or computer systems that have degraded or caused the loss of GOES imagery. Data is currently entered manually via an automated query user interface. There are 21 possible symptoms (e.g., No Data), and 73 possible causes (e.g., Sectorizer - World Weather Building) of an anomalous event. The determination of an event's cause(s) is made by the on-duty computer operator, who enters the event in a paper-based daily log, and by the analyst entering the data into the reporting system. The determination of the event's cause(s) impacts both the operational status of these systems, and the performance evaluation of the on-site computer and communication operations contractor.

The Anomaly Reporting Expert Assistant System (AREAS) is an interactive, rule-based demonstration prototype using backward chaining goal-directed inference. Upon input of a new event's symptom, AREAS queries a database of prior events with associated symptoms and causes, and then suggests possible causes to the analyst. AREAS reasons with the archived events, a rule-based representation of the satellite, communication, and computer subsystem's physical relationships, heuristics acquired from resident domain experts, and a mean best fit of prior events with the new event. Whether the analyst confirms AREAS' suggested cause or enters a new one, the event, with its related attributes, is entered into the database and thus provides an up-to-date environment in which AREAS can operate. AREAS includes a help system designed to assist new users and it provides technical information, with graphical representation, on the GOES, communication and computer subsystems.

Key Words: Knowledge-Based System, Rule-Based, Backward Chaining, Goal-Directed Inference, Environmental Satellite Systems, Anomalous Event Diagnosis, Intelligent Database

INTRODUCTION

The National Environmental Satellite, Data, and Information Service (NESDIS) oversees the operation of civilian satellite systems used for Earth-observation, and the creation and maintenance of global databases in the physical and life sciences. NESDIS provides products and services derived from environmental data that are applied to the protection of people and property, national economic systems, and the development and distribution of food, energy and other natural resources on a national and international level.

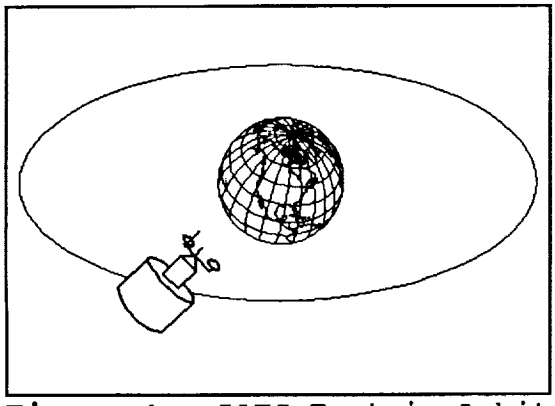


Figure 1 - GOES East in Orbit

NESDIS is responsible for the operation and maintenance of the Geostationary Operational Environmental Satellite (GOES), located at 112° West as shown in Figure 1 above, and the GOES Data Distribution System (GDDS). It is staffed with experienced meteorologists, oceanographers, computer specialists, and administrative personnel, as well as employees new to the environmental satellite domain. A contractor, PRC, Inc., provides communications and computer operations support for the GDDS.

Why Artificial Intelligence?

NESDIS determined to evaluate the potential of Artificial Intelligence (AI) tools and techniques in response to the challenge of sensing, communicating, processing, analyzing and distributing ever-increasing volumes of environmental data and products. This increase is due to the larger number of ground-based data collection systems, satellites in orbit with improved sensors, and additional data shared by other organizations, both public and private, in the United States and foreign nations.

OBJECTIVES

Four objectives were established for the development and demonstration of the Anomaly Reporting Expert Assistant System (AREAS) prototype. They were:

1. Develop a help system for anomaly reporting.
2. Increase personal knowledge of GDDS.
3. Retain valuable GDDS expertise.
4. Demonstrate the ability of AI to improve administrative and operational tasks.

CURRENT ANOMALY REPORTING SYSTEM

As a result of a computer generated error message or other indicator, a computer operator documents the problem in the paper-based Environmental Satellite Distribution/Interactive Processing Center (ESD/IPC)

Daily Log. At the conclusion of a shift, the shift supervisor synopsizes the Daily Log entries into multiple summary reports including the Operational Problem report. A combined hardcopy daily report, including the Daily Log and Operational Problem report, among others, is then distributed to management and staff. Each morning, contractor and NESDIS personnel meet for a short discussion of the most critical issues encountered in the previous 24 hours. On a daily basis a NESDIS staff member evaluates the Daily Log and Operational Problem report and enters appropriate information into the Anomaly Reporting System (ARS). The staff shares this task on a weekly, rotating basis.

The ARS queries the user for the following information:

```
Julian Date
Satellite
Zulu Time
Symptom(s)
Number Products Affected
Probable Cause(s)
Responsible Division
Number Expected
Number Actuals
Number External Lost
```

The 21 possible symptoms and 73 possible causes are available to the staff in hardcopy or in an on-line text file. For example:

```
SYMPTOM CODES
CODE #    ENTRY

01 ..... DATA EARLY
.
.
.
14 ..... NO DATA
.
.
.
21 ..... WRONG DATA
```

```
CAUSE CODES
CODE #    ENTRY

01 ..... AIR COND/HEATING
.
.
.
53 ..... SECTORIZER-WWB
.
.
.
73 ..... VIRGS
```

Accompanying the hardcopy symptom and cause list is a GDDS Diagram, part of which is shown in Figure 2. This diagram is not available on-line in the ARS.

The complete diagram (not shown here) outlines the flow and processing of data for the major communications and computer systems from the GOES spacecraft to NOAA's facilities at Wallops Island, Virginia, and Suitland and Camp Springs, Maryland. With this information, along with other available documents, assigned staff must evaluate the Daily Log and Operational Problem report and determine the cause of the anomalous event. After determination of the problem's cause and input of the data, a daily report is produced for dissemination.

ANOMALY REPORTING EXPERT ASSISTANT SYSTEM

1.0 Problem Identification

A loss of expertise was recently suffered due to the retirement and promotion of several employees. New employees needed access to the lost expertise in order to accurately determine the cause of anomalous events and prevent future occurrences, if possible. The current ARS has

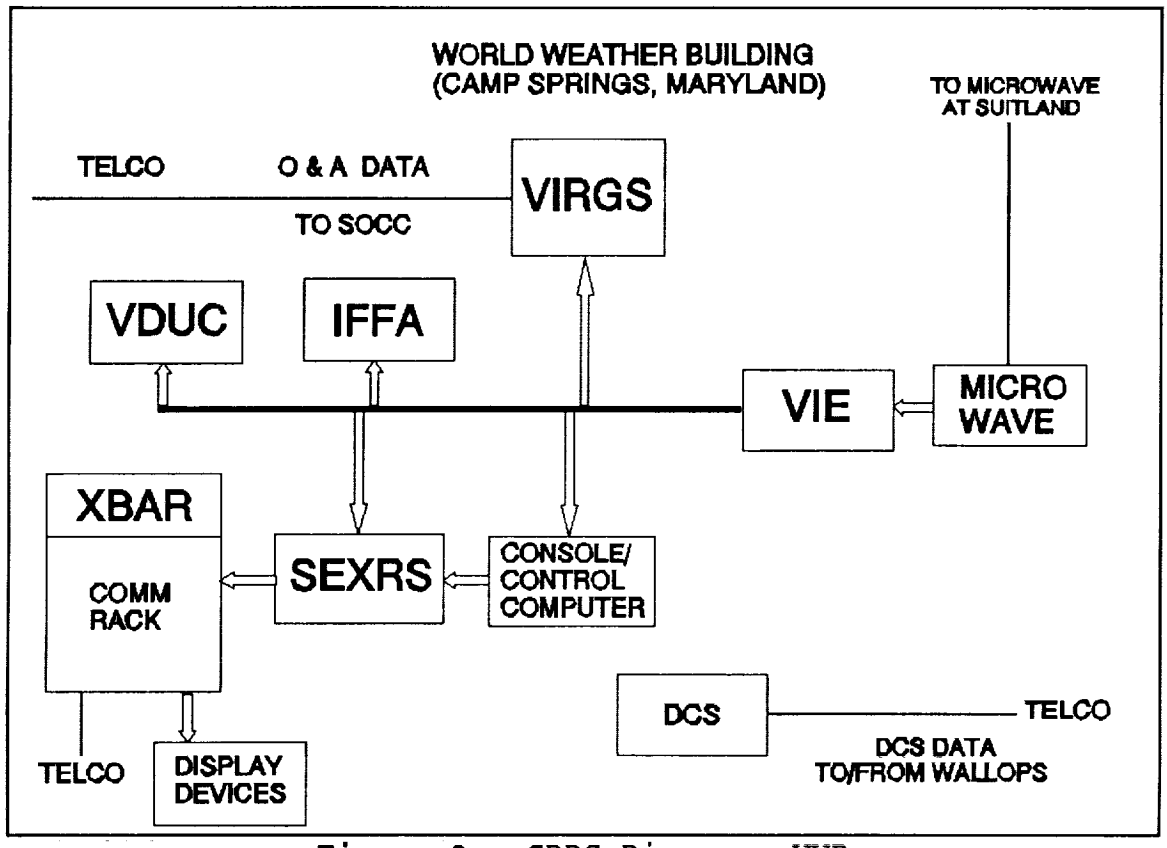


Figure 2 - GDDS Diagram: WWB

no help system and several of the new employees have limited knowledge of the GDDS.

2.0 Knowledge Acquisition

Domain knowledge was acquired through interviews with domain experts, one of whom has since retired. Extensive GDDS documentation, including the *GDDS Operations and Maintenance Contract*, was reviewed. In addition, the current Anomaly Reporting System, ESD/IPC Daily Logs and Operational Problem reports, Daily and Weekly ARS reports were also analyzed.

3.0 Analysis and Design

The analysis and selection of knowledge representation and the development tool along with the system design have been integrated within a single category. The intent is to emphasize the real world environment in which all three issues are frequently considered concurrently, especially during initial prototyping.

3.1 Knowledge Representation

Evaluation of the existing data indicated that an attribute/ value representation scheme combined with rule-based processing would be sufficient for initial prototype development. Since the current ARS uses a query/response interface it was decided to use backward chaining, goal directed inference to emulate the existing process.

3.2 Tool Selection

The criteria for tool selection, in addition to those identified above, were low cost, a simple development environment, and a short learning curve. The tool selected was Level5 Expert System Software (DOS Version 1.3) by Information Builders, Inc. This expert system shell fit the identified small system prototype requirements: a default query/response interface, symbolic representation used in an if/then rule-base environment, and the need to perform simple calculations.

3.3 System Design

The system design stressed modularity for ease of development, explanation, use, and maintenance. The shell's default user interface was employed for the selection of menu items and the input of numeric data. Individual knowledge base modules were used for each of the primary data items required for inferencing. The help system's customized graphic and narrative explanation screens were integrated through Level5's explanation function and separate knowledge base modules. The help system focused on the three major components of the GDDS: the satellite, data communications, and computer processing subsystems. A simple mean statistical analysis was provided through symptom specific knowledge base modules. This architecture is demonstrated in Figure 3., below.

4.0 Prototyping

Modularity was a key issue during the rapid prototyping of AREAS since the knowledge engineering process was being used as a learning tool for the GDDS environment. A large number of small, easily modified knowledge bases were initially prototyped to establish the relationship among various data elements, particularly between symptoms and causes, and to model the physical subsystems.

4.1 Knowledge Sources

4.1.1 Heuristic Knowledge

Heuristic knowledge was obtained from the domain experts through their explanation of Daily Log and Operational Problem report entries and GDDS processing. For example, the hardware element designation "RTIR" (i.e., RealTime InfraRed) is not specified in the sectorizer subsystem node of the Wallops version of the GDDS Diagram, but it was identified by one of the domain experts as important in distinguishing between sectorizers at the World Weather Building (WWB) and those at Wallops Island. This knowledge was then incorporated into the rule base of AREAS.

4.1.2 Documentation

A number of different documents were used as primary knowledge sources. *NESDIS Programs - NOAA Satellite Operations* identified the organization's mission and major systems used in carrying out that mission. The *GDDS Operations and Maintenance Contract* was indispensable in identifying subsystems and

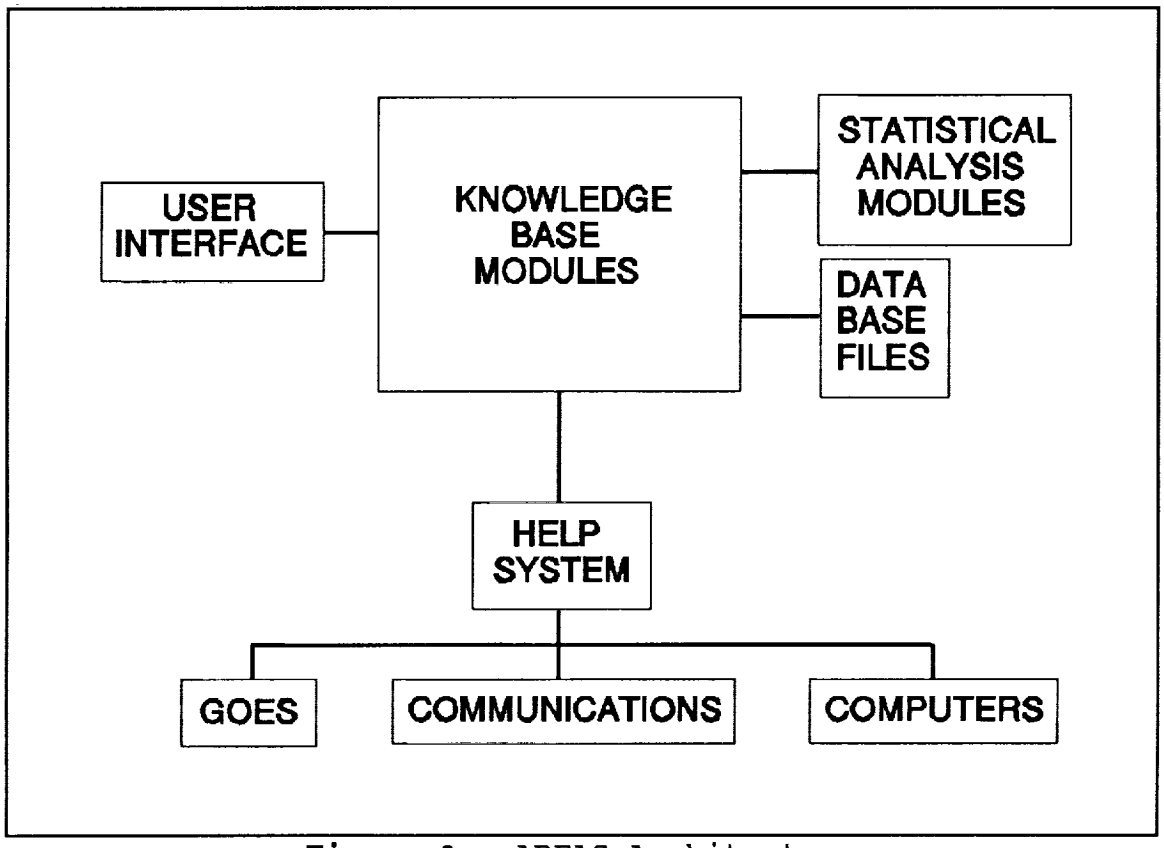


Figure 3 - AREAS Architecture

their constituent components. A memorandum to all the organizations responsible for the GDDS operation explained the use of ARS as, in part, an instructional tool. It included the GDDS Diagram, of which the WWB segment at Camp Springs, Maryland is shown in Figure 2, and the lists of possible symptoms and causes. The memorandum's express purpose was to establish a common framework in which to identify and respond to anomalous events.

4.1.3 ARS Database

Evaluation of the ARS data base provided input to the data type classifications used in AREAS, as provided by the expert system shell: numeric, attribute/value, and string.

4.2 Process of Discovery

Since one of the objectives of building AREAS was to gain additional insights into GDDS, AREAS had to be able to represent GDDS physical relationships among its subsystems and components. For example, the following "identify symptom" rule represents the interpretation of, and the relationship between, the shift supervisor's comment in the remarks column of the Operational Problem Log and the identified symptom.

```
RULE identify symptom
IF remark IS Short-SZ Halted in RCV
THEN symptom identified
AND symptom IS Degraded Data
```

The remark, "Short - SZ Halted in RCV," means the sectorizer's

processing of the imagery product's data set during transmission was terminated while in receive mode. The symptom is thus classified as degraded data (i.e., by definition 50 percent or more of the complete image was produced and was of animation quality). The next rule, "identify responsible organization," establishes the relationship between the identification of the satellite and a specific sectorizer and the organization responsible for its operation.

```
RULE identify responsible
     organization
IF symptom identified
AND satellite IS GOES East
AND hardware element IS Sectorizer
    6A11
THEN responsible division
     identified
AND responsible division IS SSD
```

4.3 Help System

The help system provides query specific information in narrative and graphical formats of crucial areas of the GDDS. If the user doesn't understand a particular query, such as "What was the responsible division?" a help screen is available with additional information explaining the physical system relationships and the organizations responsible for their oversight. Mutually supportive information from different documents is merged as well in help screens. For example, a glossary of terms such as the one shown below was merged with the GDDS Diagram in Figure 2, above.

COMM RACK	Communications Rack
DCS	Data Collection System
IFFA	Interactive Flash Flood Analyzer
O & A	Orbit and Attitude
SEXRS	Sectorizers
SOCC	Satellite Operations Control Center
TELCO	Telephone Data Lines
VAS	VISSR Atmospheric Sounder
VDUC	VAS Data Utilization Center
VIE	VAS Interface Electronics
VIRGS	VISSR Image Registration and Gridding System
VISSR	Visible and Infrared Spin - Scan Radiometer
XBAR	Crossbar Switch

4.4 Statistical Analysis

The initial objective of a statistical analysis of the ARS data base was to provide the user with background information as to the apparent relationships between symptoms and causes. This was accomplished through a simple mean analysis of the type and number of causes attributed to each symptom. This analysis, coupled with the rule output reviewed above, produces a diagnosis as shown below in Figure 4. The user is then at liberty to accept or reject the diagnosis.

4.5 Introduction of Knowledge-Based Systems

The focus on simplicity of the prototype's development, architecture, purpose, and operation was important. These issues had to be easily explainable to use AREAS as an introduction of knowledge engineering concepts. A basic approach was taken in knowledge acquisition, representation, search, and inference for this purpose.

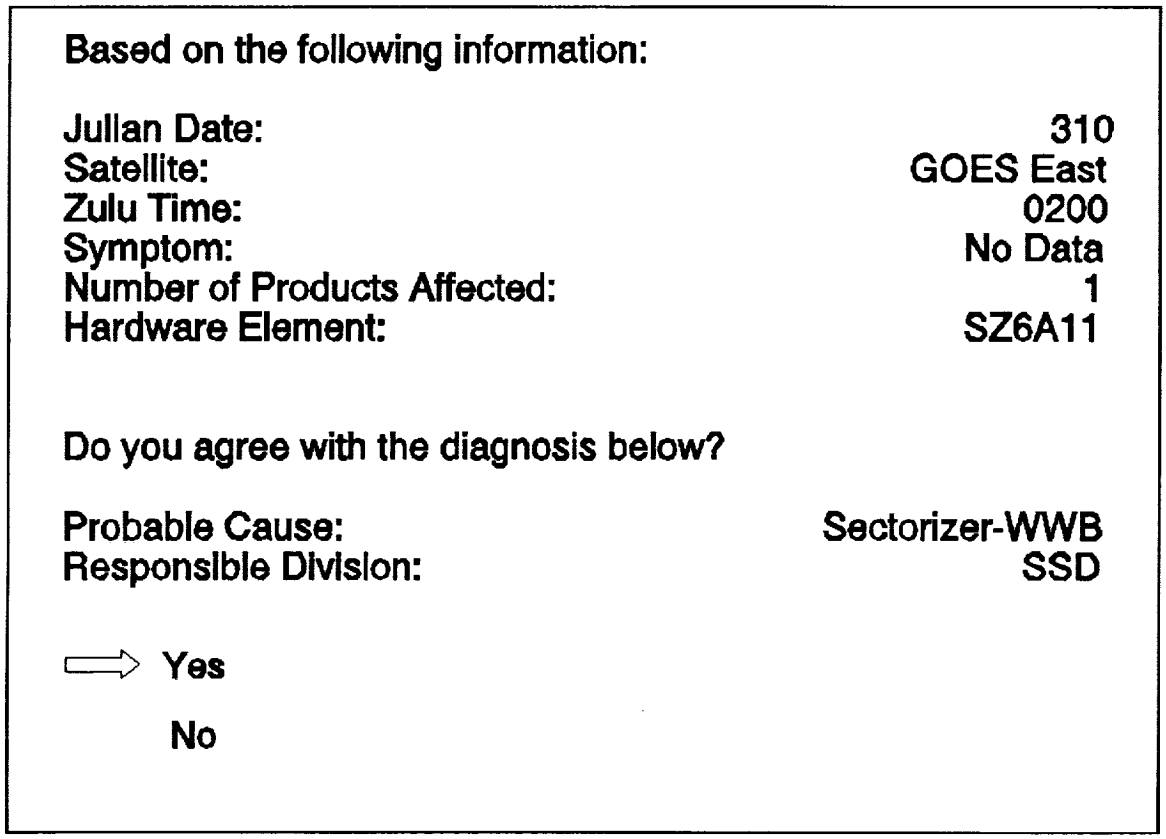


Figure 4 - Diagnosis Screen

5.0 Verification and Validation

Verification was performed through analysis of shell-produced knowledge trees linking all the goals, rules and attributes in a given knowledge base in a logical order of precedence starting with the top-level goal. Each logical path through a knowledge base was also manually derived and tested.

Initial validation was performed by comparing archived results of domain experts' analyses to system generated conclusions. Subsequent validation was conducted by domain experts through the evaluation of results from test data sets processed by AREAS.

6.0 Test and Evaluation

The qualitative test and evaluation of the AREAS demonstration prototype focused on its potential use as a help system in identifying the symptoms and causes of anomalous events. Feedback from user surveys indicated:

- a positive reaction to the display of statistical data but a need to further highlight only the most prevalent symptom/cause ratios;

- a desire to have AREAS identify individual GDDS components and their output of specific products (i.e., products are currently assigned identification codes and are logically linked to specific hardware elements within GDDS);

- approval of the graphics used but a request for more detail in representing GDDS subsystems and components;

- the need to allow multiple symptom identifications for a single anomalous event (e.g., the symptoms Data Incomplete and Degraded Data can be specified for a single event in the current ARS), and the ability to generate multiple symptom/cause records per report; and

- support for the ability to easily review input prior to data base update and subsequent report generation.

CONCLUSION

Part, but not all, of each objective was accomplished in the development and demonstration of AREAS:

1. Develop a help system for anomaly reporting.

The current ARS has no help feature. One of the expressed purposes for users and new employees using the ARS is to train them in the nomenclature and processes of the GDDS. One of the primary objectives of AREAS was the linkage of relevant narrative and graphical information to specific user queries. Based on user feedback AREAS has made an important step in identifying user needs in the successful analysis of anomalous events in GDDS.

2. Increase personal knowledge of GDDS.

The author is a novice with satellite-based systems but experienced in knowledge-based systems development. Through the development of AREAS and preparation of this paper, he was able to take advantage of the process of discovery highlighted by the knowledge engineering process to gain a better understanding of the GDDS.

3. Retain valuable GDDS expertise.

The retirement and promotion of several employees who were very experienced in the GDDS created a potential problem for new employees assigned to anomalous event tracking and analysis. Part of their expertise, through the knowledge engineering process, was captured for use through AREAS.

4. Demonstrate the ability of AI to improve administrative and operational tasks.

The only automated tool available within ARS to search the database requires the user to possess a clear idea of what is being searched for and familiarity with the data types and structures employed. The purpose in providing the user with a simple mean analysis of the data represents the initial step in providing ready access to analytical tools and results. These tools, when augmented by heuristic rules to constrain the search space, provide a reasonable method of diagnosis to assist the user in making decisions. In the future these results may identify potential trends in specific GDDS subsystems and hardware elements, as well as processing methodologies.

AREAS, with its focus on simplicity, provides the

opportunity to introduce knowledge-based systems concepts, development and use to employees in a direct, hands-on way. It highlights the value of knowledge engineering as a process of discovery. It also demonstrates the ability to harness the knowledge of disparate sources of information and provides a focus for that knowledge on problem-solving in the domain of anomalous event diagnosis for environmental satellite systems.

REFERENCES

Cote, D. "Planning Begins for Major Upgrade of NOAA's Data Management System." *Earth System Monitor* 3(1) (September 1992):1-2.

Harmon, H. & King, D. *Expert Systems*. New York: John Wiley & Sons, Inc., 1985.

Mairs, R. "Anomaly Reporting System." Memorandum (May 27, 1988), Satellite Services Division, National Environmental Satellite, Data, and Information Service, Camp Springs, Maryland.

Miller, D. & Hay, D. "Final Report of the Artificial Intelligence Techniques Study for Environmental Products and Computer System Operations." (September 1992). U.S. Department of Commerce Service Order Number 40AANE201402. Greenbelt, Maryland: COMSO, Inc.

Morelli, R. "Using Knowledge Engineering to Teach Science." *IEEE Expert* 5(4) (August 1990): 74-78.

National Environmental Satellite, Data, and Information Service. *NESDIS Programs: NOAA Satellite Operations*. Washington, DC: U.S. Government Printing Office: 1985-472-036, 1985.

PRC, Inc. *GDDS Operations and Maintenance Contract: Technical Management Proposal*. Department of Commerce Solicitation Number 52-DGNE-9-00067. McLean, Virginia: PRC, Inc., 1989.

Rao, K. & Holmes, H. & Anderson, R. & Winston, J. & Lehr, P. (Eds.). *Weather Satellites: Systems, Data, and Environmental Applications*. Boston, Massachusetts: American Meteorological Society, 1990.

Image/Data Classification/Interpretation

The probabilistic neural network architecture for high speed classification of remotely sensed imagery

Samir R. Chettri
Hughes-STX at NASA/Goddard Space Flight Center
Greenbelt, MD 20771

Robert F. Cromp
Code 930.1
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

In this paper we discuss a neural network architecture (the Probabilistic Neural Net or the PNN) that, to the best of our knowledge, has not previously been applied to remotely sensed data. The PNN is a supervised non-parametric classification algorithm as opposed to the Gaussian maximum likelihood classifier (GMLC).

The PNN works by fitting a Gaussian kernel to each training point. The width of the Gaussian is controlled by a tuning parameter called the window width. If very small widths are used, the method is equivalent to the nearest neighbour method. For large windows, the PNN behaves like the GMLC.

The basic implementation of the PNN requires no training time at all. In this respect it is far better than the commonly used Backpropagation neural network which can be shown to take $O(N^6)$ time for training where N is the dimensionality of the input vector. In addition the PNN can be implemented in a feedforward mode in hardware. The disadvantage of the PNN is that it requires all the training data to be stored. Some solutions to this problem are discussed in the paper.

Finally, we discuss the accuracy of the PNN with respect to the GMLC and the Backpropagation neural network (BPNN). The PNN is shown to be better than GMLC and not as good as the BPNN with regards to classification accuracy.

1 Introduction

High performance computers and sophisticated sensors are responsible for the explosive generation of data for scientific, industrial and commercial uses. NASA faces the same data glut with its current and future missions (including the Earth Observing System and Tropical Rainfall Measuring Mission platforms). At NASA's Goddard Space Flight Center, the Intelligent Data Management group (IDM), within the Information Science and Technology Office (ISTO), has been investigating and developing data and information management systems that can handle the archiving and querying of data produced by Earth and space missions with fast response times. This work has resulted in an Intelligent Information Fusion System (IIFS) for handling and archiving terabyte-sized spatial

databases; and Scheduler/Planner Under Deadlines (SPUDS) to guarantee that response times are maintained [CCS92]. Our current data source of applications is remotely sensed images. However, IIFS and SPUDS are not inherently limited to this data source, having been conceived as general purpose tools.

IDM's research into neural networks has been ongoing since 1989 [CHC89] starting with research into the applicability of the backpropagation paradigm to remotely sensed images. This work has continued, resulting in comparisons of backpropagation with conventional Gaussian Maximum-likelihood classification [CCB92]. Within IIFS/SPUDS, the neural networks act as high speed, low level image classifiers with higher level domain knowledge being provided by decision trees and expert systems. The combination of IIFS and SPUDS provides a scientist with the means to access a database based on image content at varying levels of resolution.

For IDM's purposes, the data glut problem can be divided into two parts. The first part deals with efficient characterization of the data and subsequent archival processes. The second part deals with efficient querying of the data based on platform, content, and spatial and temporal constraints. This paper will deal with the characterization of satellite images and the attendant problems. In particular we discuss the pros and cons of the Probabilistic Neural Network (PNN) with respect to high speed data classification.

In the following text, the PNN is first described. Next, we discuss the advantages of the PNN with respect to backpropagation neural networks (BPNN) and Gaussian Maximum Likelihood Classifiers (GMLC). As a way of addressing the shortcomings of the PNN, we introduce Kohonen's Learning Vector Quantization (LVQ) which helps increase the feedforward speed of the PNN. We then apply the PNN to an image of the Blackhills in South Dakota and discuss the quality of the output. We conclude with a summary of the main results of the paper and reveal our future research plans.

2 The Probabilistic Neural Network (PNN) architecture

The roots of the PNN lie in the histogram evaluation techniques that date back to 1661 ([TT78], [TK76]). Whereas the histogram uses rectangular boxes and quantizes the data axes, the kernel method chooses not to quantize the data axes, instead placing a kernel at each data point in multidimensional space. For an illustration of this process see [Sil86]. In the following discussion, the density estimates are obtained from a set of n observed data vectors $\mathbf{X_1}, \ldots, \mathbf{X_n}$. The actual density is denoted by $f(\mathbf{x})$ and the estimate of the density by $\hat{f}(\mathbf{x})$.

The multivariate estimate of density [Sil86] with kernel K and window width σ is written as

$$\hat{f}(\mathbf{x}) = \frac{1}{n\sigma^d} \sum_{i=1}^{n} K\left\{\frac{1}{\sigma}(\mathbf{x} - \mathbf{X}_i)\right\}. \tag{1}$$

The kernel function satisfies

$$\int_{R^d} K(\mathbf{x}) d\mathbf{x} = 1. \tag{2}$$

Usually the kernel is a unimodal, everywhere positive function. The use of non positive kernels is still an open research question.

The Gaussian kernel (also discussed in the next section) can be written as

$$K(\mathbf{x}) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}). \tag{3}$$

The kernel function and the smoothing parameter are the two choices to be made in the case of density estimates using the kernel method. Research has shown that the choice of kernel does not greatly vary the density estimate [Sil86]. All things being equal, it is then desirable to choose kernels based on their computational properties. We will address this issue in section 2.4.

2.1 Discriminant functions

Given K classes, let $f(\mathbf{X} \mid S_k)$ be the probability density function (pdf) associated with the measurement vector $\mathbf{X}$, given that $\mathbf{X}$ is from class k. Let $P(S_k)$ be the *a priori* probability of class S_k. We can use the **maximum** *a posteriori* (MAP) decision rule to identify the class to which $\mathbf{X}$ belongs. It can be stated as follows ([And72]):

$$\textit{Decide } \mathbf{X} \in S_k \textit{ iff } f(\mathbf{X} \mid S_k)P(S_k) \geq f(\mathbf{X} \mid S_j)P(S_j),\ j = 0, 1, \cdots M-1\ ,$$

where the products $f(\mathbf{X} \mid S_k)P(S_k)$ correspond to discriminant functions, and there are M classes for which discriminant functions are defined. As stated, the MAP rule consists of evaluating the discriminant functions and selecting the maximum as the winner.

Estimating the density function is a key problem in MAP estimation. If the underlying density of each class were known, the problem would be an easy one. In fact, the Gaussian Maximum Likelihood Classifier (GMLC) simplifies the problem of density estimation by *assuming* that $f(\mathbf{X} \mid S_k)$ is a multi-variate normal pdf whose parameters (the mean vector and the variance-covariance matrix) can be determined by samples conditioned on class S_k.

The probabilistic neural net (PNN) was designed by Specht [Spe90] using Parzen's [Par62] kernel function:

$$f(\mathbf{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[-\frac{(\mathbf{X} - \mathbf{W}_{ki})^T(\mathbf{X} - \mathbf{W}_{ki})}{2\sigma^2}\right]. \tag{4}$$

Note that Parzen's kernel is the same as the Gaussian kernel of equation (3). In equation (4) $\mathbf{W}_{ki}$ is the i^{th} training pattern from the $0 \leq k^{th} \leq M-1$ category, P_k is the total number of training patterns in class k, d is the dimension of the training pattern $\mathbf{W}_{ki}$, and σ is a "smoothing According to Specht, a small value of σ caused the density to have modes at the sites of the training samples. Increasing σ causes smoothing of the surface around the modes. In the limiting case, the pdf is Gaussian regardless of the true nature of the underlying distribution. This may seem to be a problem; however, according to Specht, " it is not difficult to find a good value of σ, and ... the misclassification rate does not change dramatically with small changes in σ."

2.2 PNN implementation details

The PNN can be implemented using a feed-forward network. An overview of the PNN is shown in Figure 1. There are four layers. The input layer fans out the input d dimensional vector which has

to be placed in one of M classes. Each node in the input layer is connected to every node in the pattern layer and input vector components are transformed by means of a weight $W_{i,j}$ connecting the i^{th} input node to the j^{th} pattern node. The pattern layer is subdivided into sets of nodes. Each set of nodes does the processing for a particular class. Since there are M classes, there are M sets of pattern nodes. The output of each pattern node set is sent to a node in the summation layer, thus there are M nodes in the summation layer. Finally, the outputs of the summation layer nodes are sent to the decision layer which obtains the maximum output O_k, $k = 0, \ldots, M-1$, and assigns the input vector X to class k.

In equation (4) $-\frac{(\mathbf{X}-\mathbf{W}_{ki})^T(\mathbf{X}-\mathbf{W}_{ki})}{2\sigma^2}$ is exponentiated. This product can be written as $\mathbf{X}^T\mathbf{W} - 1$ if both input and weight vectors are converted to unit vectors, as shown in Figure 2 (a). After the dot product is completed, 1 is subtracted from the total and this is multipied by σ^{-2} after which the exponentiation is performed. At the end of this step one of the terms in the sum of equation (4) has been evaluated.

If the input and weight vectors are not converted to unit vectors, then the architecture of the PNN as shown in Figure 2 (a) can be changed to reflect this. It should be mentioned here that using unit vectors changes the kernel evaluation from a dot product and two vector subtractions to a single dot product and a scalar subtraction. The disadvantage to the dot product method is that magnitude information, that may be useful during the classification process, is lost. On the other hand if our vectors all contain integers, the kernel evaluation process may be done efficiently using integer computations and the dot product method dispensed with entirely.

A summation layer node contains an adder that sums up the outputs of all the pattern nodes in a particular set and then multiplies the output by $\frac{1}{(2\pi)^{d/2}\sigma^d P_k}$ as shown in Figure 2 (b). Thus the summation layer represents the summing process of equation (4).

The decision layer obtains the maximum of the summation layer outputs, and the class to which a given input vector X belongs is finally output.

The PNN is trained by first converting the training exemplars to unit vectors. Next each connection between the input node and a pattern node is assigned a weight which is nothing but an element from the unit training vector. Thus the number of pattern layer nodes corresponds to the number of training vectors and each weight between input and pattern layer nodes corresponds to an element of a training vector. Once training has been done, the network is ready for use in feed–forward mode. The only input parameter from the user in feed–forward mode is σ. A good heuristic method for selection σ is described in [KF72]. In this method the smoothing parameter is given by $\sigma = (d^{-1}\mathrm{tr}\,[\mathbf{C}])^{1/2}N^{-\alpha/d}$. Here, $\mathbf{C}$ is the covariance matrix estimated from the data, d is the dimensions of the data, tr is the trace of a matrix, N is the number of samples and $0 < \alpha < 0.5$. Computing $\mathbf{C}$ would make the PNN training time identical to the Gaussian maximum likelihood estimator, thereby eliminating its main advantage. In [MAC+92], Radial Basis Functions (of which PNN's are a part) are used to reduce the number of hidden nodes by obtaining the covariance $\mathbf{C}$ matrix of samples and also to obtain the widths of the kernel functions. While the RBF approach of [MAC+92] is useful, we have found that σ can take on a reasonably large range of values without seriously affecting accuracy, hence our adherence to the PNN paradigm. A discussion of the results pertaining to our choice of σ is given in section 3.2.

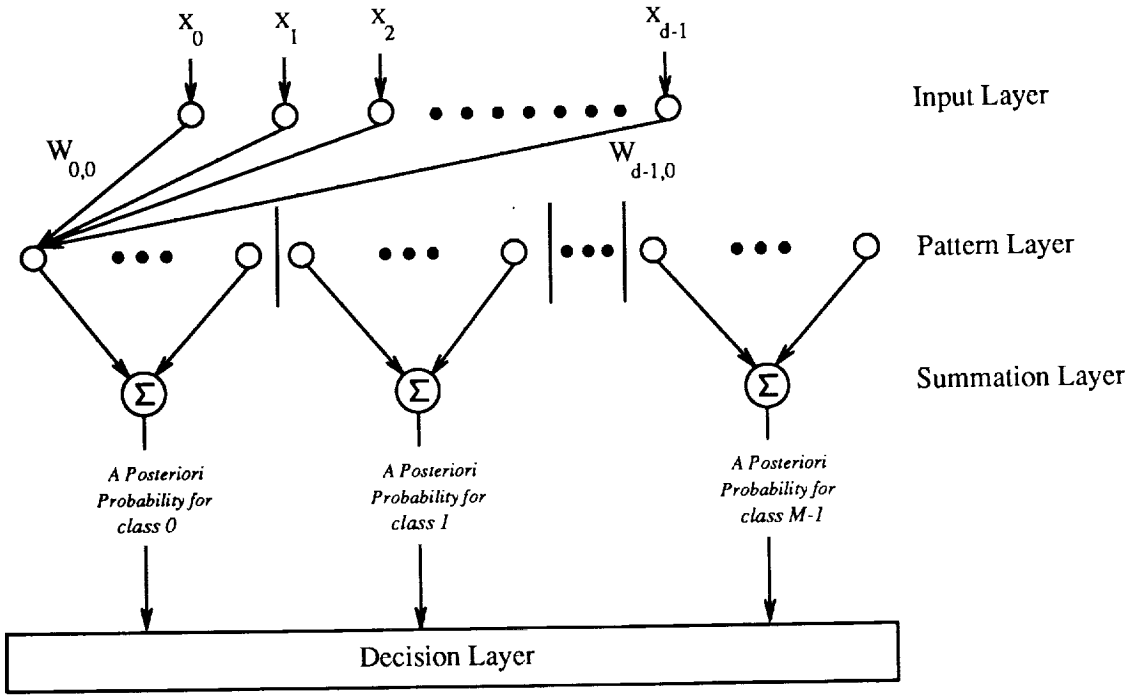


Figure 1: Feed forward implementation of Specht's discriminant analysis method

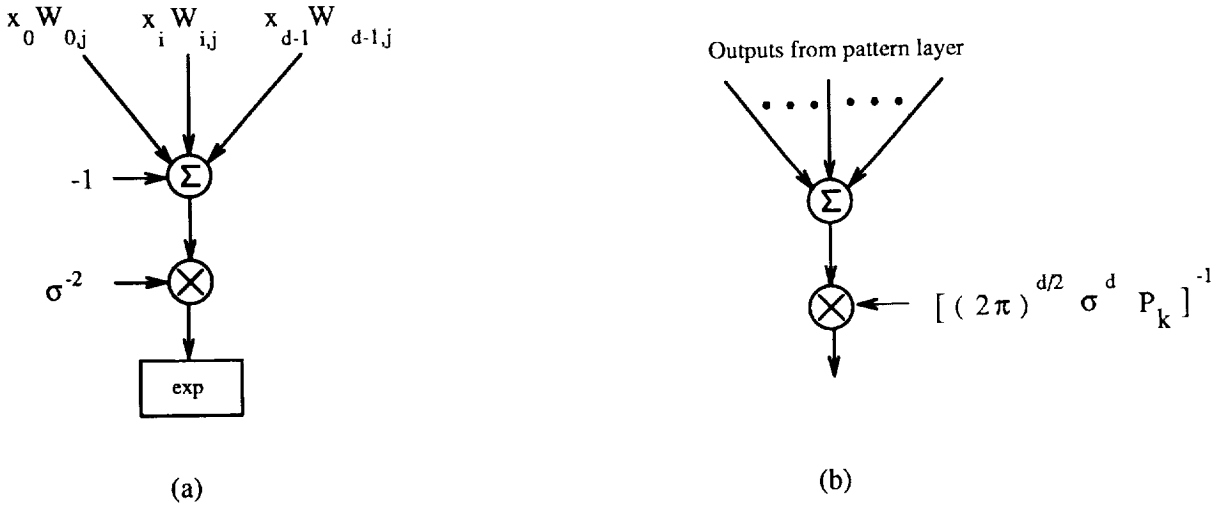


Figure 2: Details of pattern and summation layers

2.3 PNN advantages and disadvantages

In this subsection we discuss the advantages and disadvantages of using PNNs. In particular we focus on training time, retraining time, robustness to weight modification, computational load and memory requirements.

Advantages

1. Training time for the PNN is proportional to the total number of data vectors. In backpropagation the training time is roughly $O(d^6)$ [CCB92] where d is the dimensionality of the input vector. Also, the weights in the resulting backpropagation network do not bear any relationship to the training data and therefore are difficult to interpret. Depending on the flavor of the BPNN chosen, there are several parameters whose values have to be selected through heuristic means. These can affect the accuracy and generalization capability of the net.

   For example, a BPNN of the type described in [HKP91] requires that we randomly initialize the weights. The learning rate and the momentum are two additional parameters to be chosen. Different choices of these free parameters lead to different neural networks with different classification abilities. With the PNN there is only one free parameter σ to be chosen and with the negligible training time, many different nets can quickly be constructed with different values of σ and the best one chosen.

2. Retraining the PNN is easy since the hidden layer can be pruned or enlarged on demand. When a new data vector is received, it can be inserted as a node in the appropriate position in the pattern layer, and the weight connections are made from this node to the input layer. This is an $\mathcal{O}(d)$ process and an attractive feature when compared to BPNN, since the BPNN must be retrained (though not from scratch [SR87]) when new data arrive or when data from the original training set is removed.

3. The PNN is robust to weight removal. Infact, the weight removal and adjustment is the basis for the pruned PNN (PPNN). Our studies (which are also backed up by other recent work [Bur91]) indicate that the number of nodes in the pattern layer can be reduced by considerable amounts and yet give very accurate results. For further discussion on this see section 3.2. In contrast, due to the compact nature of the BPNN, weight deletion/node removal may severely impact classification accuracy.

Disadvantages

1. Since the entire training set is kept (and not an encoded version of it, as in the BPNN), the size of the hidden layer is very large as compared to the BPNN. This would be a shortcoming in computing environments where memory is scarce.

2. Processing speed is slower than BPNN since each input vector has to be evaluated over the entire training set. In a previous publication [CCB92], we have shown that the classification speed for a single input vector for the BPNN is $\mathcal{O}(d^2)$, where d is the length of the input vector. For the PNN, the speed is $\mathcal{O}(MP_kd^2)$, where M is the total number of classes and P_k is the number of exemplars in class S_k. On the surface, this would make the PNN and the BPNN have the same execution time. However, according to Kalayeh and Landgrebe [KL83],

for accurate classification P_k is proportional to d. Hence, in practice the network would have a speed that is $\mathcal{O}(d^3)$.

From our discussion, it is evident that the PNN would be very well suited for exploring dynamic environments. Such environments are commonplace in many scientific investigations of data. One of the goals of IDM's IIFS/SPUDS is the *high speed* classification of data into discipline specific indices for potential users of the system. In its computationally inefficient form, the PNN is incompatible with these goals. In the next section we address this major disadvantage of the PNN.

2.4 Speeding up feed–forward implementations of the PNN

The major disadvantage of the PNN is that *all* the training data are retained in the form of weights. This data can grow extremely large, making feed-forward evaluation of input vectors impossible in real time. One interesting way to prune the hidden layer is described in [Bur91]. In this paper, the author suggests that the data in the hidden layer be pruned using the Learning Vector Quantization algorithm of Kohonen [Koh89]. Unlike [Bur91], where the pruned PNN was applied to simulated bivariate uniform and Gaussian distributions, we apply it to higher dimensional data from real distributions. The LVQ method can be implemented as a feed–forward neural network working in both supervised and unsupervised mode [Sim90]. In the following paragraphs, we give a brief description of the algorithm in its supervised form. For more details the reader is referred to [HKP91].

In LVQ we are given a set of input vectors $\mathbf{X}_i^k$ where $\mathbf{X}_i^k$ represents the i^{th} vector from class S_k, $i = 1, \ldots P_k$ and P_k is the number of training patterns in class S_k. We select M_k vectors per class to represent the P_k vectors where $M_k \ll P_k$. We choose the minimum $M_k < P_k$ such that an acceptable level of accuracy is obtained when the P_k vectors are replaced by the M_k vectors in feedforward mode. This boils down to a trade off between computational efficiency and accuracy and will depend on the user's application. In this network, the initial weight vectors $\mathbf{w}_i^k$ are randomly chosen and a training vector is applied to the neural net input. For each weight vector a set of distances $\left|\mathbf{w}_i^k - \mathbf{X}_i^k\right|$ is calculated and the smallest one (denoted by $\mathbf{w}_{i^*}^k$) is chosen from this set. Next we move this weight closer to the input vector by the following update rules:

$$\Delta \mathbf{w}_{i^*}^k = \begin{cases} +\alpha(t)[\mathbf{X}_i^k - \mathbf{w}_{i^*}^k] & \text{if class is correct} \\ -\alpha(t)[\mathbf{X}_i^k - \mathbf{w}_{i^*}^k] & \text{if class is incorrect} \end{cases} \tag{5}$$

In equation (5) $\alpha(t)$ is the learning rate at time (or iteration level) t. The value of $\alpha(t)$ decreases as the number of iterations in the learning process increases. A common choice is $\alpha(t) = t^{-1}$. The number of iterations is denoted by t_{max}, and its range is $500 \leq t_{max} \leq 10000$. Our practical experience indicates that choosing t_{max} in the range above is sufficient for convergence. In fact, using $t_{max} > 500$ did not lead to great increases in accuracy.

Now the key point in the LVQ pruning of the PNN is that $M_k \ll P_k$, i.e., the number of prototype vectors M_k is much less than the number of vectors P_k in the hidden layer, yet gives an adequate representation of all the P_k vectors in that class. Hence, the time for feedforward classification of input vectors can be decreased, and the memory requirements reduced. We have performed experiments in pruning the PNN so as to improve the feedforward speed. These results

Table 1: Distribution of data, Blackhills, South Dakota

	Training No. of pixels	Entire image No. of pixels	Class name USGS - Level I
0	453	6676	Urban
1	478	42432	Agricultural
2	464	16727	Rangeland
3	482	194868	Forested Land
4	0	0	Water bodies
5	0	0	Wetland
6	368	1441	Barren
7	0	0	Tundra
8	0	0	Perennial snow and ice

are presented in section 3.2, Table 3 and indicate that pruning the PNN is a viable computational scheme.

3 Application of the PNN to remote–sensing

In this section we describe the data on which we tested our PNN, discuss the selection process that we employed for the training and testing data, elaborate upon the training and testing methodology used, and finally, present results for the basic PNN and the LVQ–pruned version of the PNN.

3.1 Description of data set

The data set that was used for training and testing the PNN is called the Blackhills data set, generated by the Landsat 2 multispectral scanner (MSS) (see Figure 3). This data set was previously used to compare backpropagation neural networks with Gaussian maximum likelihood classification in [CCB92]. The spectral bands are 0.5 - 0.6μm (green), 0.6 - 0.7μm (red), 0.8 - 1.1μm (near-infrared). These bands correspond to channels 4 through 7 of the Landsat sensors. There are 262,144 pixels corresponding to a 512×512 image size, and each pixel represents approximately $79m \times 79m$ on the ground. The image region covers a range of latitudes from $44°15'$ to $44°30'$ and longitudes from $103°30'$ to $103°45'$; the images were obtained in September 1973. The ground reference data set was also provided in the form of United States Geological Survey level II land use/land cover data [AHRW76]. Since we were only interested in level I classification, the different classes were conglomerated into the various higher level classes in the hierarchy; the distribution of pixels is shown in column three of Table 1. For example, from Table 1 we know that there are a total of 6676 pixels in the urban class, 453 of which were used for training the PNN.

3.2 Training and testing the PNN

The ground reference data set was viewed on a display device to get an idea of the spatial distribution of the ground reference data pixels. According to [Ric86], a minimum sample size of 60 pixels is necessary for accurate classification. Also, according to [Cam87], a large number of smaller training sites should be used rather than a few large ones. Following these recommendations, we formed training sets from the Blackhills data set. The distribution of training samples is summarized in column two of Table 1.

The PNN classifier is derived from the training group and the error estimate obtained from the test group. This method is known as the "holdout" or H method of estimating errors. We do not use the *leaving-one-out* method of training and testing as described in [WK89] since this method is extremely time consuming and only leads to marginally better accuracy [KC68] in testing for large data sets (which is the case for us).

Results for testing the trained PNN on the image are shown in the contingency table (Table 4). Each entry C_{ij} in the matrix represents the number of times a pixel in class i was put into class j. C_{ii} is the number of correct classifications in class i. In Table 4 the left-hand side set of values represents the classification results of the Unpruned PNN (UPNN) while the pruned PNN (PPNN) results are on the right. The percent correctly classified (PCC) for the UPNN is 0.697, while it is 0.737 for the PPNN. While a 4% difference might seem small, for a 512×512 image this amounts to approximately 10,000 additional pixels being correctly classified. The future generation of Earth orbiting platforms will transmit terabytes to petabytes of data, so small percentage changes in accuracy of classification will lead to large absolute changes in classification accuracy.

A closer look at Table 4) indicates that for forest and urban land cover classes (categories 0 and 3), the PPNN performs better whereas in the other cases, it is not as good. This can be explained on the basis that categories 0 and 3 have less spread in their data vectors (i.e., they are clustered very tightly), hence representing them by a smaller set of vectors leads to no degradation in classification ability. However the samples of the other classes (1, 2, 6) have a larger in-class variability and hence their multidimensional spatial clusters are not compact leading to poorer representation by a smaller set of vectors.

To study the change in the classification ability of the UPNN with σ, we varied σ in increments of 2 from 2 through 12, with the entire training set being used. The results are presented in Table 2. The results indicate that there is some variability in the classification with σ and that as σ grows larger, the UPNN PCC tends toward the Gaussian Maximum Likelihood Estimate with PCC = 0.653 as discussed in [CCB92].

Another test that we performed was to see the variability of the neural net accuracy as we changed the number of hidden nodes in the PPNN. In this test, $\sigma = 4$ since that was the case for which we got maximum PCC in the unpruned PNN. These results are shown in Table 3. We found that as the number of nodes increased there was a general increase in the accuracy of the PPNN. The four node case is an anomaly, since some classes were classified well while other classes were classified extremely poorly (to the extent of having less than 25 pixels put in the rangeland class, i.e., about 0.1% of the total number of pixels).

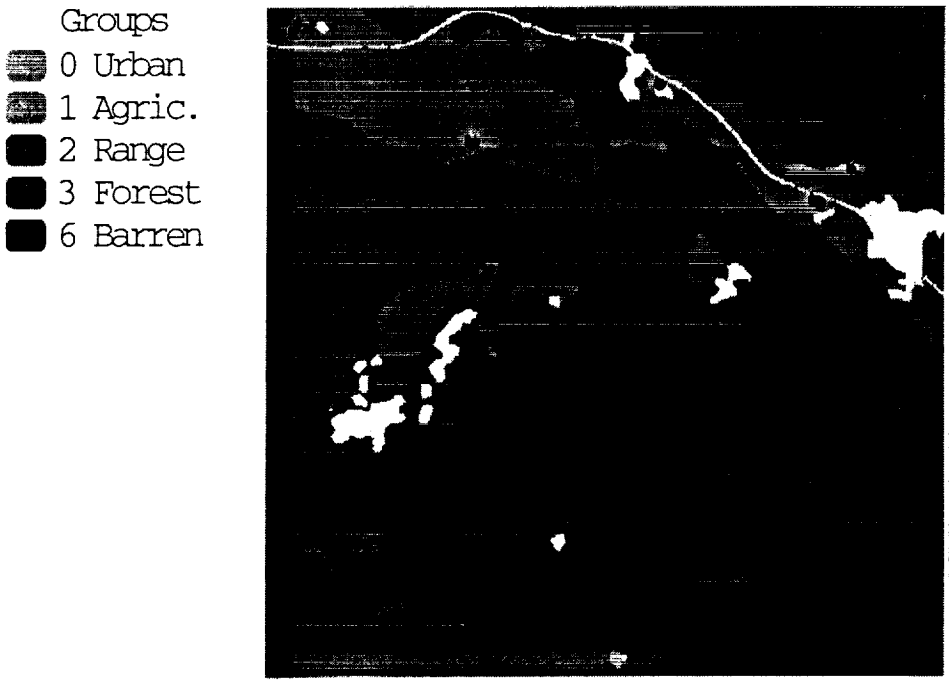


Figure 3: Ground reference data for the Blackhills image

Table 2: Variation of PCC with σ

σ	2	4	6	8	10	12
PCC	0.677	0.697	0.685	0.674	0.670	0.667

Table 3: Variation of PCC with number of nodes/class

nodes/class	4	10	20	40	80
PCC	0.744	0.720	0.721	0.737	0.742

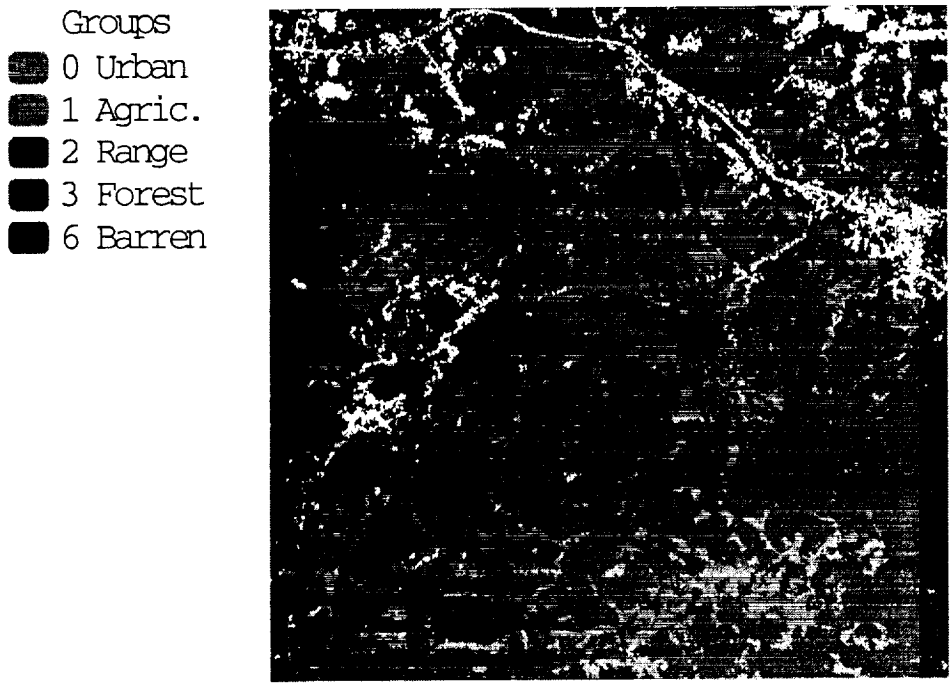


Figure 4: PNN classified Blackhills dataset

Table 4: Contingency table for PNN; Raw PNN on left with $\sigma = 4$ (PCC = 0.697); PNN-LVQ combination on right with $\sigma = 4$ (PCC = 0.737)

	0	**1**	**2**	**3**	**6**	Total pixels	**0**	**1**	**2**	**3**	**6**
0	**0.410**	0.165	0.228	0.153	0.044	6623	**0.509**	0.090	0.147	0.172	0.083
1	0.162	**0.418**	0.378	0.035	0.007	41954	0.215	**0.09**	0.147	0.172	0.083
2	0.110	0.286	**0.532**	0.071	0.002	16263	0.114	0.245	**0.381**	0.129	0.132
3	0.009	0.064	0.139	**0.785**	0.003	194386	0.010	0.070	0.041	**0.861**	0.018
6	0.196	0.118	0.128	0.177	**0.381**	1073	0.257	0.065	0.056	0.534	**0.088**

4 Concluding remarks and future work

In this research, we have described the Probabilistic Neural Network (PNN) and have applied it to remotely sensed imagery. The accuracy obtained by the PNN is better than the Gaussian Maximum Likelihood Classifier (GMLC) and not as good as the Backpropagation Neural Network (BPNN). On the other hand, the training time in the PNN is very small when compared to the other two methods. In addition the network is robust to weight changes and has very small retraining time, making it highly suitable for highly variable and dynamic environments. A modified version of the PNN (LVQ–PNN or PPNN) was discussed and compared with the raw PNN. For the chosen data set, the PPNN performed better than the raw PNN.

Future work includes additional research on the PNN and using the PNN in applications, as described next.

4.1 Planned extensions to the PNN

As has already been discussed, it is sensible to automate the pruning of the PNN. In this process the number of nodes in each class would be allowed to grow or decrease independently of each other such that the PCC per class would be optimized.

In addition, the PNN algorithms described throughout the paper are all readily adaptable for a parallel architecture implementation, most likely on a machine such as the MasPar MP-1, a 16,384 processing element SIMD machine.

4.2 Incorporation of the PNN into a metastrategy

The Intelligent Data Management group has developed a number of automatic characterization algorithms drawn from backpropagation networks, PNN, Adaptive Resonance Theory (ART) networks [CG87], decision trees, Fourier analysis and wavelet theory. Each of these methods has its own unique strengths and weaknesses, and there are cases where one may falter while another excels. We plan to attempt to develop a metastrategy that draws on its knowledge of each of these techniques to produce a hybrid characterization algorithm that performs at least as well as any single one of these components [Fin90].

5 Acknowledgements

A number of people within and without the IDM group have contributed to this work. The authors would like to thank William J. Campbell, George Fekete, Robb Lovell and Nicholas Short, Jr. for their help during this research.

References

[AHRW76] J. R. Anderson, E. E. Hardy, J. T. Roach, and R. E. Witmer. A land use and land cover classification system for use with remote sensor data. Geological Survey Professional Paper 964, United States Government Printing Office, Washington, D.C., 1976.

[And72] H. C. Andrews. *Introduction to Mathematical Techniques in Pattern Recognition.* Wiley–Interscience, New York, 1972.

[Bur91] P. Burrascano. Learning vector quantization for the probabilistic neural network. *IEEE Trans. on Neural Networks*, 2(4):458–461, 1991.

[Cam87] J. B. Campbell. *Introduction to Remote Sensing.* Guilford Press, New York, 1987.

[CCB92] S. R. Chettri, R. F. Cromp, and M. Birmingham. Design of neural networks for classification of remotely sensed imagery. In *1992 Goddard Conference on Space Applications of Artificial Intelligence*, pages 137–149. National Aeronautics and Space Administration, 1992.

[CCS92] R. F. Cromp, W. J. Campbell, and Jr. Short, N. M. An intelligent information fusion system for handling the archiving and querying of terabyte-sized spatial databases. In *International Space Year Conference on Earth and Space Science Information Systems.* American Institute of Physics, 1992.

[CG87] G. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing ART architecture in a nonstationary world. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[CHC89] W. J. Campbell, S. E. Hill, and R. F. Cromp. Automatic labeling and characterization of objects using artificial neural networks. *Telematics and Informatics*, 6(3–4):259–271, 1989.

[Fin90] N. V. Findler. *Contributions to a Computer-Based Theory of Strategies.* Springer-Verlag, Berlin, 1990.

[HKP91] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, California, 1991.

[KC68] L. Kanal and B. Chandrasekaran. On dimensionality and sample size in statistical pattern recognition. In *Proc. Nat. Electron. Conf.*, pages 2–7, 1968.

[KF72] W. L. G. Koontz and K. Fukunaga. Asymptotic analysis of a nonparametric estimate of a multivariate density function. *IEEE PAMI*, 21:967–974, 1972.

[KL83] H. M. Kalayeh and D. A. Landgrebe. Predicting the required number of training samples. *IEEE Trans. on Patt. Anal. and Mach. Intelligence*, 5:664–667, 1983.

[Koh89] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

[MAC+92] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5:595–603, 1992.

[Par62] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:308–319, 1962.

[Ric86] J. A. Richards. *Remote Sensing Digital Image Analysis, an Introduction.* Springer-Verlag, Berlin, 1986.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[Sim90] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations.* Pergammon Press, New York, 1990.

[Spe90] D. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[SR87] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:367–372, 1987.

[TK76] M. E. Tarter and R. A. Kronmal. An introduction to the implementation and theory of nonparametric density estimation. *The American Statistician*, 30:105–112, 1976.

[TT78] R. A. Tapia and J. R. Thompson. *Nonparametric Probability Density Estimation.* Johns Hopkins University Press, Baltimore, 1978.

[WK89] S. M. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Eleventh Int. Joint Conference on Artificial Intelligence*, pages 781–787. American Association of Artificial Intelligence, 1989.

IMAGE ANALYSIS BY INTEGRATION OF DISPARATE INFORMATION

Jacqueline Le Moigne
Information Science and Technology Office and
Center of Excellence in Space Data and Information Sciences
Mail Codes 930.4 and 930.5
NASA Goddard Space Flight Center
Greenbelt, MD 20771
Tel: (301) 286-8723
Fax: (301) 286-5152
lemoigne@nibbles.gsfc.nasa.gov

ABSTRACT

Image analysis often starts with some preliminary segmentation which provides a representation of the scene needed for further interpretation. Segmentation can be performed in several ways, which are categorized as pixel-based, edge-based, and region-based. Each of these approaches are affected differently by various factors, and the final result may be improved by integrating several or all of these methods, thus taking advantage of their complementary nature.

In this paper, we propose an approach that integrates pixel-based and edge-based results by utilizing an iterative relaxation technique. This approach has been implemented on a massively parallel computer and tested on some remotely sensed imagery from the Landsat-Thematic Mapper (TM) sensor.

1. INTRODUCTION

After pre-processing of some original data, image segmentation is the process which generates a spatial description of an image as a set of specific parts, regions or objects. The "segmented" output is then utilized by a higher-level image interpretation process. There is no single standard approach to segmentation which would be "successful" for any type of data, but some general methods have been developed based on the two main characteristics of regions or objects in an image:

(1) each region or object exhibits an internal uniformity with respect to some image property (e.g., gray level, color, texture),

(2) each region or object presents some contrast with its surroundings.

These two properties lead to three different types of segmentations, pixel-based, region-based and edge-based segmentations.

Each of these approaches is affected differently by various factors. Pixel-based methods form their decision only based on the information given at each pixel, while the two other types of segmentation take into account the information contained in the surrounding pixels. Usually in a pixel-based approach, all of the original information is utilized, thus avoiding a selection process. Such methods are also easier to integrate in a learning process, but their main

drawback is that they do not take into account spatial information. Conversely, edge- and region-based approaches base their decision on spatial information. Edge-based methods measure the variation of intensity between pixels belonging to different objects; they may produce excellent results for unevenly illuminated images, but they also can be very sensitive to noise. Region-based approaches, which measure the internal uniformity of some intensity or texture function, often produce spurious segmentation under non-uniform lighting conditions, but are usually less sensitive to noise.

In general, these three types of segmentation may be improved by integrating them and by taking advantage of their complementary nature. We previously proposed an approach that integrates region segmentation and edge detection results by interpreting a binary tree representation (Le Moigne, 1992), thus producing a refined region segmentation. This algorithm has been tested on Landsat-TM data. The integration of edge and region data may also be performed by a relaxation method and has been proposed in (Le Moigne, 1989); in the work described in this paper,we refine this relaxation method for the purpose of integrating edge and classification information and we implement it on a massively parallel computer, the MasPar MP-1. Then, we test this approach on remotely sensed imagery, such as Landsat-TM data.

2. RELAXATION TECHNIQUES

a. Overview

A large number of iterative relaxation schemes have been proposed to improve the results given by such basic processes as edge detection, region segmentation or pixel classification (Davis, 1981; Hummel, 1987; Faugeras, 1981; Peleg, 1980; Zucker, 1977). The principle of these algorithms is to utilize contextual information for iteratively changing the initial labeling of the objects in a scene toward optimal labeling.We will concentrate on relaxation methods for which the decisions at each point are taken in a probabilistic fashion. This general class of relaxation techniques is described in (Davis, 1981). Let us assume that we have a set of N objects $\{O_1, O_2, \ldots, O_N\}$ (e.g., the pixels) to be labeled into one of L classes $\{C_1, C_2, \ldots, C_L\}$ and that $p_i^n[C_k]$ is the probability that the object O_i is assigned to the class C_k at the iteration n. The principle of the relaxation algorithms, then, is to build a series of probability sets $\{p_i^n[C_k] ; 1 \le i \le N; 1 \le k \le L\}$, where each new iteration step, n, adjusts the probabilities according to the contextual information, and these probabilities satisfy the conditions (1) and (2):

$$\text{for every i, for every k,} \quad 0 \le p_i^n[C_k] \le 1, \qquad (1)$$

and

$$\text{for every i,} \quad \sum_{k=1}^{L} p_i^n[C_k] = 1. \qquad (2)$$

Thus, the problem is to define the updating formula for $p_i^n[C_k]$. The simplest updating (which we use in this work) is described below.

The relaxation scheme assumes that the class assignments of each object depend on the class assignment of the "other" objects; the "other" objects can be defined, for example, by neighboring objects. Therefore, we define

$c[i,k;j,l]$ as the compatibility coefficient between the object O_i with the label C_k and the object O_j with the label C_l. We assume that the $c[i,k;j,l]$'s belong to the range interval $[0,+1]$ and are positive if there is compatibility and equal to 0 if there is high incompatibility. The coefficient $c[i,k;j,l]$ can be defined as a conditional probability, $p(i \in C_k / j \in C_l)$, which provides a probabilistic framework. Let $q_i^n[C_k]$ be the global compatibility for the object O_i with the label C_k; it is defined by

$$q_i^n[C_k] = \frac{1}{V[i]} \sum_{j=1}^{j=V[i]} \sum_{l=1}^{l=L} c[i,k;j,l]\, p_j^n[C_l] \quad (3)$$

where $V[i]$ is the number of objects in the neighborhood of object O_i. Then $q_i^n[C_k]$ is the "increment" which is applied to update $p_i^n[C_k]$ and compute the new probability set $\{p_i^{(n+1)}[C_k]\}$ (see (Davis, 1981) for details):

$$p_i^{n+1}[C_k] = \frac{p_i^n[C_k] \times q_i^n[C_k]}{\sum_{l=1}^{l=L} p_i^n[C_l] \times q_i^n[C_l]} \quad (4)$$

This multiplication still ensures that the conditions (1) and (2) are satisfied. Besides, if the global compatibility of the object O_i with the label C_k is higher than all the compatibilities of O_i with the other labels C_l, then the $P_i^n[C_k]$'s increase relative to the other $p_i^n[C_l]$'s. That means that this scheme provides an overall improvement of the labeling but it does not guarantee the convergence toward stable labeling.

Other relaxation schemes (Faugeras, 1981; Peleg, 1980) utilize different formulas or different frameworks. For example in (Faugeras, 1981), another way of updating the probabilities $p_i^n[C_k]$ is given; the principle of this other method is to minimize a criterion by the "projected gradient" optimization method. The criterion measures the ambiguity and the consistency of the current labeling at each step. This algorithm provides us with a converging sequence of probabilities P^n. Starting from an initial point P^0, the method converges toward a local minimum in the vicinity of P^0.

b. Utilizing Relaxation to Integrate Disparate Information

We now describe how a relaxation technique, such as the one described above, can be utilized to integrate knowledge from edge detection and pixel classification.

Previously, Zucker and Hummel (Zucker, 1977) simultaneously used edge and region data in a relaxation process for labeling dots. Their goal was to provide a low-level description of the roles of dots in cluster analysis. Unlike our approach, they used these two types of data "equally", i.e., both edges and regions defined the labels and the initial probabilities. There were ten different labels, with eight edge labels at various orientations, one "region" label called "interior point" label, and one "noise" label.

In our approach, the labels are all region labels, and edge data are utilized to update the labeling by way of the compatibility coefficients. This approach tends to be more general and this algorithm can be very easily applied to the fusion of various types of data. In our work, the

definition of the labels and the initial probabilities are provided by a neural network pixel classification (Chettri, 1992), then the relaxation process updates the initial labeling by using some edge detection results, e.g., from a Canny operator (Canny, 1986). These standard techniques, pixel classification and edge detection, can easily be changed without altering the definition of the overall relaxation algorithm.

Once the initial probabilities are given, the fusion is realized by computing the compatibilities between neighboring pixels. Coefficients $c[i,k;j,l]$ in formula (3) represent the compatibilities between object O_i with the label C_k and object O_j with the label C_l. If a relaxation is performed only on regions, these coefficients represent the compatibilities of neighboring pixels belonging to given regions. If both regions and edges are considered, both information can be integrated into the compatibility coefficients by utilizing the following equation:

$$c[i,k;j,l]= p_r(C_k/C_l) \times F_{k\&l}(i,j) \quad (5)$$

where $p_r(C_k/C_l)$ is the "region-probability" (independent of edges) of i belonging to the class C_k if the neighbor j belongs to the class C_l; this probability could be estimated from some previous ground truth data. The last term in Eq. (5), $F_{k\&l}(i,j)$, is a function that varies in [0,1] and is:

- closer to 1 if $C_k = C_l$ (or, more generally, if regions C_k and C_l are similar in tone) and i is not an edge point, or if regions C_k and C_l are different and i is an edge point,
- closer to 0 if $C_k = C_l$ and i is an edge point, or if regions C_k and C_l are different and i is not an edge point.

For example, F can be defined by:

$$F_{k\&l}(i,j) = K_{k\&l} \times Mag(i) + (1- K_{k\&l}) \times (1- Mag(i))$$

where $K_{k\&l}$ is equal to 0 if k=l and equal to 1 if $k \neq l$, and Mag(i) is the magnitude of the gradient computed at the point i and normalized between 0 and 1.

The magnitude of the gradient at neighboring point j would also be taken into account and we could utilize the following formula:

$$F_{k\&l}(i,j)= K_{k\&l} \times Mag(i) \times Mag(j) + (1- K_{k\&l}) \times (1- Mag(i) \times Mag(j)).$$

This updating has not been implemented yet, but will be considered in future work.

Therefore, both region and edge information participate simultaneously in the updating of the labeling. Results are presented in Section 4.

3.PARALLEL IMPLEMENTATION

Computation time is the main concern of relaxation techniques. However these techniques are characterized by parallel local processing which can be easily implemented on an architecture that favors computations between adjacent pixels (Fishler, 1987). The method discussed in this paper has been implemented on a MasPar MP-1. The MasPar Parallel Processor is a fine-grained, massively parallel SIMD architecture, with 16,384 parallel processing elements arranged in a 128x128 matrix and connected by an eight nearest neighbors interconnection network.

The parallel implementation of the relaxation algorithm is straightforward, with the quantities $p_i^n[C_k]$ and $q_i^n[C_k]$ computed in parallel at each pixel. Timings are given in Table 1 for various size images and various numbers of labels.

4. RESULTS

Figure 1 shows the results of this algorithm on a test image. In this example, we assume that the "ideal" image (or "ground truth") is composed of two distinct regions and we choose some compatibility coefficients which reflect this assumption; the initial probabilities present three classes, two classes corresponding to the two "ideal" regions and one "artefact" class. The edge image presents strong edges at the border between the two "ideal" regions and the edge magnitudes decrease with the distance to this border. For this example, we notice that if the relaxation algorithm is utilized without any edge information, one of the labels "takes over" the whole image after only three iterations. If the edge information is integrated in the relaxation process, the two regions are still separated after 10 iterations and the labeling seems to be stable.

Figures 2 to 5 present results of the algorithm applied on a Landsat-TM scene ("Washington D.C. region") shown at the top left corner of Figure 2. Initial probabilities for this scene have been obtained from a classification into seven labels, utilizing a probabilistic neural network (see (Chettri, 1993) for details). The 7 labels correspond to the classes "urban", "agriculture", "rangeland", "forest", "waterbodies", "wetland", and "bareland".

The algorithm described in section 2 was applied to this initial classification, using two different numbers of labels; first, we grouped these 7 labels into the 3 classes "urban", "agriculture", and "other". Figures 2 and 3 show the results without and with the edge information. When no edge information is utilized (Figure 2), one of the labels (label 0) has "disappeared" after 20 iterations. When the edge information is taken into account (Figure 3), the three labels are still present after 30 iterations, and the labeling seems to stay stable after 20 iterations.

Then, the 7 labels defined previously are considered, and we obtain similar results: see Figures 4 and 5. When no edge information is utilized, only three labels are left after 30 iterations and one of the three is covering most of the image; but when the edge information is integrated in the updating formula, all the initial labels are still represented after 30 iterations and the labeling seems to stabilize after 20 iterations. Also, the results obtained after 10 or 30 iterations can be compared to the ground truth data shown at the top left corner of Figure 5: qualitatively, we can observe an overall improvement of the segmentation (e.g., suppression of small isolated pixels or groups of pixels), but some features (e.g., a road) have been regrouped with a neighboring region. A quantitative evaluation of these results will be performed later.

The previous results show the importance of the edge information and how the integration of edge- and the pixel-based segmentations can improve the final result.

Other similar results have also been obtained on AVHRR data and will be presented at the conference.

5. CONCLUSION

The results presented in this paper show how the integration of complementary information, such as pixel- and edge-based techniques, can improve the final segmentation.

More work needs to be done, especially in the definition of the initial probabilities, in the choice of the compatibility coefficients, and in the quantitative evaluation of the results. Also, the results presented in section 4 seem to show that, when the edge information is utilized, the relaxation process becomes "stable" after a certain number of iterations: this issue of "convergence" will be studied, and different schemes, such as the one presented in (Faugeras, 1981) will be investigated.

Acknowledgments

The author wishes to thank James C. Tilton and Samir R. Chettri for the original data that were used in this work, as well as Samir R. Chettri for the neural network classification results that were used to build the initial probabilities of Figures 2 to 5.

REFERENCES

Canny, J., (1986), "Computational Approach to Edge Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-8, 679-698.

Chettri, S.R., (1993), "The Probabilistic Neural Network Architecture for High Speed Classification of Remotely Sensed Imagery," submitted to 1993 Goddard Conference on Space Applications of Artificial Intelligence.

Davis, L.S., & Rosenfeld, A., (1981), "Cooperating Processes for Low-Level Vision: A Survey," Artificial Intelligence 17, 245-264.

Faugeras, O.D., & Berthod, M., (1981), "Improving Consistency and Reducing Ambiguity in Stocchastic Labeling: An Optimization Approach," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 4, 412-424.

Fishler, M.A., & Firschein, O., (1987), "Parallel Computer Architectures for Computer Vision," in Readings in Computer Vision: Issues, Problems, Principles and Paradigms, edited by M.A. Fishler and O. Firschein, Morgan Kaufmann Publishers, Los Altos, California, 768-772.

Hummel, R.A., & Zucker, S.W., (1987), "On the Foundations of Relaxation Labeling Processes," in Readings in Computer Vision: Issues, Problems, Principles and Paradigms, edited by M.A. Fishler and O. Firschein, Morgan Kaufmann Publishers, Los Altos, California, 585-605.

Le Moigne, J., (1989), "Data Fusion by Relaxation," in Proc. of Intelligent Autonomous Systems Intern. Conference, Amsterdam, December 11-14, 368-377.

Le Moigne, J., & Tilton, J.C., (1992), "Refining Image Segmentation by Integration of Edge and Region Data," IGARSS'92, Houston, Texas, May 26-29.

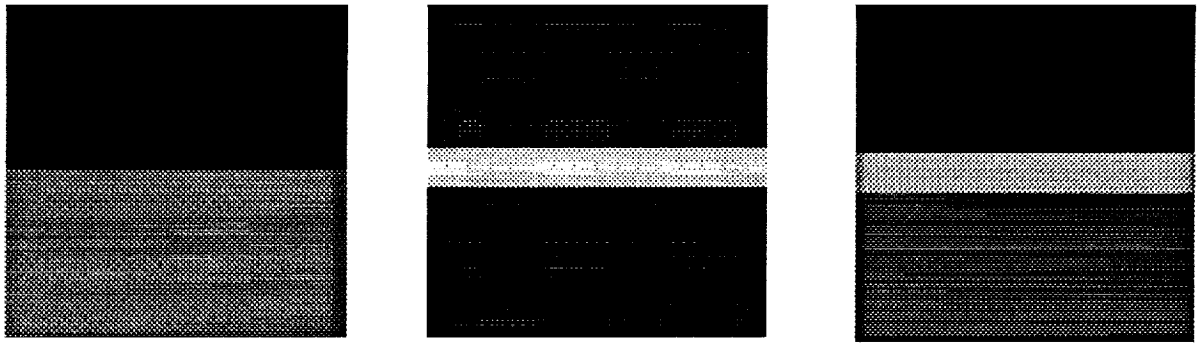
Peleg, S., (1980), "A New Probabilistic Relaxation Scheme," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 4, 362-369.

Zucker, S.W., & Hummel, R.A., (1977), "Computing the Shape of Dot Clusters, I: Labelling Edge, Interior, and Noise Points," University of Maryland Computer Science Center Technical Report No 543.

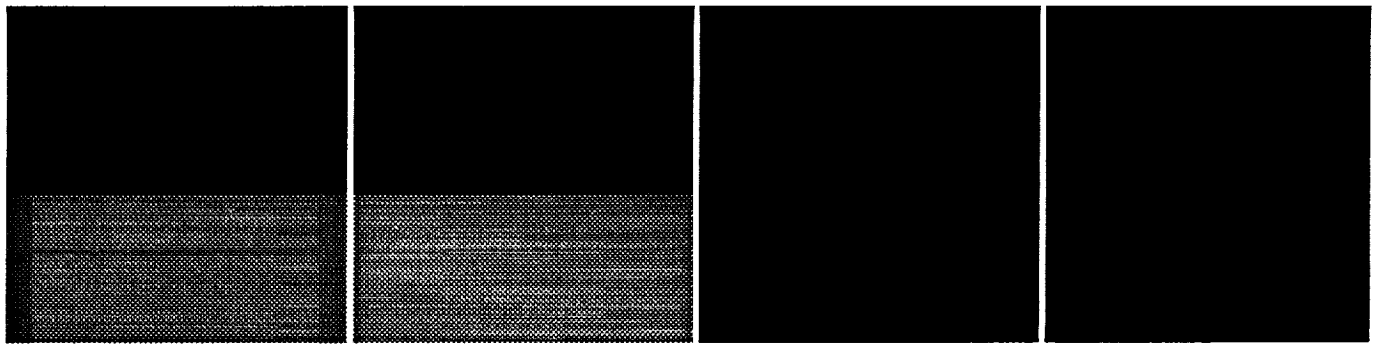
Image Size	# Labels	Edges ?	Time per Iteration
256*256	7	no	0.36
256*256	7	yes	0.41
256*256	3	no	0.10
256*256	3	yes	0.11
512*512	2	no	0.17
512*512	2	yes	0.17

Table 1

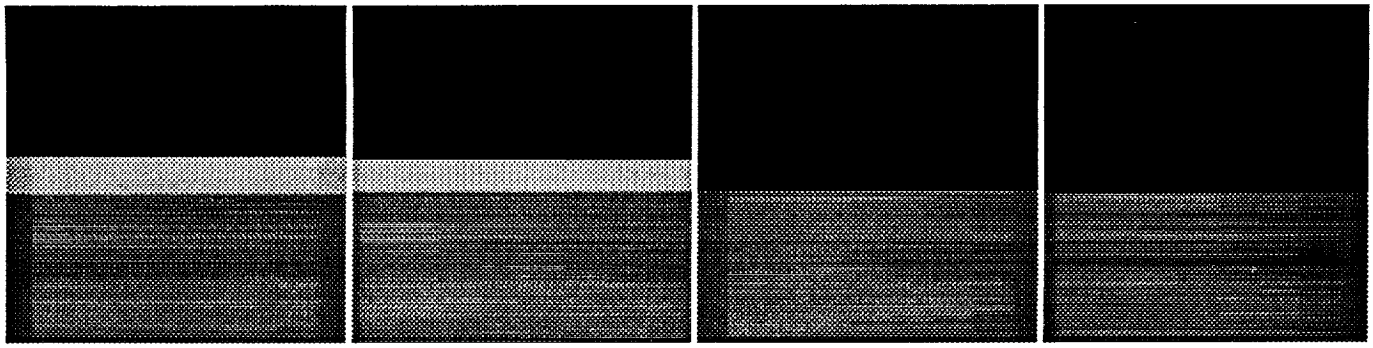
Timings Obtained for One Relaxation Iteration on a MasPar MP-1



"Ideal" Edges Initial Probabilities



Iter 1/ no edges Iter 2/ no edges Iter 3/ no edges Iter 10/ no edges



Iter 1/ edges Iter 2/ edges Iter 3/ edges Iter 10/ edges

Figure 1
Results of our Method on a Test Example

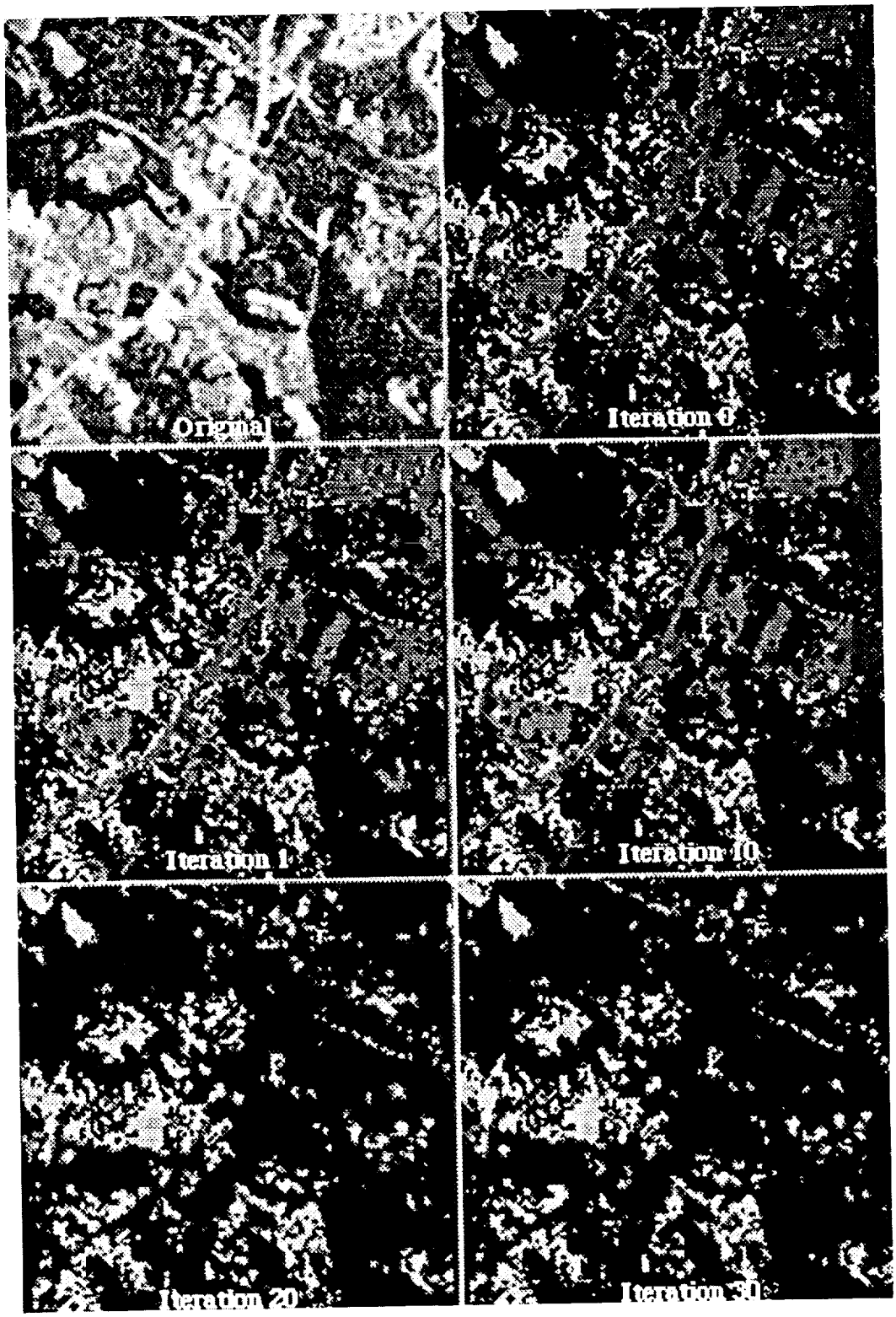


Figure 2
Relaxation Results Without Edge Information on a Landsat-TM scene (3 Labels)

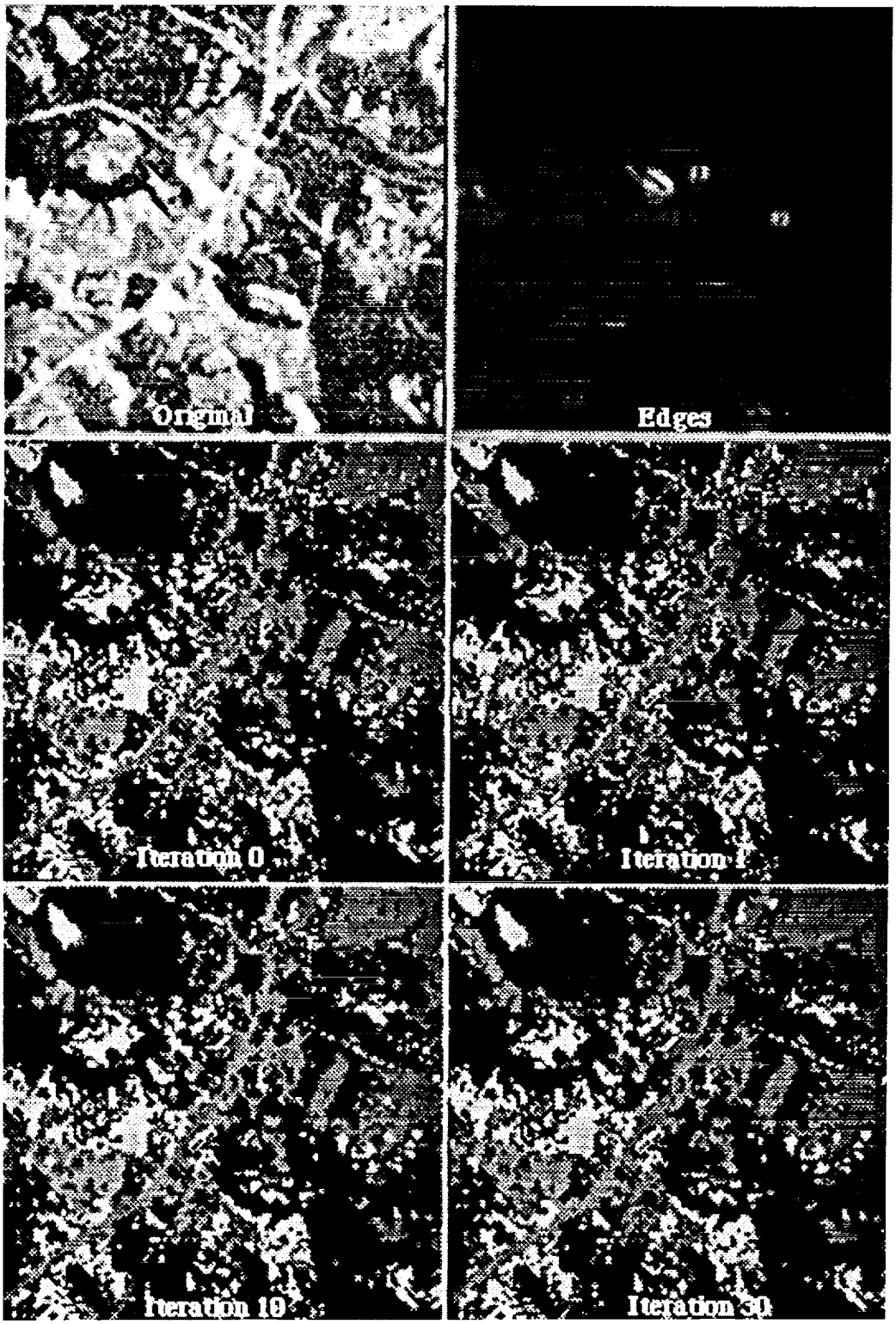


Figure 3
Results of the "Relaxation With Edges" Method on a Landsat-TM scene (3 Labels)

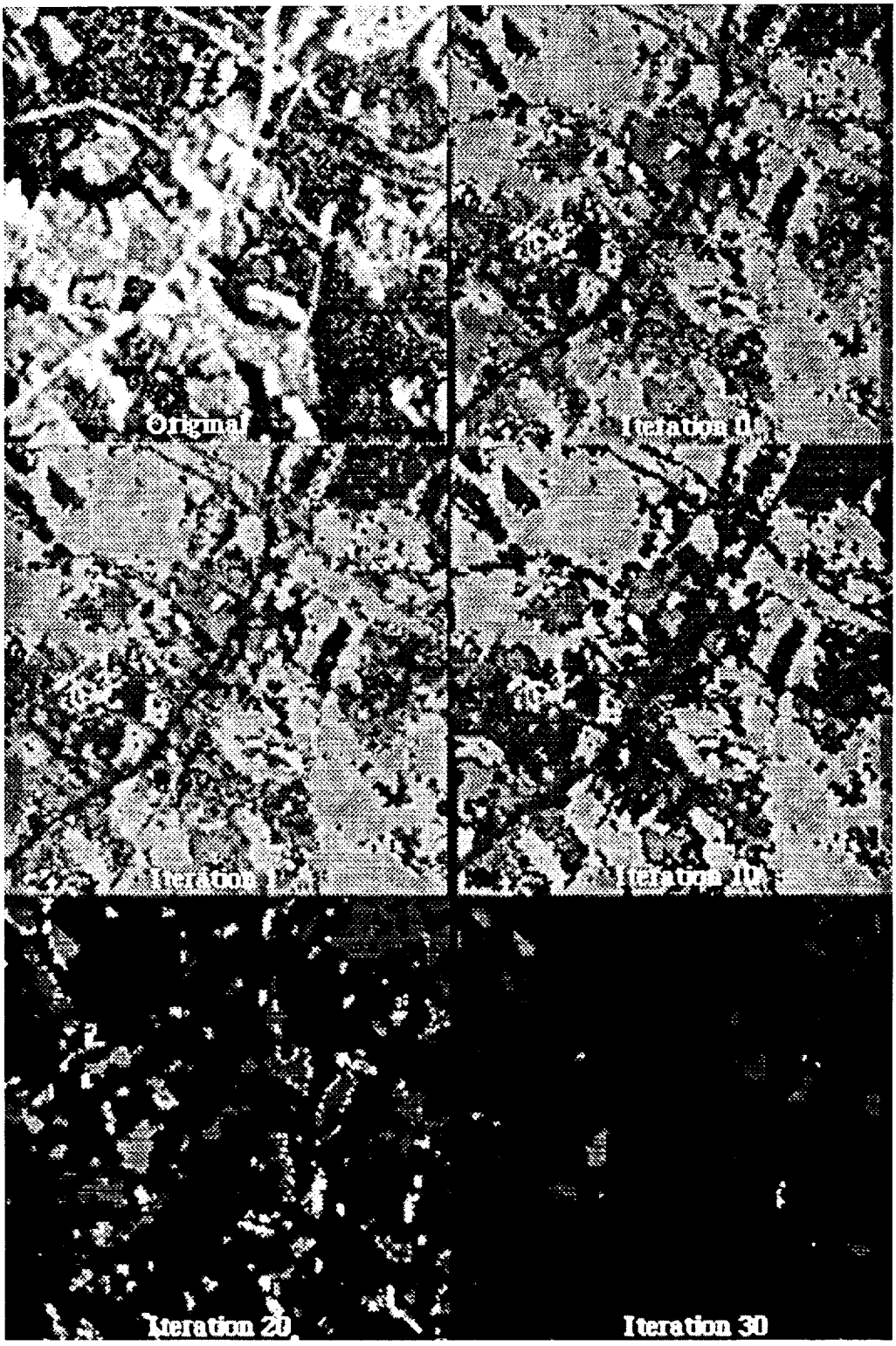


Figure 4
Relaxation Results Without Edge Information on a Landsat-TM scene (7 Labels)

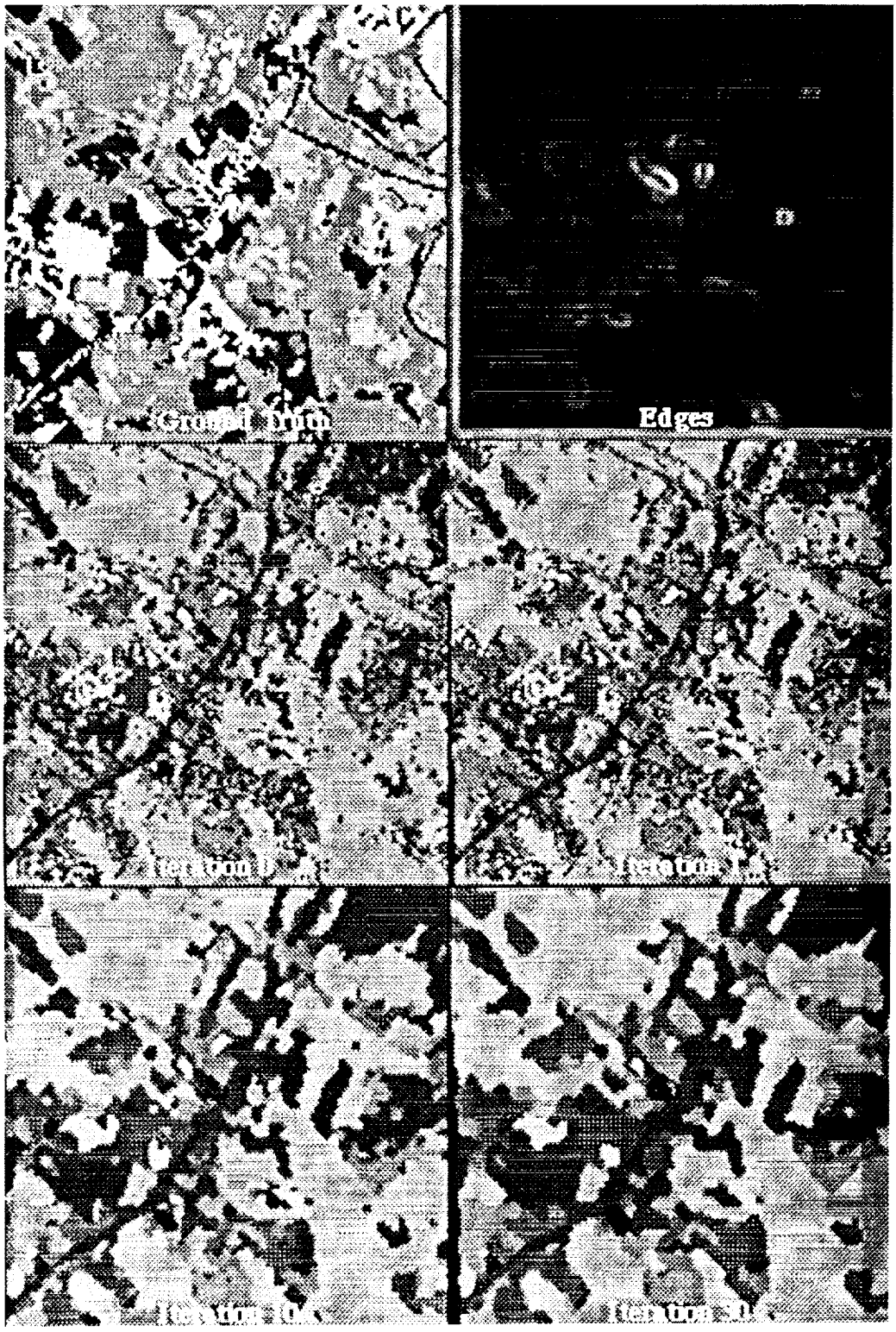


Figure 5
Results of the "Relaxation With Edges" Method on a Landsat-TM scene (7 Labels)

DATA FUSION WITH ARTIFICIAL NEURAL NETWORKS (ANN) FOR CLASSIFICATION OF EARTH SURFACE FROM MICROWAVE SATELLITE MEASUREMENTS

Y. M. Fleming Lure[+], Norman C. Grody[*], Y. S. Peter Chiou[+], and H. Y. Michael Yeh[+]

[+] Caelum Research Corporation, Silver Spring, MD 20901 (Tel) (301) 593-1748, (FAX) (301) 593-3951

[*] National Oceanic and Atmospheric Administration (NOAA), Washington, D. C. 20233

ABSTRACT

A data fusion system with artificial neural networks (ANN) is used for fast and accurate classification of five earth surface conditions and surface changes, based on seven SSMI multi-channel microwave satellite measurements. The measurements include brightness temperatures at 19, 22, 37, and 85 GHz at both H and V polarizations (only V at 22 GHz). The seven channel measurements are processed through a convolution computation such that all measurements are located at same grid. Five surface classes including non-scattering surface, precipitation over land, over ocean, snow, and desert are identified from ground-truth observations. The system processes sensory data in three consecutive phases : (1) pre-processing to extract feature vectors and enhance separability among detected classes; (2) preliminary classification of Earth surface patterns using two separate and parallely acting classifiers: back-propagation neural network and binary decision tree classifiers; and (3) data fusion of results from preliminary classifiers to obtain the optimal performance in overall classification. Both the binary decision tree classifier and the fusion processing centers are implemented by neural network architectures. The fusion system configuration is a hierarchical neural network architecture, in which each functional neural net will handle different processing phases in a pipelined fashion. There is a total of around 13,500 samples for this analysis, of which 4% are used as the training set and 96% as the testing set. After training, this classification system is able to bring up the detection accuracy to 94% compared with 88% for back-propagation artificial neural networks and 80% for binary decision tree classifiers. The neural network data fusion classification is currently under progress to be integrated in an image processing system at NOAA and to be implemented in a prototype of a massively parallel and dynamically reconfigurable Modular Neural Ring (MNR).

1. INTRODUCTION

Artificial neural networks (ANN) have demonstrated capabilities for robust pattern classification in the presence of noise and object-to-background sensory uncertainty, and have found applications in environmental monitoring including land cover determination, vegetable mapping, soil survey, etc., or multichannel satellite imagery. This paper presents a data fusion system with artificial neural networks which will utilize multichannel SSMI satellite imagery, to combine supervised trainable and self-organized neural network architectures with specific knowledge-based classification techniques, with reference to fast and accurate classification of the earth surface. This neural approach is intended to compensate for different classification techniques by using the data fusion method and to reduce the lengthy training time required in a supervised learning network. The overall neural network data fusion system, which will be described in more detail, can also be seen as a four-layered supervised network which is composed of several modular and hierarchical networks. In this paper, we will start with a background discussion of the measurement used in this study. The data fusion classification system will be presented. Hardware implementation of each component in a Modular Parallel Ring (MPR) will also be discussed. Some experimental results will be presented and a summary will be given.

2. BACKGROUND

The SSMI instrument, flown on board the Defense Meteorological Satellite Program (DMSP) polar orbiting satellites, is a seven-channel conically-scanning microwave radiometer, measuring brightness temperatures at 19, 22, 37, and 85 GHz. All measurements are obtained with dual polarizations (H and V) except for 22 GHz channel. The 19 and 22 GHz channels are mainly responsive to variations in temperature and water vapor at large spatial scale. The 37 and 85 GHz channels, due to the scattering effects at high frequencies, respond to precipitation at smaller scale. Polarization measurements have been used to infer the wind speed, precipitation, and snow cover over the land and ocean. The spatial resolution (field of view) of the different channels decreases in proportion to the wavelength (inverse with frequency). It provides unique signatures for identifying surface features and obtaining the temperature and condition of the Earth's atmosphere. In comparing the measurements at different frequencies, effects due to different spatial resolutions are minimized by convolving all measurements to the 55-km resolution of the lowest-frequency channel (Grody, 1991). This enables one to investigate the spectral variations without having to consider the effects of spatial inhomogeneity on the different channel measurements. The measurements (brightness temperature, sometimes called antenna temperatures) used in this study were made between November 1988 and January 1989 and covers the entire northern hemisphere. The data was identified and confirmed by "ground truth" as five different data sets corresponding to five different surface classes: non-scattering medium (Non-Sm), precipitation over the ocean (R-Ocean), snow cover land (Snow), precipitation over the land (R-Land), and the desert (Desert). Each class has different samples ranging from 445 to 5535 and there is a total of over 13,034 samples. Table 1 illustrates some SSMI measurement classification characteristics including SSMI measurements, surface features and their corresponding samples. The brightness temperatures are normalized within the range of (-1, +1), denoted as X_i, and the desired output classes are represented by mutually orthogonal vectors, denoted as C_j.

Table 1 SSMI classification characteristics

SSMI	Channel frequencies and polarizations						
	19 H $T_H(19)$	19 V $T_V(19)$	22 V $T_V(22)$	37 H $T_H(37)$	37 V $T_V(37)$	85 H $T_H(85)$	85 V $T_V(85)$

Surface features:	Non-Sm	R-Ocean	Snow	R-Land	Desert
Number of samples:	4294	505	5535	2255	445

3. DATA FUSION CLASSIFICATION SYSTEM

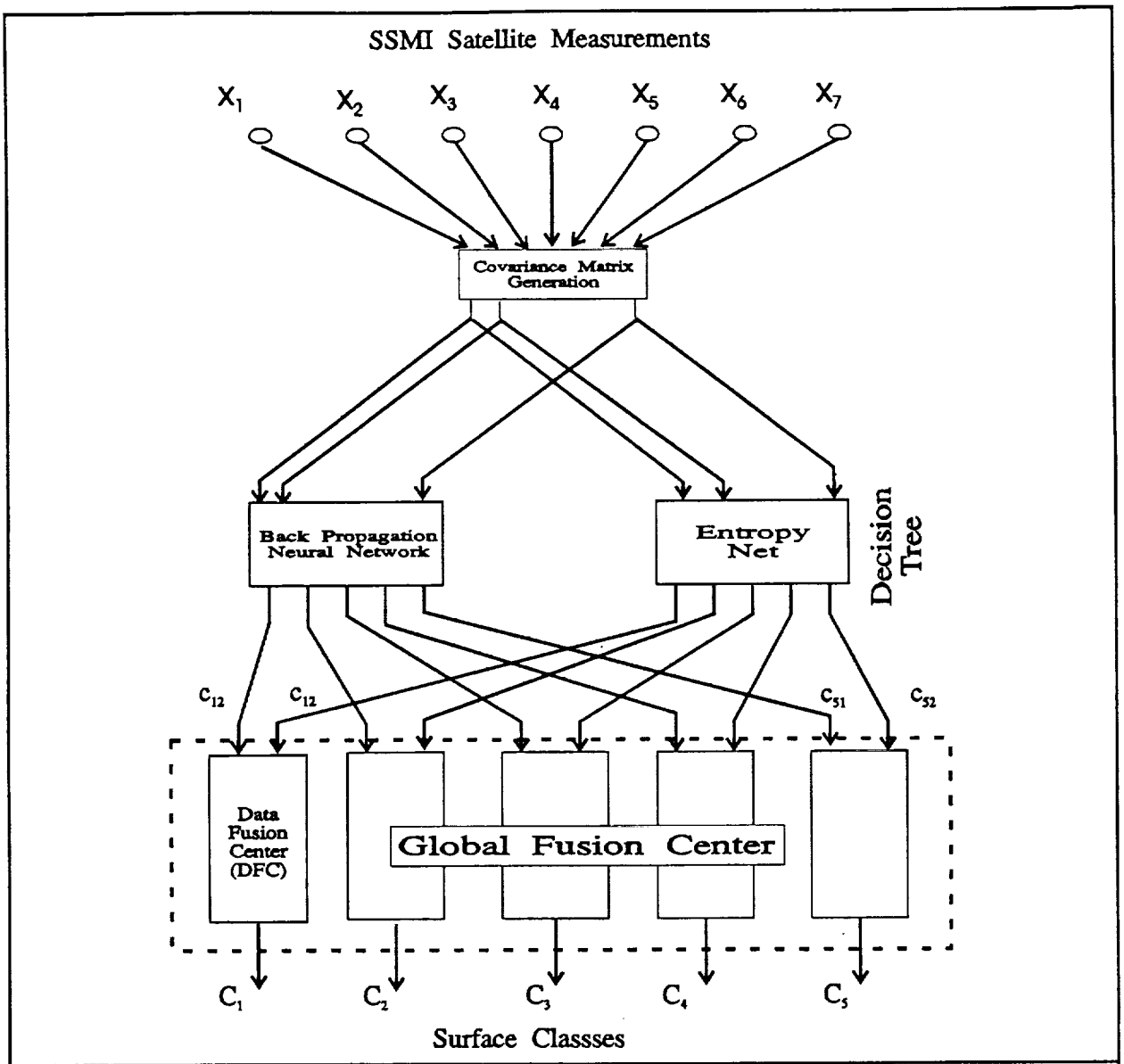


Figure 1. Data Fusion System with Artificial Neural Networks for SSMI Measurements

Although existing neural network paradigms have demonstrated excellent capabilities in learning and generalization, efficient training and determination of internal topology (such as number of hidden neurons) still remain challenging tasks. This data fusion classification system implemented with ANNs provides an alternative approach to attack these problems and can be easily implemented in hardware. Basically, this system treats each classifier as a different sensor and fuses each classification result to obtain the optimal or better results. The term "optimal" is used such that the probability of error is minimized in the likelihood ratio test. The sizes and connections of intermediate layers (or hidden layers) can be determined based upon the desired data flow.

This fusion classification system will process sensory data in three consecutive phases, as follows: (1) pre-processing, aimed at extracting feature vectors and at enhancing separability among detected classes; (2) preliminary classification of Earth surface patterns at two separate and parallely acting classifiers: back-propagation ANN (BP ANN) and a binary decision tree (BDT); and (3) fusion of classification results performed at global fusion center (GFC) from different classifiers and imagery to obtain the optimal decision. The configuration is a hierarchical neural network architecture, in which each functional neural net will handle different processing phases in a pipelined fashion.

3.1 Pre-processing

Pre-processing for SSMI imagery includes mainly the generation of (7 x 7) covariance matrices from measured brightness temperatures at each pixel. Information about pixel brightness temperatures, covariance matrix elements, and desired surface class definitions is collected in a feature vector for the supervised training of a neural network classifier. It has been demonstrated that increasing the elements of the feature vector by adding more relevant parameters, derived nonlinearly from original features, can reduce the number and size of hidden layers, and can also reduce the training time (Marks, et al., 1988). Since the covariance matrix evaluation involves the manipulation of two matrices, the operations involved are suitable to neural network implementation by feed-forward topologies, by merely assigning two manipulated matrices to the weights and input vectors of the back propagation neural architecture, as has been investigated.

3.2 Preliminary Classification

3.2.1 BP ANN Classifier

A three-layer (one hidden layer) supervised back propagation (BP ANN) algorithm is used to train the network to become a feed forward pattern recognition engine (Rumelhart and McClelland, 1991) to learn the input feature vectors corresponding to different output classes. There are 14 input neurons corresponding to SSMI measurements as well as to their covariance matrix, 60 hidden neurons, and 5 output neurons representing 5 surface conditions. It takes around 40 and 160 epoches to train the BP ANN classifier to learn up to 90% and 100% accuracy of the training data set, respectively. With a fully-trained BP ANN, the classification accuracy can reach up to 88% (Lure, et al., 1992a, 1992b). For the data fusion classification system, the BP ANN is only trained to a "satisfactory" accuracy (e.g., 75%). Such a "partially" trained ANN only takes around 50% of the training time required in fully-trained nets. A single fully-trained network can only reach a certain detection accuracy limit whereas a combination of several networks such as this one can reach even higher precision since the fusion processor will make an optimal decision based on the statistics of preliminary classification accuracy.

3.2.2 BDT Classifier

The BDT classifier is constructed to implement Grody's global classification algorithms as in Figure 2 (Grody, 1991). They are designed to analyze global coverage of satellite data sets and to classify based on the physical characteristics of measurements and on surface types. This technique performs a hierarchical tree-structured decision procedure through the evaluation of polynomial functions of input feature elements and through thresholding. The special topology of BDT classifiers used for surface condition classification based on SSMI measurements is drawn from the so-called Entropy Net architecture (Sethi, 1990). This architecture includes a two-layered topology, of which the lower layer performs arbitrary mapping of thresholding operations, while the upper layer performs logical operations (e.g. AND, OR) which allow us to convert the hierarchical decision procedure into a fully parallel process. The weight vectors between the layers are determined from

the coefficients of polynomial functions of the decision tree functions. The logical operations, such as AND, OR, NOR, and NAND, are implemented by using a simple BP ANN architecture with sigmoid transfer functions (Lippmann, 1987). A striking advantage of the neural implementation architecture is that it allows us to specify the number of neurons needed in each layer, along with the desired output. This, in turn, leads to an accelerated progressive training procedure that also allows each layer to be trained separately.

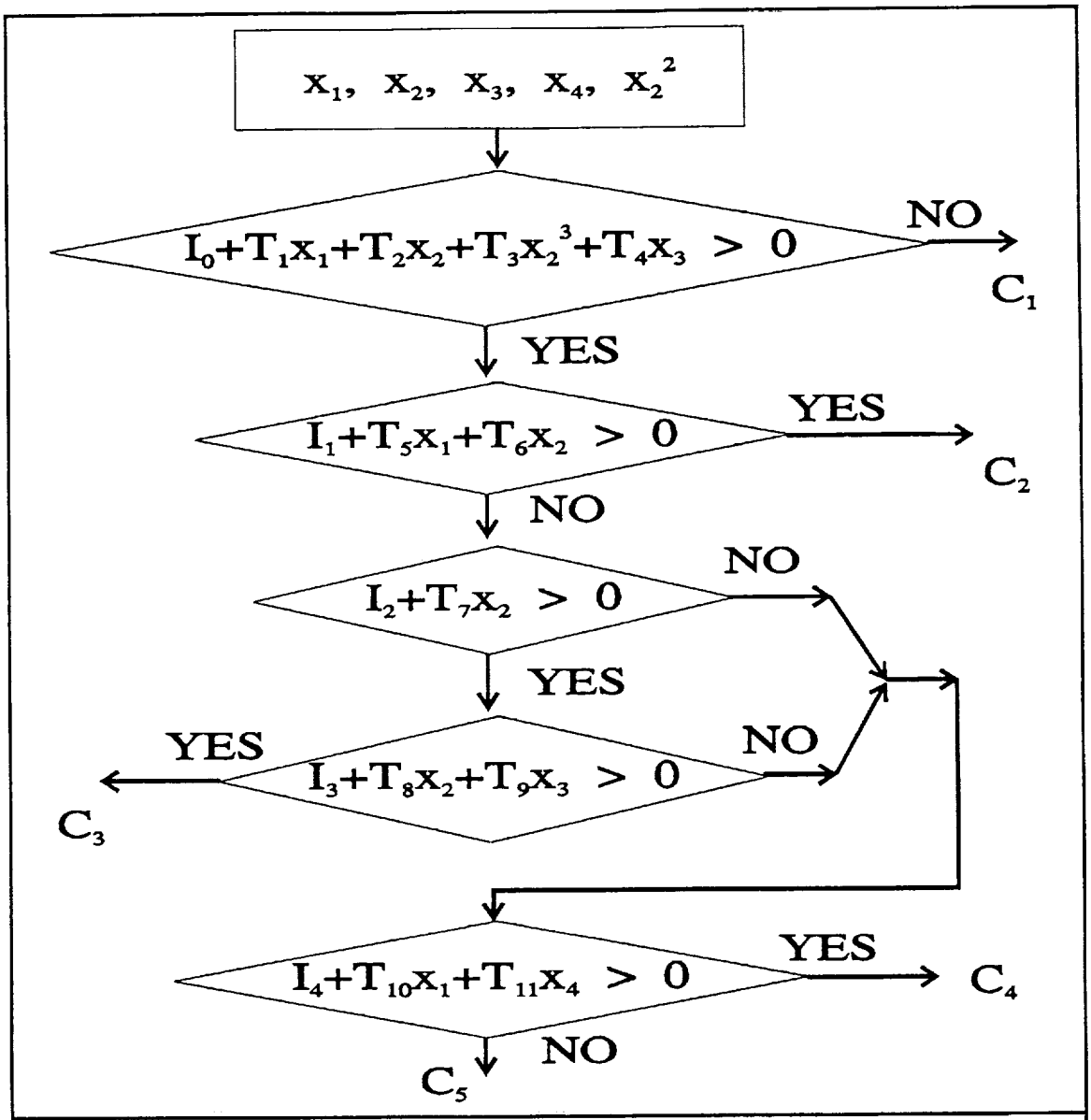


Figure 2(a)

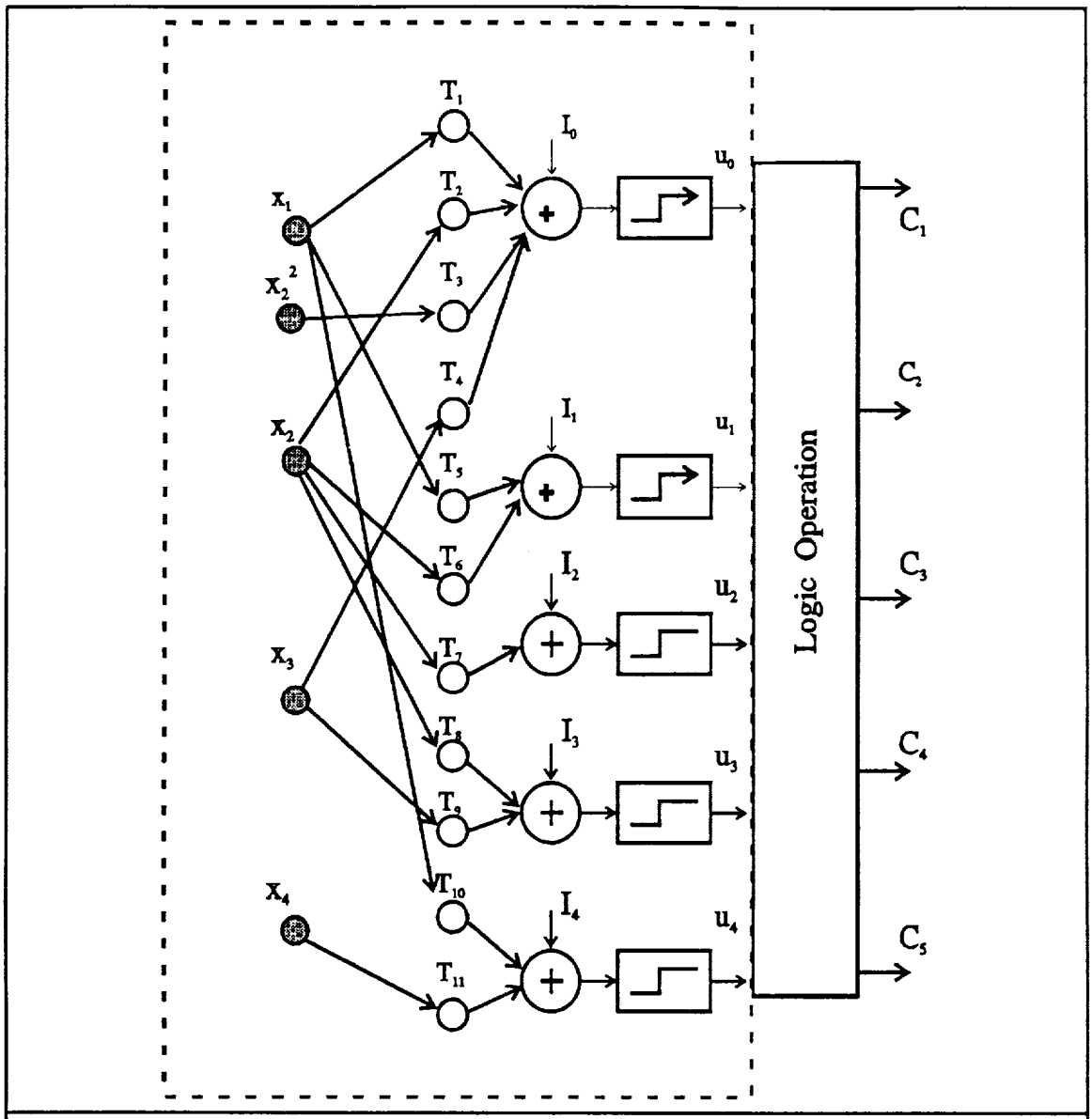


Figure 2(b)

Figure 2. (a) BDT Classifier and (b) its Neural Implementation. X_i's denote the SSMI measurements; T_i's denote the higher order polynomial coefficients in (a) and weights in (b); and I_i's denote constants in (a) and biases in (b), respectively.

There are 5 neurons corresponding to 4 selected SSMI measurements and to one element of the covariance matrix (X_1, X_2, X_3, X_4, and X_2^2), and 5 output neurons for each surface class. The individual decision from both BP ANN and BDT modules are sent to the global fusion center (GFC) for the final decision. The two-trainable-layered BP neural net for logical operation is trained based upon the data derived from known logic relationships from the decision tree. As for other neural networks for logic operations, it only takes a few epoches for them to learn the desired patterns.

3.3 Fusion Processing

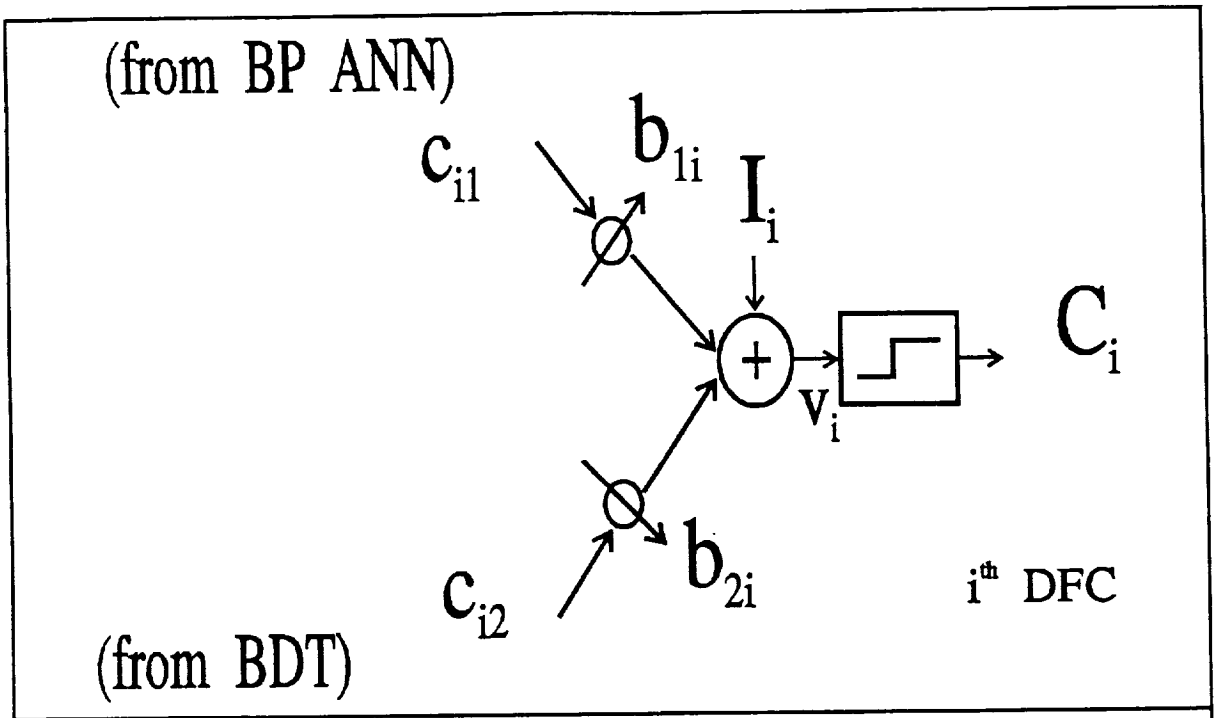


Figure 3. Data Fusion Center. c_{il}s denote classification results from preliminary classifications for class i; b and I denote the weights and bias of an entropy net; and C_i denotes the optimal decision for class i.

The fusion processing involves global fusion center (GFC) operations, which integrate results from both BP ANN and BDT classifiers. The GFC is composed of several different data fusion centers (DFC), each of which corresponds to different types of output classes as in Figure 3. A self-adjusted or self-trained learning algorithm is used in each DFC to set the optimal decision rules such that the total probability of detection is maximized. This data fusion scheme, also called distributed-detection scheme, corresponds to a two-layered network of nonlinear threshold elements, e.g., binary or sigmoidal functions (Tenney, 1981). The decision operation, weights and bias of these elements are obtained as

$$v_i = I_i + \sum_{j=1}^{n} b_{ji} u_{ij}$$

$$b_i = \log\left(\frac{(1-P_{M_i})(1-P_{F_i})}{P_{F_i}\,P_{M_i}}\right)$$

and

$$I = -2\log\left(\frac{P(H_0)}{P(H_1)}\right) + \log\left(\prod_{i=1}^{n}\frac{(1-P_{M_i})}{P_{F_i}}\right) + \log\left(\prod_{i=1}^{n}\frac{P_{M_i}}{(1-P_{F_i})}\right)$$

where n denotes the number of classifiers ($n = 2$), P_{M_i} represents missed detection in the *ith* classifier, P_{F_i} represents a false alarm in the *ith* classifier, $P(H_1)$ denotes the probability that the desired class is present, and $P(H_0)$ denotes the probability that the desired class is absent.

The probability functions Ps are obtained during training by comparing individual classification results with the desired class. The fusion networks are trained by self-adapting off-line stochastical information to form the detection system. The stochastic information including a priori probability, the probability of false alarm, and of missed detection is obtained by comparing classification results from individual classifiers with ground-truth data. The approximation rules are obtained from the nonlinear combination of the statistics of previous classification results from individual classifiers.

4. HARDWARE IMPLEMENTATION

The neural network data fusion system for real time processing is implemented in a prototype of a massively parallel and dynamically reconfigurable Modular Neural Ring (MNR) architecture (Ligomenides, et al., 1991), which is capable of maintaining a high performance for digital and neural applications. The MNR architecture is composed of multiple primitive processing rings (pRing) embedded in a global communication structure and is interfaced to a host workstation as in Figure 4. It is a multiple-SIMD (single instruction multiple data) architecture. Each of the pRings consists of 40 processing elements (PE) that are capable of mapping any number of neurons. It has been shown that the MNR provides very highly efficient hardware utilization and very low communication delay overhead. The achieved speed/capacity performance is increased linearly with the number of processing elements, without upper limit.

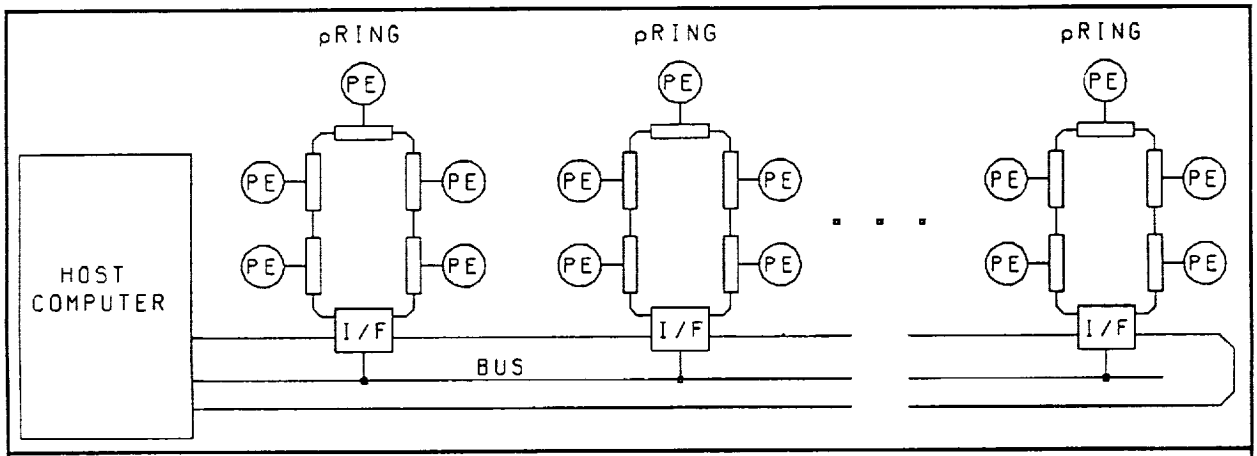


Figure 4. Bussed-pRing Architecture. PE denotes processing element which is used to implement operations in neuron(s).

Covariance matrix evaluation, involving the manipulation of two matrices, is performed by merely assigning two manipulated matrices to the weights and input vectors of the feed forward neural architecture. Two pRings are used to implement the BP ANN module: one for handling the 16x64 weight matrix of input-hidden connection and one for the 64x16 weight matrix of hidden-output connections. The third pRing is used for the parallel implementation of the BDT, which handles a 16x16 weight matrix. Since some weights are not utilized (for example, the input-hidden connection in BP ANN only requires a 14x61 weight matrix), they are filled with zero weights to satisfy hardware implementation requirements. The operation and performance of the hardware-based networks remain almost unchanged. Once the training is finished, the weights and bias are then stored in the memory of each PE for future processing. Both BP ANN and DBT operations are performed at the MNR architecture simultaneously. The individual decision from each operation is then fed to the data fusion center (DFC) for final optimum decision performed at the host computer.

5. CLASSIFICATION RESULTS

There are a total of 13,034 samples of data used in this study. Each of five different classes contains from 400 to 5,000 different samples. We used 500 samples of data as training sets which represent 3.8% of the total samples. Each training set, obtained randomly from the total data set, consists of an equal number of samples from five different classes. The rest of the samples (over 96%) are used for testing the network and the classification results are shown in Table 2. Once the BP ANN is trained either fully or partially, it is used to perform the classification. The classification accuracies, using the fully-trained BP ANN classifier (i.e., all training patterns are recognized by this BP ANN), are 82%, 98%, 97%, 78%, and 79% for non-scattering medium, precipitation over ocean, snow, desert, and precipitation over land, respectively (Lure, et al., 1992). The classification accuracies are 99%, 56%, 81%, 57%, and 70% for each surface class. Note that the class of non-scattering medium represents the surface which can not easily be specifically identified as any of the other four surfaces. The overall accuracy for BP ANN approach is around 88% whereas it is around 80% for BDT classifier. The preliminary results show that the neural network data fusion system improves the classification accuracy for all classes by around 4% from BP ANN's results. The overall accuracy of neural network data fusion is improved to 94%. Even without fully-trained being (e.g., 75% of training set are learned correctly by BP ANN) the overall classification accuracy can still achieve similar classification accuracies. From the coefficients of the data fusion center, it is also found that the BP ANN plays a more important role in classifying the non-scattering medium, snow, and desert; whereas the BDT is more dominant in classifying the other two surfaces. The significance of each SSMI measurement to classification of each of five surface types can also be obtained through the linearization procedure of the weights described in the previous study.

Table 2. Classification Results from BDT Classifier, BP ANN, and Data Fusion System

ALGORITHM	Non-sm	R-Ocean	Snow	Desert	R-Land	Overall
BDT	99%	56%	81%	57%	70%	80%
BP ANN	82%	98%	97%	78%	79%	88%
ANN FUSION	86%	98%	97%	84%	83%	94%

6. SUMMARY

In this research effort, a data fusion system with artificial neural networks is presented to classify surface types based on the SSMI measurements. Both back propagation ANN (BP ANN) and binary decision tree (BDT) classifiers are used for this study. Seven SSMI measurements (brightness temperature at 19, 22, 37, and 85 GHz for H and V polarizations, except V for 37 GHz) at each image pixel are extracted as an input feature vector. Five surface types including non-scattering medium, precipitation over the ocean, snow cover land, precipitation over the land, and the desert are used as target patterns. After training by using less than 4% of the samples, both BP ANN and BDT are able to perform the classification over 13,000 samples. The training for this data fusion system is performed progressively. The BP ANN, first module of entropy net, and logical operation net, are trained seperately. Once these are trained, each data fusion ceter network is trained seperately. The overall accuracy for the BP ANN and the BDT approaches 88% and 80%, respectively. The neural network data fusion system which fused the individual decision from the BP ANN and the BDT improved the overall accuracy to 94%. The significance of the contribution from either approach is determined based on the coefficients of the data fusion center. The fusion system is currently implemented in a massively parallel and dynamically reconfigurable hardware neural network (Modular Neural Ring) for real time parallel processing and integrated in an image processing system at NOAA/NESDIS. The data fusion classification system not only preserves

the advantages of both BP ANN and BDT classifiers (for example, the capability of physical interpretation of input feature space from the BDT classifier and robust classification from the BP ANN), but also reduce the pitfall of individual classifiers (for example, brute-force training of the BP ANN module and sensitivity to noise of the BDT module).

ACKNOWLEDGEMENTS

This work is partially supported by NOAA/NESDIS and Air Force Rome Development Center under SBIR Phase II contract: F30602-89-C-0130. The SSMI data set was provided by Hughes Aircraft Co. through NOAA/NESDIS. The authors are grateful for the discussion with P. A. Ligomenides and L. Jump of Computer Systems and Architecture Group at Caelum Research Corporation. The authors are also thankful for the comments by the reviewers. This paper is submitted to IEEE Transaction of Neural Network. Due to page limitation, a more detailed information can be found in the submitted paper.

REFERENCES:

Grody, N.C., 1991, "Classification of Snow Cover and Precipitation Using the Special Sensor Microwave Imager, J. Geophy. Res., Vol. 96, No. D5, 7423-7435.

Ligomenides, P. A., L. B. Jump, and Y-S. Chiou, 1991: "A Reconfigurable Ring Architecture for Large Scale Neural Networks", Proc. of Conf. Fuzzy and Neural Syst. and Vehi. Appli. 1991, Tokyo, Japan.

Lippmann, R. P., 1987, "An Introduction to Computing with Neural Nets". IEEE Acous. Spec. Signal Proc. Magazine, 4-22.

Marks II, R. J., L. E. Atlas, and S. Oh, 1988: "Generalization in Layered Classification Neural Networks", ISCAS'88, IEEE, 503-506.

Lure, Y.M.F., N.C. Grody, Y.S.P. Chiou, and H.Y.M. Yeh, 1992a: "Classification of Earth Surface from Special Sensor Microwave Imager (SSMI) Using Artificial Neural Network (ANN) Data Fusion," IEEE Proc. IGARSS '92, May 26-29, Houston, TX.

Lure, Y.M.F., Y.S.P. Chiou, H.Y.M. Yeh, and N.C. Grody, 1992b: "Hardware-based Neural Network Data Fusion for Classification of Earth Surface Conditions", Invited Paper, 26th Annual Asilomer Conf. on Signal, System, and Computers, Oct. 26-29, 1992, CA, IEEE Compu. Society.

Sethi, I.K., 1990: "Entropy Nets: From Decision Trees to Neural Networks." Proc. of the IEEE, Vol. 78, No. 10, October 1990.

Tenney, R. R., and N. R. Sandell, 1981: "Detection with distributed sensors", IEEE Trans. Aeros. Elec. Sys., pp 501-509.

Rumelhart, D. E., J. L. McClelland, 1986, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge, MA.

Neural Networks for Atmospheric Retrievals

Howard E. Motteler
NRC, Code 930
NASA/GSFC
Greenbelt, MD 20771

J. A. Gualtieri
USRA, Code 902.2
NASA/GSFC
Greenbelt, MD 20771

L. Larrabee Strow
Department of Physics
UMBC
Baltimore, MD 21228

Larry McMillin
NOAA/NESDIS
5200 Auth Road
Camp Springs, MD 20746

Abstract

We use neural networks to perform retrievals of temperature and water fractions from simulated clear air radiances for the Atmospheric Infared Sounder (AIRS). Neural networks allow us to make effective use of the large AIRS channel set, and give good performance with noisy input. We retrieve surface temperature, air temperature at 64 distinct pressure levels, and water fractions at 50 distinct pressure levels. Using 728 temperature and surface sensitive channels, the RMS error for temperature retrievals with 0.2K input noise is 1.2K. Using 586 water and temperature sensitive channels, the mean error with 0.2K input noise is 16%. Our implementation of backpropagation training for neural networks on the 16,000-processor MasPar MP-1 runs at a rate of 90 million weight updates per second, and allows us to train large networks in a reasonable amount of time. Once trained, the network can be used to perform retrievals quickly on a workstation of moderate power.

1 Introduction

The next generation of NASA earth viewing satellites on Earth Observing System (EOS) platforms will produce a deluge of raw data that must be processed into products that describe the state of the earth and its atmosphere over time. Satellite instruments that probe the atmosphere measure radiances over a number of channels, and this information must be "inverted" to obtain information about the atmospheric state, such as the temperature, humidity, and composition.

The Atmospheric Infrared Sounder (AIRS) [3], currently under development, should provide both higher accuracy and vertical resolution than the present operational sounders (HIRS/MSU) [10], and lead to higher forecasting skill and a long term accurate measure of climate change. The AIRS instrument will contain upwards of 4000 channels at a much higher spectral resolution than the currently operational HIRS instrument, which

has 20 channels. The optimum use of these data for atmospheric sounding in a cost effective way may require completely new techniques, as existing methods for current instruments may not be transferable in a straightforward manner. Traditional retrieval (or inversion) techniques are computationally intensive, especially non-linear techniques that require several iterative calculations of the channel radiances. It is estimated that the AIRS will require one of the most computationally intensive data systems on EOS.

To address these new computational challenges, we have implemented a backpropagation training algorithm on the Maspar MP-1 at Goddard Space Flight Center to train neural networks to perform atmospheric retrievals of temperature and water profiles from simulated clear air radiances for the AIRS instrument. [The problem of cloudy atmospheres is a topic of future work not treated here.] These neural networks allow us to make effective use of the large AIRS channel set, give good performance with noisy input data, and allow for very fast processing even with very large numbers of channels.

We have found that the backpropagation code maps very well to the Maspar, and we have obtained network training rates of 93 million connection updates per second (CUPS) in single precision [1]. Once such a network has been trained on the Maspar, it can be downloaded to a workstation where the time to obtain retrievals is the time to perform three matrix multiplies – of order less than 0.5 sec with a thousand input channels. (On the Maspar the retrieval time is at least an order of magnitude faster).

The accuracy of the results obtained with our neural networks are quite competitive with other retrieval methods. Using 728 temperature and surface sensitive channels, and with 0.2K std noise added to the input brightness temperatures, the neural network has an overall RMS error retrieving 64 pressure levels of 1.22K. Using 586 water, surface, and temperature sensitive channels, and with 0.2K std noise added, the neural network has an overall error retrieving 50 pressure levels of 16% [2].

In order to better understand retrieval performance, we perform a sensitivity analysis of trained networks. This analysis is useful in selecting what sets of channels are to be used, in a process of iterative refinement, and in many cases shows a close correspondence to plots of weighting functions (discussed in the next section).

In the sequel we describe the atmospheric retrieval problem, show how we use neural networks to solve the problem, describe the datasets used in training the networks, and present a number of representative results. We also describe the method of sensitivity analysis for evaluating the effectiveness of input sets to a neural network.

2 Atmospheric Retrievals

The problem of atmospheric retrievals [7], [5] (the "inverse problem") is to take as input the radiances at a specified set of frequency channels measured by a sensor on a satellite above the top of the atmosphere and compute the temperature or water profiles of the atmosphere (as a function of pressure) that gave rise to those radiances.

Associated with the inverse problem is the "forward problem" of computing the radiances

at the top of the atmosphere generated by layers of molecules in local thermal equilibrium from the surface up through the atmospheric column in the sensor's field of view. (We refer to this column as a temperature profile.) Assuming a plane parallel atmosphere in local thermodynamic equilibrium and negligible scattering, and no instrument function one can write the monochromatic radiance at nadir at the top of the atmosphere as

$$R_\nu = \epsilon_\nu B_\nu(T_s)\tau_\nu(P_s,[T(P)]) + \int_{\ln P_s}^{\ln \bar{P}} d\ln P \frac{d\tau_\nu(P,[T(P')])}{d\ln P} B_\nu[T(P)]$$

where ϵ_ν is the emissivity of the surface s, and the contribution of reflected radiation which is negligible at most frequencies of interest has not been included. $B_\nu(T)$ is the Planck function for emitted radiance of a blackbody at frequency ν and temperature T,

$$B_\nu(T) = 1.19\,x\,10^{-5}\frac{\nu^3}{\exp[1.439\nu/T]-1}.$$

The quantity $\tau_\nu(P_s,[T(P')])$ is the atmospheric transmittance from the surface at pressure P_s to the top of the atmosphere at pressure $\bar{P}$ which is the fraction of photons of frequency ν emitted at the surface P_s that arrive at the sensor at altitude $\bar{P}$. The quantity $\frac{d\tau_\nu(P,[T(P')])}{d\ln P}$ is the *weighting function* for the frequency ν and when multiplied by $d\ln P$ describes the fraction of photons of frequency ν emitted in the layer between pressure P and $P+dP$ that reach the top of the atmosphere. Fig. 1 [3] shows a few of the several thousand weighting functions available from the AIRS instrument and indicates how a weighting function can be associated with a narrow vertical region of the atmosphere. The notation $(P,[T(P')])$ as the argument of $\frac{d\tau_\nu}{d\ln P}$ is used to stress that it is *functional* of the profile $T(P')$ between $\bar{P}$ and P and a *function* of P.

Present retrieval systems are most easily classified as being either linear regression techniques or non–linear iterative techniques. Both techniques can use varying amounts of statistics for regularizing their solutions, as well as varying amounts of the forward problem radiative transfer. The linear regression approach is dependent on a very good first guess in order to be in the linear regime for the regression. The non-linear iterative method does not require such a good first guess, but does require time-consuming forward problem calculations. In addition, it is not clear if the non-linear iterative approach can coherently use all the information in the AIRS channel radiances without numerical problems. It may also be possible to iterate the linear regression approach, however this would result in the need to iteratively calculate the forward problem for a very large number of channels, introducing a very heavy computational burden.

3 Neural Networks

We use a three-layer feed-forward neural network, batch trained with a modified backpropagation algorithm [6], [8] with an adaptive learning rate. This network can be represented as

$$\mathbf{Y} = F_3(\mathbf{W}_3F_2(\mathbf{W}_2F_1(\mathbf{W}_1\mathbf{X}+\mathbf{B}_1)+\mathbf{B}_2)+\mathbf{B}_3),$$

where each F_i maps matrices to matrices, element by element, by applying a *transfer func-*

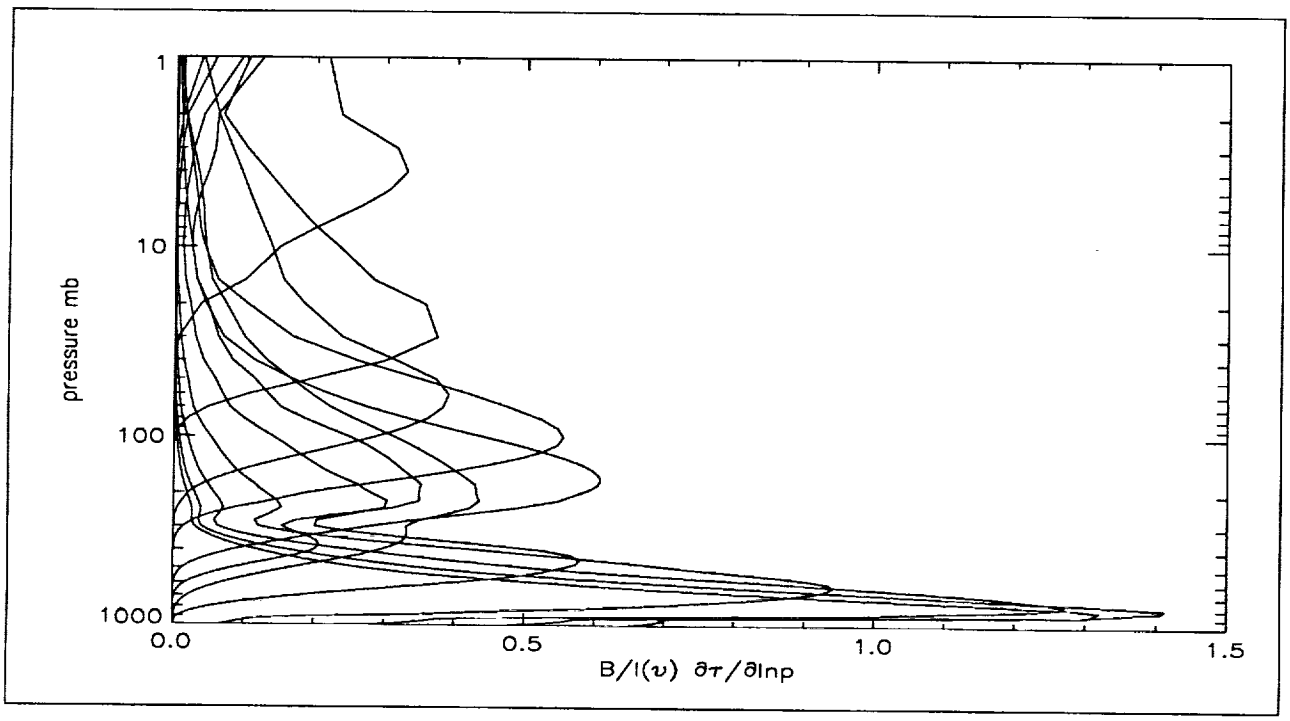


Figure 1: Representative weighting functions for the AIRS instrument. The x axis is a measure of the weighting function (where I(ν) is the radiance) and the y axis is pressure in mb.

tion to each matrix element and the matrices shown in boldface type are combined by matrix multiplication and addition. The mapping F_i is often referred to as a *layer*, with the weight matrices representing connections between layers. We use the hyperbolic tangent as a transfer function in the first two layers, and a linear function in the third. The input matrix $\mathbf{X}$ is of size (row $\times$ col) $n_{in} \times n_{training}$ and the output $\mathbf{Y}$ matrix is of size $n_{out} \times n_{training}$. The $\mathbf{W}_i$ are *weight matrices* of size respectively $n_1 \times n_{in}$, $n_2 \times n_1$, and $n_{out} \times n_2$. The $\mathbf{B}_i$ are *bias matrices* of respective sizes $n_1 \times n_{training}$, $n_2 \times n_{training}$, and $n_{out} \times n_{training}$ composed of single bias column vectors of respectively size n_1, n_2, and n_{out} replicated $n_{training}$ times to build the bias matrices. The quantities $n_{in}, n_1, n_2, n_{out}$, and $n_{training}$ are the number of input units (frequency channels), the number of first layer hidden units, the number of second layer hidden units, the number of output units (pressure levels), and the number of examples in the training set.

The networks we use for temperature retrievals have one input component for each instrument channel, and one output component for each AIRS pressure level. The first layer has between 90 and 108 transfer functions, the second between 60 and 72 transfer functions, and the output layer has a linear function for each pressure level. For water retrievals we have used 90 transfer functions in the first layer and 60 in the second layer.

Back-propagation training is a variation of gradient descent, in which weight and bias vectors are incrementally adjusted in an attempt to match the network output with a

set of training examples. This training set is a set of pairs, where each pair is an input together with the desired output. A single presentation of all the training data and corresponding weight and bias adjustment is called an *epoch*. Training consists of a sequence of epochs, and typically continues until the sum-squared error is acceptable or some resource limit is encountered. Training is a computationally intensive process for non-trivial networks. Although training is slow, applying a trained net is very fast, with the runtime being dominated by the time for the three matrix-vector multiplies.

It is convenient in the case of temperature retrievals to convert radiances R_ν to brightness temperatures Θ_ν according to the relation $B_\nu(\Theta_\nu) = R_\nu$ [9]. The brightness temperature is the temperature a blackbody would be at to produce the radiance R_ν. By doing this the large dynamic range of radiances is reduced to a much smaller dynamic range of brightness temperature. Further, each element of the input and output vector pairs are scaled to be differences from the mean values over the training set, and are divided by the standard deviation of the training set. This "normalizes" the inputs and outputs to a useful dynamic range for the transfer functions used.

We have developed a backpropagation code for the 128 × 128 processor MasPar MP-1 at the Goddard Space Flight Center in mpl (Maspar's parallel extension of C), which makes extensive use of the Maspar linear algebra library. This code efficiently handles the virtualization needed to map very large networks of many tens of thousands of weights and biases across the 16384 processing elements of the machine. Originally the code was written completely in double precision (64 bits) but since the results were found to be highly immune to noise in the data sets, a single precision version is now being used. Profiling tests show the code spends 95% of the time performing matrix multiplications, for which the Maspar routines are highly optimized. We are observing execution rates of 93 million weight updates a second [1] on typical datasets.

4 Datasets for Training

Datasets for training and testing are generated from the set of 1761 TIGR profiles [4] of temperature and water using the radiative transfer equation, to obtain corresponding radiances for the entire AIRS channel set. Thus the physics of the problem is built in by (1) the judicious selection of a large representative set of profiles and (2) the radiative transfer equation that gives the matching radiances. The TIGR profiles have been interpolated from the original 40 levels to either 66 TOVS pressure levels (for earlier experiments) or 64 TOVS pressure levels (as used in the AIRS science teams "write test"). The retrieved quantities are the temperatures and water amounts in the 64 intervening slabs with an additional element for the surface temperature, which may be different from the lowest slab. The surface emissivity is assumed to be one, for these experiments.

Our general method is to partition a dataset into training and extrapolation sets. The net is trained on the training set, and is then tested with the extrapolation set, both with and without noise; the noise inputs have a normal distribution and 0.2K standard deviation.

5 Results

In this section we present representative results for several profile and channel sets. In general, training runs were stopped when the RMS training error stopped showing significant improvement; this occurred after on the order of 100,000 epochs. Once network parameters (adaptive learning parameters, sizes of hidden layers, and initial distributions) are fixed in a useful range, different sets of random initial weights typically have a small effect on final RMS error. When the full set of TIGR profiles is divided into training and extrapolation sets of approximately equal size (with representatives from all latitudes in both sets) exchanging training and extrapolation subsets also has a small effect. The result for all the runs discussed are summarized in Table 1.

In run 150, the 880 even numbered TIGR profiles were used for training and the 881 odd numbered TIGR profiles were used for testing the network. Input to the net is brightness temperature for 666 AIRS channels, selected for surface and air temperature sensitivity. Output is surface temperature and air temperature at 66 distinct pressure levels. The network has 108 hyperbolic tangent transfer functions in the first hidden layer, and 72 hyperbolic tangent transfer functions in the second hidden layer. After 140,000 epochs, RMS training error is 1.20K, RMS extrapolation (testing) error is 1.26K, and RMS extrapolation error with 0.2K std noise is 1.44K. These results are shown in Fig. 2. After 100,000 epochs of further training with noisy data (0.2K std noise added to the input data), RMS training error is 1.22K, RMS extrapolation error is 1.23K, and RMS extrapolation error with 0.2K std noise is 1.37K

In the upper plot of Fig. 2, the temperature retrieval error at the surface and at each of 66 pressure levels is shown. In the lower plot, the same set of errors is presented as 11 groups of 6 pressure levels (the surface is still distinct, and is not grouped with any pressures levels). We do not have a completely satisfactory explanation for the small 'oscillations' in the 66 level plot. This pattern of fine variations appears across a wide range of training sessions and channel sets. (Note the similarity between these small scale variations in the Fig. 2 and Fig. 3 plots.) One possible explanation is that these variations correspond to variations in the numbers of weighting functions available at different pressure levels. Another possibility is that these may be an artifact of the fast transmittance code (as supplied by JPL for the AIRS science teams "write test") that we use to generate brightness temperatures. This is a matter for further investigation.

A sensitivity analysis of run 150 (discussed in the next section) is shown in Fig. 4. This analysis, together with similar results from other runs using the same channel set, indicated that channels with wavenumbers roughly between 750 and 1200 were not being used by the network. This information, together with the relatively high error above the 50mb pressure level suggested changes to the channel set, which were incorporated in run 170.

In run 170, the 880 even numbered TIGR profiles were used for training and the 881 odd numbered TIGR profiles were used for testing the network, as before. Input to the net is brightness temperature for 728 AIRS channels, selected for surface and air temperature sensitivity, taking into account previous sensitivity analysis. Output is surface temperature and air temperature at 64 distinct pres-

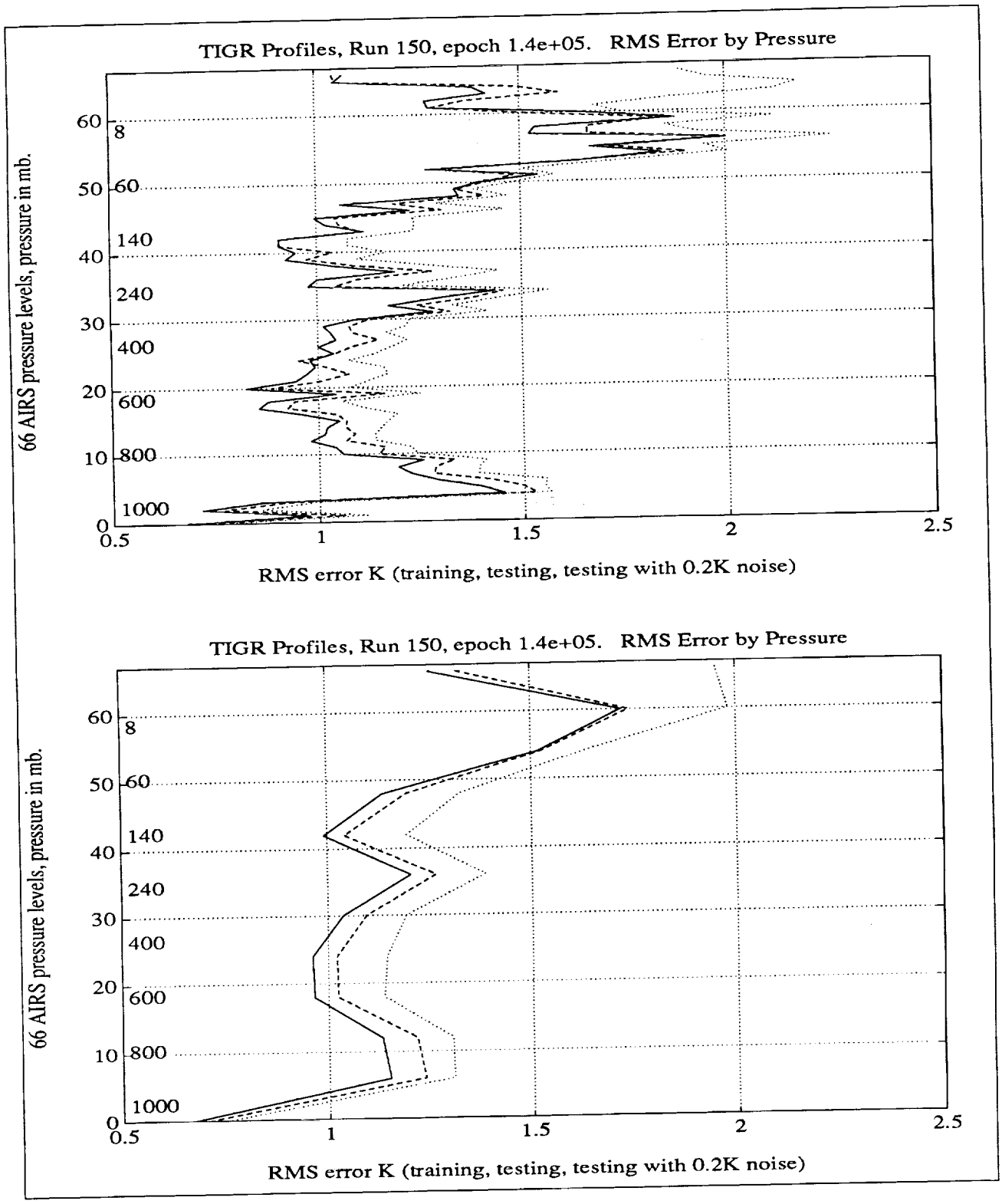


Figure 2: RMS temperature errors for run 150.

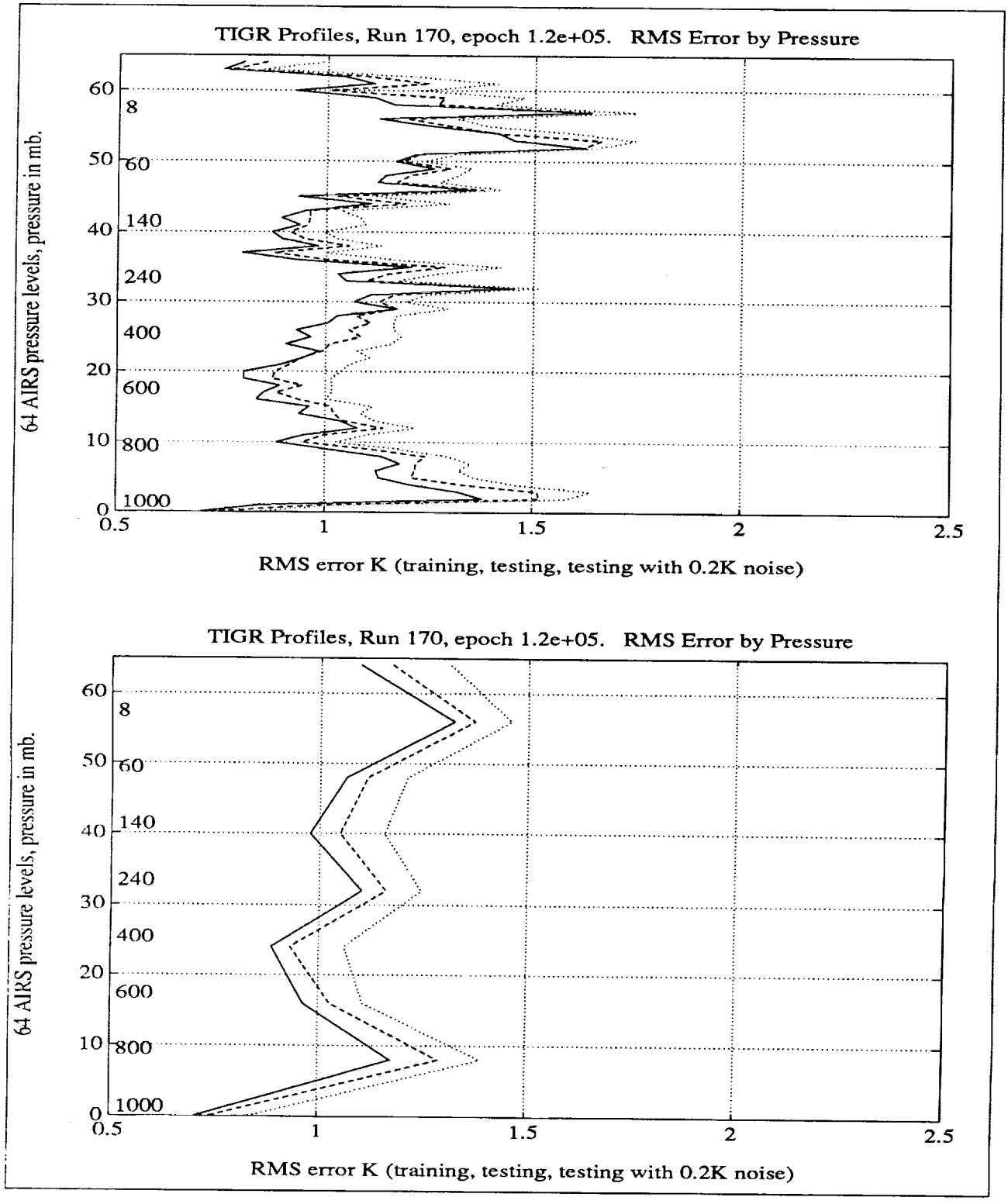


Figure 3: RMS temperature errors for run 170.

Run	Net Size	Epoch	RMS errors (a)		
			train	test	noise
150	666 × 108 × 72 × 67	240,000	1.22K	1.23K	1.37K
170	728 × 108 × 72 × 65	160,000	1.02K	1.09K	1.22K
90	586 × 90 × 60 × 50	50,000	13.2%	15.0%	15.9%

Table 1: Summary of runs discussed.

sure levels.[1] The network is the same size at the network for run 150. After 160,000 epochs, RMS training error is 1.02K, RMS extrapolation error is 1.09K, and RMS extrapolation error with 0.2K std noise is 1.22K. These results are shown in Fig. 3. A slight improvement in noise performance of this network could probably be realized by further training with noisy data.

A sensitivity analysis of run 170 is shown in Fig. 5. Note that the 'flat spot' (the large group of unused middle channels) is much reduced, but that there are still some unused channels.

Fig. 6 shows some initial results for water retrievals. Input to the net is brightness temperatures for 586 AIRS channels, selected for both water and temperature sensitivity. The same set of TIGR profiles were used as in runs 150 and 170, while the network was slightly smaller, with 90 transfer functions in the first hidden layer and 60 in the second. After 50,000 epochs, overall error for the first 50 pressure levels (expressed as percentages) is 13.2% training error, 15.0% extrapolation error, and 15.9% extrapolation error when 0.2K std noise is added.

As with more traditional methods of interpolation, neural networks can both under- and over-fit. High training error or inability to converge on the training set is a sign of underfitting, while poor performance on new data is a sign of over-fitting. The close correspondence between training and extrapolation errors on all the runs, and appropriate smoothness of retrieved profiles, suggest that the size of our hidden layers is not too large, and that we are not overfitting. It may be possible to use larger hidden layers to improve training and also (though to a lesser degree) extrapolative behavior.

6 Sensitivity Analysis

Once a network has been trained we can obtain a measure of its dependency on the input

[1] We switched from 66 to 64 pressure levels to match conventions used for the AIRS science team "write test."

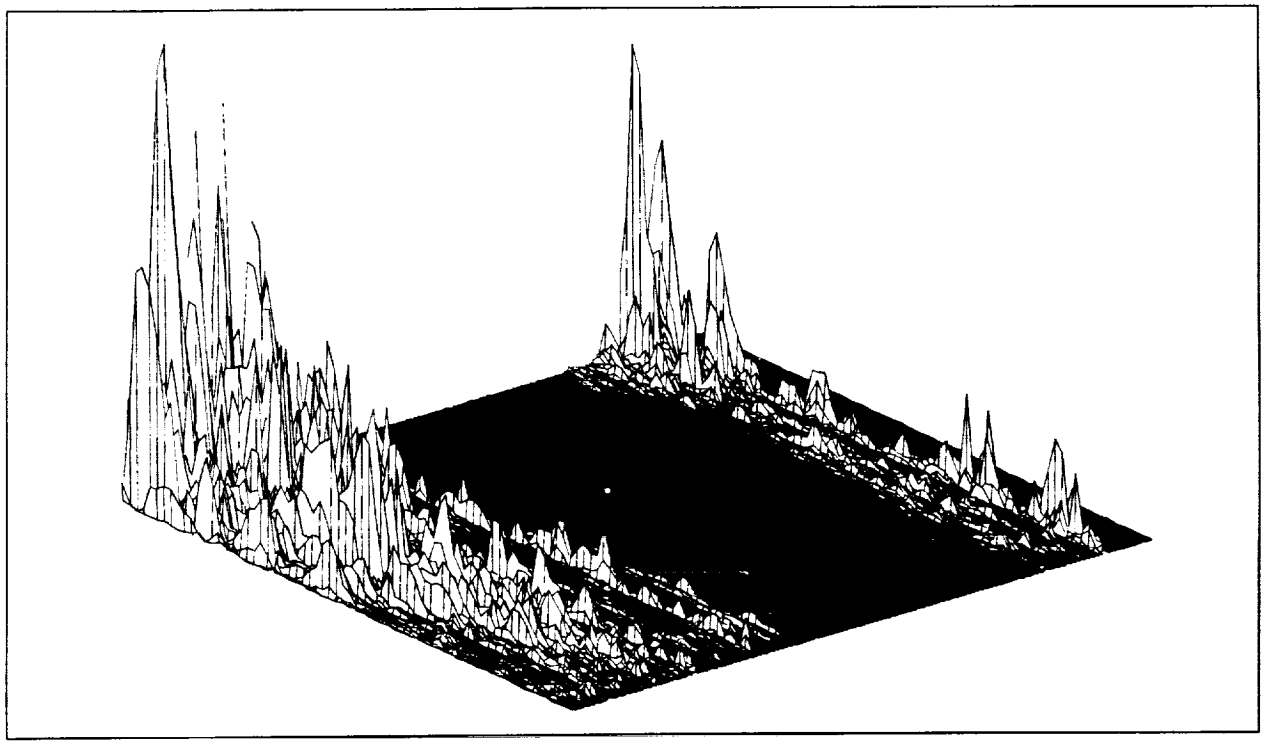


Figure 4: Sensitivity plot for run 150.

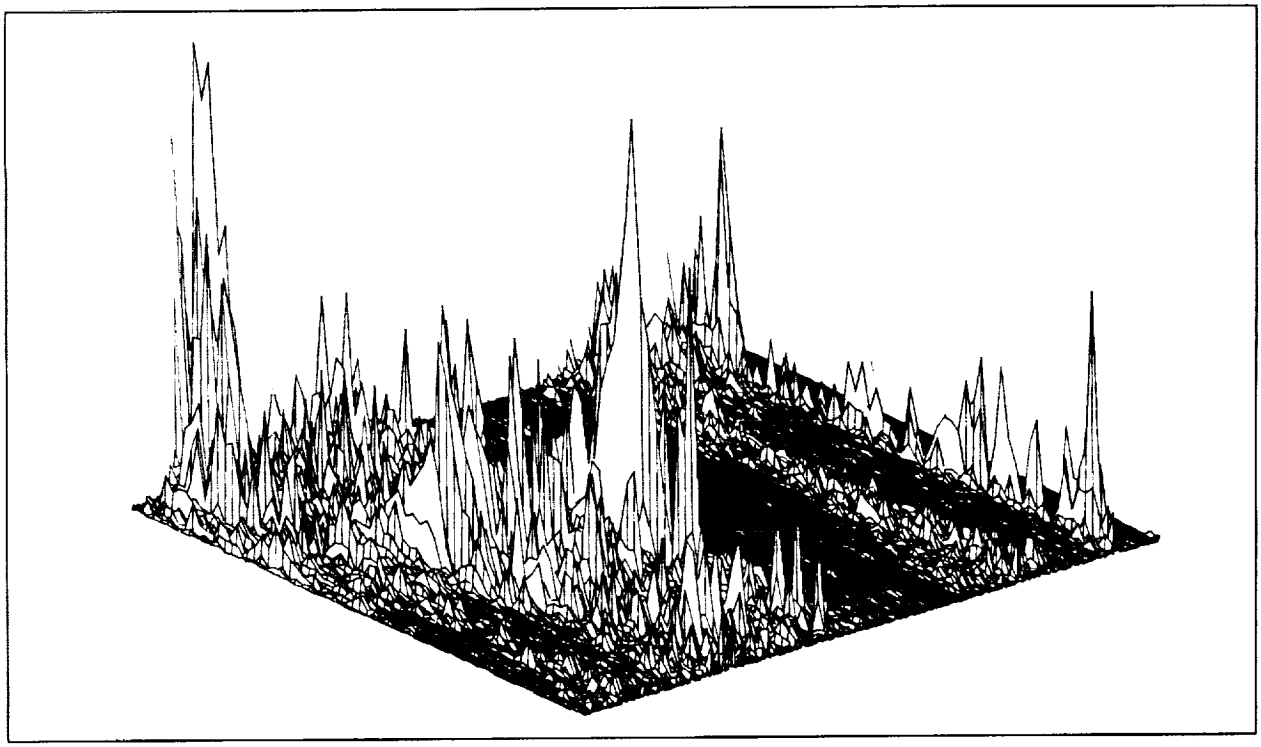


Figure 5: Sensitivity plot for run 170.

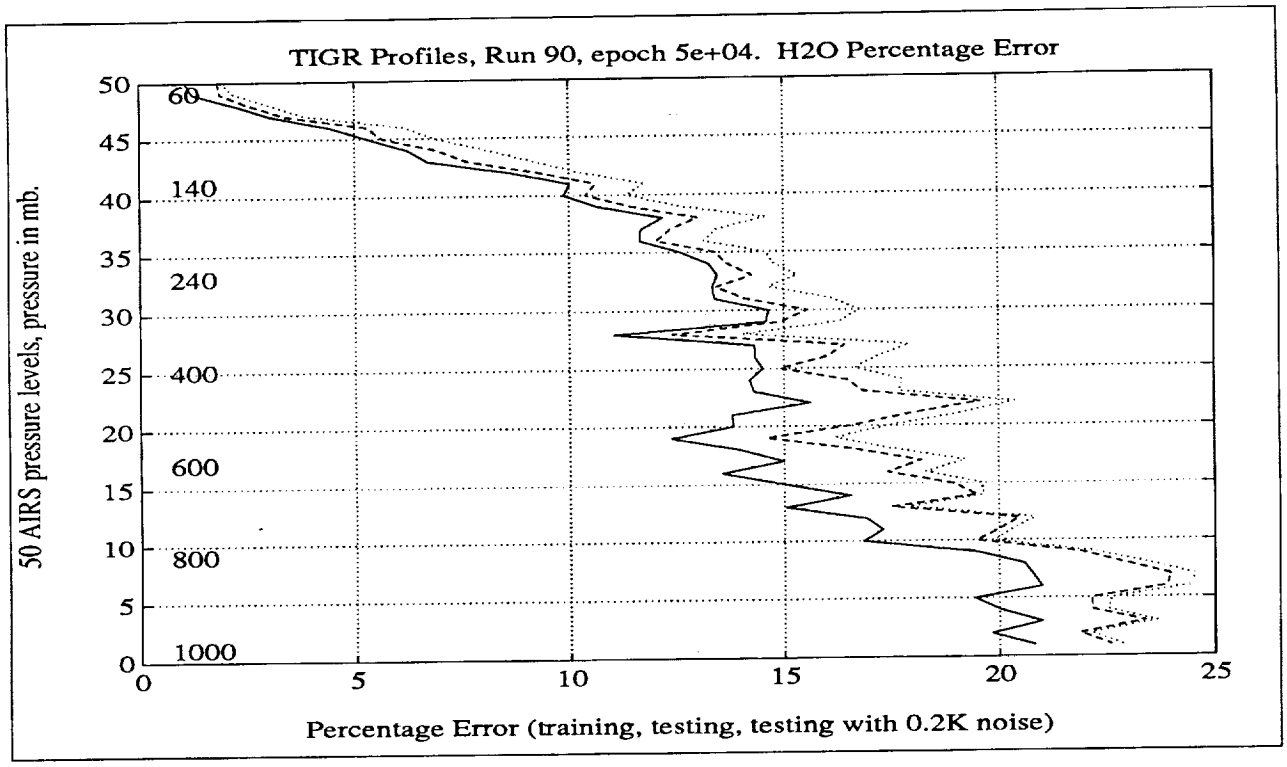


Figure 6: % errors for H_2O for run 90.

channel set by computing the Jacobian matrix of the partial derivatives of outputs with respect to inputs evaluated at a representative sample of profiles. In particular we have computed numerically by differences the quantity

$$S_{ij} = \sum_{\gamma} (\frac{\Delta T_i^{\gamma}}{\Delta \Theta_j^{\gamma}})^2 / N_{\gamma}$$

where γ indexes over the set of profiles in the dataset, N_{γ} is the number of profiles in the dataset, and Δ is the difference operator. If S_{ij} is large then on average over the set of all TIGR profiles frequency channel j has a large effect on temperature (water) in pressure layer i, while if it is small then the network has found little dependence of frequency channel j on the temperature (water) in pressure level i.

In the plots of sensitivity analysis Figs. 4 and 5, channels run from left to right, with the lower wavenumbers to the left. Pressure levels run from front to back, with the surface at the back of the plot. The z axis represents sensitivity (the sum square of partials), averaged across all the training profiles.

For many channels, sensitivity peaks correspond to weighting function peaks. The sensitivity plot looks much more 'noisy' and this is to be expected. (The sensitivity plot for an untrained net looks much like uniform noise.) In effect, the net has discovered its own representation for the weighting functions, where information from groups of channels is used to retrieve information about a particular pressure level. We conjecture that the 'noisy looking' sensitivity plot is inseparable from the network's good performance on noisy input.

7 Conclusions

We have demonstrated an application of backpropagation neural networks to the retrieval of accurate atmospheric temperature and water profiles, using the hundreds of channels of spectral information that will be available on the AIRS instrument. The prohibitive cost of training such large networks with large training sets is ameliorated by an effective mapping of the algorithm to the parallel architecture of the Maspar MP-1. The neural network allows us to make effective use of the large AIRS channel set, especially for better noise performance. Once the network is obtained it can be used to obtain very fast retrievals even with many input channels on modest computational platforms.

A sensitivity analysis of the network suggests ways we can refine the choice of channels used by the network. In principle, one could take the entire AIRS channel set, train a net for (say) temperature retrievals, perform a sensitivity analysis on the resultant net, get a smaller set of temperature sensitive channels, and use the smaller channel set to train a second net.

There are a number of directions for further work. Our present results indicate it is likely that a somewhat larger net may have errors below 1K. It may be that *simultaneously* retrieving temperature and water using a large combined channel set will give even better results than so far obtained. The retrieval of other atmospheric parameters, such as O_3, are promising areas for further investigation, as are the potential application of neural nets to cloudy atmospheres.

Acknowledgement: The authors would like to thank Alain Chedin and Milt Halem for helpful discussions of this problem.

References

[1] connections updates per second = (#weights + #biases) × #epochs × #training exemplars/ (cpu time).

[2] Water error = $\sum_i \sum_j Q_i^j / N_{levels} N_{profiles} \times 100\%$, where $Q_i^j = (1 - \frac{\text{q\_obs}_i^j}{\text{q\_calc}_i^j})$, where i is the level index and j is the profile index, and where q_obs and q_calc are observed and calculated water density in units of g/cm^2.

[3] Atmospheric Infrared Sounder: Science and measurement requirements. Technical Report D6665 Rev. 1, Jet Propulsion Laboratory, 1991.

[4] A. Chedin, N. A. Scott, C. Wahiche, and P. Moulinier. The improved initialization inversion method: A high resolution physical method for temperature retrievals from satellites of the tiros-n series. *Journal of Climate and Applied Meteorology*, 24:128–143, 1985.

[5] A. Deepak, H.E. Fleming, and J.S. Theon. *RSRM '87: Advances in Remote Sensing Retrieval Methods.* Deepak Publishing, 1988.

[6] Robert Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Publishing Company, New York, NY, 1990.

[7] C. Rodgers. Retrieval of atmospheric temperature and composition from remote measurements of thermal radiation. *Rev. Geophys. Space Phys.*, 14:609–624, 1976.

[8] Patrick K. Simpson. *Artificial Neural Systems.* Pergamon Press, Inc., Elmsford, New York, 1990.

[9] J. Susskind, J. Rosenfield, and D. Reuter. An accurate radiative transfer model for use in the direct physical inversion of HIRS2. *J. Geophys. Res.*, 88:8550–8586, 1983.

[10] J. Susskind, J. Rosenfield, D. Reuter, and M. T. Chahine. Remote sensing of weather and climate parameters from HIRS2/MSU on TIROS–N. *J. Geophys. Res.*, 89:4677–4697, 1984.

CLASSIFYING MULTISPECTRAL DATA BY NEURAL NETWORKS

Brian A. Telfer, Harold H. Szu
Naval Surface Warfare Center, Code R44
Silver Spring, MD 20903
btelfe@ulysses.nswc.navy.mil

Richard K. Kiang
Earth Science Data Operations Facility
NASA Goddard Space Flight Center
Greenbelt MD 20771
rkiang@favente.gsfc.nasa.gov

ABSTRACT

Several energy functions for synthesizing neural networks are tested on 2-D synthetic data and on Landsat-4 Thematic Mapper data. These new energy functions, designed specifically for minimizing misclassification error, in some cases yield significant improvements in classification accuracy over the standard least mean squares energy function. In addition to operating on networks with one output unit per class, a new energy function is tested for binary encoded outputs, which result in smaller network sizes. The Thematic Mapper data (four bands were used) is classified on a single pixel basis, to provide a starting benchmark against which further improvements will be measured. Improvements are underway to make use of both subpixel and superpixel (i.e. contextual or neighborhood) information in the processing. For single pixel classification, the best neural network result is 78.7%, compared with 71.7% for a classical nearest neighbor classifier. The 78.7% result also improves on several earlier neural network results on this data.

INTRODUCTION

In the past several years, a general awareness of the environmental crises has gradually taken place among the world's nations. We wish to address automated surveillance technology for environmental issues. Global warming, ozone depletion, large-scale deforestation, extinction of species are just a few of the issues that could lead to serious consequences to all inhabitants on the Earth, in a scale that will respect no national or political boundaries. To understand and quantify the anthropogenic impact on the environment, and to predict the eventualities if the deteriorating trend is not reverted, consistent and long-term monitoring of the global environment is needed. Through the Earth Probes and the Earth Observation System (EOS), NASA's Mission to the Planet Earth will continue to provide the essential measurements.

The amount of measurements from the Mission to the Plant Earth, however, will be unprecedented. For example, the first EOS AM platform alone will generate more than one terabyte (TB) data a day, compared with the 5 TB from the entire 12 years of AVHRR Pathfinder data. To timely process, analyze, store, and disseminate the satellite measurements and extracted information to a worldwide user community presents a formidable challenge, and demands innovative analytical methods and advanced computing and data communication technologies.

Among the contemporary information sciences, neural networks have proven to be a versatile technique for input-to-output mapping, without the constraint of formulating the exact relationship between the two. In addition, contextual and neighborhood knowledge can be easily included. In the past few years, neural networks have been applied to classifications of remotely sensed data (e.g., Campbell et al. 1989, Decatur 1989, Benediktsson et al. 1990, Liu et al. 1991, Bischof et al. 1992, Kiang 1992). In these studies, spectral data and ground truth are input to multilayer perceptron networks with one or more hidden layers, and networks are extensively trained off-line by minimizing a least-mean-squares (LMS) energy function with back-propagation (Werbos 1974, Rumelhart et al. 1986). It has been shown that the performance of neural network techniques is superior to classical techniques for systems operating in real-time.

It is well documented that minimizing the LMS energy function produces a neural network that approximates the Bayesian *a posteriori* probabilities (the

probability of a class given a particular input vector) of classes of data represented by a training set (see Richard & Lippmann 1991, for a review). Given an infinitely large training set and a network with sufficient functional complexity, the approximation error becomes negligible, and the misclassification error converges to the Bayes rate. While this property makes the LMS energy function attractive, there is an important qualification. The functional complexity needed for approximating the *a posteriori* probabilities is greater than that needed for approximating only class boundaries. Thus, if we are only interested in the classification of an input, rather than its *a posteriori* probability, a neural network that estimates probabilities will be needlessly complex. The additional complexity is a disadvantage both from the principle of parsimony (using the smallest number of weight parameters to increase generalization [see, e.g., Barron & Cover 1991]) and from the hardware implementation standpoint. Therefore, we test energy functions that minimize the misclassification error directly (Szu & Telfer 1991, Telfer & Szu 1992a), rather than indirectly via approximating the *a posteriori* probabilities. We call these Minimum Misclassification Error (MME) energy functions.

We first formulate these energy functions and provide a two-feature example that illustrates the concept. The Landsat Thematic Mapper data is described and results are presented for classifying on a pixel-by-pixel basis. These results are intended to provide a benchmark for further improvements that make use of both subpixel and superpixel (contextual) information. The paper concludes by discussing these research directions.

ENERGY FUNCTION FORMULATION

The commonly used σ-LMS energy function is given by

$$E_{\sigma-LMS} = \sum_{n=1}^{N}\sum_{k=1}^{K}[d_{nk} - \sigma(o_{nk})]^2, \qquad (1)$$

where d_{nk} is the desired output (normally set to 0 or 1) of the k-th output unit for the n-th training vector, σ is a sigmoidal function [we use $\sigma(z) = 1/(1 + exp(-z))$], and o_{nk} is the output of the k-th output unit for the n-th training vector, before the sigmoidal nonlinearity is applied. With one output unit per class, and $d_{ck} = 1$ for training vectors from class c, $d_{ck} = 0$ otherwise, minimizing $E_{\sigma-LMS}$ produces outputs that approximate the Bayesian *a posteriori* probabilities. An input vector is then classified according to the largest output value. However, for practical applications (finite training sets and networks with limited functional complexity), $E_{\sigma-LMS}$ function is not guaranteed to minimize misclassification error (Barnard & Casasent 1989).

A more natural energy function for classification simply counts the number of training vectors that the network misclassifies. The formulation of this counting function varies depending on the output encoding. For a two-class problem, a single output unit suffices, with positive outputs indicating one class and negative outputs indicating the other. A counting function for this network is given by (Szu & Telfer 1991, Telfer & Szu 1992a)

$$E_{MME} = N - \sum_{n=1}^{N} step(d_n o_n), \qquad (2)$$

where d_n is the desired sign of the actual output o_n and $step(z) = 1$ if $z \geq 0$; $step(z) = 0$ otherwise. (Eq. 2 thus uses a sharp membership function; a fuzzy logic version would be an obvious extension.) When the desired sign is the same as that of the actual output o_n, the n-th training vector $\mathbf{x}_n$ is correctly classified, the step function equals 1, and the number of misclassifications E_{MME} is reduced by one. When the desired output sign and actual output sign differ, $\mathbf{x}_n$ is misclassified, the step function equals 0, and E_{MME} is not reduced. To minimize an energy function with gradient descent, the energy function must be differentiable. Although the step function in Eq. 2 is not differentiable, it can be approximated by a sigmoidal function that is gradually steepening. As the magnitudes of the network weights increase, the magnitudes of the network outputs o_n also increase, and the sigmoid behaves more and more like a step function required by Eq. 2.

For multiple classes, if there is one output unit per class and an input is classified based on the largest output, an appropriate counting function, called the Classification Figure of Merit (CFM) (Hampshire & Waibel 1990), is given by

$$E_{CFM} = N - \sum_{n=1}^{N} \sigma(o_{max} - o_{other}), \qquad (3)$$

where o_{max} is the output from the unit that should have the maximum value (corresponding to the training vector's class) and o_{other} is the largest value of the other output units. Here the step function has been replaced by a sigmoid with the above justification. For a correct classification, $o_{max} - o_{other} > 0$ and

$\sigma(o_{max} - o_{other}) \rightarrow 1$, and the number of misclassifications is reduced by 1. For a misclassification, $o_{max} - o_{other} < 0$ and $\sigma(o_{max} - o_{other}) \rightarrow 0$, and the number of misclassifications is not reduced. A proof showing that minimizing E_{CFM} does give the desired result is given in (Hampshire & Pearlmutter 1991).

With multiple classes, the outputs may also be binary encoded labels, in which case the outputs are passed through a threshold rather than a maximum detector. An advantage of binary encoded outputs over one output unit per class is that fewer output units are required. For example, for 16 classes, one output unit per class requires 16 output units, but binary encoded outputs require only 4 output units. In addition, error correcting codes can be used as class labels. For example, a Hamming code (Lin & Costello 1983) with 7 output units can encode 16 classes and correct a single error in the output units. Such an error correcting approach increases classification accuracy and has been shown to improve associative memory performance (Liebowitz & Casasent 1986, Casasent & Telfer 1992). A new MME energy function to minimize misclassification error for binary encoded outputs is given by

$$E_{MME} = N - \sum_{n=1}^{N} \sigma[\sum_{k=1}^{K} \sigma(d_{nk} o_{nk}) - K + 0.5]. \quad (4)$$

The summation over k equals the number of correct output units for the n-th training vector. If all are correct, the summation equals K, and the outer sigmoid becomes 1, which reduces the number of incorrect misclassifications by 1. If there are one or output errors, the summation over k equals at most $K - 1$ (for a single output error) and the outer sigmoid becomes 0, and the misclassification count is not reduced. Note that in this case of multiple classes, E_{MME} must determine from all the output units whether a classification is correct or not. It is not sufficient to simply sum the errors from each output unit individually by summing Eq. 2 over multiple classes.

2-D EXAMPLE

Before considering the Thematic Mapper data, we consider a simpler two-class example of synthetic data with two features. This allows the class boundaries to be easily visualized to provide insight into LMS and MME energy functions. Since the data set is much smaller than the Thematic Mapper data, it also allows more detailed study.

Two classes with equal *a priori* probabilities are drawn from concentric circular uniform distributions with radius $\sqrt{2}/2$ (class 1) and 1 (class 2). The Bayes rate (minimum error) is 0.25, with a circular boundary of radius $\sqrt{2}/2$. The training set consists of 1000 vectors from each class and is shown in Figure 1. (The class boundaries shown in Figure 1 will be discussed shortly.) The test set consists of 5000 vectors from each class. The larger test set is needed to increase the confidence levels of the results.

The following study considered L_1 and L_2 norm versions of the two-class E_{MME}. More details are provided elsewhere (Telfer & Szu 1992b). The method described in the formulation section is the L_1 version. Multilayer perceptrons with two layers of weights and varying numbers of hidden units were tested for σ-LMS, MME L1 and MME L2. The procedure was to randomly initialize the weights to values between ± 1, first train each network for 200 iterations (epochs) using σ-LMS, and then using that result as a starting point, train for 800 iterations using the three energy functions. The motivation for the initial 200 iterations was to move the networks into a reasonable area of weight space which could then be tuned further by each energy function. This was found to produce better results than simply starting with each energy function from random weights. Other random weight magnitudes were also tried to ensure that the best results possible from each energy function were being measured. A conjugate gradient method (Fletcher 1987) was used (restart cycle of 5) with a simple inexact line search in implementing the backpropagation algorithm. For each number of hidden units, ten initial sets of random weights were constructed. In an attempt to discount runs that became stuck in local minima, only the run that gave the minimum training set error for each energy function was included in the results.

Figure 2a plots the performance of each energy function vs. number of hidden units. The MME energy functions produce excellent results with only three hidden units, and as more hidden units are added, they descend to essentially identical training set errors of 0.246 for MME L2 and 0.248 for MME L1 with 8 hidden units. Since it was plain that the MME energy functions were reliably finding minimum error networks, their hidden units were not increased beyond 8. For σ-LMS, the training set error also slowly decreased with increasing numbers of hidden units, but consistently remained higher than the MME training set results and the Bayes rate. With 16 hidden units, σ-LMS still gave 0.259 error, over 1%

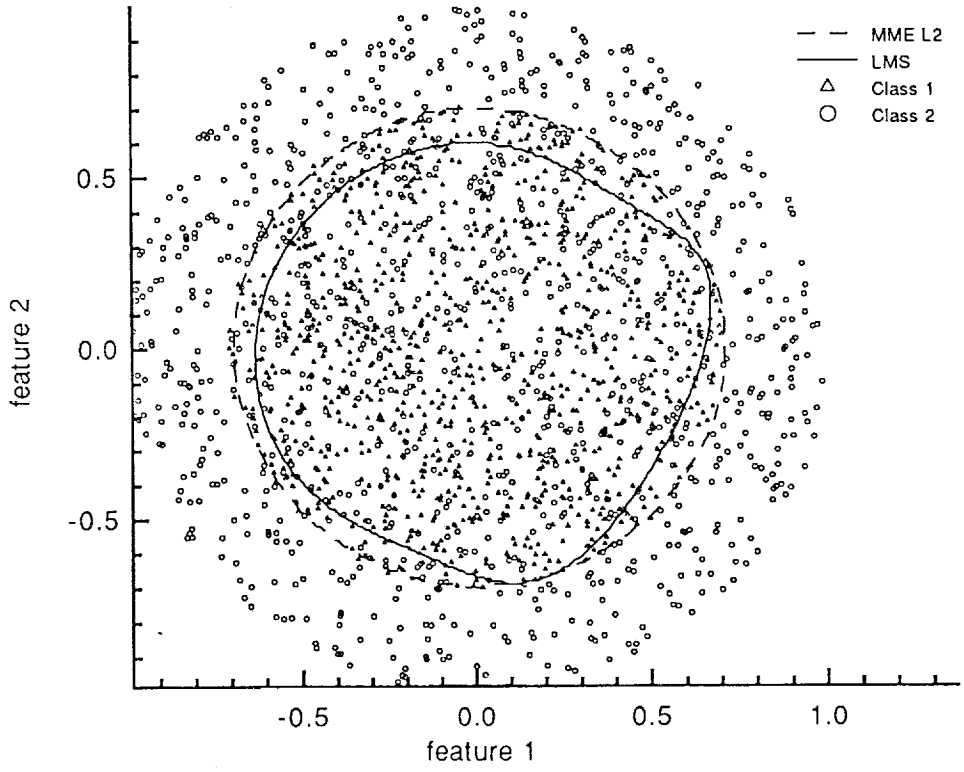


Figure 1: Training set for 2-D case with class boundaries found by σ-LMS and MME L2 networks with four hidden units.

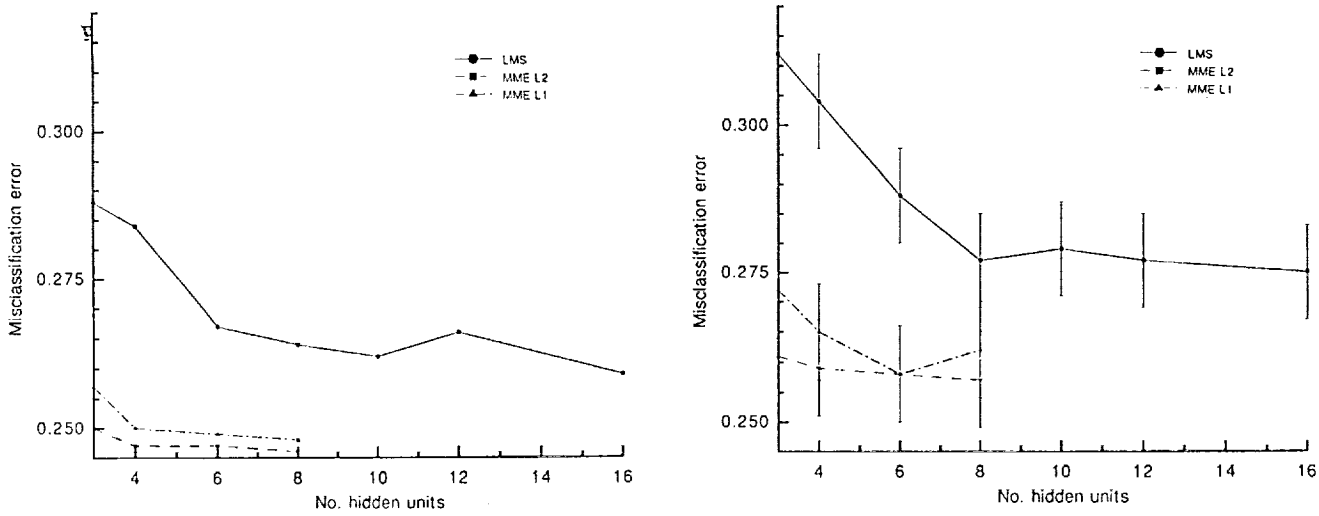


Figure 2: (a) Training set and (b) test set results for different energy functions vs. number of hidden units.

higher than for the MME energy functions. For an infinitely large training set, σ-LMS would converge to the Bayes rate, but this result does not hold for a finite training set.

These results are also reflected in the MME L2 and σ-LMS class boundaries with four hidden units, plotted in Figure 1. The MME L2 boundary is clearly almost exactly the desired circle, while the σ-LMS boundary is consistently inside the optimal boundary.

Of course, the more important question is how the networks performed on the test set. These results are plotted in Figure 2b with 95% confidence intervals (Highleyman 1962). The test set errors are all higher than the respective training set results as expected. The MME test set results are still lower than the σ-LMS results, and the results are statistically significant (although there is a slight overlap between the MME L2 and σ-LMS results at 8 hidden units, even this is still significant with a high but less than 95% confidence level). Even with 16 hidden units, the σ-LMS result is still significantly (in the statistical sense) worse than all but 4 of all 8 MME results with 8 or fewer hidden units. Thus, for this example, σ-LMS requires roughly five times the number of hidden units of the MME energy functions (16 vs. 3) to give equal test set performance.

LANDSAT EXAMPLE

Description of Data

Landsat-4 Thematic Mapper (TM) data taken in July 1982 over an area in the vicinity of Washington, D.C. were used in this study. The TM is a 7-band instrument, with spectral coverages 0.45-0.52 (TM1), 0.52-0.60 (TM2), 0.63-0.69 (TM3), 0.76-0.90 (TM4), 1.55-1.75 (TM5), 10.40-12.50 (TM6), and 2.08-2.35 (TM7). The ground Instantaneous Field-of-View (IFOV) is 30m except for the thermal bands (TM6), which is 120m. As the infrared and the thermal bands had not yet cooled off after launch, only the first four bands are usable.

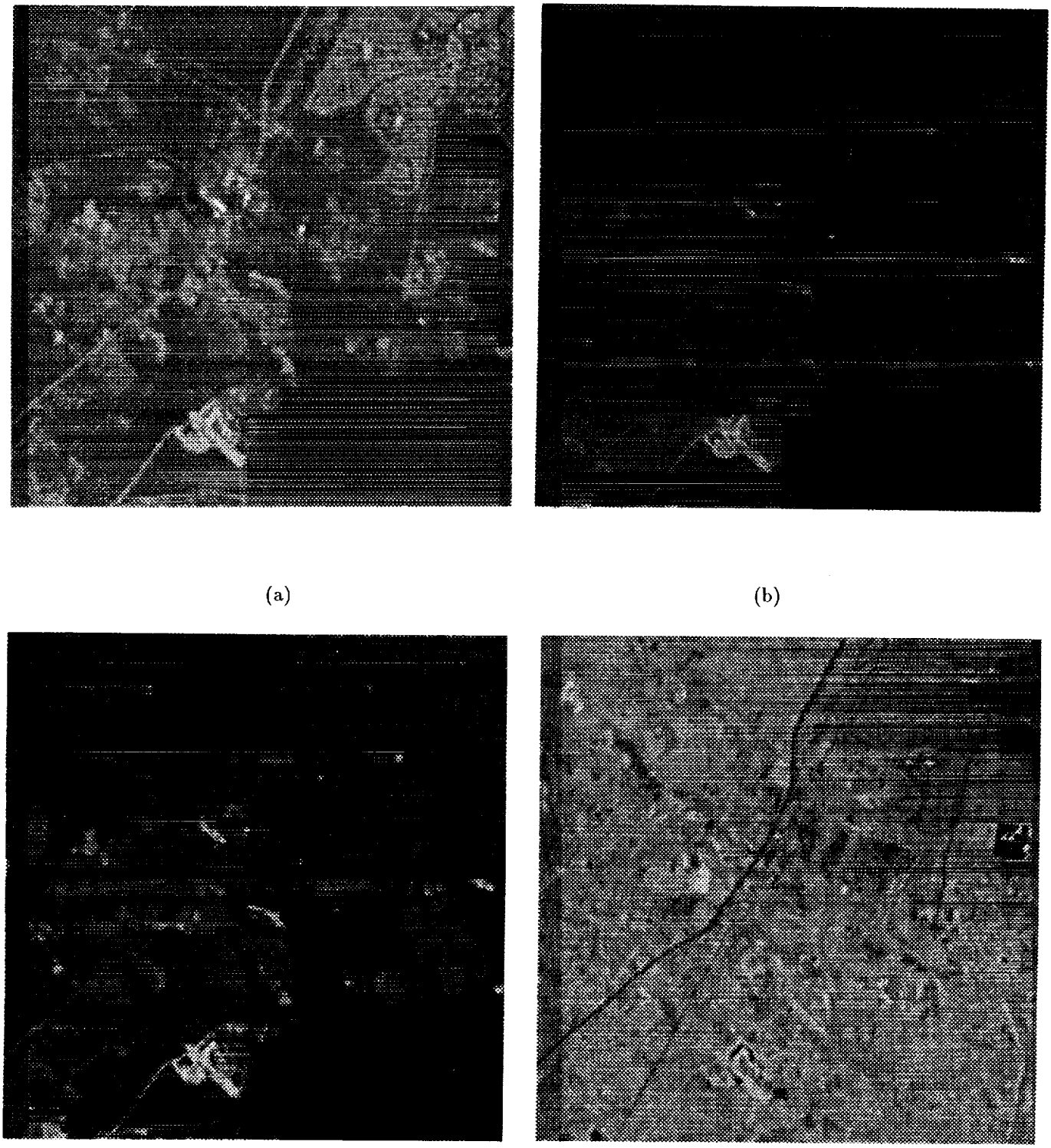
The ground truth consists of 17 categories, and were obtained through photointerpretation of color infrared aerial photographs and subsequent field visits (Williams et al. 1984). Specifically, the categories are (1) water, (2) miscellaneous crops, (3) standing corn, (4) corn stubble, (5) shrubland, (6) grassland or pasture, (7) soybeans, (8) bare soil/cleared land, (9) mostly hardwood dense canopy, (10) mostly hardwood less dense, (11) mostly conifer, (12) mixed wood, (13) asphalt, (14) single-family residential area, (15) multi-family residential area, (16) industrial or commercial area, and (17) bare soil/plowed fields.

In general, ground truth contains information categories instead of spectral categories. As the IFOV is broad enough to cover multiple ground categories, there are natural overlaps among the spectral signatures for these categories. Since the neural networks in this study perform classifications based on spectral data alone, whether the information categories correspond to distinct spectral categories should be examined, in order to estimate the intrinsic discriminability among the categories.

To achieve this objective, the spectral signatures for all categories are computed. The signatures consist of mean vectors and covariance matrices. A number of measures, such as divergence and Mahalanobis distance, could be used to estimate the separability among multi-dimensional clusters. In this study, we compute the ratio of between-class variance to within-class variance along the Fisher optimal discriminant vector (Duda & Hart, 1973). From the ratios, it is concluded that some information categories are heavily overlapped with others, and that the 17 information categories should be combined into 6 categories, following the land use and land cover classification system of Anderson et al. (1976). These six categories are: (1) urban or built-up land, (2) agricultural land, (3) rangeland, (4) forest land, (5) water, and (7) bare soil/cleared land. Notice that there is no Category 6 (wetland) in this data. In Anderson's system, Category 7 is barren land, such as salt flats, beaches, bare rock, etc. Since bare soil/cleared land (Category 17 in the ground truth data) does not exactly fit the definition, the original description in the ground truth is used instead.

To give an idea of the terrain types present, Figure 3 shows the four bands of the 256×256 image (slightly cropped for display purposes). Roads are clearly visible. A housing development is at the upper right. Fields are visible in the center of the image. The dark areas are primarily forest.

The area for which ground truth exists (a roughly 150×150 area in the center of Figure 3) has 21,952 pixels, with pixels placed alternately into training and test sets, giving 10,976 pixels for each. The number of pixels in each class is give in Table 1. Since each pixel contains four spectral bands, each feature vector contains four features, with an additional element set to one to provide a bias term. Each of the four



(a) (b)

(c) (d)

Figure 3: Four bands of Thematic Mapper data: a) TM1, b) TM2, c) TM3, d) TM4.

Class Name	No. Pixels
Urban	2754
Agric	1670
Range	3184
Forest	13781
Water	28
Bare	535

Table 1: Class distribution of Landsat data.

spectral features was normalized to have zero mean and standard deviation of 0.75.

Procedure and Results

Multilayer perceptrons with two layers of weights and twelve hidden units were tested for $E_{\sigma-LMS}$, E_{CFM} and E_{MME}. (Networks with fewer hidden units were also tried but found to perform slightly worse.) For the network structure of one output unit per class, six output units were used, while for binary encoding, three output units were used. (Two of the possible eight codes were unused.) The procedure was to randomly initialize the weights to values between ± 1, first train each network for 500 iterations (epochs) using σ-LMS, and then using that result as a starting point, train for 1000 iterations using the three energy functions. A conjugate gradient method was used (restart cycle of 5) with a simple inexact line search.

The resulting classification accuracies are given in Table 2. We first consider the results for one output per class. Although CFM improved the σ-LMS training set accuracy by 1%, the test set results are identical. The small training set improvement indicates that σ-LMS is finding class boundaries very close to the minimum error boundaries. The excellent σ-LMS performance can be explained by the large training set size and apparently relatively small functional complexity needed to represent the *a posteriori* probabilities in this case.

For the binary coded outputs, the σ-LMS outputs esimate the probabilities that the outputs are 1 given the input. This can be seen to perform worse than MME, which improves accuracy by 2.2% for the training set and 1.2% for the test set. The difference in the test set result is significant with an 88% confidence level. (The 95% confidence level is $\pm 0.75\%$.) There is no statistically significant difference between the two test set results for one output per class and MME

Energy Function	Accuracy (%)		Output Encoding
	Train	Test	
σ-LMS	78.1	78.7	1/class
CFM	79.1	78.7	1/class
σ-LMS	76.4	76.9	binary code
MME	78.6	78.1	binary code

Table 2: Classification accuracies for Landsat data.

binary encoding, but these three results do differ significantly from the σ-LMS binary encoding result. Although the saving in weights by binary encoding is not large in this example, for larger numbers of classes, the savings becomes significant. In addition, the binary encoding performance would be improved by using error correcting codes.

For comparison, a classical nearest neighbor classifier (Duda & Hart, 1973) gave 71.7% test set accuracy. Also, seven previous neural network tests (with various network architectures and sizes) on this data set have given test set accuracies between 71.6% and 78.4% (Kiang 1992, Hwang et al. 1993). Our best result of 78.7% is statistically better than all but one of these previous results (78.4%), and was obtained with a much smaller network – 132 weights for our network vs. about 640 weights required for the radial basis function network giving 78.4%. The fact that this previous result is similar to our best results suggests that this could be the best possible accuracy that can be obtained by classifying single pixels. Further accuracy improvements can be obtained by making use of subpixel information and by classifying based on a neighborhood of pixels. We discuss this in the next section.

There have been neural network based studies (e.g. Bischof et al. 1992) in which classification accuracies are higher than ours. However, it must be pointed out that a direct, fair comparison among these studies may not be possible. As known in remote sensing applications, classification accuracies are highly dependent on the ground types involved, the sensors' resolutions, the seasons when the measurements were taken and the environmental conditions. In general, discrimination among various kinds of vegetation covers is rather difficult.

DISCUSSION OF FUTURE WORK

The 78.7% classification accuracy for single pixel classification should be regarded as a starting point to benchmark further improvements that involve both subpixel and superpixel information. In addition to this work, a method for improving the training set is also discussed.

Since the Thematic Mapper pixel footprint is 30m, the spectra from different landuse types can be mixed in a single pixel. In related work (Shimabukuro & Smith 1991), mixture components are estimated using conventional least squares techniques in order to estimate ages of eucalyptus areas. Neural network apporaches remain to be tested. Since a neural network trained by LMS estimates *a posteriori* probabilities, these can be used as mixing proportions to provide subpixel classification results. For example, it was observed in the LMS classification results described above that many of the pixels along a road passing through forest had large outputs corresponding to both urban (manmade) and forest. Rather than classifying the pixel as urban (road) *or* forest based on only the single largest output, it seems more appropriate to classify the pixel as a certain fraction urban/road *and* a certain fraction forest based on the two largest mixing components. Simply classifying based on the largest output was observed to create many discrepancies with ground truth. For example, the groundtruth marks only discontinuous stretches of the road as urban and the rest as forest. The LMS neural network classifies (based on largest output) the entire stretch of road as urban, but also has a high second largest output for forest. Thus, making use of subpixel mixtures should improve results. Mixture information provides general information about a pixel, but does not indicate the physical region within the pixel occupied by a particular ground type. Super-resolution theory appears promising for physically locating ground types within pixels based on the classifications of nearby pixels.

Conversely, since land use occurs in patches larger than the 30m pixel size, it seems clear that information from neighboring pixels should also increase classification accuracy. Several such ideas for making use of context have been tested with conventional classifiers (Mohn et al. 1987 [tests several prior approaches], Lee & Philpot 1991, Jeon & Landgrebe 1992) and a neural network approach (Bischof et al. 1992). The neural network approach combines spectra from the pixel to be classified and from neighboring pixels into a single feature vector. The neural network then learns from the training set how much weight should be placed on information from neighboring pixels in classifying the central pixel. Bischof et. al. demonstrated a 5% improvement with this method vs. single pixel classification. We are currently testing this contextual technique with our new MME energy functions. A two-pass hybrid spectral/spatial approach is also planned to overcome projection registration and distortion problems.

Lastly, editing the training set should also help improve results. As noted elsewhere (Williams et al. 1984), any minor errors registering groundtruth with the Thematic Mapper data could result in mislabeled samples. Therefore, training samples near class boundaries in the image should be deleted.

ACKNOWLEDGEMENT

The support of this work by by the NSWCDDWO Independent Research Program and an Office of Naval Research Young Navy Scientist Award is gratefully acknowledged. The authors thank Gerald Dobeck for fruitful discussions involving the MME energy function for binary encoded outputs.

REFERENCES

Barnard, E., D. Casasent (1989), "A Comparison Between Criterion Functions for Linear Classifiers. *IEEE Trans. Systems Man and Cybernetics*, Vol. 19, pp. 1030–1041.

Barron, A., T. Cover (1991), "Minimum Complexity Density Estimation," *IEEE Trans. Information Theory*, Vol. 37, pp. 1034–1054.

Benediktsson, J.A., P.H. Swain, O.K. Ersoy (1990), "Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data," *IEEE Trans. Geoscience and Remote Sensing*, Vol. 28, p. 540.

Bischof, H., W. Schneider, A.J. Pinz (1992), "Multispectral Classification of Landsat Images using Neural Networks," *IEEE Trans. Geoscience Remote Sensing*, Vol. 30, pp. 482–490.

Campbell, W.J., S.E. Hill, R.F. Cromp (1989), "Automatic Labeling and Characterization of Objects Using Artificial Neural Networks," *Telematics and Information*, Vol. 6, p. 259.

Casasent, D., B. Telfer (1992), "High Capacity Pattern Recognition Associative Processors," *Neural Networks*, Vol. 5, pp. 687–698.

Decatur, S.E. (1989), "Application of Neural Networks to Terrain Classification," *Proc. International Joint Conf. Neural Networks*, p. I-284.

Duda, R., P. Hart, (1973), *Pattern Classification and Scene Analysis*, New York: John Wiley and Sons.

Fletcher, R. (1987), *Practical Methods of Optimization*, New York: John Wiley and Sons.

Hampshire, J., A. Waibel (1990), A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Trans. Neural Networks*, Vol. 1, pp. 216–228.

Hampshire, J., B. Pearlmutter (1991), Equivalence Proofs for Multi-layer Perceptron Classifiers and the Bayesian Discriminant Function. In Touretzky, Elman, Sejnowski, and Hinton, eds., *Proceedings of the 1990 Connectionist Models Summer School*, San Mateo, CA: Morgan-Kaufmann, pp. 159-172.

Highleyman, W. (1962), "The Design and Analysis of Pattern Recognition Experiments, *Bell System Technical Journal*, Vol. 41, pp. 723-744.

Hwang, J.N., S.R. Lay, R.K. Kiang (1993), "Robust Construction Neural Networks for Classification of Remotely Sensed Data," submitted to *World Conference on Neural Networks*, Portland OR.

Jeon, B., D.A. Landgrebe, "Classification with Spatio-Temporal Interpixel Class Dependency Contexts," *IEEE Trans. Geoscience Remote Sensing*, Vol. 30, 663–672.

Kiang, R.K. (1992), "Classification of Remotely Sensed Data using OCR-Inspired Neural Network Techniques," *Proc. 1992 International Geoscience and Remote Sensing Symposium*, Houston TX, pp. 1081–1083.

Lee, J.-H., W.D. Philpot (1991), "Spectral Texture Pattern Matching: A Classifier for Digital Imagery," Vol. 29, pp. 545–554.

Liebowitz, S., D. Casasent (1986), "Error Correction Coding in an Associative Processor," *Applied Optics*, Vol. 26, pp. 999-1006.

S. Lin & D.J. Costello (1983), *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall.

Liu, Z.K., J.Y. Xiao (1991), "Classification of Remotely Sensed Image Data Using Artificial Neural Networks," *Int. J. Remote Sensing*, Vol. 12, pp. 2433-2438.

Mohn, E., N.J. Hjort, G.O. Storvik, "A Simulation Study of Some Contextual Classification Methods for Remotely Sensed Data," *IEEE Trans. Geoscience Remote Sensing*, Vol. 25, pp. 796–804.

Richard, M., R. Lippmann (1991), "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities," *Neural Computation*, Vol. 3, pp. 461–483.

Rumelhart, D.E., G.E. Hinton, R.J. Williams (1986), "Learning Representations by Back-Propagating Errors," *Nature*, Vo. 323, pp. 533–536.

Shimabukuro, Y.E., J.A. Smith (1991), "The Least-Squares Mixing Models to Generate Fraction Images Derived from Remote Sensing Multispectral Data," *IEEE Trans. Geoscience Remote Sensing*, Vol. 29, pp. 16–20.

Szu, H., B. Telfer (1991), "Minimum Misclassification Error Performance Measure for Layered Networks of Artificial Fuzzy Neurons," *Proc. International Joint Conference on Neural Networks*, Vol. II, p. A-916.

Telfer, B. H. Szu (1992a), "Implementing the Minimum-Misclassification-Error Energy Function for Target Recognition," *Proc. International Joint Conference on Neural Networks-Baltimore*, vol. IV, pp. 214–219.

Telfer, B., H. Szu (1992b), "Energy Functions for Minimizing Misclassification Error with Minimum-Complexity Networks," submitted to *Neural Networks*.

Werbos, P. (1974), "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. thesis in applied mathematics, Harvard University.

Williams, D.L. et. al. (1984), "A Statistical Evaluation of the Advantages of Landsat Thematic Mapper Data in Comparison to MSS Data," *IEEE Trans. Geoscience and Remote Sensing*, Vol. 22, pp. 294–302.

Knowledge Engineering

THE WORKPLACE DISTRIBUTED PROCESSING ENVIRONMENT

Troy Ames
NASA/Goddard Space Flight Center
Data Systems Technology Division
Greenbelt, Maryland 20771
(301) 286-5673
ames@kong.gsfc.nasa.gov

Scott Henderson
CTA Incorporated, Space Systems Division
6116 Executive Boulevard, Suite 800
Rockville, Maryland 20852
(301) 816-1218
scott@cta.com

Abstract

Real time control problems require robust, high performance solutions. Distributed computing can offer high performance through parallelism and robustness through redundancy. Unfortunately, implementing distributed systems with these characteristics places a significant burden on the applications programmers. Goddard Code 522 has developed *WorkPlace* to alleviate this burden. WorkPlace is a small, portable, embeddable network interface which automates message routing, failure detection, and re-configuration in response to failures in distributed systems. This paper describes the design and use of WorkPlace, and its application in the construction of a distributed blackboard system.

1. The Dilemma

WorkPlace was developed as part of the Intelligent Ground System (IGS) project within Goddard Space Flight Center's Data Systems Technology Division with funding from NASA Code R. The IGS project is exploring the use of multiple knowledge-based systems in the satellite control center, particularly in the area of platform monitoring and fault diagnosis. The current practice is to introduce isolated expert systems into operations. Our objective is to achieve a more comprehensive system that involves many expert systems that communicate and cooperate with each other and with conventional components of the control center.

We faced two technological hurdles in achieving this objective. First, we needed to provide a flexible, open mechanism for data exchange which would support multiple platforms and heterogeneous applications. Second, we needed to develop an architecture for expert systems which would accommodate the asynchronous nature of a distributed cooperative environment. This paper describes the solutions to these problems that we have developed.

2. The Blackboard Solution

Blackboard systems represent the standard metaphor for distributed problem solving in AI. That metaphor describes a team of experts who cooperate to solve some problem. These experts communicate by writing partial solutions on a blackboard. The posting of a partial solution by one expert triggers the activity of an expert with a related expertise. Together these experts progressively evolve the partial solutions into a solution to the top level problem. Thus the blackboard architecture is based on:

- a universally accessible space for posting partial solutions (the blackboard),
- partitioning of that space into multiple levels of abstraction,
- and the opportunistic application of Knowledge Sources (experts) to that space to further the current level of understanding.

Traditional implementations of the blackboard approach use shared memory within a uni-processor for the information space. When the posting of a new partial solution triggers multiple knowledge sources, conflict resolution strategies serialize the execution of those sources and their access to the blackboard. This limits performance.

Several systems have been described in the literature which provide different approaches to parallelizing blackboard systems. One family of approaches is based on the use of a multi-processor architecture. Hearsay II (Fennell and Lesser, 1976) provides a central blackboard which is written to by concurrent experts executing on a simulated multi-processor. The experts (or knowledge sources) are further de-coupled into precondition and action parts which can execute concurrently within their own copy of relevant portions of the blackboard (known as *contexts*). The central blackboard provides node

C-3

and region locks to mediate reading and writing by the concurrent knowledge sources. The CAGE system (Nii, Aiello, and Rice, 1989) also provides a central blackboard, and can be executed with inter-knowledge source, intra-knowledge source, and intra-rule parallelism on a simulated multi-processor. Similarly, CAGE provides locks to deal with concurrent reads and writes to the central blackboard. Extensive simulation experiments have been performed on this architecture to measure the relative effects of these different forms of parallelism on the performance of a representative problem. Polygon (Nii, Aiello, and Rice, 1989) departs from the central blackboard theme by parallelizing the nodes which would normally reside on the central blackboard. Direct communication between nodes obviates the need for a global data structure. Data coherence is handled through the use of "smart" slots in the nodes which decide when a new value is better than the existing value of the slot using local heuristics. Polygon also runs on a simulated multi-processor.

A second family of approaches is built on concurrent processes in one or more conventional computers communicating through Inter-process Communication (IPC) mechanisms. The transaction processing blackboard described in (Ensor and Gabbe, 1988) provides a central blackboard which mediates the interaction between satellite blackboards which operate concurrently. The central blackboard uses a transaction processing metaphor to mediate reading and writing by satellite blackboards. This system was implemented on a network of Symbolics Lisp Machines and provides a nice model of loosely coupled groups of closely coupled experts. The COPS system (Leao and Talakdar, 1988) extends the OPS5 production system to provide fact exchange between independent OPS5 processes. Remote writing is available through addressed IPC messages, but appears to be used primarily for instantiating new processes. Normal fact exchange is accomplished through the use of "ambassador" rules. Ambassador rules can be thought of as parasites which are inserted into remote COPS processes to watch for fact patterns and report detections back to the originating system. A subset of the COPS processes are designated as blackboards. These central repositories exist primarily as intermediaries between non-rule-based applications (which can not accept ambassador rules) and the other OPS5-based applications.

3. WorkPlace Architecture

Our work falls into the second family of approaches, attempting to bring the cooperation available in blackboard systems to an environment of physically distributed conventional computers. Unlike COPS and the Ensor and Gabbe system, WorkPlace places no constraints on the processing formalism used in communicating nodes, and supports a range of interfaces to TCP/IP[1]. Cooperation is built on a flexible event distribution mechanism rather than shared memory. This mechanism uses a Publish/Subscribe/Sample metaphor, providing an exceptionally simple application interface. Cast in terms of the blackboard metaphor WorkPlace offers:

- a common catalog of facts with a selectively replicated fact space,
- and parallel application of knowledge sources and transformers to that space to further the current level of understanding.

From the application's point of view there are four operations necessary to participate in the WorkPlace environment. First, the application must provide a handler for facts received over the network. The implementation of this handler is entirely up to the application. Second, the application must regularly call a ProcessEvents() function to allow the communications software to keep in contact with the rest of the group. Information destined for the application will be caught during this call and passed to the application's fact handler. Third, the application must inform the agent of its remote information needs. These needs can change dynamically throughout the life of the program. Finally, the application must explicitly make information available which might be of interest to other members of the group.

The remainder of this section explains these operations in more detail, and presents some of

1 WorkPlace currently supports UNIX and Macintosh interfaces to TCP/IP. Support for VMS may be added in the future. Intermediate blackboards are not required to integrate heterogeneous applications since no assumptions are made about the nature of those applications.

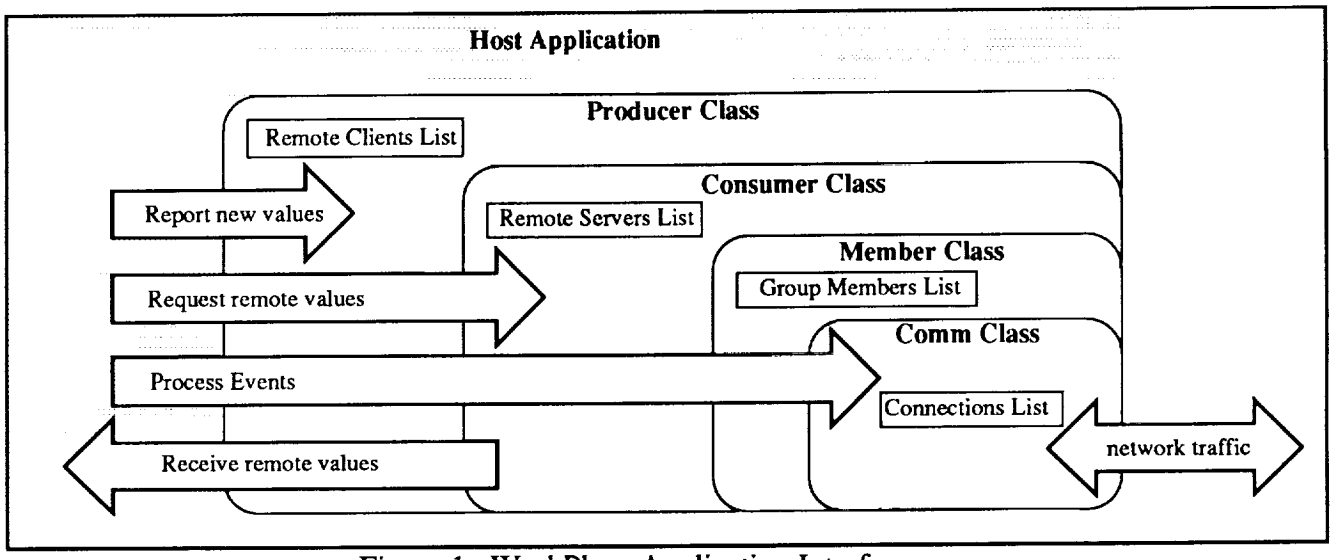


Figure 1. WorkPlace Application Interface

the ramifications of our implementation. An overview is shown in figure 1.

3.1 Membership In The WorkPlace

For applications to exchange information they must know of each other's existence and location. The list of existing applications and their locations can be thought of as membership information. WorkPlace acquires this membership information dynamically. The only static information required is the name of the group and the address of at least one member.

Dynamic membership means that the full roster and address lists are determined during the execution of the applications. The simplest approach is to use a centralized data server as an information clearing house. This server accepts connections from remote applications for either receiving or delivering information. All information produced by an application is forwarded to this data server for selective distribution to client sites. The down side of this approach is that a given piece of information is transmitted twice if a client exists for that information, and once if that information is not currently needed. The benefit is that a complete history of the products of the system is available. This centralized data server also represents a single point of failure for the environment: if the server goes down information flow stops.

The approach taken in the WorkPlace environment is to provide every application with the ability to accept and request connections from remote applications for either receiving or delivering information. The environment then becomes an association of peer nodes. That association is born with the appearance of a founding member, and ceases to exist when the last member exits. During the life of an association, any of its members, including the founding member, may leave the group and may later return. This fully distributed and dynamic design provides four benefits over centralized and static ones:

- Reduced vulnerability to individual node failures.
- Direct transmission of desired information from producer to consumer.
- No forwarding of unused information.
- The ability to add, delete, or move processes on an ad-hock basis without unnecessarily disrupting the execution of retained processes.

3.2 Product Exchange in The WorkPlace

From the application's perspective, product exchange is simply a matter of packaging information and reporting it, or receiving an information product and unpackaging it. The actual routing of information between agents occurs asynchronously with no involvement by the application. Allowing the agent to derive this information dynamically provides the flexibility necessary to accommodate changes in computational resources (e.g. processor or link failures), changes in computational load, and run

time changes to the applications suite (e.g. application crashes, upgrades to existing applications, and addition and removal of diagnostic processes).

Information is packaged as a value with a unique identifier. Identifiers are broken down further into an object specifier and an attribute specifier, as in "the float value (object specifier) temperature (attribute specifier) is 71.3 °F (value specifier)." Thus the conceptual package for an information product is a triple of the form <ObjectName>, <AttributeName>, <AttributeValue>. Services are provided for constructing attribute values from integer numbers, floating point numbers, character strings, or nested lists of these atomic types. The units in which a value is cast are assumed to be known to the receiver a-priori.

Reporting information is referred to as *Publishing* in WorkPlace. Each time the application publishes information, the agent caches the value reported. If there are registered clients for that information product, a message is forwarded to each of those clients specifying the new value. When the application publishes an information product whose identifier is different from any previously published by that application, the agent makes an announcement to all active members of the group. If the identifier has never been published by any other member of the group, the announcement identifies the object name, attribute name, and a more computationally efficient identifier for the product to be used in subsequent transactions. Otherwise the announcement simply notes the new source for that information identifier. If an application ceases to produce some information product, it can announce this fact through the *UnPublish* method. This removes the application's name from the producers list of the indicated product for every active member of the group.

The application requests remote information products by subscribing to information products. The embedded agent contacts known producers of that product and registers subscriptions with them. The agent also records the request so sources of that information which appear in the future can also be contacted. Two variants of the subscription method exist: *SubscribeToAll*, and *SubscribeToAny*. *SubscribeToAll* places subscriptions for the requested products with every agent known (now or in the future) to be capable of producing those products. When an application expects a single source for a piece of information, it can use the *SubscribeToAny* variant. SubscribeToAny places a single subscription, per product, with a random agent known to be capable of producing that product. If the selected product source stops publication of that product, quits the group, or displays anomalous behavior, then the agent will automatically move the subscription to an alternative source. A measure of fault tolerance is afforded through this mechanism by intentionally providing redundant copies of an information source on separate hardware. The flow of product updates can be halted by invoking the *UnSubscribeTo* method. This is useful when throughput disparities force the receiver to sample the data stream. An application can subscribe to and un-subscribe to a product or products an arbitrary number of times. The only overhead of *SubscribeTo* and *UnSubscribeTo* invocations is a short message to the selected supplier(s) of the product.

The application does not receive the value of a subscribed product until the value of that product changes. To obtain the current value of an information product, the application invokes the *SampleAll* or *SampleAny* agent methods. One, or more sources may or may not exist for the requested products. If no sources are known, then the supplied default value is returned to the application's product handler. If only one source is known, then a sample request is forwarded to that source. That source's agent responds to the request with the last cached value for the product. The local agent receives that value and returns it to the application by way of the application's product handler. If multiple sources are known for the requested product, then a sample request is forwarded to a randomly selected source for the "Any" case, or to each source in the "All" case.

3.3 The Network Interface

In reality the WorkPlace agent implements only the bookkeeping and protocol necessary to track group membership and information product sources and subscriptions. The actual network interface is implemented in the OSCAR (Open System for Coordinating Automated Resources) Agent class over which the WorkPlace agent is layered.

The OSCAR Agent class provides a common Application Programmer Interface (API) for network communications over several operating systems. Within this class a suite of standard routines for writing messages, reading messages, and getting connection status is defined. Subclasses implement the actual routines for specific protocols and operating systems. Currently the VMS, UNIX, and Macintosh operating systems are supported by OSCAR. New protocol implementations are added as peers in the suite of supported protocols. The OSCAR agent class selects among these protocols when a send request is made based on information it has on the location and type of the destination agent. The OSCAR agent also monitors each communication path for connection requests from remote agents.

4. Integration of a Distributed System

The IGS project has developed a testbed to evaluate and demonstrate the functionality of distributed knowledge-based systems within a control center setting (see figure 2). This testbed incorporates a spacecraft simulator, command scheduler, user interface, and three knowledge-based diagnostic systems. These testbed applications are integrated through the WorkPlace software. Our operational goal is to evolve the diagnostic components of the testbed into a platform diagnostic system for the first EOS spacecraft, due to be launched in late 1998.

The object-oriented spacecraft simulator accepts commands and generates telemetry data. A command scheduler acts as the bottleneck through which spacecraft commands are forwarded. Three knowledge-based systems interpret the telemetry stream in real time to monitor the state of spacecraft subsystems. When anomalies are detected, these systems provide explanation and advice to the user interface and optionally post suggested fixes with the command scheduler. The user interface depicts a graphical hierarchy of the spacecraft and ground components, where the user can zoom down into lower levels of detail when a problem is detected. An intelligent front end to the user interface filters and synthesizes related fault warnings to reduce information overload.

Integrating the Spacecraft Simulator

The job of the testbed is to control and monitor an object-oriented simulation of the EOS A spacecraft The model is composed of an electrical power system, thermal bus system, HIRIS (High-Resolution Imaging Spectrometer) instrument payload, and platform manager. The EOS spacecraft model is augmented by a model of the sun and the space-to-ground

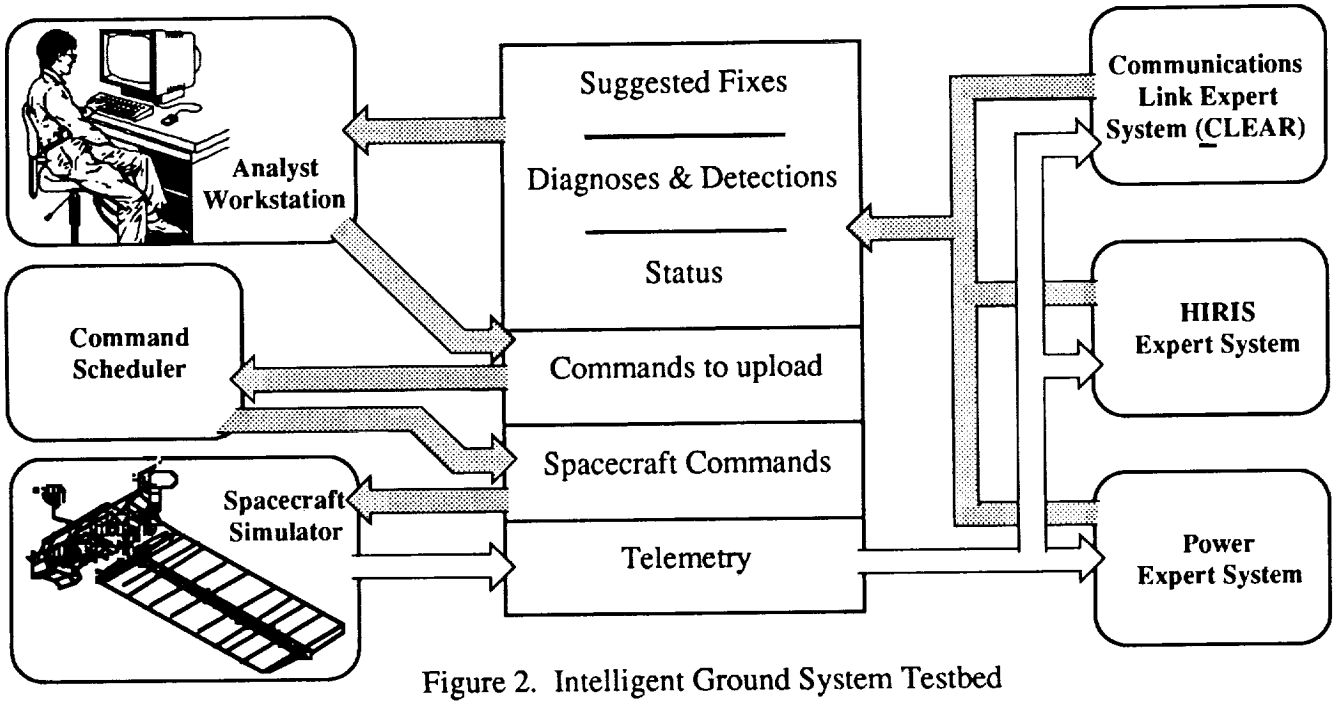


Figure 2. Intelligent Ground System Testbed

communications link. The simulation demonstrates electrical, thermal, and mechanical aspects of the spacecraft's behavior over time, with realistic responses to up-loaded commands and partial equipment failure. The simulator design is based on the connection manager architecture proposed by the Software Engineering Institute for flight simulators, as reported in (Lee, 1989), and the suggestions for object-oriented simulation in (Zeigler, 1990).

To operate successfully within the context of the testbed, the simulator needed to be able to receive commands and report telemetry. This was achieved by creating two new object types within the simulation environment. The first type received binary and serial information from other objects within the simulator, and converted and published that information as information products. These objects were then directly wired to the telemetry sources in the model. The second object type received information products and converted them to serial and binary signals. These signals were then connected to the spacecraft's command handler to allow remote control of the spacecraft. Lastly, the cyclic executive for the simulation was altered so that the embedded agent's *ProcessEvents* method could be called between simulation cycles.

Integration with Clips

The knowledge-based components of the testbed include three rule-based systems: the Communications Link Expert Assistance Resource (CLEAR), a power bus monitoring and diagnostic system (PowerFDIR), and a monitoring and diagnostic system for the spacecraft's HIRIS instrument (HirisFDIR). CLEAR was taken from an operational NASA communications fault diagnostic system. The other two systems where developed by the task expressly for the testbed.

Each of these systems is written in the "C" Language Integrated Production System (CLIPS), an expert system shell developed by Johnson Space Center (Giarrantano, 1991). We produced a distributed version of CLIPS version 5.1 (IGSClips) by embedding a WorkPlace agent. The integration required the addition of six new functions mirroring the distributed data agent methods: Publish, SubscribeToAny, SubscribeToAll, SampleAny, and SampleAll. One additional right-hand side function, GenNetSymbol, was added so that applications could generate network-unique identifiers. A call to the OSCAR ProcessEvents() function was inserted after each rule-execution cycle to service the network connection. Product updates arriving during this call are asserted into the fact base in the form (InPort <Object Name> <Attribute Name> <Attribute Value>). We have not yet evaluated the performance costs to CLIPS of the ProcessEvents call between cycles when no network information is pending.

IGSClips is similar in concept and operation to the COPS system (Leao and Talakdar 1988) described earlier. The main difference lies in the migration of group and communications management code out of the production system and into a separate module (the WorkPlace agent). Because IGSClips does not rely on ambassador rules, direct cooperation between the simulator and the diagnostic agents was possible.

The Impact of Asynchronous Operation

Two basic diagnostic architectures are present in the testbed: a shallow reactive architecture, and a deeper model-based architecture. While the deeper architectures are able to take more information into account in making their diagnoses, their integration into the asynchronous environment was more difficult.

CLEAR was implemented with the help of a domain expert who, through personal experience, was able to impart rules which related surface features (telemetry values) almost directly to diagnoses. CLEAR was ported to the testbed with only minor modifications necessary to publish its diagnoses and to format incoming telemetry to be accepted by the existing rule base.

The PowerFDIR and HirisFDIR diagnostic systems could not benefit from the compiled knowledge of a domain expert. Instead, these diagnostic systems keep two models of the spacecraft subsystem they monitor. The first model maintains the expected state of the subsystem based on a known initial state, the command stream that has been sent to the subsystem, and the known behavior of the subsystem in response to commands. The second model maintains the current state of the subsystem based on telemetry from that subsystem. Anomaly detection results from a comparison of these two models.

Temporal Coherence

Early experiments with the testbed showed that when the simulator was running on a separate platform from that which the diagnostic systems were running on, it was possible to swamp the model-based diagnostic systems with telemetry. In some cases the diagnostic systems were making recommendations for conditions which no longer existed, and in others the systems exhausted their ability to buffer incoming telemetry and crashed. Our solution was to change the diagnostic systems so that they in effect sampled the telemetry stream and then reacted to that sample. This de-coupled the processing rate of the diagnostic systems from the production rate of the telemetry source.

Unfortunately, we could not use the WorkPlace *Sample* operation to do this. The telemetry data is logically partitioned into sets which represent a snapshot of the spacecraft at a particular point in time. If the diagnostic system sampled two telemetry points which were not from the same frame, it would not be able to build a coherent picture of the current state of the reporting subsystem. Instead, when a diagnostic system wants to sample the telemetry stream, it places subscriptions for the telemetry points it needs. The diagnostic system then throws away all the telemetry it receives until the start of a new frame is detected (e.g. $element_0$ arrives). The system then caches all the subsequent data points until the end of the frame is detected, at which point the subscriptions are revoked. Now if the rule-based system is fast enough, it operates as it had previously. If at any time it is not fast enough, each diagnostic cycle only processes the most recent set of telemetry available.

This elaborate behavior on the part of the application simulates the sampling of a frame of information. If the WorkPlace agent embedded in the simulation knew that a given subset of data was part of a larger product, then that agent could take steps to guarantee the temporal coherence of the data made available to the group for sampling. At this point we have not extended the WorkPlace agent to support this, so the burden remains on the client application.

Non-monotonicity

Cooperation allows independent systems to leverage each other's expertise. A power failure on one power bus affects all the subsystems which are drawing power from that bus. The PowerFDIR has the expertise to identify the bus power failure, but the HirisFDIR does not. Through cooperation, the HirisFDIR can use external information generated by the PowerFDIR to distinguish an external power failure from an internal power distribution problem. Unfortunately, there is no guarantee that the helpful information will arrive before the subsystem monitor makes its diagnosis. The only solution we have at this time is for the subsystem monitor to retract its diagnosis when better information becomes available.

Summary

We have described our solutions to two technological hurdles standing in the way of cooperative knowledge based systems. The first, WorkPlace, provides an open system for fact exchange within a heterogeneous environment. We think that the generality of this tool makes it suitable for a wide range of applications, and that its support for "hot spares" makes it unique. Second, we have described some of the complications which have arisen from asynchrony, and how those complications have constrained the basic architecture of agents within the distributed cooperative environment. Taken together, these solutions demonstrate a viable design for physically distributed cooperative systems, and provide key tools for use in their implementation.

References

Adler, R.M. (1992). Coordinating Complex Problem-Solving Among Distributed Intelligent Agents. *1992 Goddard Conference on Space Applications of Artificial Intelligence*, CP 3141, 47-57.

Buckley, B., and Wheatcraft, L. (1992). Distributed Expert Systems for Ground and Space Applications. *1992 Goddard Conference on Space Applications of Artificial Intelligence*, CP 3141, 59-70.

Dominy, R. (1991). *Open System For Coordinating Automated Resources(OSCAR) Programmers Guide.* Technical report to NASA Goddard Space Flight Center.

Ensor, J.R., and Gabbe, J.D. (1988). Transactional Blackboards. In *Readings in Distributed Artificial Intelligence*, eds. Bond and Gasser. Morgan Kaufmann Publishers, Inc.

Fennell, R., and Lesser, V.R., (1976). Parallelism in Artificial Intelligence Problem Solving: A case study of Hearsay II. In *Readings in Distributed Artificial Intelligence*, eds. Bond and Gasser. Morgan Kaufmann Publishers, Inc.

Giarrantano, J. (1991) *CLIPS Reference Manual..* NASA Johnson Space Center

Kai Li. 1986 'Shared Virtual Memory on Loosely Coupled Multiprocessors'. Ph.D Thesis, Yale University, Department of Computer Science.

Leao, L.V., and Talukdar, S.N. (1988). COPS: A System For Constructing Multiple Blackboards. In *Readings in Distributed Artificial Intelligence*, eds. Bond and Gasser. Morgan Kaufmann Publishers, Inc.

Lee, K. J., et. al. (1989). *An OOD Paradigm for Flight Simulators, 2nd Edition.* Technical Report of the Software Engineering Institute.

Nii, H.P., Aiello, N., and Rice, J. (1989). Experiments on Cage and Poligon: Measuring the Performance of Parallel Blackboard Systems. In *Distributed Artificial Intelligence, Volume II.* eds. Gasser and Huhns. Morgan Kaufmann Publishers, Inc.

Rossomando, P.J. (1992). The Achievement of Spacecraft Autonomy Through the Thematic Application of Multiple Cooperating Intelligent Agents. *1992 Goddard Conference on Space Applications of Artificial Intelligence*, CP 3141, 87-103.

Zeigler, B. P., (1990). *Object-Oriented Simulation with Hierarchical, Modular Models.* Academic Press Inc.

Model-Based Reasoning for System and Software Engineering The Knowledge From Pictures (KFP) Environment

Sidney Bailin, Frank Paterra, and Scott Henderson
CTA Incorporated*

Walt Truszkowski**
NASA/Goddard Space Flight Center

1. Introduction

This paper presents a discussion of current work in the area of graphical modeling and model-based reasoning being undertaken by the Automation Technology Section, Code 522.3, at Goddard. The work was initially motivated by the growing realization that the knowledge acquisition process was a major bottleneck in the generation of fault detection, isolation, and repair (FDIR) systems for application in automated Mission Operations. As with most research activities this work started out with a simple objective: to develop a proof-of-concept system demonstrating that a draft rule-base for a FDIR system could be automatically realized by reasoning from a graphical representation of the system to be monitored. This work was called Knowledge From Pictures (KFP) (Truszkowski *et. al.* 1992). As the work has successfully progressed the KFP tool has become an environment populated by a set of tools that support a more comprehensive approach to model-based reasoning. This paper continues by giving an overview of the graphical modeling objectives of the work, describing the three tools that now populate the KFP environment, briefly presenting a discussion of related work in the field, and by indicating future directions for the KFP environment.

2 Graphical Modeling as a Basis for Answering Questions: KFP Concept

By way of introducing the major concepts in the current KFP environment we describe an approach to modeling a system that allows one to perform the following functions:

- Verify the correctness of a system design
- Simulate the behavior of a system
- Monitor the behavior of a system

Each of these functions amounts to answering certain questions about the system being modeled. We therefore view verification, simulation, and monitoring as different forms of *querying a system model.* In its current state of development, our models can be used to answer the following types of questions:

1) Under what conditions will event **E** or state **S** occur? This can be asked at design time for verification, or at run-time for explaining an observed event **E** or state **S** (monitoring).

* Mailing address: CTA Incorporated, 6116 Executive Boulevard, Suite 800, Rockville, MD 20852. E-mail: sbailin@cta.com, fpaterra@cta.com, scott@cta.com.

** Mailing address: NASA/Goddard Space Flight Center, Code 522, Greenbelt Road, Greenbelt, MD 20771. E-mail: wtruszkowski.520@postman.gsfc.nasa.gov.

2) What will occur as a consequence of state S and/or event E? This can be asked at design time for verification, or during system testing (simulation).

3) Will state S_2 occur as a consequence of state S_1 and event E? This can be asked at design time for verification, during system test (simulation), or at run-time to explain the observation of state S_2 (monitoring).

2.1 The Graphical Language

The modeling language represents a system as a set of components that are connected together via input and output ports. Each component has a set of output ports, which transmit information or physical resources (e.g., heat, power) to other components; and a set of input ports, which receive such resources from the output ports of other components. In addition, each component has an internal state, which is represented through one or more stores or variables. A component may also contain sub-components, which in turn are connected with each other. Thus, there is no conceptual difference between a system and a component: a component is a system consisting of its sub-components, and any system may be used as a component within a larger system.

For example, Figure 1 shows a model of the temperature control subsystem of a spacecraft instrument. The purpose of this subsystem is to control the temperature of the lens. Heat provided to the lens influences the temperature sensor, which in turn sends a digital temperature signal to the temperature driver. The function of the temperature driver is to turn the heater and cooler on and off as needed. The heater and cooler in turn influence the lens by passing or reducing heat, respectively.

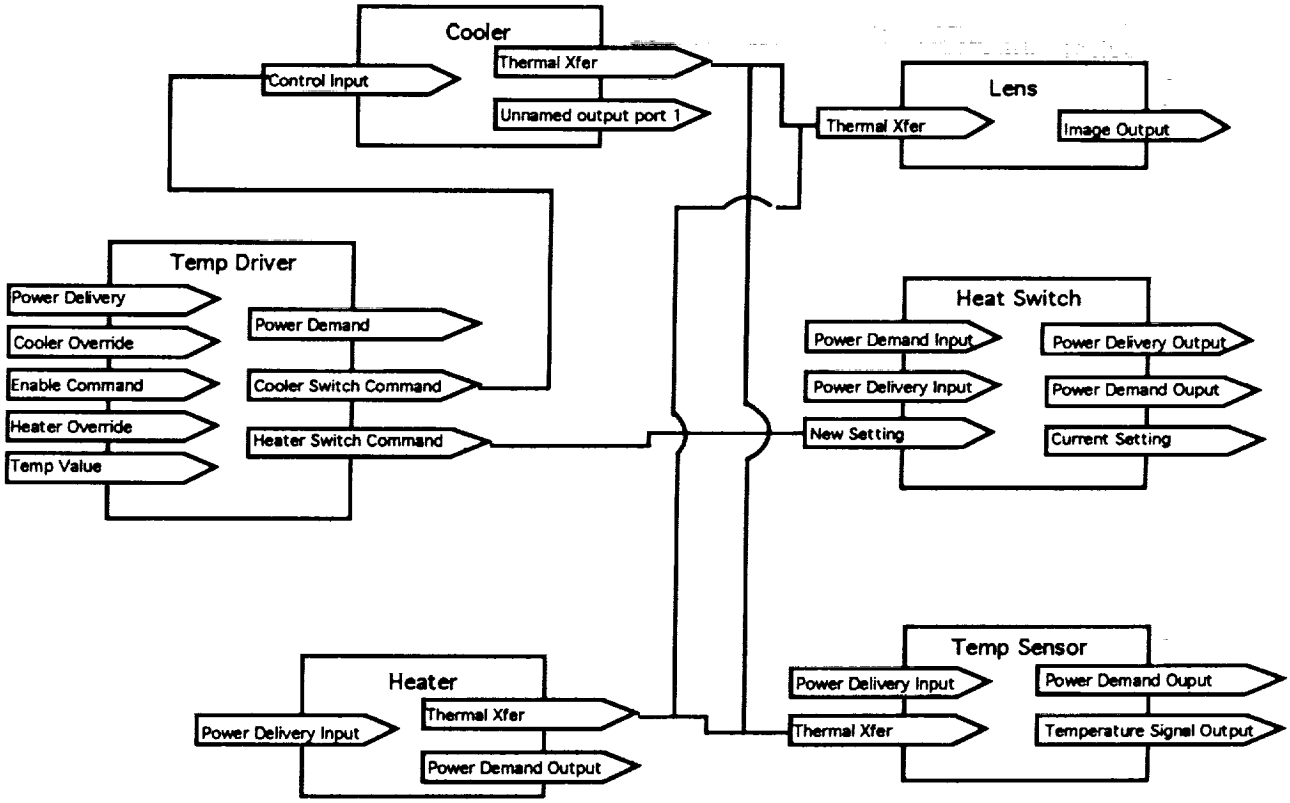


Figure 1: Temperature Control Subsystem Model

Each component in the model has a behavior description that specifies how its internal state and outputs change in accordance with changing input (and its previous state). To describe behaviors we use a tabular representation similar to that advocated by Parnas *et al* (1990). The tabular form allows us to accommodate both continuous functions and discontinuities. As Parnas observed, discontinuities are the major problem in specifying system behavior as a mathematical function—functional expressions in mathematics are typically continuous. At the other end of the continuous/discontinuous spectrum, a finite state machine is well-suited to describing a discrete set of behaviors, but is not suited to specifying new states and outputs as a continuous function of inputs and previous state.

The tabular representation is a blend of these two approaches. Each row in the table represents a nominalized, or abstracted, state, within which the system's behavior may be described as a continuous function of the current input and the specific (non-abstracted) state. Different rows in the table correspond to different abstracted states, in which the behavior is characterized by different functions. For example, the following table specifies the behavior of the temperature driver in the model shown in Figure 1:

Table 1: Behavior of Temperature Driver

Present State	Influence	Next State	Action
Idle	T > MaxTemp	CoolerOn	Set Cooler Signal
Idle	T < MinTemp	HeaterOn	Set Heater Signal
CoolerOn	T <= MaxTemp-Delta	Idle	Drop Cooler Signal
HeaterOn	T >= MinTemp+Delta	Idle	Drop Heater Signal

In this case the next-state and action functions are discrete-valued and are constant within each row of the table. The input port of the temperature driver is, of course, real-valued, but in the behavior table it is described in terms of three abstracted states:

T > MaxTemp, T < MinTemp, and (implicitly) MinTemp <= T <= MaxTemp

2.2 Querying the Models

Let us see how such models can be used to answer the types of questions posed above:

Under what conditions will event **E** *or state* **S** *occur?* To answer this we need to perform backward chaining through the state transitions and connections described in the model. We begin with the "fact" (whether hypothetical or observed) that event **E** or state **S** has occurred. This is treated as a goal in our backward chaining search. Typically **E** or **S** will describe the internal states and/or outputs of one or more components. We therefore look for state transitions in these components that would result in **E** or **S**. Each of these transitions will be predicated on a previously occurring internal state and input event. These previous states and input events therefore become subgoals. Input events of one component translate into output events of another component via the connections specified in the model. Similarly, the internal states that were pre-conditions of **E** or **S** become subgoals; they can be established by either assuming them as initial conditions, or tracing them back via still earlier transitions to previous states.

What will occur as a consequence of state **S** *and/or event* **E***?* Answering this requires that we perform forward chaining through the state transitions and connections described in the model. We being with the "fact" that the system is in state **S** and/or that event **E** has just occurred, and we proceed to execute the state transitions that occur as a result (as specified

in the behavior description of the model). Values that occur at output ports must be propagated over to whatever input ports they are connected to, and subsequent state transitions must then be carried out. Forward chaining therefore amounts to "executing" the model.

Will state S_2 occur as a consequence of state S_1 and event E? This type of question can be addressed by either forward or backward chaining, and in both cases there is a possibility of inconclusive results. We can execute the model starting with the occurrence of event **E** in state S_1, and check for the system entering into state S_2. If there is feedback in the model—i.e., there is a cyclical connection between some components C_1 -> C_2 ... -> C_k -> C_1—it may not be possible to limit the execution time within which S_2 must occur. We can, alternatively, treat S_2 as a goal in backward chaining, as in the first type of query—but with the additional constraint that the initial conditions arrived at must be consistent with S_1 and **E**. Here too, if there is feedback in the system, there is a possibility of infinite regression. We can artificially limit such searches by placing a bound on the number of transitions executed and the number of connections traversed. If the execution of transitions and the flow of resources over connections are viewed as taking time, rather than being instantaneous, then such a bound corresponds to a time limit within which S_2 is required to occur.

3 Tools for Querying the Models - the KFP Environment

In the KFP environment we have developed tools to provide answers to each of the types of questions posed above. The *Formal Interconnection Analysis Tool* (FIAT) is intended to be used for verifying designs. It performs backward chaining to answer questions (1) and (3) at design time. *The Multiple Aspect Simulation Tool* (MAST) executes the models. MAST can be viewed as a tool for simulating a model or, depending on the context, for implementing the model as a software system. The Diagnostics Inferred from Graphics (DIG) tool generates rules that backward chain through the model at run-time. DIG is intended to be used for system monitoring. In this section we describe the user interface through which the models are specified, and then show how FIAT, MAST, and DIG process the models for their respective purposes.

3.1 The User Interface

Figure 2 shows the main selection panel of the environment. Operations for managing the model library are provided within the Load and Save Libraries menu.. The Edit Components menu brings up the editor with the selected component displayed, or with a work space for creating of a new model.

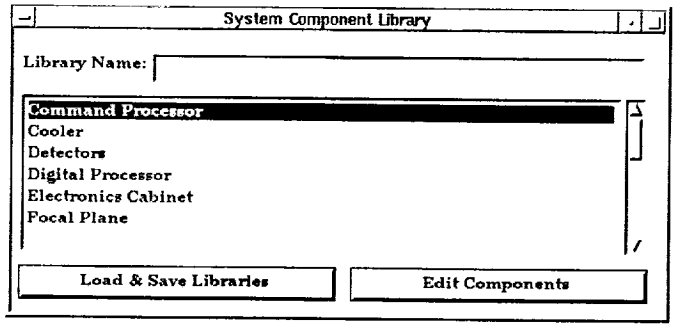


Figure 2: KFP Main Selection Panel

Figure 3 shows the model editor with an example system defined. The components of the system, shown as boxes, are *CmdProcessor*, *DigitalProcessor*, *Heat Relay*, *Heater*, *Cooler*, *PowerSupply*, and *Sensor*. Each component has ports, shown as large arrows. Connections between ports are displayed as arcs. As shown in the figure, a menu containing commands for editing and querying the model pops up in response to a middle-button mouse click in any "white space" area of the diagram:

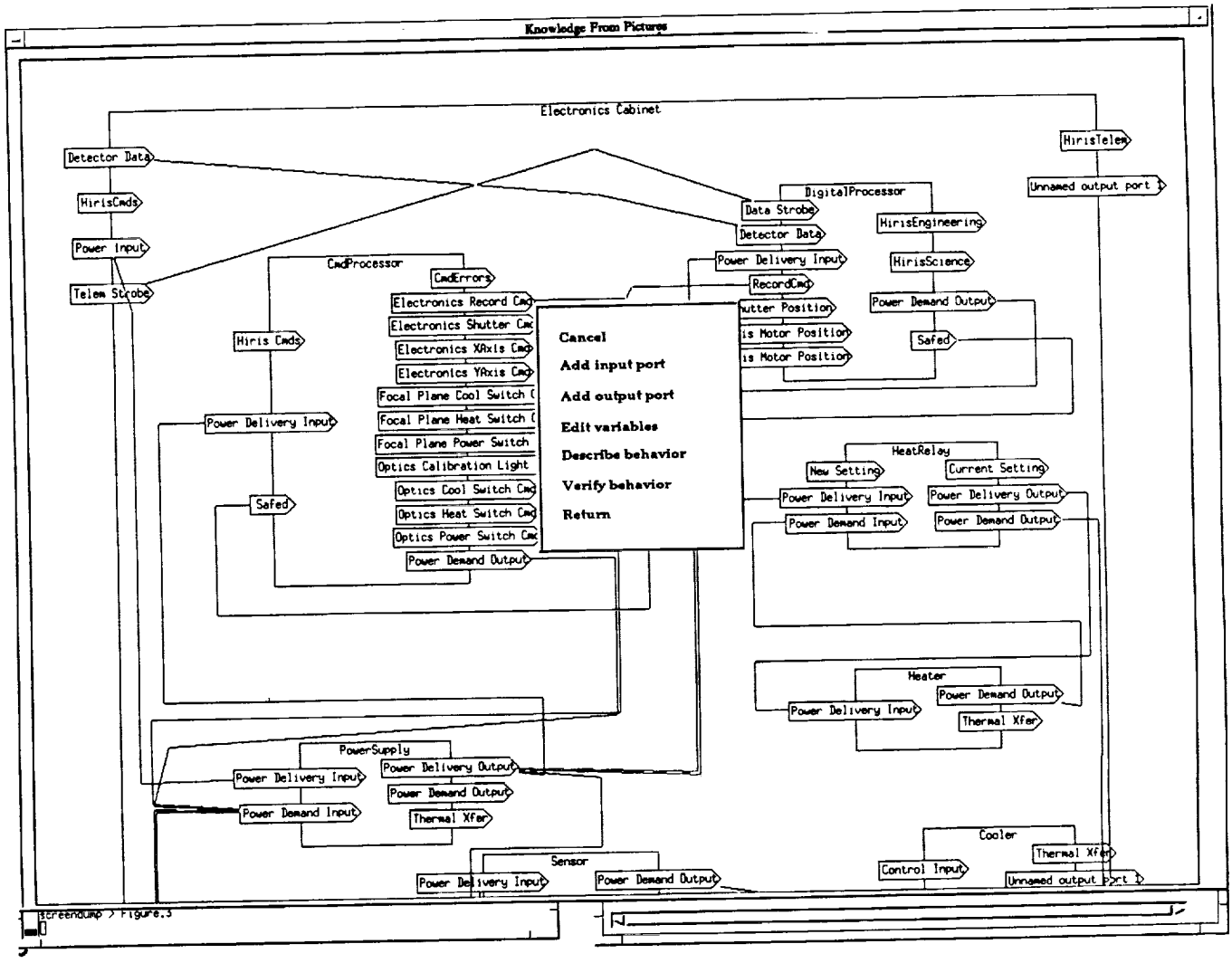


Figure 3: KFP Model Editor

The **Describe Behavior** selection is used to add, modify, or remove behavior states and transitions of the currently displayed system. As shown in Figure 4, the **Mappings** panel displays the state transitions that are currently defined. The **Transition** panel enables the user to modify an existing transition description, or to create a new one.

The *Starting State* is the state that the object is in before the transition occurs. The *Trigger* is the variable assignment or input influence that causes a state transition to occur. The *Ending State* is the state to which the object transitions. The **Add** buttons under *Starting State* and *Ending State* are used to add conjunctive conditions to these states' definitions. When the **OK** button is pressed, the behavior definition is added to the object's description, and is available to the specific tools that are used to query the model.

After adding all the components, behaviors, connections, and ports that are needed to define a system, the analyst selects the appropriate menu option to invoke one of the specific tools described below. For example, **Verify Behavior** (in Figure 3) invokes the Formal Interconnection Analysis Tool.

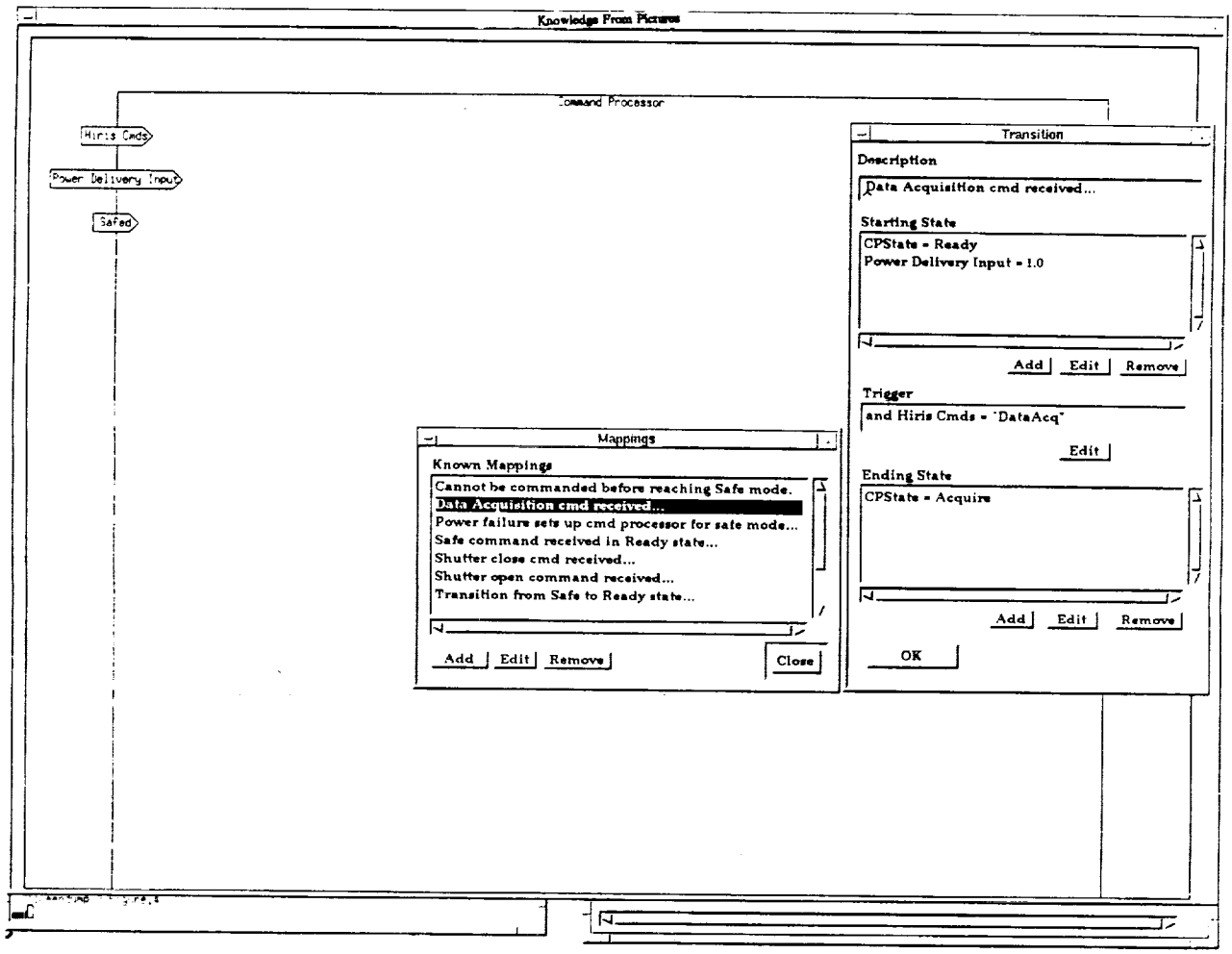


Figure 4: Component Behavior Definition

3.2 The Formal Interconnection Analysis Tool

FIAT uses a planning algorithm, implemented in Prolog, to chart a path from a (partially) specified initial state to a specified end state. Steps in the path are either the transfer of a value along a connection between ports, or the execution of a transition within a component. The planning algorithm works backwards from the specified end-state until it arrives at conditions that are specified in the initial state, or are consistent with the specified initial state.

FIAT is invoked by stating a goal to the planner. The planner then determines how to arrive at a situation in which this goal is true. A typical goal is of the form

<time-tag>: <goal-condition>

indicating that at the time designated by <time-tag>, the condition <goal-condition> is true. The time tag can be a numeric expression, a symbolic expression (e.g., containing a variable *t*), or one of the keywords START, END. A typical goal condition is **A.B.C.D** = *V*, indicating that the variable/port **D** within the component **A.B.C** has the value *V*.

Backchaining algorithm. The planner responds to a goal in one of the following ways, which are listed in order of priority:

- Finds a way to show that the goal is established. For example, a goal of the form START: <goal-condition> is established if <goal-conditrion> is implied by the user-specified initial conditions. A goal of the form T: <goal-condition> is established if the planner can show that <goal-condition> is implied by the user-specified initial conditions *and* that it is not affected by the user-specified initial event.
- Tries backchaining to create one or more subgoals. Backchaining takes one of two forms, either through a connection or through a state transition, depending on whether the goal refers to an input port, a state variable, or an output port.
- Adopts the goal as an additional assumption of the plan. The goal condition must be *consistent with* (though not necessarily implied by) the user-specified initial conditions.

Synchronizing and checking consistency of subplans. FIAT processes a list of subgoals by achieving each subgoal independently. This is not sufficient in general, since the subplans may interact. Moreover, the subplans may be of different length, requiring that they be synchronized with each other. In a general planning context, checking the consistency of an arbitrary set of subplans can be computationally intensive, since one must consider arbitrary interleavings of the individual steps of the plans. In our domain, however, it is not necessary to consider arbitrary interleavings. Instead, we synchronize in one of two ways:

- *Without time.* In this approach, all time tags are of the form START or END. All transitions and propagations of values along connections are assumed to occur instantaneously. This approach can only be used for models without feedback. In such models, given any two components $\mathbf{C_1}$ and $\mathbf{C_2}$, either $\mathbf{C_1}$ is "upstream" from $\mathbf{C_2}$ or *vice versa*. FIAT can therefore assume that no changes occur to a component **C** until all components upstream from **C** have been processed. When viewed recursively, this implies that all upstream components have stabilized in their resulting states by the time **C** undergoes a transition. Thus, each component undergoes at most one state transition, from its initial state (START) to the END state, which results from the influence of its upstream neighbors.

- *With time.* In this approach, every possible state transition and every connection in the model is annotated with a numeric or symbolic delay value, which indicates the length of time consumed by the transition or by propagation over the connection. FIAT uses this information to tag each step of a subplan with its time of occurrence in relation to the time of the ultimate goal.

Once the subplans are synchronized, FIAT can check their consistency by comparing, at each step in the plan, the values of the ports and state variables affected at that step. In general such comparisons are difficult because the values may be symbolic rather than numeric. For example: the value of a state variable V of a component C at time t may be specified, in one subplan, as a polynomial expression E_1 in the current values at the input ports $\mathbf{I_1}$ and $\mathbf{I_2}$ of C. In another subplan, the value of V at time t may be specified as another polynomial expression $\mathbf{E_2}$ in $\mathbf{I_1}$ and $\mathbf{I_2}$. To verify the consistency of these subplans, FIAT must establish the equality of $\mathbf{E_1}$ and $\mathbf{E_2}$. This is a theorem-proving problem and cannot be solved in general. Thus, depending on the complexity of the behavior specifications, the plan returned by FIAT may not be a conclusive proof that the goal state can be reached. Expanding FIAT's theorem-proving power in order to handle complex behavior specifications is an important goal of our research.

3.3 The Multiple Aspect Simulation Tool

The multiple aspect simulation tool (MAST) is used to "execute" the graphical models. We use the term "simulation" because typically, in our environment, the diagrams are used to model physical (electro-mechanical) systems. If, however, the model simply represented the components of a software system, then the resulting MAST code would be an implementation of that system.

MAST is based on a generalization of the connection management approach described in (Lee *et al* 1990). In that approach, communication between components is achieved through the operation of a *connection manager*, which is responsible for visiting each updated output port of each component and propagating its value to the necessary input ports (those to which the output is connected). In our generalization of this approach, each *type* of connection has its own connection manager. Currently MAST contains connection managers for the following types of connections:

- Power
- Thermal
- Digital
- Image

Each of these types of connections requires its own form of processing, e.g., the frequency with which values are updated—hence the use of separate connection managers (and the name "multiple aspect"). Another deviation from Lee *et al* is that the connection managers in MAST are global, i.e., they range over all components in the model. In the original approach, each subsystem, sub-subsystem, etc. has its own connection manager, which handles the connections between objects in that subsystem, sub-subsystem, etc. The use of global managers provides more flexibility in determining the order in which components should be visited.

The major benefit of using connection managers is that each component in the simulation remains independent of all other components. The components influence each other strictly through the flow of information over the connections defined in the graphical model, and these connections are implemented by means of connection managers. This simplifies the

construction of the simulator from the graphical model: all that needs to be done is to generate data descriptions of the components and their connections, and code implementing the behavior (i.e., the state transitions) of each individual component. The connection manager code, which is driven by the component descriptions, remains constant from one model to another and is simply linked in.

The entire simulation is driven by an executive, which is another fixed block of code that is linked together with the connection managers and component definitions.

3.4 The Diagnostics Inferred from Graphics Tool

DIG generates an expert system to monitor the system described by the graphical model. The expert system consists of a set of facts and rules in the C-Language Integrated Production System (CLIPS—see Giarrantano, 1991). The generated rules solve the fault monitoring problem as three subproblems: Detection, Isolation, and Recovery.

In the generated rules, connections between components of a system are used to isolate a failed component. The fault is detected when an alarm condition occurs. An example of such a condition would be a temperature-sensitive object operating outside of its design temperature range. Figure 1 showed a system in which such a fault may occur. The lens component is temperature sensitive and will register an alarm when its sensor reads above or below defined thresholds. In this example the only component involved in the alarm condition is the lens itself; however, in a more complex system one might also need to check other components, such as the quality of communication signals being received, before it is known that an alarm condition exists.

The cause of an alarm could be one of many failed components. DIG uses the connections between components as well as their known behavior states to identify the component that has suffered a fault. The values of each component's state variables are considered along with its current inputs to determine if it is operating according to its defined behavior. Both influence and behavior information are represented by CLIPS facts in the generated expert system.

Each alarm condition is represented by a CLIPS rule that uses facts about the state of the components contributing to the alarm to determine whether the condition exists. When an alarm is detected, a search begins for the faulted object causing the alarm. This search is performed by two rules generated for each object. The first rule compares the object's current state and inputs to its behavior specification; if these do not match, then the fault is occurring in that object. If the fault is found, a fact is asserted to begin the recovery phase. If no fault is detected, the second rule fires and uses the connection information to identify the next object to be examined. The connection paths form a collection of chains of objects that either directly or indirectly influence the components contributing to the alarm.

After a fault has been detected and isolated, the recovery phase begins. At present the recovery phase consists solely of notifying the operator, who can then take corrective action.

We have recently developed a run-time user interface for the generated expert system, which uses an animated version of the graphical model to display system status to the user. The run-time user interface itself is independent of the monitored system, and works in conjunction with any rule-base generated by DIG (and the corresponding diagram). The animation works as follows: when an alarm occurs, the component to which the alarm is attached is highlighted in red. During the ensuing fault isolation process, components that

"check out" are highlighted in green; a component in which the fault has been isolated is identified by pointing to it with red arrows. The user can therefore follow the fault isolation process by observing the successive highlighting of components in the display. When the state causing the alarm changes to a satisfactory state, the highlights are cleared and the components are restored to their usual display mode.

4 Related Work

Our approach to model-based engineering is closely related to work on executable specifications in software engineering and to model-based diagnostics in artificial intelligence. In this section we briefly review these two research areas in so far as they bear on our work.

4.1 Executable Software Specifications

The trend towards ever higher levels of languages in software engineering has led to the use of diagrams as executable specifications. Numerous tools developed in the research community, and a small number that are commercially available, either interpret diagrams or generate exectuable code on the basis of an implied operational semantics for the diagrams. The syntax and semantics of the diagrams varies widely, from dataflow approaches to state-based representations (see, for example, Zave and Schell, 1986; Jensen, 1987; Wang, 1988; Pulli, 1989). In our work, both MAST and DIG act as code generators that are guided by graphical models.

Harel (1992) makes a point quite close to ours by suggesting that such tools are more than curiosities, or even productivity enhancers. They represent, rather, a significant shift in the level of abstraction at which engineers can, and should, think about software. Two open issues in this shift concern the degree to which diagrams can accurately represent the intended functions of a software system, and the performance levels that can be achieved with automatically generated code. The first issue—semantic richness—depends on the modeling approach used, including the way in which diagrams are interpreted operationally, the amount and kinds of text-based annotations permitted, and (importantly) the domain of applications for which the software is intended. For example, dataflow models are amenable to a wide range of operational interpretations (see, for example, Bewtra *et al*, 1992); the semantics implied by MAST are well suited to simulation systems, but may not be appropriate for systems in which messaging plays a more essential role than dataflow.

The second issue—performance—is one that Harel sees as being progressively addressed as more work is done in the area of executable specifications. The chief use of such tools today is for the execution of functional prototypes of a system; the production system can then be developed with adequate performance by means of more conventional methods. Harel sees this changing, however, as we become more skillful at generating code from high-level models.

4.2 Model-Based Diagnostics

Model-based reasoning has become an important alternative to the conventional fault-based approach to diagnostics which was first demonstrated in the MYCIN system (Hayes-Roth *et al*, 1983). The fault-based approach uses a symptom/explanation structure, typically encoded in rules, to offer possible diagnoses of an observed problem. The limitations of this approach are now well-known: faults must be explicitly encoded in the knowledge base in order to be recognized, there is no sound method of representing uncertainty, and the validity of the knowledge base is difficult to establish.

Model-based reasoning employs a more direct representation of the rules that govern a system's behavior (Struss, 1992). The model-based approach treats the knowledge base as a description of the structure and behavior of the system being analyzed. Valid behavior is then characterized in terms of the states of observable elements of the system, and the relations that must hold between these states. Invalid behavior consists of any violation of these constraints, rather than being characterized by some finite set of faults identified *a priori*.

The model-based approach also admits a theoretically sound representation of uncertainty. The field of causal modeling applies Bayesian probability to the causal relationships between aspects of a system's state (Lemmer and Kyburg, 1992). The distinction between this and the fault-based approach is subtle but important. The fault-based approach draws a direct relationship between sets of symptoms and possible diagnoses. Causal modeling relates partial observations of a system's state to possible extended descriptions of the system's state. The range of possible explanations is much wider than in the fault-based approach, and is less susceptible to the biases that can easily enter unnoticed into a symptom/explanation structure.

Although the DIG tool generates a knowledge base in the form of production rules, the form of reasoning performed by these rules is clearly model-based.[1] There is no explicitly defined fault set—only a description of admissible and inadmissible states of the system. The rules are used to isolate the problem on the basis of the known relationships between components.

The open-endedness of model-based diagnostics has an analog in our approach to model-based engineering. As described in Section 2, we view the process of engineering as a process of creating, querying, and modifying models. In this context, open-endedness means that the questions that can be answered are not limited to some pre-defined set. This is a significant departure from the common practice in software engineering of using "canned" methodologies, which in essence prescribe a certain set of questions to be answered about a system under development. The model-based approach to system and software engineering provides a basis for adapting a method to the needs and constraints of a given project. Adaptation is achieved by tailoring the questions that will be asked about the system models. Of course, changing the questions may require enhancing or otherwise changing the models. By placing the emphasis on querying models, however, our approach encourages a scientific mindset in developing systems, as opposed to a mechanical "cookbook" approach.

5. Future Directions

Currently the KFP environment is a stand-alone set of tools for model-based graphical reasoning. In the coming year this environment will be integrated into a version of the Generic Spacecraft Analyst Assistant (GenSAA) workbench. GenSAA is designed to support rapid development and application of real-time expert systems in the Mission Operations domain. This experimental integration of the two environemnts (KFP and GenSAA) will provide an opportunity to more fully evaluate the anticipated benefits that will be derived from embedding a model-based graphical reasoning capability in a workbench for the real-time development of expert systems.

---

[1] We make this observation because model-based reasoning is often contrasted with rule-based reasoning.

References

Giarrantano, J., 1991. CLIPS Reference Manual. NASA Johnson Space Center, Houston, TX.

Harel, D., 1992. Biting the silver bullet: Toward a brighter future for system development. *IEEE Computer,* January 1992.

Hayes-Roth, F., Waterman, D., and Lenat, D., eds., 1983. *Building Expert Systems.* Addison-Wesley Publishing Company.

Jensen, K., 1987. Computer tools for construction, modification, and analysis of Petri nets. *Advances in Petri Nets, Part II,* ed. W. Brauer, W. Reisig, and G. Rozenberg. Lecture Notes in Computer Science, Vol. 255, pages 4-19. Springer-Verlag, New York, NY.

Lee, K. *et. al.*, 1990. An OOD paradigm for flight simulators, 2nd edition. Technical Report of the Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Lemmer, J. and Kyburg, H., 1992. *An Investigation of Independent Causality as a Basis for Uncertain Prediction and Inference.* Internal memorandum, CTA Incorporated, Rome, NY, November 16, 1992.

Montalvo, F., 1986. Diagram understanding: associating symbolic descriptions with images. *IEEE Computer Society Workshop on Visual Languages*, held in Dallas, TX, June 25-27, 1986. IEEE Computer Society Press, pages 4-11.

Musen, M., Fagan, L., Shortliffe, E., 1986. Graphical specification of procedural knowledge for an expert system. *IEEE Computer Society Workshop on Visual Languages*, held in Dallas, TX on June 25-27, 1986. IEEE Computer Society Press, pages 167-178.

Parnas, D., Asmis, G., and Madey, J., 1990. Assesment of safety-critical software. Technical Report 90-295, ISSN 0836-0227. Telecommunications Research Institute of Ontario. Queens University, Kingston, Ontario.

Pulli, P., 1989. Pattern-directed real-time execution of SA/RT specifications. *Proceedings of the Euromicro Workshop on Real Time*, June 1989. IEEE Computer Society Press, Los Alamitos, CA.

Struss, P., 1992. Knowledge-based diagnosis - an important challenge and touchstone for AI. *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 863-874. John Wiley and Sons, Chichester, England.

Truszkowski, W., Paterra, F., and Bailin, S., 1992. Knowledge from pictures. *Technology 2002 Conference*, December 1-3, 1992, Baltimore, MD. Proceedings to be published by NASA in 1993.

Wang, Y., 1988. A distributed specification model and its prototyping. *IEEE Transactions on Software Engineering,* Vol. 14, No. 8, pages 1090-1097. August 1988.

Zave, P. and Schell, W., 1986. Salient features of an executable specification language and its environment. *IEEE Transactions on Software Engineering,* Vol. 12, No. 2, pages 312-325. February 1986.

Inferring Heuristic Classification Hierarchies from Natural Language Input*

Richard Hull, Fernando Gomez
Department of Computer Science
University of Central Florida
Orlando, FL 32816
hull@cs.ucf.edu (407) 823-2366

Abstract

A methodology for inferring hierarchies representing heuristic knowledge about the check out, control, and monitoring sub-system (CCMS) of the space shuttle launch processing system from natural language input is explained. Our method identifies failures explicitly and implicitly described in natural language by domain experts and uses those descriptions to recommend classifications for inclusion in the experts' heuristic hierarchies.

1 Introduction

It is becoming generally accepted that most experts organize their problem-solving knowledge into a hierarchy of concepts [Gomez and Chandrasekaran, 1984; Clancey, 1985]. This hierarchical organization of knowledge is not explicitly used by the experts during the solution of problems, but rather is used in an implicit form. The task of the knowledge acquisition programs is to extract this hierarchical organization from the experts by making explicit to them the steps they need to visit in arriving to solutions. In other words, the goal of the knowledge acquisition interface is to make explicit the hierarchy of concepts. A well known knowledge acquisition methodology to acquire hierarchical knowledge from experts is that of repertory grids [Boose and Bradshaw, 1988; Boose, et al., 1989; Gaines and Shaw, 1988]. The repertory grid methodology elicits categorizations, called *constructs*, from the expert by asking him/her to rank numerically elements of the domain according to how well they satisfy a given construct.

Although this methodology has achieved considerable success, the problem of construct selection remains one of the most serious bottlenecks in the repertory grid methodology. If the constructs are provided to the domain expert by the knowledge engineer, the method works reasonably well because the task of the domain expert consists of filling in the cells of the grid with the appropriate values. However, in most cases the key aspect of the knowledge acquisition task is the acquisition of the constructs themselves from the domain expert. In this regard, elicitation techniques face strong limitations due to the fact that the linguistic aspect and contextual knowledge associated with the constructs are difficult to handle by elicitation techniques alone.

Our own research has been addressing this problem by studying the automatic construction of *constructs* or categorizations from natural language input. In [Gomez and Segami, 1991], the reader may find a description of linguistic constructions whose underlying structures are hierarchical categorizations. In this paper, however, we study the problem of *inferring* classifications from natural language sen-

*This research is being funded by NASA-KSC Contract NAG-10-0058

tences, rather than that of directly mapping into hierarchical structures. In order to provide some motivation for the problem we are facing, Figure 1 contains a portion of the heuristic hierarchy acquired from domain experts using our present interface. The problem we have experienced with our present interface is similar to the acquisition of *constructs* in the repertory grid methodology. If a good portion of the heuristic hierarchy is provided to the domain expert by the knowledge engineer, he/she can continue from there without considerable difficulty. However, building the hierarchy from scratch by the domain expert is a different matter altogether. Then, the main idea is to ask the expert to describe a given problem (a CCMS computer error in our application), infer some categorizations from the natural language description, and ask the expert to select the relevant one(s). This is basically the main idea that we explore in this paper in the context of the CCMS space shuttle network.

The remainder of this paper is organized into 6 sections. Section 2 describes the problem domain and our original knowledge acquisition interface. Section 3 describes the relationship between the interface, the Natural Language Component (NLC), and the Classification Suggestion Module (CSM). Section 4 explains the structures passed from the NLC to the CSM. Section 5 describes how the CSM infers classifications. Section 6 provides an overview of the NLC. Section 7 gives the authors' conclusions and lists future work to be done.

2 Automatic Knowledge Acquisition Interface (AKAI)

OPERA (Expert System Analyst) is an expert system whose task is to improve the operations support of the computer network in the space shuttle launch processing system at Kennedy Space Center[Adler, et al., 1989]. OPERA functions as a consultant to systems engineers by suggesting probable causes and recommending diagnostic and operational advisories regarding network error messages generated by the check out, control, and monitor subsystem (CCMS). Because OPERA only has information on approximately 10% of the 1500 error messages generated by the CCMS network, some type of knowledge acquisition tool is needed. During the past several years we have worked to develop a knowledge acquisition interface for OPERA. The result of this effort has been the creation of the Automatic Knowledge Acquisition Interface or simply AKAI.

It became apparent to us as we worked on the interface that while OPERA is not based on classification problem-solving, AKAI could make use of classification hierarchies [Gomez, et al., 1992a]. Two distinct types of classification hierarchies were identified and are now used by the interface: heuristic hierarchies and factual hierarchies. Heuristic hierarchies represent heuristic problem-solving knowledge of the domain. Each expert has his/her own ideas about how this knowledge is organized depending on their personal experience. Factual hierarchies represent hard or factual knowledge about the physical structure of physical objects. A factual hierarchy for the CCMS network was constructed and is currently being used by the interface. Because of the static nature of the CCMS network, the factual hierarchy is rarely modified. Of primary concern to us is the acquisition of the heuristic knowledge possessed by CCMS experts. Therefore, the focus of our research now is acquiring and constructing heuristic hierarchies, with the goal of AKAI being to acquire probable causes and advisories from systems engineers as efficiently as possible.

Towards this goal, user friendly features such as pull-down menus, mouse selectable text, and a wealth of functions to reorganize the hierarchy were incorporated in AKAI. Beta testing revealed, however, that naive users still had difficulty during the initial stages of heuristic hierarchy construction for the reasons stated above. In an effort to address this problem,

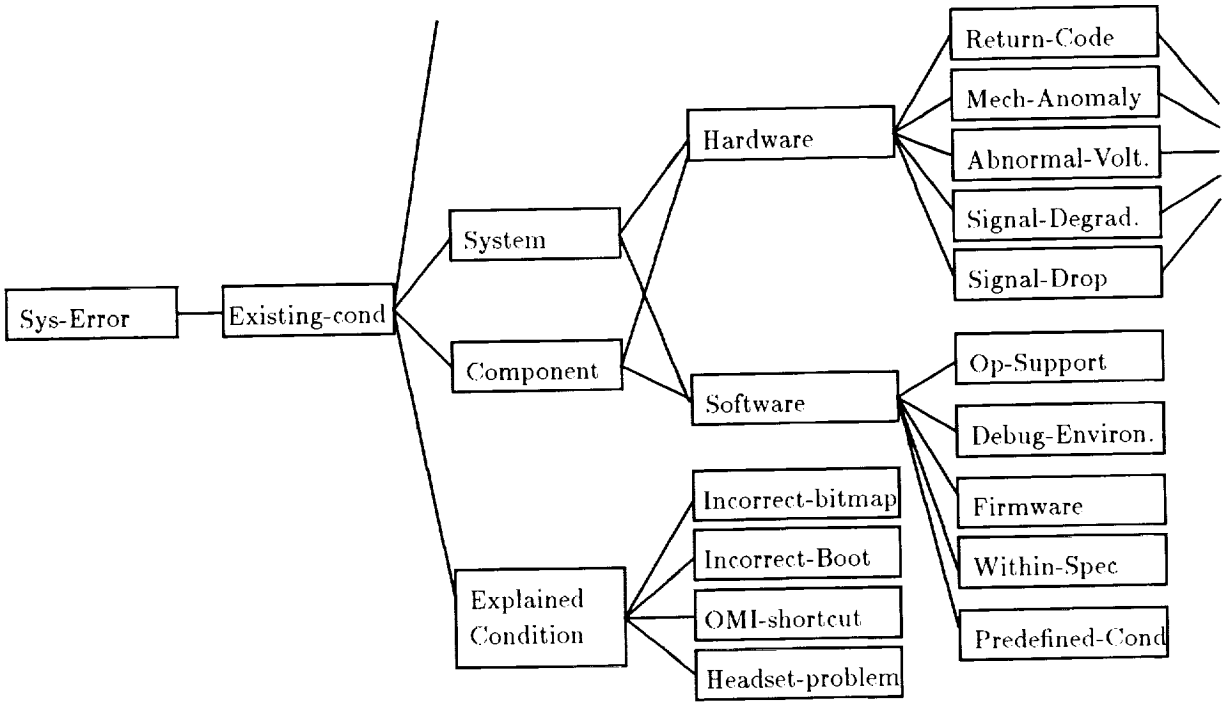


Figure 1: A Portion of a Heuristic Hierarchy for the CCMS Domain

we have added a natural language component (NLC) and a classification suggestion module (CSM).

3 The Improved Knowledge Acquisition Interface

The operation of the interface, graphically displayed in Figure 2, has changed only slightly due to the addition of the NLC and the CSM. The NLC is constructed around SNOWY [Gomez & Segami, 1989, 1990, 1991]. SNOWY is responsible for parsing (determining the syntactic constituents of the sentence) and interpreting (constructing the logical form of the sentence), and then forming (mapping the logical form of the sentence into SNOWY's representation language). The NLC is called by the interface during error categorization. At this time, the expert is asked to place the error message he/she has chosen to describe in his/her heuristic hierarchy. During the first stages of hierarchy construction there is a good chance that the appropriate category for the error message currently being described is not already in the heuristic hierarchy. In the original interface, the expert was expected to know, and was asked for, the name of an appropriate category. This was often a problem in the initial stages, and the experts caught in these situations tended to provide unsound categorizations.

The interface has since been enhanced to help unsophisticated users add new error categories to their heuristic hierarchies. If a user is unsure of how to classify an error, he/she is asked to provide a short description of what he/she knows about the error. This description typically consists of two or three sentences detailing relevant information about the message. The text is saved and passed to the NLC. The NLC enlists SNOWY to parse, interpret, and form the sentences. If SNOWY can make sense of the expert's description, the output of the formation phase is then passed to the CSM. The CSM uses the formation output to recommend categories to the expert. If one or more of these recommendations are selected by the

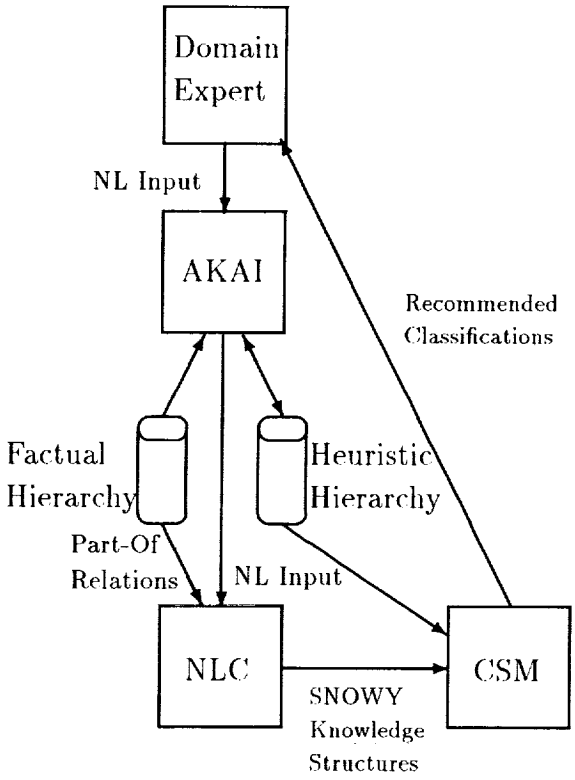


Figure 2: System Design

expert as acceptable, the problem of classifying the error is solved and the suggested error categories as well as the error message are placed in the expert's heuristic hierarchy. The interface then prompts the user for the probable causes of the error message and its operational and diagnostic advisories (this function of AKAI was not changed by the addition of the NLC and CSM).

On the other hand, if the expert is not satisfied with the CSM's recommendations or if SNOWY is unable to understand the expert's description, we may still be able to make a reasonable suggestion by postponing the classification of the error message until the probable causes have been entered by the domain expert and examined by the interface. We strongly believe that the probable causes represent an excellent source of text that is understandable by SNOWY and will provide classifications worth recommending. Most of the probable cause data that has been collected so far is of the form "$\mathcal{X}$ has failed," where $\mathcal{X}$ is a component of the CCMS network. SNOWY is quite capable of understanding sentences in this form. The classifications suggested by the CSM for these sentences are usually relevant because experts commonly use failed component names as category names within their heuristic hierarchies. If this process fails, however, the NLC and CSM are deactivated and the user falls back on the features of the original interface.

One may then question why the interface bothers to ask the user for a textual description of the error when analysis of the probable causes appears to provide suitable suggestions. We have found that additional text is needed if we are to make suggestions other than failed component suggestions. The system would not be able to make suggestions like "initialization failures" or "on line failures" if we only called the NLC with probable cause text. Classifications of this type are present in the heuristic hierarchies of the Grumman personnel first consulted to test the interface. Therefore, we must provide the interface with additional texts which could lead to recommendations other than failed components.

The operation of the enhanced interface is identical to the original after the error message has been placed within the heuristic hierarchy. The code of the original interface, therefore, was disturbed only slightly, and the users of the interface did not need to re-learn how to operate the system.

4 Input to the CSM

Before addressing the details of the CSM we must describe the structures which it takes as input. The formation phase of the NLC maps the logical form constructed by the interpreter into the knowledge representation structures of the representation language KL-SNOWY through the use of formation rules. The formation algorithm is called to form clauses as they finish the interpretation phase. The most

embedded clause of a sentence is formed first, the second most embedded is formed second, and so on until the main clause is formed. The structures, called object structures and relation structures, are used by the CSM to make recommendations. Together these two types of structures form the kernel of KL-SNOWY. There is a significant advantage to having SNOWY apply its formation phase to the logical form produced by the interpreter. This will become apparent during our discussion of the Classification Suggestion Module in section 5, if one understands the structures of the representation language. Therefore, it is important that the semantics of object and relation structures is clear.

4.1 Object Structures

Object structures represent knowledge about physical and abstract objects. Some physical objects are trains, tools, mountains, geese, etc., and some abstract objects or ideas are sets, states, properties, and relations. Conceptual relations representing knowledge about the object are represented as slots in the object structure frame (see the box surrounding the object structure for *CPU* in Figure 3).

These relations will either describe the object in some way or attribute actions to it. In the CPU object structure example, the slot "(process (data ($more (@a3))))" represents an action attributed to the concept CPU, and "(made-of (silicon ($more (@a2))))" represents a description. The relation structure names, *@a3* and *@a2*, point to relation structures that contain additional information which is not stored directly under CPU but elsewhere in SNOWY's long-term memory (LTM). In general, concept relations are represented in object structures as:

```
relation (@a1)                  monadic
relation (concept1 (@a1))       diadic
```

All concepts must have a unique name in memory so that the knowledge about them can

Object Structure

CPU
(is-a (electrical-component))
(part-of (computer ($more (@a1))))
(made-of (silicon ($more (@a2))))
(process (data ($more (@a3))))

@a3
(instance-of (action))
(args (CPU) (data))
(pr (process))
(actor (CPU (q (?))))
(theme (data (q (?))))

Figure 3: A Portion of the Concept *CPU* Acquired by SNOWY from Natural Language Input.

be integrated in a single place. Therefore, we need a method for dealing with concepts which are not explicitly named in the sentence. An example of such a sentence is "The adapter in the FEP returned an invalid status." The subject of the sentence, "the adapter in the FEP," is a complex concept which must be given a dummy name (a gensym) to uniquely identify it in LTM. The structure is called an *x-structure*. We use a *characteristic-features* slot to specify the necessary and sufficient conditions describing this new concept. For this complex concept, the representation would be:

```
(x1 (cf (is-a (adapter))
        (part-of (FEP))))
```

The meaning of this is that the x-structure *x1* is a sub-class of adapter, whose members all have the feature of being a part of a front-end processor (FEP). This feature is "characteristic" because it is shared by every member of the class *x1*. Complex concepts can arise from natural language constructs such as existentially quantified sentences, complex noun phrases, and restrictive qualifiers (relative clauses and prepositional phrases).

4.2 Relation Structures

Relation structures represent knowledge about instances of conceptual relations. Each structure contains a verbal concept, its cases and their fillers, the quantification of each filler, an *instance-of* slot indicating whether the relation is a description, action, proposition (embedded relation) or cf-structure, and an optional *truth-value* slot which indicates whether the relation is believed to be true or false by SNOWY. In the absence of a *truth-value* slot the statement is taken as true by default. For example, the relation structure, *@a3*, that represents "CPUs process data" is shown at the bottom of Figure 3.

The first slot, *instance-of*, indicates that *@a3* is an instance of an action relation. The *args* slot lists the arguments of the relation. If the relation is monadic, the *args* slot will contain a single concept. If the relation is diadic, as is the case in this example, the *args* slot contains two concepts, and so on. The *pr* slot contains the verbal concept or primitive. Following the verbal concept are its thematic cases. Each case is filled by a "quantified" concept from the argument list. The quantifier of an argument is the filler of its *q* sub-slot. In *@a3*, both the quantifiers for CPU and data are unknown, represented by a question mark. This reflects the fact that from the statement "CPUs process data" it is not clear if all CPUs process all data or only some CPUs process some data. Other possible fillers of the *q* slot are: most, many, all, cardinal adjectives, and numerals.

Creation of relation structures is normally handled by the formation algorithm. This algorithm constructs structures from the logical form by collecting the thematic cases identified by the interpreter for sentence clauses. In certain sentences, however, the formation algorithm must be overridden or postponed because the verbal concept requires an unusual construction to be formed. To handle these special cases, we use formation rules which are briefly discussed in section 6.

5 The Classification Suggestion Module (CSM)

The task of the Classification Suggestion Module (CSM) is to take the output from the formation phase of SNOWY and produce a list of error message classifications that can be suggested to the user. To accomplish this task, the CSM scans the output of the formation phase of SNOWY looking for certain constructions that are likely to lead to plausible suggestions. The CSM looks for the following constructions: negated relations and relations that indicate failures, descriptive relations which explicitly or implicitly indicate failed components, and complex noun phrases describing failed components. After a set of suggestions is identified, the CSM attempts to prioritize them based upon an analysis of the expert's heuristic hierarchy. This prioritized list of suggestions is then presented to the expert. Additionally, if the expert selects one or more of the suggestions, the CSM will attempt to engage the expert in a dialog whose purpose is to elicit more information. The sections below discuss each of the constructions relevant in identifying possible suggestions, the prioritization task, and the elicitation of additional information.

5.1 Relation Structures

The CSM identifies relation structures containing negated verbal concepts or with verbal concepts that indicate failures. Consider for example the formation of the sentence "The FEP failed to detect an acknowledgement from the i/o adapter," which contains a negated verbal concept.

The CSM scans the formation output for relation structures, such as @a27 below, and examines their truth-value slots. If the truth-value slot indicates that a verbal concept is explicitly negated, as **become-aware** is in the example below, we save the relation structure. The system can then use the cases of these structures to generate plausible classification suggestions (see the following section).

```
@a27
   (truth-value (f))
   (args (fep) (acknowledgement))
   (pr (become-aware))
   (actor (fep (q (constant))))
   (theme
      (acknowledgement (q (?))))
   (instance-of (proposition))
```

Example 1

Verbal concepts that implicitly indicate failures are also identified. In the sentence, "The option plane microcode crashed," the verb *crashed* indicates a failure. This is immediately obvious to the CSM because of the verbal concept that *crash* is mapped to during formation.

```
option-plane-microcode
   (is-a (microcode))

@a30
   (args (option-plane-microcode))
   (pr (fail))
   (actor
      (option-plane-microcode
         (q (constant))))
```

Example 2

The verb rules for the verb *crash* map it to the verbal concept **fail**. Other verbs which are mapped to the verbal concept **fail** are *break*, *collapse*, and *fail*. Because SNOWY is able to determine the underlying meaning of these verbs, the CSM has an easy time selecting negated relations and relations indicating failures.

5.2 Case Roles as Plausible Classifications

Some of the cases of these relation structures, such as actor, theme, at-loc, and at-time, lead to plausible classifications. In Example 1 above, the relation structure @a27 has two case slots: the actor case, filled by *fep*, and the theme case, filled by *acknowledgment*. These two cases suggest two possible error message classifications. One possible classification is the class of error messages generated by "fep failures". Because all the relation structures selected by the CSM denote failures, the actor of each relation represents a component that has failed to accomplish some task.[1] That failed component may also be responsible for generating other error messages. Therefore, it makes sense to recommend a class of error messages caused by the failed component. For this example, the CSM would save the classification "fep failure" as a possible classification to be recommended to the expert.

Another possible classification is "acknowledgement failures".[2] This supports the notion that the theme case of failure relations may lead to plausible classifications, when the original sentence is a "fail to" construction. In the sentence "The common data buffer failed to update the system configuration table," the theme case, filled by "the system configuration table," may potentially represent a category of errors. While the actor case represents "what" failed, the theme case describes the component that failed to be acted upon. Consequently, one might think that the theme case is not as likely a source of classifications as the actor case. We can, however, conceptualize a class of error messages which indicates the failure of some component to update the system configuration table. Each member of the class would have similar operational advisories instructing systems engineers in how to handle the failed update. Therefore, the CSM saves the theme case fillers of negated relations as possible classifications.

At-time cases can also lead to plausible classifications. These cases indicate *when* a failure occurred, which may be very signif-

[1] We must recognize that if the expert describes failures of irrelevant components, the system will make necessarily irrelevant recommendations which the expert may ignore.

[2] These failures are so common they are referred to as NOACKs.

icant. For example, consider the sentence "The FEP failed to respond during initialization." The prepositional phrase "during initialization" tells us that the failure occurred during the process of initialization. In general, if the filler of the at-time case is a process, we recommend that filler as a possible classification. For the example above this gives "initialization failures". It is our belief that the fillers of the at-time case should almost always be processes. This is because it makes little sense to use a time NP (a noun phrase specifying a time) except in certain situations.[3]

At-loc cases can lead to plausible classifications. For example, "The transmitter/receiver failed in the HIM" is a sentence in which the at-loc case, filled by "the HIM," represents a possible category of error messages. Because the failure occurred within the HIM, we can infer that the transmitter/receiver is located within the HIM and therefore may be a sub-part of the HIM. The HIM, which is the larger object, is likely to have other sub-parts which may fail. This means that the class of "HIM failures" is likely to be a good category of error messages. One should note that the object and its sub-part(s) form a part-of hierarchy. Discussion of how part-of hierarchies can be used to help prioritize suggested classifications can be found in section 5.5, *Part-Of Hierarchies.*

[3]In most cases, we would not expect to see a sentence with an at-time case filled by a time NP, such as, "The FEP failed to respond to the HIM at 10 pm." Obviously the expert giving such a description does not realize that he/she has described a specific error event, while what we are after is a more general description of the error. However, it may make sense to write, "The FEP fails to respond to the HIM during the winter". This sentence would lead to the classification, "winter failures," which seems plausible. In the cases where the at-time filler is a time NP, the CSM asks the expert, "Is this the only time that this error occurs?" If the expert responds with an affirmative answer, the system retains the filler as a possible classification.

5.3 Descriptive Relations and Noun Phrases

Concepts that have negative properties may lead to plausible classifications. If the expert mentions a defective component within his/her error message description, that component is likely to contribute to the error. The CSM identifies descriptive relations that indicate faulty components, as in "the i/o adapter is not operational," "the HIM may be down," or, "the HIM is unable to reset the status register." In these cases, the predicate adjective is examined to see if a failure is present. Predicates that explicitly or implicitly describe negative properties of network components provide strong indications that the components they modify have failed. Explicitly negated predicates are those that clearly indicate a negation, either by inclusion of the adverbs *not* and *no*, or through the use of negative prefixes. Some examples of explicitly negated predicates are *abnormal*, *unable*, *disabled*, *uninhibited*, and *incapable*. Important features, such as negative prefixes, are stored in a lexicon for each word. For example, the word *abnormal* has the following feature:

```
abnormal
   (neg-prefix (normal))
```

The neg-prefix slot tells us that *abnormal* contains a negative prefix affixed to the root word *normal.*

The representation of descriptive relations that denote negated properties is exactly the same as the representation of negated actions discussed in an earlier section. For example, the output from the formation phase for the sentence "the i/o adapter is abnormal" is

```
@a39
   (truth-value (f))
   (args (i/o-adapter) (normal))
   (pr (has-property))
   (descr-subj (i/o-adapter
      (q (constant))))
   (descr-obj (normal (q (?))))
   (instance-of (description))
```

Notice that the descriptive relation **has-property** is negated. The meaning of the relation structure, @a29, is "the i/o adapter does not have the property of being normal." The CSM can determine that this structure denotes a negative property by examining the truth-value slot in search of an "f". A more difficult sentence to handle would be "the i/o adapter is not abnormal." In this case, the formation phase realizes that there is a double negation. The final structure, therefore, will not have a truth-value slot filled by "f", and we will not recommend "i/o-adapter failures" as a category of error messages.

Some predicates may indicate a failure or negation but are not explicitly negated. Examples of this type of predicate are *defective*, *down*, and *broken*. In these cases, the meaning of the predicate adjective is needed if we are to determine that a failure has occurred. Currently, a sub-hierarchy within SNOWY's LTM maintains knowledge of properties.

The CSM also identifies complex noun phrases that indicate faulty components, as in "the defective HIM..." or "the failed data bus...." This is accomplished by examining the x-structures of complex noun phrases for negative properties. If the x-structure of a complex noun phrase has a negative property, the CSM will save the super-concept of the x-structure as a possible classification. From the sentence "All further polling will cease pending component fault isolation of the failed HIM," we would like to recommend "HIM failures" as a possible classification. The relevant portion of the representation provided to the CSM by the formation phase is

```
x1
   (cf (is-a (HIM)) (@a41))

@a41
   (args (x1) (defective))
   (pr (has-property))
   (descr-subj (HIM (q (constant))))
   (descr-obj (defective))
```

Defective indicates a failure so the CSM saves the super-concept of x1, HIM, as a possible classification.

5.4 Prioritizing Recommendations

Once a set of candidate classifications has been determined from a sequence of text, the CSM orders the candidates from highly recommended to least recommended. Several orderings are possible.

- If it can be determined that the user's heuristic hierarchy is structured based upon component/sub-component relationships, then failed components should be highly recommended.
- If it can be determined that the user's heuristic hierarchy is structured based upon process/sub-process relationships, then verbal concepts that represent processes or at-time slot fillers which are processes should be highly recommended, e.g., "the microcode fails during initialization," or, "the microcode failed to initialize."
- If nothing about the user's hierarchy can be determined, then fall back on the structure of the factual hierarchy which is a structural one, i.e., failed components should be highly recommended.

By prioritizing the classifications, the most relevant classifications (determined heuristically using the rules above) can be presented to the expert as such. This helps when the set of possible classifications is large.

5.5 Part-Of Hierarchies

There may also be a hierarchical relationship between several of the candidate classifications, especially when the candidates are selected from text describing probable causes. For example, the probable causes for error 141 are:

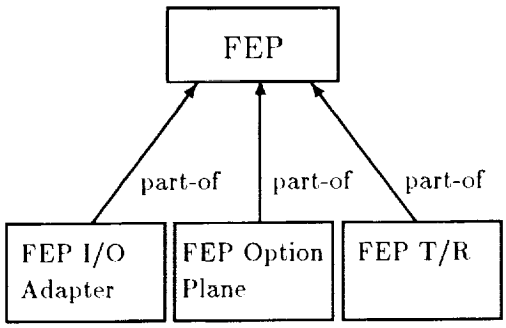


Figure 4: FEP Part-Of Hierarchy

1. `FEP i/o adapter failed.`
2. `FEP option plane failed.`
3. `I/O adapter port on the 4-port controller failed.`
4. `FEP transmitter/receiver failed.`

This leads to the failed component hierarchy shown in Figure 4. Probable causes 1, 2, and 4 describe the failures of sub-parts of a FEP. Determining that the FEP is related to i/o adapter, option plane, and transmitter/receiver by the has-part relation is the job of the noun-noun interpretation algorithms contained within SNOWY. That is, SNOWY is responsible for determining the meaning of complex noun phrases such as "FEP option plane," which is, "an option plane that is part of a FEP." The CSM simply has to look for a *part-of* relation under each of the sub-parts to recognize that a hierarchy exists. The existence of a part-of hierarchy provides strong evidence that the root of the hierarchy should be an error message category. In fact, it makes sense for the system to recommend the entire hierarchy to the expert.

5.6 Eliciting Additional Information

Up to this point, we have discussed how the NLC understands natural language input and how the CSM uses that understanding to identify and prioritize relevant categories of errors for presentation to the expert. The knowledge acquisition task does not end, however, when the expert selects a suggested classification. When experts accept suggested classifications, the CSM will "keep them talking" by prompting them with questions designed to trigger their recall of additional error message classifications. These questions prompt the expert for the names of similar messages that they feel would fall under the suggested category. The CSM also asks the user for other categories of errors that may be similar to the suggested category.

6 An Overview of the Natural Language Component

The NLC is an application of SNOWY. SNOWY is a system which integrates problem solving, knowledge acquisition, and information retrieval. In [Gomez & Segami, 1989] it was shown that, "in order for SNOWY to understand text, it needs to start with a minimum set of concepts which categorizes the world into states, actions, collections, etc." This *a priori* set of concepts, or ontology, is organized into a hierarchy based upon *is-a* relationships. The hierarchy is part of SNOWY's LTM. This LTM maintains the information that SNOWY has gathered from natural language input.

Each sentence presented to SNOWY undergoes three phases: a parsing and interpretation phase, a formation phase, and a recognition and integration phase. Because the recognition and integration phase is primarily concerned with updating SNOWY's LTM, which is unnecessary for our task, we only call upon SNOWY to parse, interpret, and form the expert's natural language input. These three processes are described below.

Parsing a sentence involves identifying its syntactic structures. The parser used by SNOWY is called WUP, which stands for word usage parser [Gomez 1989]. The underlying philosophy of WUP is that the syntactic usages of words play a greater role in parsing than is generally admitted. Discussion of how the usages are implemented and the details of the

operation of the parser will not be conducted in this paper, however. For our purposes, it is enough to know that the parser identifies the syntactic categories of the sentence. The syntactic categories used by WUP are: subject, verb, object, indirect object, prepositional phrase (PP), predicate, subordinate clause, and conjoined clause. The parser is not responsible for determining the attachment of prepositions, the verbal concept underlying the main verb, or the meaning of complex noun phrases. That is the duty of the interpreter.

6.1 Interpretation

The interpretation process is responsible for constructing the logical form from the syntactic constituents identified by the parser. This logical form represents the semantics of the sentence independent of any context. As each constituent of a sentence is identified, it is sent to the interpreter. It is important to point out that a constituent need not be interpreted the first time that it is seen by the interpreter. In fact, there are many cases where the interpretation of a particular constituent must be postponed until all the constituents of the sentence have been read. The constituent could be a noun phrase representing the subject or object of the sentence, in which case the interpreter must determine the meaning of the noun phrase. If the constituent is a verb, the interpreter must determine the underlying verbal concept that the verb represents. If the constituent is a prepositional phrase, the interpreter must determine its attachment and its meaning. Each of these three types of interpretation has its own set of interpretation rules. We will discuss each of the three types of interpretation and then culminate the interpretation section with a discussion of how this fits in with the domain at hand.

6.1.1 Noun Phrases

Interpreting noun phrases requires a great deal of knowledge of the meanings of nouns and adjectives. This is evident in the noun phrase "arthropod legs," which is the subject of the simple sentence "Arthropod legs are jointed." We can make sense of this phrase only because we know very well that arthropods, such as spiders and crustaceans, have legs. This knowledge allows us to determine that the NP above means "the legs that are part of arthropods." Without any knowledge of *arthropod* or *leg* we would be unable to determine a relationship between these two nouns. Similarly, knowledge of the adjective "wooden" is necessary to determine the meaning of the NP, "wooden legs," which is "legs made of wood." SNOWY stores knowledge of nouns as relations under their corresponding LTM concepts in SNOWY's concept hierarchy. Knowledge of adjectives is stored as interpretation rules. The noun phrase interpretation algorithm uses this knowledge when considering each pair of items in a given NP.

6.1.2 Verb Rules and Verbal Concepts

The interpreter algorithm makes use of verb rules to establish the underlying verbal concepts of sentences. These verbal concepts represent the meaning of the verb in the sentence. Below are the verb rules for the verb *dump*:

```
A Portion of the Verb Rules for the
            Verb Dump

(dump
  (((dump) (dumps) (dumped) (dumping)
    (has dumped) (had dumped))
      ((obj
         ((if part-of obj computer)
            (primitive-is transfer-data)
            (semantic-role-of-is obj
               from-loc)))))))
```

The "obj" slot contains a verb rule which will be tried when the parser passes the object constituent to the interpreter. This rule chooses the verbal concept **transfer-data** and marks the object as filling the from-loc case of this verbal concept in the event that the object of

the sentence is a part of a computer. Therefore, this rule would be used when SNOWY is interpreting a sentence like "Dumping the CPU registers would help isolate..." The interpretation of the main clause of this sentence would be "somebody transfer-data from the CPU registers (from-loc) to some unknown location (to-loc)."

While verbs like *dump* have very clear meanings, other verbs can be quite ambiguous. The verb *go* is reported to have 63 different meanings [Hirst, 1992]. We can side step this problem in most cases, however, because the domain of the incoming natural language is restricted to CCMS network error message descriptions.

Once the verbal concept has been determined, the interpreter attempts to fill the thematic cases of the verbal concept. Interpretation is now said to be driven by the verbal concept in the sense that we will attempt to place each of the other constituents within its framework. Thematic cases or roles show how noun phrases are related to the verbal concepts of sentences. Some of the most common thematic cases used by KL-SNOWY are: actor, theme, instrument, at-loc, from.loc, to-loc, at-time, init-time, end-time, descriptive-subject, and descriptive-object.

6.1.3 Prepositional Phrases

Interpreting prepositional phrases involves selecting the proper attachment (what sentence constituent is modified by the prepositional phrase) and establishing the meaning of the modification. Meanings and attachments are established by the verbal concept and interpretation rules under the given preposition [Gomez et al., 1992b]. Verbal concepts claim prepositional phrases through preposition rules (P-rules) stored under them. Noun phrases claim prepositional phrases through P-rules stored under the preposition.

6.1.4 Interpretation in the CCMS Network Domain

While interpretation of arbitrary text is currently an open problem, we can use the fact that we know the domain of the incoming discourse and the task of the CSM to limit the scope of the interpretation so that it is manageable. For instance, we have found that a significant percentage of the noun phrases used in error message descriptions and in the probable causes indicate specific components of the CCMS network.[4] The following is a table of some of the most common noun phrases in this domain:

Table 1: Common Noun Phrases in the CCMS Domain

active cpu	i/o card
common data buffer	data bus
error message	GSE FEP
ground data bus	LDB FEP
FEP option plane	PCM FEP
GSE data bus	standby cpu
GSE microcode	i/o adapter
HIM status register	option plane
system config table	

The semantics of these noun phrases can be captured by a few noun phrase interpretation rules. For instance, the phrase "data bus" is taken to mean a "bus for transporting data," where in this case *bus* is not a vehicle which makes frequent stops, but is a physical structure for transporting data and control information. Because we know the domain of the natural language input we will simply ignore the vehicle meaning of *bus*. A rule stored under the concept "data" will build the following interpretation when the noun phrase is interpreted:

```
(bus (transport (data)))
```

Another rule will look for part-of relationships between the nouns in noun phrases. This

[4]The components may be hardware components or software programs and data structures.

rule captures the meaning of phrases like "GSE data bus," "GSE FEP," "GSE microcode," and "FEP option plane." The interpretations of these four phrases are:

```
((bus (transport (data)))
    (part-of (GSE)))
(FEP (part-of (GSE)))
(microcode (part-of (GSE)))
(option-plane (part-of (FEP)))
```

Of course, to determine these part-of relations we must know *a priori* the physical structure of the CCMS network. This *a priori* information has been assembled by a knowledge engineer and is stored in AKAI's factual hierarchy. Therefore, we can determine these relations simply by consulting the factual hierarchy.

Verb rules need to be provided for the verbs commonly used in error descriptions and probable causes. A list of the verbs, for which verb rules were added, is given in Table 2. Each of these verb rules must specify a verbal concept.

Table 2: Verbs needing New Verb Rules

activate	fail	poll
command	generate	reset
detect	initialize	respond
dump	isolate	

Table 3 lists the new verbal concepts created for this domain.

Table 3: New Verbal Concepts

activate	initialize
command	isolate
become-aware	poll
fail	reset
fail-negation	respond
generate	transfer-data

Because the verb rules and verbal concepts added to SNOWY are dependent on knowing the LTM categories for nouns commonly used in the CCMS domain, it was necessary to add them to SNOWY's *a priori* hierarchy. Table 4 is a partial list of the concepts that were added to SNOWY's LTM.

Table 4: New Concepts added to LTM

acknowledgement	LDB
adapter	microcode
board	option-plane
buffer	PCM
bus	register
card	signal
computer	switch
cpu	transceiver
FEP	transmission
HIM	uplink
i/o	

6.1.5 Formation Rules

Formation rules are stored under verbal concepts. When the formation algorithm is activated, it looks to see if the verbal concept selected by the interpretation process has any special formation rules stored under it.[5] If formation rules are found, the normal formation algorithm is overridden and the system attempts to fire them. If a rule fires successfully, its consequent list is evaluated, effectively taking over the task of formation. Let us now discuss an example of a formation rule written by the authors to handle a special construction used in the CCMS domain.

Negated relations may come from sentences which use the "fail to" construction, or from sentences with explicitly negated verbs. The "fail to" construction is one in which the main verb of the main clause is *fail* and *fail* is followed by an embedded clause beginning with the word *to*. The representation of sentences using this type of construction is a relation structure representing an embedded clause whose verbal

[5]This discussion assumes that the interpreter was able to determine a verbal concept. In the event that no verbal concept was selected, the formation phase will be unable to construct a relation structure and is aborted.

concept has been negated. The formation rule responsible for creating this structure, shown below, is stored under the verbal concept **fail-negation** in the *f-rules* slot.

```
(fail-negation
    (is-a (description))
    (subj (thing (descr-subj)))
    (obj  (proposition (descr-obj)))
    (f-rules
        (fire-all
            ( t (negate-relation)))))
```

This rule calls a LISP function, called *negate-relation*, to negate the embedded clause. Take for example the sentence "The FEP failed to detect a response from the i/o adapter." We would like to end up with KL-SNOWY structures that represent that the FEP did *not* become aware of a response from the i/o adapter. Therefore, the task of the negate-relation function is to place an *f* in a *truth-value* slot of the relation structure associated with the embedded relation "[FEP] detect a response from the i/o adapter."

7 Conclusions and Future Work

We have shown how natural language input can be used to infer classifications suitable for inclusion into the heuristic hierarchies of AKAI, in a real world environment. We are currently in the early stages of the implementation of these ideas. Very little work needs to be done on the NLC, per se, because SNOWY is a working system. The bulk of our effort is, therefore, focused on implementing the CSM. Nevertheless, there are several data files used by SNOWY that must be scaled up if the enhancements of AKAI are to work "outside the lab."

One such data file is SNOWY's lexicon. To address this problem, a machine-readable dictionary created by the Summer Institute for Linguistics, called Englex, is being adapted for use by SNOWY. Specifically, entries in Englex are being converted into a format assimilable by SNOWY and added to SNOWY's lexicon. Englex contains morphological data for approximately 11,000 nouns, 4000 verbs, and 3400 adjectives, as well as adverbs, acronyms and abbreviations, proper nouns, prepositions, determiners, conjunctions, quantifiers, etc. Especially useful are markers indicating negative prefixes and nominalizations for nouns. By incorporating these words into SNOWY's lexicon, we hope to minimize the problem of encountering unknown words during the parsing of an expert's description.

Other data that will need to be expanded are SNOWY's verb rules and verbal concepts, interpretation rules for interpreting complex noun and prepositional phrases, and new formation rules for handling special sentence constructions. At first glance this task may seem quite daunting, but because we are receiving natural language input constrained to the domain of CCMS network messages, we can expect a limit to the diversity and complexity of the incoming text. This claim is supported by an analysis of the text that makes up the probable causes and advisory data currently stored in OPERA.

While extension of the NLC involves data, work on the CSM requires coding changes. It is important to note that the complexity of implementing the CSM is significantly reduced by the robustness of SNOWY's representation. Determining failures and their related cases is a simple task, assuming that SNOWY has been able to create the appropriate structures. This underscores the importance of an adequate representation for the purpose of acquiring knowledge.

Acknowledgements

We would like to thank R. Bruce Hosken and William Verhagen of Grumman Technical Services, KSC, for providing us with a set of typical error message descriptions, and the referees who made valuable comments.

References

Adler, R., Heard, A., & Hosken, B. (1989) An Expert Operations Analyst (OPERA) for a Distributed Computer Network. *AI Systems In Government (AISIG)*. Washington D.C.

Boose, J. and Bradshaw, J. (1988). Expertise Transfer and Complex Problems: using Aquinas as a knowledge acquisition workbench for knowledge-based systems, in J. Boose and B. Gaines *Knowledge-Based Systems,* vol. 2, Academic Press: London.

Boose, J., Shema, D. & Bradshaw, J. (1989). Recent Progress in Aquinas: a knowledge acquisition workbench. Journal of Knowledge Acquisition, 1, 185-214.

Clancey, W.J. (1985). Heuristic Classification, *Artificial Intelligence*, 27, 289-350.

Gaines, B.R., and Shaw, M. L.G. (1981). New Directions in the analysis and interactive elicitation of personal construct systems, in M.L.G. Shaw (ed). *Recent Advances in Personal Construct Technology*, Academic Press: New York.

Gomez, F. (1989) WUP: A Parser based on Word Usage. UCF-Tech-89-2, Dept. of Computer Science, University of Central Florida, Orlando, FL, 32816.

Gomez, F. & Chandrasekaran, B. (1984). Knowledge organization, and distribution for medical diagnosis. In W. Clancey & E. Shortliffe, Eds. *Readings in Medical Artificial Intelligence*. Reading, MA: Addison-Wesley.

Gomez, F., Hull, R., Karr, C., Hosken, R.B., & Verhagen, W. (1992a) Combining Factual and Heuristic Knowledge in Knowledge Acquisition, *Telematics and Informatics*, 9(6), Dec., 1992.

Gomez, F. & Segami, C. (1989) The Recognition and Integration of Concepts in Understanding Scientific Texts. *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, 51-77.

Gomez, F. & Segami, C. (1990) Knowledge acquisition from natural language for expert systems based on classification problem-solving methods. *Knowledge Acquisition*, Vol. 2, 107-128.

Gomez, F. & Segami, C. (1991) Classification Based Reasoning. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3), 644-659.

Gomez, F., Segami, C., & Hull, R. (1992b) Prepositional Attachment, Prepositional Meaning, Verb Meaning, and Thematic Roles. UCF-Tech-92, Dept. of Computer Science, University of Central Florida, Orlando, FL, 32816. (Submitted for Publication)

Hirst, Graeme. (1992) Semantic interpretation and the resolution of ambiguity. Cambridge University Press.

Multi-Viewpoint Clustering Analysis[1]

Mala Mehrotra
Vigyan Inc.
30 Research Drive
Hampton, VA 23666-1325

Chris Wild
Dept. of Computer Science
Old Dominion University
Norfolk, VA 23529-0162

Abstract

In this paper, we address the feasibility of partitioning rule-based systems into a number of meaningful units to enhance the comprehensibility, maintainability and reliability of expert systems software. Preliminary results have shown that *no single structuring principle or abstraction hierarchy is sufficient to understand complex knowledge bases.* We therefore propose the Multi-View Point - Clustering Analysis (MVP-CA) methodology to provide multiple views of the same expert system. We present the results of using this approach to partition a deployed knowledge-based system that navigates the Space Shuttle's entry. We also discuss the impact of this approach on verification and validation of knowledge-based systems.

Keywords domain knowledge, primary view, secondary view, conceptual clustering.

Introduction

Knowledge-based systems owe their appeal to the promise of utilizing expertise in the domain knowledge for the solution of difficult, poorly-understood, ill-structured problems. However, they must be subjected to rigorous verification and validation (V&V) analyses before they can be accepted into real-world critical applications. Unfortunately, expert systems do not lend themselves to the traditional V&V techniques for highly reliable software. There is a need to formulate an acceptable set of V&V techniques which can assure their quality. Better knowledge-acquisition techniques as well as better management, understanding and enhancement of the knowledge base is critical to the success of such V&V activities.

The difficulty in the V&V of large knowledge-based systems arises due to a number of reasons. Firstly, rapid prototyping and iterative development form key features of any expert system development activity. This has led to the development of ad-hoc techniques for expert system design without any software engineering guidelines. Moreover, due to the data-driven nature of expert systems, as the number of rules of an expert system increase, the number of possible interactions between the rules increases exponentially. The complexity of each pattern in a rule compounds the problem of V&V even further. As a result, large expert systems tend to be incomprehensible,

[1]This research was supported through Phase-I SBIR Grant - NAS9-18706 from NASA Johnson Space Center, Houston, TX.

difficult to debug or modify, and almost impossible to verify or validate.

Compounding the problem further is the fact that most expert systems are built without much regard to defining the requirements or specifications upfront. As any software, conventional or knowledge-based, becomes more complex, common errors are bound to occur through misunderstanding of specifications and requirements. Therefore, it is our belief that even if a software life cycle stresses specifications and requirements upfront, that will not be enough to guarantee the right product for complicated systems. There are bound to be ambiguities and interpretational problems. What is needed is a complementary tool that is capable of exposing such ambiguities and misinterpretations so that corrective action can be taken before it is too late in the software life cycle. Having a semi-automated means for capturing and structuring the meta-knowledge in a rulebase and cross-checking it with the specifications and requirements at various stages of the software life cycle could certainly help in this effort.

Conventional software yields more easily to verification efforts because control is explicitly represented as procedures which can be structured to encapsulate run-time abstractions. Modules can be designed in conventional software, each consisting of a manageable unit with a well-defined interface. Furthermore, procedures can be grouped into packages or objects which share an internal data structure. These units can then be subjected to unit/integration testing techniques.

Due to the declarative style of programming in knowledge-based systems, the generation of clusters to capture significant concepts in the domain seems more feasible than it would be for procedural software. By using knowledge-based programming techniques one is much closer to the domain knowledge of the problem than with procedural languages. The control aspects of the problem are abstracted away into the inference engine (or alternatively, the control rules are explicitly declared). The existence of a model of the domain would benefit the analysis of other knowledge-based systems within that domain by providing seeds for cluster formation. In addition, the use of a domain model to assist in the development of new knowledge-based systems is a promising research direction.

Existing research indicates that misunderstandings of the domain are a primary cause of systems failures [5, 12, 19]. Often small oversights or misunderstood interactions between sources of expertise lead to catastrophic failures. Techniques, methodologies and supporting tools are therefore needed to manage a complex system from multiple viewpoints and discover subtle interrelating concepts that are so critical for assuring the reliability of these systems. Even though language support for systems structuring has long been recognized as a key aspect of modern software and knowledge engineering, it is our contention that *no single structuring can simultaneously capture all the important concepts in complex knowledge-based systems.* We believe that techniques, methodologies and supporting tools are needed to manage a complex system from multiple viewpoints and that the discovery of subtle interrelating concepts is critical for assuring the reliability of these systems.

In this paper, we propose the concept of Multi-Viewpoint Clustering Analysis (MVP-CA) and show it as a feasible and effective technique towards structuring a rulebase for capturing its explicit as well as its implicit knowledge. The extraction of implicit, previously unknown, yet potentially useful in-

formation from the rulebase can have considerable impact on various stages of the life cycle of knowledge-based systems software. It can expose various design pitfalls during construction of the rulebase and the functional limitations of the software during its operation, as well as the subtle interrelationships between subgroups of rules that could prove very valuable in the maintenance of the system. It is our contention that the understanding of any large knowledge base will require that it be viewed from several different, possibly orthogonal viewpoints. MVP-CA provides an ability to discover significant structures within the rulebase by providing a mechanism to structure both hierarchically (from detail to abstract) and orthogonally (from different perspectives). Moreover, transfer of expertise from one problem domain to another related domain would be facilitated through the factoring of common aspects across the domains. Hence software reuse can be exploited through multiple structuring of a knowledge-based system.

First, we give an overview of our approach, followed by the methodology used to generate meaningful partitions. Next, we present the results of applying this methodology to a deployed expert system for navigation. We discuss some of the related work in this area and finally give our conclusions.

MVP-CA Overview

Our research efforts address the feasibility of automating the identification of rule-groups in knowledge-based systems software, to reflect the underlying subdomains of the problem. We prove the feasibility of MVP-CA (Multi-Viewpoint Clustering Analysis) methodology by building an MVP-CA tool to structure a few CLIPS[2] [3] knowledge-based systems along several viewpoints and showing that no single structuring principle or abstraction hierarchy is sufficient to understand complex knowledge bases.

Our approach utilizes clustering analysis techniques to group rules which share significant common properties and to identify the concepts which underlie these groups. Cluster analysis is a kind of unsupervised learning in which (a potentially large volume of) information is grouped into a (usually much smaller) set of clusters. If a simple description of the cluster is possible, then this description emphasizes critical features common to the cluster elements while suppressing irrelevant details. Thus, clustering has the potential to abstract from a large body of data, a set of underlying principles or concepts which organizes that data into meaningful classes. The knowledge acquisition process therefore involves "mining" the rule base for interesting concepts shared among the rules. The quality of clustering is related to two competing factors: intra-group **cohesiveness** and inter-group **coupling**. Informally, one can say that a group (or a cluster) is cohesive if all the items clustered together are somehow related or similar. Two groups are highly coupled if they share many similar properties and they are loosely coupled (possibly decoupled) if they share few (or no) similar properties. It is interesting to note that the qualities which define a good cluster are precisely those which define a good modular functional decomposition of a problem.

Preliminary experiments with the MVP-CA tool exposed significant natural structures within different knowledge bases. For example, consider ONAV (Onboard Navigation Expert System) [1], an expert system deployed on the shuttle to navigate dur-

[2]C Language Production System

ing re-entry. The file structure of ONAV provides one partitioning of the whole system. Not only did we find this generally accepted partitioning of ONAV, but we also found less obvious, more subtle interrelationships that existed across these primary clusterings. In this paper we present some of our results of applying the MVP-CA tool to ONAV. Misunderstandings of subtle interactions contribute most to the unreliability of knowledge-based systems [10]. Hence any methodology that exposes these relationships will contribute towards the V&V of large knowledge-based systems.

To illustrate the need for multiple viewpoints, consider an expert system for selecting the appropriate wine to complement a dinner. Even such a relatively small rulebase can be structured from several different viewpoints, as shown in Figure 1. Very broadly, the knowledge base can be divided into knowledge about the problem domain (selecting the appropriate wine) and knowledge about the control domain. The control knowledge breaks up further into user interface (how to question the user) and overall control strategies (balancing user preferences against experts' opinion through various phase control rules). Printout statements that ask the user for input or control the phasing of control rules belong to the control domain.

Similarly, knowledge about the problem domain, to aid in the selection of an appropriate wine for a meal, can be further subdivided into three major subdomains: types of food, wine properties and varieties, and a model of the customer's preferences. These domains are further subdivided into various subaspects. All these reflect different viewpoints of the same rule base. Within the food subdomain there are partitionings of taste of food, style of food, ingredients, etc. This is a hierarchical partitioning under the food subdomain. An orthogonal viewpoint in the wine subdomain is the interaction of wine properties with meal qualities. Similarly there are different aspects of the problem from the customer's viewpoint. In addition, there are rules which overlap subdomains or pass information to rules in other subdomains (data dependency relationships). Thus the same rule can be part of one subdomain and at the same time create information for use by rules in other subdomains, such as interface rules that specifically combine concepts from two subdomains (e.g., the relationship between beverage and the style of food.) There is an added value in using the MVP-CA tool for exposing substructures within the abstract groups formed, through hierarchical partitionings generated by it. The hierarchies represent viewpoints at different levels of conceptual abstraction.

MVP-CA Methodology

The methodology used for MVP-CA is summarized graphically in Figure 2. In the *Cluster Generation Phase* the focus is on generating meaningful clusters through statistical and semantics-based measures. In the *Cluster Analysis Phase* the focus is on performing a statistical and functional analysis of the output generated from the previous phase. Results of a statistical analysis of the output data feed back as better constraints on the parameters for grouping to improve the quality of subsequent clusterings. A functional analysis of the clusters captures the key concepts conveyed by the clusters generated. *Concepts* are meaningful patterns in the rulebase along with their associated attributes. A set of key concepts constitutes a single *viewpoint.* Multiple *clusterings* present multiple viewpoints on the rule base.

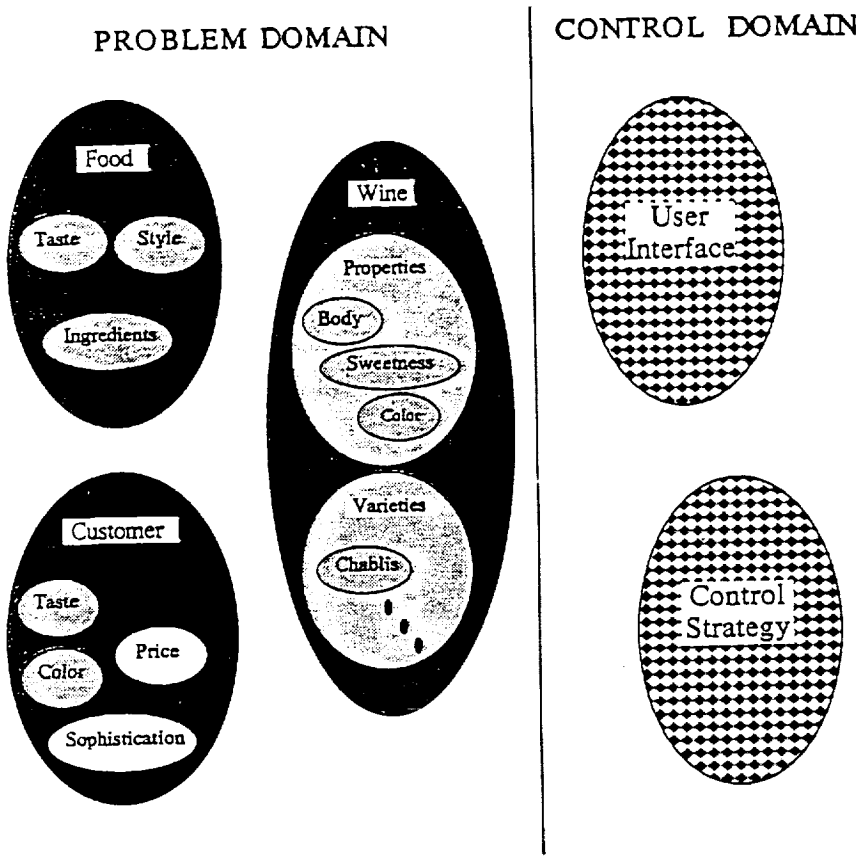


Figure 1: A Multi View Point of the Wine Rule Base

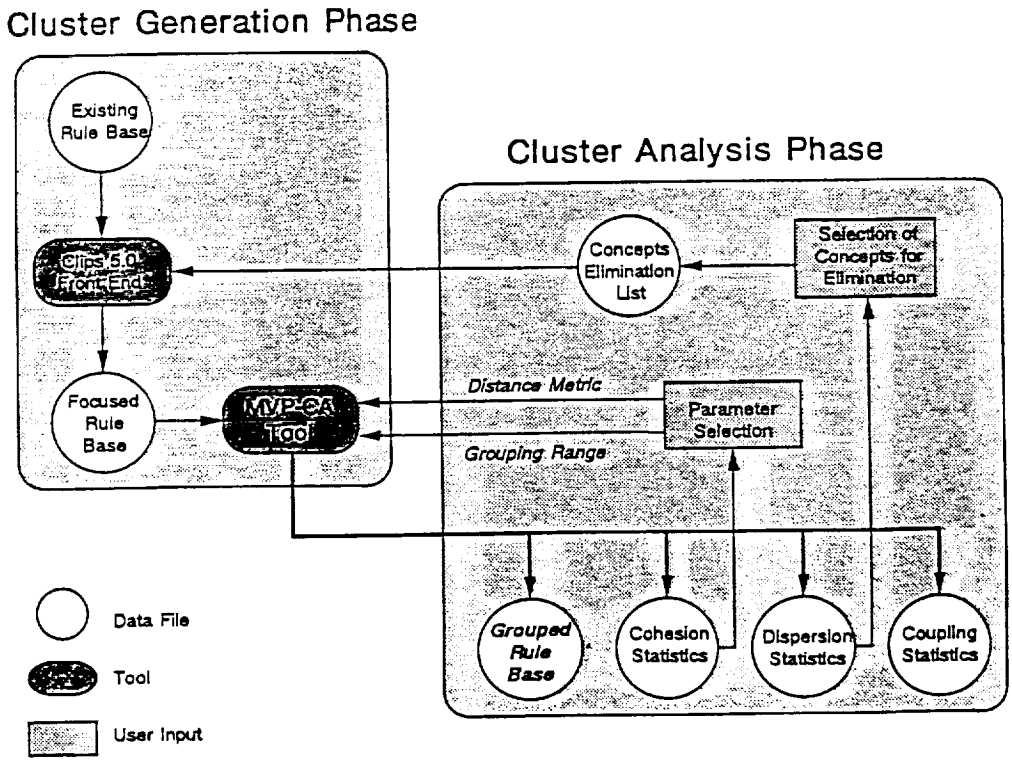


Figure 2: Phase-I Data Flow Diagram

A two-step procedure is utilized for extracting multiple viewpoints of a rulebase. First, form the best cluster possible using various measures, such as dispersion, cohesion and coupling. The overall dispersion of a pattern p is

$$disp(p) = \sum_{i=1}^{n_C} disp_{G_i}(p)$$

where n_C is the number of groups for clustering C and $disp_{G_i}(p) = 1$ if $p \in G_i$ and is 0 otherwise. Coupling is defined in terms of the inter-group distance, $D(i,j)$ as follows:

$$D(i,j) = \sum_{r_k \epsilon G_i} \sum_{r_l \epsilon G_j} \frac{d(r_k, r_l)}{n_i * n_j}$$

where n_i and n_j are the number of rules in groups G_i and G_j, respectively and $d(r_k, r_l)$ is the distance between rules r_k and r_l defined according to a distance metric selected by taking into account the nature of the rule base application [15]. For a given clustering, C, the cohesiveness measure is an index of the similarity of rules belonging to the same group. Cohesiveness of a rule r_k with respect to the group G_i that it belongs to is the average number of concepts($cncp$) it shares with the other rule members in the group G_i.

$$coh_{G_i}(r_k) = \sum_{\substack{(r_l \epsilon G_i) \\ (r_k \neq r_l)}} \frac{\mid 2 * comm\_cncp(r_k, r_l) \mid}{\mid cncp(r_k) \mid + \mid cncp(r_l) \mid}$$

Our clustering algorithm starts with all rules in their own clusters. At each step of the algorithm, the two groups which are most similar are merged together to form a new group. This pattern of mergings forms a hierarchical cluster from the single-member rule cluster to a cluster containing all the rules. One can look at this clustering near the "best" clustering points. Deciding which level in the hierarchy forms the "best" clustering of the rules requires an analysis of the cohesiveness of each cluster (the intragroup similarity) versus the coupling between groups (the intergroup similarity). When group cohesiveness is plotted against number of groups, plateau regions are generated signifying stable values for cohesiveness in certain ranges of number of groups. These regions represent optimal partitionings for a particular level of conceptual abstraction. Insight into concepts dominating the various clusters can be obtained through an examination of the groups at select points on the plateau regions. A hierarchical view of the rulebase can then be generated by repeating the above procedure for different plateau regions on the cohesiveness plots.

Next, with this "best" cluster, form a concept focus list - to either sharpen a current viewpoint or expose an alternate viewpoint. The concept focus list is formed from dispersion statistics of patterns. Dispersion is based on shared concepts - i.e. how a single concept is dispersed among the clusters. Low dispersion concepts are likely to represent concepts which characterize the clusters they are in. In fact, high dispersion concepts may interfere with the generation of highly cohesive clusters. Removing these concepts before clustering can help define the clusters more distinctly - a process which we call "sharpening". However, high dispersion concepts may also represent legitimate alternate structurings of the knowledge base. By selectively removing the low dispersion concepts, it is possible to reveal subtle alternate viewpoints - a concept we have termed multi-viewpoint clustering analysis [17, 16]. Thus the MVP-CA methodology provides a mechanism for comprehending complex knowledge-based systems through structuring them both hierarchically (from detail to abstract) and orthogonally (from different perspectives) leading to discovery of signif-

icant structures within the rule base.

Experimental Results

In this section we present some of the results obtained to date with the deployed knowledge-based system ONAV. Other results using animal classification and wine selection (available as part of the CLIPS 5.1 release) expert systems have been presented in [16].

Even with extensive comments and a tool such as CRSV[3] [2], the conceptual dependencies of rules across files cannot be easily determined. Not having any experience with Shuttle mission terminology, the rulenames were our only guide for understanding the domain in this knowledge-base. After clustering this rulebase several times using different criteria, we began to understand more of the subtle interrelationships. A graphical user interface, currently under development, would allow us to navigate through the rulebase and document the insights generated by the partitioning, thus fully utilizing the MVP-CA methodology. We document below our understanding of ONAV based on the natural partitionings set up by the developer as well as different groupings generated through the MVP-CA tool. We also show some of the interrelated concepts uncovered by this tool.

ONAV is an expert system developed at NASA Johnson to help navigate re-entry of a space craft. It has 387 rules divided across 16 files reflecting the various stages of navigation: ascent, entry and landing. The largest file *tacan.r* contains 127 rules. Monitoring of the space shuttle through ONAV entails updating some state vectors in the files *state.r*, *3state.r* and *hstd.r*. Measurements of velocity and acceleration are calculated through sensor readings from various devices such as the inertial measurement unit (*imu*), drag unit(*drag*), barometer unit(*baro*), tactical air navigation unit(*tacan*) and microwave scan beam landing system(*msbls*). The readings go through a Kalman filter and the state vector is updated through different types of line replacement units (*lru*) attached to the different devices. The computers onboard perform the necessary integrations on the corrected readings to obtain accurate values of velocity and position.

During landing, readings from different sources have to be tallied so that the positioning of the shuttle can be as accurate as possible before it hits the runway. During ascent the shuttle relies mainly on the inertial measurement unit readings, since an accurate positional value is less critical. All the *lrus* feed data to both the primary avionics system software(PASS) as well as to the backup flight system(BFS). Each of these systems have different selection schemes for determining the quality of data received. Ground-based radar stations resolve any conflicting values for the position of the shuttle and are used to aid in isolating malfunctioning equipment on board. Fidelity of the data is monitored through the status of a number of different flags. Rules in *telemetry.r* and *operator.r* determine which of the readings and updated state vectors are reliable at any point in time and give the operator power to override any decision. *Tables.r* provides general information on the *lru* configurations onboard, the fault matrix to be used for identifying the *imu* component that has failed, and a definition of the quality ratings to be used for the different state vectors and data readings. Runway selections are checked out in the file *runway.r*. Rules in *init.r*, *control.r*, and *output.r* essen-

[3]CLIPS Cross Reference Style Analysis and Verification Tool

tially accomplish the initial set up of global information during the different stages of the navigation by activating the various phase control rules, and they also handle the user interface issues.

Initial analysis of our results indicates that grouping a rulebase according to control aspects of the problem is not sufficient for understanding the problem. The static aspects of the problem can be understood only if domain knowledge can be separated from control knowledge [8, 9]. The original partitioning of ONAV into 16 files by the developer provided only a coarse partitioning based on the different phase aspects of the knowledge-based system. When the phase aspects of the rulebase were excised, it was found that rules with similar domain information were formed into a single group to give a secondary view. In order to discover the implicit interconnections between rules in different files, we combined all the files of ONAV to form one 387-rule rulebase. Since ONAV is primarily a monitoring system with some diagnostic capabilities, more meaningful partitionings were obtained when the antecedent patterns played a major role in determining the distance between rules [15].

Figure 3 shows the cohesion plot for a primary view of ONAV. The cohesion values beyond 200 groups are not plotted because there are too many single groups after that point. Consider some of the interesting plateau regions such as those around 11 and 50 groups. Partitionings generated with the primary view are more or less in accordance with the developer's partitionings in the rulebase reflecting various phase values. At 50 groups, we can see various subaspects for the *tacan* subphase - such as, tacan prediction rules, rules that put *tacan* in automatic mode, rules to determine *lru* quality, and so on - grouped in separate groups. However, at 10 groups, all these

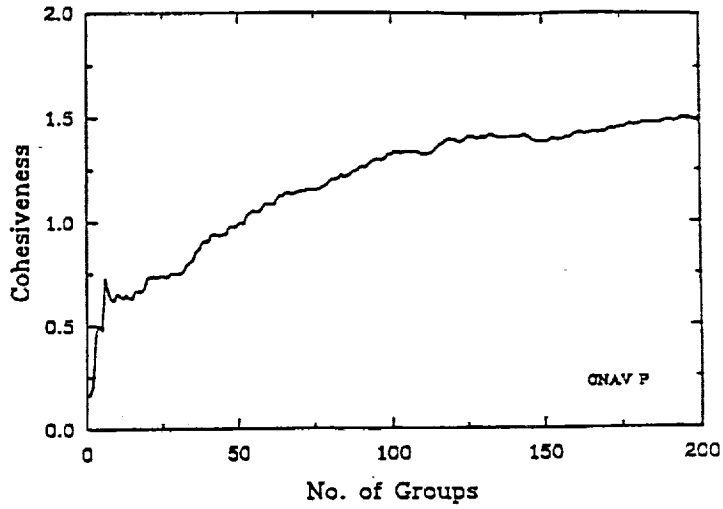


Figure 3: Cohesiveness Plot: ONAV rulebase - Primary View

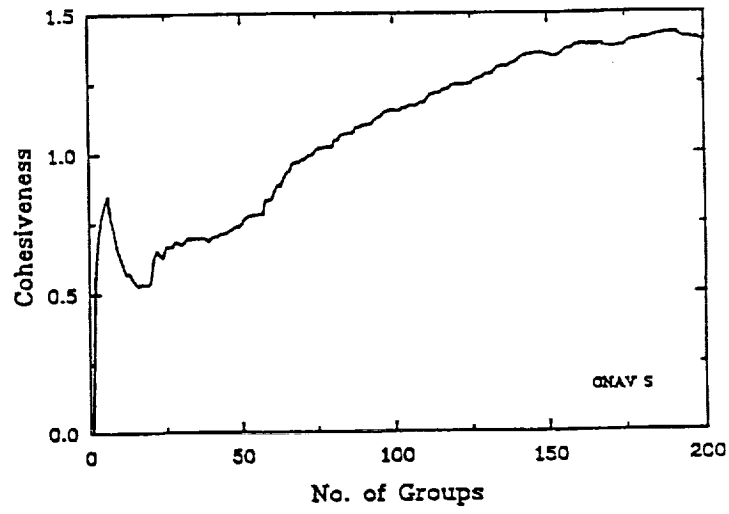


Figure 4: Cohesiveness Plot: ONAV rulebase - Secondary View

```
Group no 20:
Total number of rules in group: 15
Distance:: Min: 2.000000   Max: 7.666667   Mean: 4.284770
Cohesiveness: 0.429254   Minimum Membership: 0.033520
130     init-engaged-system-is-bfs      0.150933
131     init-engaged-system-is-pass     0.445672
134     init-system-availability-bfs-only       0.537089
135     init-system-availability-pass-only      0.566508
137     init-system-availability-both-pass-avail        0.561815
136     init-system-availability-both   0.565853
140     init-report-major-mode  0.451083
138     init-wrong-atmosphere   0.399324
139     init-right-atmosphere   0.371124
132     init-enable-msbls-sensor-lights 0.232097
133     init-enable-tacan-sensor-lights 0.295337
141     init-keep-last-ops-num  0.362514
142     init-report-abort-mode  0.501832
143     init-report-ascent-events       0.544941
223     nav-initialize  0.452685
```

Figure 5: Initialization Rules - Primary Clustering

tacan rules come together to form one group as conceived by the developer. Thus, while the original partitioning of ONAV into 16 files by the developer provided a coarse partitioning based on the different phase aspects of the knowledge-based system, there is added value in using the MVP-CA tool to expose the substructures within these abstract groups.

In the primary view, some groupings seem to have been generated based on criteria other than phase control. Initialization rules across different files come together in a group, group 20 in Figure 5, revealing initialization relationships from various phases. Initializations from other files, such as *nav-initialize* from file *state.r*, combine with this group revealing initialization relationships across files. This is an important revelation from the point of view of maintenance and verification.

In order to reveal a secondary view, we excised the concept of *phase* and *engaged-system*, which had the highest dispersion values in the primary view. The cohesion plot for the secondary view is given in Figure 4. Figures 7 and 8 give cross-sections of secondary groupings when all phase values were excised. The rule labelings generated in these files are the rulenames given by the developer originally. The numbers on the left are the rule numbers; distance between rule numbers thus gives an indication of the degree of juxtaposition of the rules in the combined rule base. Right-hand side numbers provide the cohesion value of the rule with respect to its group.

Once the phase aspect is deleted from the rulebase, other domain-dependent concepts start asserting themselves. In fact, in Figure 7, group 8 rules with similar rulenames (*hstd-same*, *hstd-bad*, *hstd-good* and *hstd-unavail*) across different files (*hstd.r* and *op-*

```
Group no 6:
Total number of rules in group: 19
Distance:: Min: 2.000000   Max: 6.000000   Mean: 3.688889
Cohesiveness: 0.443354   Minimum Membership: 0.160000
27      control-kickoff 0.385737
194     operator-stop   0.526758
201     operator-uplink-runway  0.454210
195     operator-delta-state    0.472484
196     operator-changed-delta-state    0.520917
197     operator-bfs-no-go      0.398486
198     operator-bfs-go 0.438764
199     operator-runway-selection       0.400035
200     operator-desired-runway-from-operator   0.443254
204     operator-atmosphere-change      0.375280
202     operator-toggle-tacan   0.342885
203     operator-cant-toggle    0.416190
205     gndeph-bad      0.443719
207     gndeph-same     0.490814
206     gndeph-good     0.452024
209     hstd-good       0.451576
208     hstd-bad        0.481044
210     hstd-same       0.534733
211     hstd-unavail    0.394810

Group no 12:
Total number of rules in group: 4
Distance:: Min: 2.333333   Max: 3.250000   Mean: 2.763889
Cohesiveness: 1.112825   Minimum Membership: 0.571429
42      hstd-bad        1.229437
44      hstd-same       0.884921
43      hstd-good       1.136364
45      hstd-unavail    1.200577
```

Figure 6: *Hstd* rules - Primary View

```
Group no 8:
Total number of rules in group: 24
Distance:: Min: 2.000000   Max: 9.900000   Mean: 4.437921
Cohesiveness: 0.377193   Minimum Membership: 0.000000
27      control-kickoff 0.351796
211     hstd-unavail    0.353633
194     operator-stop   0.479850
201     operator-uplink-runway  0.414068
210     hstd-same       0.492559
195     operator-delta-state    0.459732
196     operator-changed-delta-state    0.487993
197     operator-bfs-no-go      0.375855
198     operator-bfs-go 0.405531
199     operator-runway-selection       0.355490
200     operator-desired-runway-from-operator   0.391028
205     gndeph-bad      0.440268
207     gndeph-same     0.446688
202     operator-toggle-tacan   0.318074
203     operator-cant-toggle    0.389968
43      hstd-good       0.294328
206     gndeph-good     0.444678
209     hstd-good       0.472972
42      hstd-bad        0.316276
208     hstd-bad        0.487546
44      hstd-same       0.220726
138     init-wrong-atmosphere   0.090802
139     init-right-atmosphere   0.148827
204     operator-atmosphere-change      0.413935
```

Figure 7: *Hstd rules* - Secondary View

```
Group no 5:
Total number of rules in group: 4
Distance:: Min: 2.000000   Max: 4.000000   Mean: 3.000000
Cohesiveness: 1.328788   Minimum Membership: 0.013423
20      baro-aif-changed        1.176493
36      drag-aif-changed        1.653500
310     tacan-aif-changed       1.372859
179     msbls-aif-changed       1.112300
```

Figure 8: *Aif* rules - Secondary View

erator.r) come together because all of these rules deal with an incorrect input value for the *hstd* indicator. However, the *hstd* indicator is important in two subphases (*fact-assertion* and *hstd*). Once the phase component is deleted, the domain information that determines the *hstd* status pulls these rules into the same group. In the primary view these rules were in separate groups, 6 and 12, as shown in Figure 6.

It is also interesting to note that rules that share the concept of modifying the auto-inhibit-force flag (*aif*) in different phases all combine together in group 5, see Figure 8. This is a functional grouping of rules based on actions to be taken when there is a discrepancy between the previous and current values of the *aif* flag in the *barometer*, *drag*, *tacan* and *msbls* units. An orthogonal view of the rulebase comes into perspective with this grouping.

Such a view may be of immense value to the maintainer of the rulebase, since functional dependencies like these can be extremely difficult to locate across files, especially if the maintainer has not been the original developer of the system. Thus, our experimental results with the MVP-CA tool has demonstrated the feasibility of discovering significant structures within the rulebase by providing a mechanism to structure both hierarchically (from detail to abstract) and orthogonally (from different perspectives).

Related Work

Extraction of meta-knowledge for the purposes of comprehending and maintaining expert systems has been an accepted norm. In this section, we examine the role of structuring for this purpose in some well-established knowledge-based systems.

Systems such as XCON [4, 18] that have been in development for more than 10 years had to develop a new rule-based language, RIME, and rewrite XCON-in-RIME to facilitate its maintenance. XCON-in-RIME is supposed to make the domain knowledge more explicit both in terms of restructuring the rules and in terms of exposing the control structure for firing of the rules. Thus the problem space gets more hierarchically organized into different functional aspects, the problem solving method is made more explicit, a domain-specific classification is imposed on the rules and rule templates are created to serve as guides for rule creation.

Meta-Dendral [6] is a case study in the area of acquisition of domain knowledge. Meta-Dendral tries to resolve the bottleneck of knowledge acquisition through automatic generation of rule sets so as to aid the process of formation of newer scientific theories in mass spectroscopy.

TEIRESIAS [7] is built upon the MYCIN system to provide a mechanism for effective knowledge transfer. TEIRESIAS uses meta-rules to encode rule-based strategies that govern the usage of other rules. For this purpose it generates a set of rule models that are then used to guide this effort by being suggestive of both the content and form of the rules. These rule models can suggest incomplete areas of the knowledge base, provide summary explanations and help during debugging sessions. TEIRESIAS demonstrates the power of analyzing rule sets for experts especially when writing new rules. It is very helpful to see existing rules that are similar to a new rule under consideration so as to set the appropriate certainty factors in the new rule. Similarity could be suggestive of similar premises or similar conclusions. By comparing other evidence and other conclusions, the strength of the proposed rule can be estimated in the proper context. In fact,

each of the clusterings carries an extra slot indicating the context in which the rule set applies.

Although others [11, 13, 14] have attempted to cluster knowledge bases in order to abstract and structure the knowledge in them, existing approaches are limited in two major ways. First, we believe that *no one single structuring viewpoint is sufficient to comprehend a complex knowledge base.* Second, it is difficult to understand a single knowledge base isolated from an understanding of the underlying application domain. Often clues to the underlying semantic concepts are provided through descriptive names. Even then, the syntactic structure alone is rarely sufficient for managing and maintaining a complex system.

Clustering analysis can be used to reveal regularities in the knowledge base which can suggest possible subdomains of the problem. This structuring of the knowledge base is intended to capture both the explicit and the implicit knowledge in the knowledge base. *The point of interest of such an analysis should not be the clusters themselves, but the principles and ideas suggested by the clusters.* Such groups would allow one to abstract away from the point of view that *each rule is a procedure call* and look at the system from higher semantic levels. Each such group or unit can then be viewed as a procedure having a well-defined interface to other rule-groups. Once a rule base is decomposed into such "firewalled" units, studying the interactions between rules would become more tractable.

Due to the declarative style of programming in knowledge-based systems, the generation of clusters to capture significant concepts in the domain seems more feasible than it would be for procedural software. By using knowledge-based programming techniques one is much closer to the domain knowledge of the problem than with procedural languages. The control aspects of the problem are abstracted away into the inference engine (or alternatively, the control rules are explicitly declared.) Generation of a model of the problem domain can be accomplished through clustering. The existence of a model of the domain would benefit the analysis of other knowledge-based systems within that domain by providing seeds for cluster formation. In addition, the use of a domain model to assist in the development of new knowledge-based systems is a promising research direction.

Conclusions

Knowledge-based systems have the potential to greatly increase the capabilities of many aerospace applications such as Space Station, manned and unmanned spacecraft and civilian and military air transport. Automated systems that are knowledge based need to be deployed aboard these missions to reduce manpower support. Failure of such systems, however, can result in loss of life and of substantial financial investment. Hence these systems need to be highly reliable. Whereas DOD standards for conventional software have been developed, such as ADA-9x, a credible development and validation methodology for knowledge-based systems is currently lacking. Acceptance of knowledge-based systems software for critical missions is very much dependent on development of effective software engineering and validation techniques. A structured approach to management and maintenance of such systems would go a long way towards dispelling the myth that expert systems are inherently unreliable and that nothing can be done about it.

Expert systems have a wide commercial applicability. Liability issues arising out of improper functioning of such systems demand that any risk to life or property be either totally eliminated or at least minimized. Hence, it is imperative to develop rigorous and automatic testing tools for the verification and validation of knowledge-based systems. An integrated environment for expert system verification and validation, such as is proposed by MVP-CA, would overcome this barrier, opening them up for a broad range of important applications. An integrated system for performing V&V on structured knowledge bases will enhance the reliability of knowledge-based software and bridge its current gap with conventional systems.

References

[1] Knowledge Requirements for the Onboard Navigation Console Expert/Trainer System. Technical Report JSC-22657, NASA, Lyndon B. Johnson Space Center, Houston, TX., September 1988.

[2] CLIPS Reference Manual. Technical Report JSC-22948, Artificial Intelligence Center, NASA, Lyndon B. Johnson Space Center, Houston, TX., June 1989.

[3] *CLIPS Basic Programming Guide CLIPS Version 5.1.* Houston, TX., September 1991.

[4] V. E. Barker and D. E. O'Connor. Expert Systems for Configuration at Digital: XCON and beyond. *Communications of the ACM*, 32(3), March 1989.

[5] V. R. Basili and B. T. Perricone. Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, 1(27):42–52, January 1984.

[6] B.G. Buchanan. Issues of Repesentation in Conveying the Scope and Limitations of Intelligent Assistant Programs. In J.E. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence*, pages 407–425. John Wiley & Sons, 1979.

[7] B.G. Buchanan and E.H. Shortliffe. Knowledge Engineering. In *Rule-Based Expert Systems*, chapter 7, pages 149–158. Addison Wesley Publishing Co., 1985.

[8] B. Chandrasekharan. Generic tasks in knowledge-based reasoning: High-level building blocks for expert systems design. *IEEE Expert*, Fall 1986.

[9] W. J. Clancey. The advantages of abstract control knowledge in expert system design. In *Proceedings, National Conference on Artificial Intelligence*, pages 74–78, 1983.

[10] D. Hamilton and K.Kelley. State-of-the-practice in knowledge-based system verification and validation. *Expert Systems with Applications*, 3:403–410, 1991.

[11] R. J. K. Jacob and J. N. Froscher. A Software Engineering Methodology for Rule-based Systems. *IEEE Transactions on Knowledge and Data Engineering*, 1990, in press.

[12] N. Leveson. Software Safety: What, Why and How. *Computing Surveys*, 2(18):125–164, June 1986.

[13] S. Lindell. Keyword Cluster Algorithm for Expert System Rule Bases. Technical Report SD-TR-87-36, The Aerospace Corporation, El Segundo, CA., June 1987.

[14] K. Lindenmayer, S. Vick, and D. Rosenthal. Maintaining an Expert System for the Hubble Space Telescope Ground Support. In *Proceedings, Goddard Conference on Space Applications of Artificial Intelligence and Robotics*, pages 1–13, May 1987.

[15] M. Mehrotra. Rule Groupings: A Software Engineering Approach towards Ver-

ification of Expert Systems. Technical Report NASA CR-4372, NASA Langley Research Center, Hampton, VA., May 1991.

[16] M. Mehrotra, C. Wild, and D. Rosca. A Software Engineering Approach Toward Validation of Knowledge-Based Systems. Technical report, ViGYAN SBIR Phase-I Final Report, Hampton, VA., August 1992.

[17] M. Mehrotra, C. Wild, and D. Rosca. Role of clustering analysis in the verification of expert systems. In *Notes for the AAAI-92 Workshop on Verification, Validation and Testing of Knowledge-Based Systems*, July 1992.

[18] E. Soloway, J. Bachant, and K. Jensen. Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rulebase. In *Proceedings of AAAI-87*, July 1987.

[19] C. Wild, J. Chen, and D. Eckhardt. Reasoning about Software Specifications: A Case Study. *Proceedings of AIAA Computers in Aerospace VII Conference*, pages 297–306, October 1989.

Information Management

An Application of Machine Learning to the Organization of Institutional Software Repositories

Sidney Bailin and Scott Henderson
CTA Incorporated
6116 Executive Boulevard, suite 800
Rockville Maryland, 20852
(301) 816-1218

Walt Truszkowski
NASA/Goddard Space Flight Center
Data Systems Technology Division
Greenbelt, Maryland 20771
(301) 286-8821

Abstract

Software reuse has become a major goal in the development of space systems, as a recent NASA-wide workshop on the subject made clear.[1] The Data Systems Technology Division of Goddard Space Flight Center has been working on tools and techniques for promoting reuse, in particular in the development of satellite ground support software. One of these tools is the Experiment in Libraries via Incremental Schemata and Cobweb (ElvisC). ElvisC applies machine learning to the problem of organizing a reusable software component library for efficient and reliable retrieval. In this paper we describe the background factors that have motivated this work, present the design of the system, and evaluate the results of its application.

1. Repositories vs. Architectures in Reuse

The work described in this paper concerns the self-organizing capability of a repository of reusable software components. The recent trend in software reuse support technology has been away from a dependence on unguided searching of component repositories, and towards a more model-based approach [D'Ippolito '89; Simos '91; Mark '92]. In this introduction we explain the difference between the two approaches, the role of the model-based approach at NASA/Goddard, and our reasons for focusing on repository organization.

The model-based approach to reuse emphasizes the development of generic architectures within a given application domain. A generic architecture shows how systems in the domain are typically composed of reusable components. The field is moving towards knowledge-based support for generic architectures, e.g., formalizing the constraints under which components can be combined and the commitments entailed by reusing specific components. The ultimate goal is to build new systems by selecting features or specifying parameter values, thus instantiating the generic architecture with little or no additional software development.

NASA/Goddard has a long and successful history of using generic architectures, since even before the terminology became fashionable. The Standard Software System (SSS) [WEC '79]was developed in the late 1970s to provide a kernel of support for satellite operations control centers, and several missions were based upon it. The Multi-Satellite Operations Control Center Application Executive (MAE) [Sperry '85] was a successor to SSS in the mid-1980s, motivated in large part by the replacement of processor hardware and operating systems. Again, several missions were based upon MAE. Most recently, the Transportable Payload Operations Control Center (TPOCC) [Measday '91] was developed to take advantage of workstation technology and open system interconnection. The degree of parameterization and configurability has increased with each of these successive software architectures.

In a narrower domain—that of monitoring telemetry housekeeping data—the Generic Spacecraft Analyst Advisor (GenSAA) [Hughes '91] is generalizing an architecture for rule-based monitoring systems that was first illustrated in the Communications Link Expert Advisory Resource (CLEAR). GenSAA will provide a high degree of automated support in the creation of monitoring systems for different spacecraft.

---

[1] Organized by Langley Research Center and held at the Research Triangle Institute in April, 1992.

The Flight Dynamics Division of Goddard is developing the Combined Operational Mission Planning and Attitude Support System (COMPASS), which is an Ada-generic framework for creating orbit analysis systems for different spacecraft. The role for such generic architectures in the development of satellite ground software seems to be firmly established.

The need for specialized software has not disappeared, however. In order to meet the ever expanding requirements of the scientific community with available resources, new technologies must continually be inserted into ground support systems. Direct-manipulation user interfaces and knowledge-based systems are typical of the kinds of technology being introduced into control centers.

The Data Systems Technology Division has as its charter the development of new technologies to the point where they can be safely integrated into a control center. In such an environment there is a greater need to respond to unprecedented requirements, and less of a role for generic application architectures. The goal of reuse-based software development remains, however. Thus the emphasis in this group is on reuse within *horizontal domains*: these are the supporting domains that provide basic application-independent services, upon which vertical (i.e., application) domains are built (for example, databases, communications, and user interfaces are horizontal domains).

The Data Systems Technology Division's Code 522 develops a substantial portion of their software using the programming language C++. Each software project spins off a set of potentially reusable modules in the form of C++ component classes. These components implement, for the most part, application-independent functions. Because the components are produced as they are needed, their capabilities tend to be scattered over several horizontal domains. In order for developers to take full advantage of this collection—including developers who may not have contributed components—some rational method of organizing the library is required. A rational organization is one that allows developers to find what they need efficiently and reliably.

A key consideration in this challenge is that a given organization will not suffice as the contents of the repository evolve. As requirements evolve so do solution techniques. The contents of the library will change as new reusable components are developed—also as older components are removed, because their usefulness has diminished or was overestimated to begin with. As the contents of the library change, we cannot expect the same organization to remain appropriate.

Why not use the "given" hierarchy of C++ classes, which is implied by their public inheritance structure, to organize the repository? Public inheritance in C++ is supposed to represent the "is-a" relation [Meyers '91, Cargill '91]. In principle, a global inheritance hierarchy could function as the organization of the repository, and new components could be created within this inheritance framework. Our conclusion, however, is that this approach is not pragmatic. Design teams elaborate local inheritance hierarchies within a project or segment of a project to achieve maintainable code. A typical project may include a dozen independent inheritance hierarchies. The additional levels of abstraction required to relate these hierarchies to each other and to the organization's past work are typically missing because it would be expensive to develop and no additional functionality would be gained by this extra effort. When multiple individuals and multiple organizations are involved in a project, irreconcilable hierarchies may be developed to support similar functions. Adjusting source code to retrofit it into a unified inheritance hierarchy may not be economically feasible or even possible (in the case of object-code libraries). Finally, we want the organization of the repository to be malleable—to evolve as the characteristics of the components evolve. We would like to achieve such flexibility without having to revamp the C++ class definitions continually.

Thus we have been led to consider a method for incrementally defining categories of components. In the machine learning literature, such a method is characterized as an *unsupervised incremental* learning algorithm: unsupervised, because the set of available categories is not given to the algorithm *a priori*, but is developed instead by the algorithm itself; incremental, because the set of categories, as well as their boundaries, can change every time a component is added or removed from the repository [Fisher and Pazzani 91]. Unsupervised incremental learning is sometimes known as *concept formation*. ElvisC employs a concept formation algorithm called Cobweb to perform automatic classification of C++ components. A slightly different version of the algorithm is used to retrieve components that best match a given query.

2. The ElvisC Repository

ElvisC attempts to provide an organized repository with minimal construction and maintenance costs to its user community. It does so by assisting in the formal characterization of repository submissions, and automatically organizing the repository contents to reduce searching. ElvisC is composed of three major components: 1) a case base of submitted assets, 2) an extensible asset characterization language similar to that first demonstrated in TEIRESIAS [Davis '82], and 3) an unsupervised classification system derived from the Cobweb inductive concept formation system [Fisher '87].

One novelty of this work is its use of an unsupervised classification system to solve the "distance metric" problem of case-based systems: determining the closest match to a given problem specification within the existing case base when a perfect match does not exist. It does this by treating the problem specification as if it were the characterization of a new case to be classified: the bin into which this problem characterization would have been placed is the bin that contains the closest matches to the new problem. The advantage of this classification technique is that it requires no specific knowledge of the application domain beyond the n-dimensional shape[2] of the existing case base. Thus the repository is both self-organizing and ontologically open.

2.1 Acquisition and The Feature Space

Characterization of assets serves two purposes: it describes the asset for potential reusers, and it provides the basic data with which to organize the library for reuse. To assist in the organization of the library, the characterization must be able to differentiate this asset from dissimilar assets and to relate this asset to similar assets. Thus characterizations must draw from a common vocabulary which can be extended as needed to distinguish new cases from existing cases. To meet these demands ElvisC acquires its characterization language interactively from users when they submit assets. This technique was first demonstrated in TEIRESIAS [Davis and Lenat '82] for the acquisition of diagnostic rules for blood diseases.

The asset characterization language, or *feature space*, is organized as a tree of keyword schemata. This tree is rooted in the *Feature* feature. The children of *Feature* are top level characterizations of an artifact, for instance the asset's function or the environment in which the asset was defined. These children may have children themselves, each of which represents a further refinement of its parent's concept. The entire feature space below the 'Feature' feature is malleable and can be extended or edited by contributing authors. Modifications to the feature space are tracked by the interface so that old definitions can be repaired to fit the new space.

Each schema specifies the keyword name, a textual definition of the keyword, an indication of how the keyword can be refined, and prompts for selecting from existing child features or soliciting new child features. The *Feature* feature is defined as follows:

> name: "Feature"
> definition: "This is the mother of all features."
> refinement type: Alternatives
> solicitation prompt: "What new feature would you like to add?"
> child selection prompt: "What features characterize this asset?"

When a user interacts with this schema they are presented with the child selection prompt, followed by a list of the refinements currently available for refining the *Feature* feature. If the user enters the name of one of the refinements presented, they then interact with the schema for that refinement. If the user enters a name that is not on the list of existing refinements, they enter a dialog for instantiating a keyword schema as a new refinement for the *Feature* feature.

---

[2] The *dimensions* of the space are the attributes by which the cases are classified; the *shape* is determined by the relative frequencies of cases along these dimensions.

The refinement type for the *Feature* feature is *Alternatives*. This refinement type allows the selection of one or more children in the description of an asset or the specification of a query. Alternative refinements of the root feature might be *Feature* ⇒ *Language* and *Feature* ⇒ *Performance*. An asset may be characterized by either or both of these alternatives. The *Alternatives* refinement type can be thought of as an inclusive "or" in the refinement tree. The existence of alternative refinements in the feature space is what differentiates our approach from those that use the feature space as a decision tree for directly sorting assets into asset type categories.

A second refinement type is *Options*. This method of refinement allows the selection of one and only one refinement. Subsequent selection of another refinement deselects previous refinements. The features *C* and *C++* could be options for the refinement of *Feature* ⇒ *Language* . This type of refinement can be thought of as an exclusive "or" in the refinement tree.

The third refinement type is *Arguments*. This method of refinement indicates that the refinements of a feature are jointly required for the feature to be present in a characterization. For instance, if one of the options for *Feature* ⇒ *Function* was *Storage*, we might specify two arguments for *Storage*: *What* (what is being stored) and *Where* (where is it being stored). Thus an asset characterized by the storage keyword would possess *Feature* ⇒ *Function* ⇒ *Storage* ⇒ *What* ⇒ ... and *Feature* ⇒ *Function* ⇒ *Storage* ⇒ *Where* ⇒ ... attributes. This type of refinement can be thought of as an "and" in the refinement tree.

Finally, a feature may not have any type of refinement. Such a feature is said to be "terminal". In the language example above, *C* and *C++* are terminal since no further refinement of these features is deemed necessary. If at some later point it became important to distinguish *ANSI C* and *K&R C*[3], then the previously terminal *C* feature token could be refined to provide these new options.

An asset is characterized by selecting or adding progressively more detailed refinements from the feature space. The description of an asset is equivalent to the set of nodes from the feature space that were traversed during asset definition. The query specification process works identically.

2.2 The Solution Space

The solution space is composed of a case base of assets and a concept hierarchy of asset types. The asset case base is organized as a forest of inheritance trees to allow derived class assets to inherit, optionally, the features of their parent class assets, thereby simplifying asset characterization. Assets that have no parent class specified, or assets which are written in a non-object-oriented language, are treated as base class assets and stored as children of a root asset. Thus in practice the asset case base tends to be top-heavy rather than deep.

Each asset record in the hierarchy contains that asset's features (as described above) and housekeeping data. Housekeeping data encompasses all information that is required for the asset but that does not contribute to its classification. Examples of housekeeping data include the asset's name, location, and author. Asset names are guaranteed to be unique so that they can serve as identifiers in the hierarchy. The asset's location is intended to be an accessible path name for the source or object files constituting the asset. In the future we hope to use this path name to support automatic delivery of selected artifacts to clients of the repository.

2.3 Organization of the Solution Space

Earlier versions of ElvisC (then called Elvis) viewed a query to the case base as constraints on which existing cases could be retrieved. If no existing cases satisfied all the constraints, Elvis had to selectively relax some or all of those constraints until a match was found. The problem was knowing which constraints to relax.

---

[3] ANSI C refers to the American National Standards Institute's definition of the C language. K&R C refers to the definition of the C language published by Kernighan and Ritchie prior to the ANSI work.

Suppose that a user is looking for an asset written in "C" which functions as a sorted collection of pointers, any of which can be located on average in O(log n) time. The features for this request would be:

Feature ⇒ *Language* ⇒ *C*

Feature ⇒ *Function* ⇒ *Collection* ⇒ *Sorted*

Feature ⇒ *Performance* ⇒ *Average case* ⇒ *O(log n)*

Suppose that no asset in the case base fulfills all these criteria. Relaxing the language constraint one level means that any language can be used. Relaxing the function constraint means that a collection of any type can be used. Relaxing the performance constraint means that any level of performance is adequate. The combined result of all of these individual relaxations would be the union of a number of possibly disjoint sets of assets. If we could order the relaxation options *a priori* based on their expected impact on the appropriateness of retrieved assets, we could successively apply those options until we got a set of assets of reasonable size and appropriateness. Otherwise we could end up with a very large set of assets, and no clue of how to order them for appropriateness other than Hamming Distance (the number of matching attributes). The uninformed constraint relaxation approach is equivalent to a brute force exhaustive search in the feature space, beginning with the specified constraints, in order to find exemplars in the asset space. The contents of the case base provide no insight into how to direct this search other than knowing when it is done.

ElvisC uses the Cobweb inductive concept formation system to locate the closest match to a desired case. Cobweb is an unsupervised incremental concept formation system originally developed by Douglas Fisher in his doctoral work at the University of California, Irvine [Fisher '87] . A thorough description of the algorithm and its Lisp implementation in Cobweb/3 is available in [McKusick and Thompson '90]. This method uses the contents of the solution space to guide the search. As a result, smaller sets of closest matches are retrieved, and a method for rank ordering the appropriateness of the retrieved cases is available. The Cobweb algorithm appears to perform in O(log *n*) time for classification, where *n* is the number of cases in the the case base. Since the size of the case base grows slowly and probably never exceeds thousands of cases, this performance should be adequate. Furthermore, the accuracy of an inductive method should grow as the case base grows.

2.3.1 Cobweb

Cobweb takes a stream of object identifiers and their descriptions, and incrementally organizes them into a concept hierarchy. As one proceeds deeper into the hierarchy the concepts formed become more and more specific until one reaches the leaves of the hierarchy in which a concept describes the attributes of a single object. Cobweb uses relative frequencies of attributes to construct the concepts, and uses those frequencies as probabilities when finding the best concepts (bins) to house the next object seen. Thus Cobweb does not employ any domain specific heuristics to do its classification, nor does it resort to human supervision in its construction or use of these bins.

ElvisC uses Cobweb to determine the closest matches to a prospective re-user's stated requirements. We treat the features comprising the requirement as the specification of a hypothetical object to be classified. Cobweb then filters the new hypothetical object down its classification hierarchy with the additional constraint that no new bins can be created to house the concept. The bin in which this problem characterization would have been placed (the host bin) is the bin which contains the closest match to the new problem. The bin which contains the host bin (the super-host bin) contains the next closest matches for the stated requirement. ElvisC currently returns the contents of these two bins in response to a query, with the asset from the host bin at the front of the list.

2.3.2 How Cobweb Works

Cobweb uses a metric called Category Utility (CU) to determine the relative goodness of different placements of a new submission in the current case base. The version of category utility used in this implementation is:

$$\text{CategoryUtility} = \{ \sum_{k=1..nk} P(C_k) * [\sum_{i=1..ni} \sum_{j=1..nj_i} P(A_i = V_{ji} \mid C_k)^2] - [\sum_{i=1..ni} \sum_{j=1..nj_i} P(A_i = V_{ji})^2] \} + nk$$

where:

nk is the number of categories at the current level of the hierarchy
$P(C_k)$ is the probability of membership in category k relative to all other categories 1..nk
ni is the number of possible attributes
nj_i is the number of possible values for the ith attribute
$P(A_i = V_{ji} \mid C_k)$ is the probability that an attribute i has the value j given membership in category k
$P(A_i = V_{ji})$ is the probability that an attribute i has the value j for all categories 1..nk

This metric was first proposed and discussed in detail in [Gluck and Corter '85].

Cobweb categorizes a new submission to the library using a recursive algorithm. The algorithm begins at the root of the classification hierarchy which represents the bin containing all examples. At this root node the algorithm performs as follows:

1. Add the example as an exemplar of the node, updating the node's value frequencies according to the example's description.

2. If the node has no exemplars other than this new exemplar, terminate (classification is complete).

3. If the node now has two exemplars, create two child nodes (bins) to hold each of the exemplars and terminate (classification is now complete).

4. If the node now has more than two exemplars, it must have two or more child nodes. The options for placement of the new example are:

   4a: Create a new child (sub-bin) which will hold the example as its sole exemplar.
   4b: Place the exemplar in the existing child (sub-bin) which best fits the exemplar.
   4c: Create a new child (sub-bin) which will hold the exemplar by merging the two existing children which best fit the exemplar.
   4d: Replace the existing child which best fits the exemplar by its children, and then place the exemplar in one of those new children based on where it fits best.

   Each option is tried individually, with the results of option 4b used to identify the children to merge in option 4c and the child to split in option 4d. The category utility score is computed for the node after an option is performed, and then the results of the option are undone. After all options have been tried, the option which resulted in the best category utility score is selected. The algorithm, beginning at step 1, is then recursively applied to the selected sub-bin.

The $P(C_k)$ multiplier in the category utility function has the effect of limiting the number of children of a node to a range between two and some small constant (in our experience less than a dozen). Since this algorithm never considers more than the immediate children of the current node, and then selects at most one of those children for further classification, the performance of the algorithm is roughly O(log n).

2.3.3 Modifications to Cobweb

Cobweb has been applied to domains where the number of attributes is known *a priori* , and every exemplar is described by values for every attribute. In our reuse library, the attribute space is growing and exemplars are described only by a subset of that space.

The first problem is easy to solve. When our version of Cobweb is confronted with a previously unseen attribute, it updates all existing artifact descriptions with an "unknown" value for the new attribute.

The second problem is more fundamental. An attribute such as *Performance => O(1)* could be missing because the artifact was inadequately described, or because the attribute does not apply to the artifact (e.g., the artifact is a representation for calender dates, and thus a characterization of algorithm performance is not appropriate). The default implementation of CU and Cobweb will determine that two descriptions match on an attribute if both have a value of "unknown" for that attribute. This is correct for the latter case where the attribute is not appropriate to the definition of the artifacts. A modification to the calculation of CU is possible such that an unknown value for an attribute within an artifact description is treated as unique to this asset description, thus matching no other descriptions with a value of "unknown" for that attribute. This is correct for the former case where the artifact was inadequately described. Operationally, the use of unique unknowns causes a bushy classification hierarchy with a large number of bins holding a small number of exemplars. Many artifacts which we would have expected to be present in the same bin, at some level of the hierarchy, are instead only collectively present in the root node of the hierarchy. If we use non-unique unknowns these artifacts do tend to be present in the same bin at a level of the hierarchy below the root node, thus showing their similarity at that level of abstraction.

In ElvisC we treat "unknown" values for attributes as significant (non-unique) during the classification of contributed assets, but downplay the importance of unknown values during retrieval (treat them as unique). This decision is based on the assumption that retrieval requests are intentionally loosely described, but that asset descriptions are not. Since unknowns are treated as unique during retrieval, traversal of the classification hierarchy is guided more by attributes with known values than those with unknown values. Since during retrieval no actual modifications are made to the classification hierarchy, the deeper form caused by the normal (non-unique) algorithm is preserved. Finally, since the retrieval algorithm returns the one exemplar from the leaf of the classification hierarchy which best matched the request, followed by the exemplars of the parent of that leaf, a smaller and more focused set of closest matches is returned by the deeper hierarchy.

2.3.4 Archetypes and Prototypes

Cobweb has been augmented in ElvisC to keep two *synthetic artifact descriptions* at every classification bin. The *prototype description* records for each attribute the most probable value for that attribute given the contents of the bin. The *archetype description* records for each attribute the most probable value for that attribute other than the "unknown" value if there exist exemplars in the bin which have known values for that attribute.

We hope to use the archetype descriptions to assist the submitted artifact description process in guaranteeing that an artifact description is adequately specified. Once an author has made a preliminary characterization of his or her artifact, the interface will do a preliminary classification to identify an appropriate archetype. The interface will then use archetype attributes with known values which correspond to unknown values in the proffered artifact description to prompt the author for additional information to complete the description.

We hope to use the prototype descriptions as the basis for functionality-based asset browsing, an alternative to the inheritance-based asset browsing currently employed.

3. An Experiment

ElvisC is, as its full name implies, an experiment. The hypothesis is that the Cobweb algorithm can organize software components for efficient and reliable retrieval of closest matches without human supervision, and without the direct encoding of closeness information. The experimental apparatus is the ElvisC implementation itself, augmented by methods for quantifying its performance in an attempt to refute the hypothesis. The experimental subjects are the potential users of the system, and their collective solutions and problems.

There are two types of errors that could refute the hypothesis. The first type is errors of omission, where an applicable case is not retrieved in response to a query. These errors could be quantified by recording examples of retrieval requests, and comparing the responses of ElvisC to the responses of a human librarian who is familiar with the case base. Errors of omission are inversely related to the reliability of retrieval. The second type is errors of inclusion, where patently irrelevant cases are returned as the

primary candidate solutions. Such errors could also be quantified by comparing the responses of a human librarian to those of the system. Errors of inclusion are inversely related to retrieval efficiency. Of the two types of errors, errors of omission would provide a more serious challenge to the hypothesis.

In this experiment outside subjects were not available. Instead we evaluated an automatically generated classification hierarchy directly, looking for characteristics that would lead to the two types of errors. To minimize errors of omission, the classification must co-locate (at some level) assets that solve the same problem, rather than scatter them about the hierarchy. To minimize errors of inclusion, the classification must be deep enough to effectively partition small numbers of solutions. In a hierarchical classification scheme these two characteristics of good organization are themselves related: the higher the level at which similar assets are co-located, the larger the number of assets that must be inspected in detail to identify a best candidate (and hence the lower the efficiency of the overall retrieval process).

In [Bewtra and Lide '92], investigators manually characterized and classified a set of 56 source code components from a Code 522 reuse directory. In this experiment, we entered the same classes into ElvisC using the feature descriptions provided in that paper[4]. The fifty-six classes were added to eleven classes from the National Institute of Health C++ class library, which had been previously entered into ElvisC, so that in total 67 classes were described and classified. The resulting classification hierarchy was printed out and evaluated in order to determine its support for efficient and reliable retrieval of artifacts by an unfamiliar user. The description language which was developed while entering these classes and the classification hierarchy that resulted are presented in detail in [Henderson '92].

4. Results

Figure 1 presents the classification derived manually in [Bewtra and Lide '92]. Dotted boxes denote categories of C++ classes that were treated in that paper but were not included in this experiment because of time limitations. The manual classification was constructed without any knowledge of the results (indeed, the existence) of this experiment. Conversely the experiment was performed without access to the results of the manual classification.

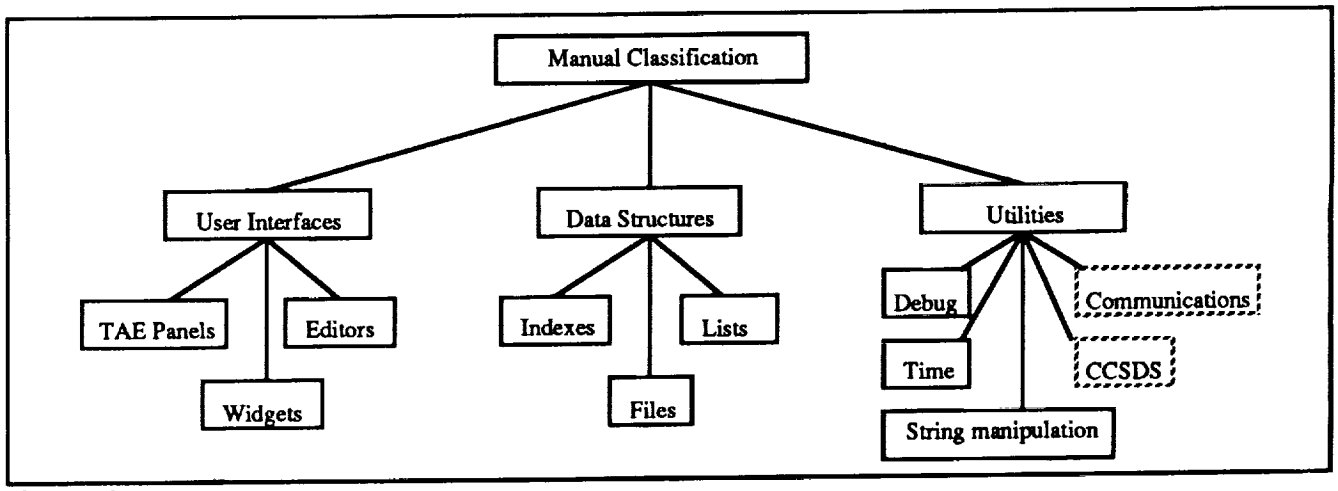


Figure 1.

It is interesting to note that this manually derived hierarchy went through several revisions during the writing of [Bewtra and Lide '92]. Since there is no performance task associated with this hierarchy, its construction is the result of subjective decisions on how to partition the C++ classes that were evaluated. It is reasonable to assume that these decisions were influenced by the authors' contact with a large number of software modules that were not part of that evaluation set.

---

[4]In some cases the provided descriptions were very terse, and several descriptions referred to base classes that were not described elsewhere in the paper. Thus we were not always able to provide distinctive or complete descriptions for each asset, and this may have diminished the effectiveness of the automatic classification.

Figure 2 presents the top levels of the automatically generated classification hierarchy. The nodes in this tree will be called *concepts* to differentiate them from the nodes in the manual classification hierarchy which we will call *categories*. Since ElvisC does not currently generate titles or descriptions for concepts, we have entered titles based on examination of the concept exemplars and characteristic features. Characteristic features were identified by a combination of two statistics. The first statistic is the probability that an exemplar of the concept has the feature. The second is the probability that an asset which has that feature is an exemplar of the concept. To be a characteristic feature for a concept, both of these probabilities had to be above 50%[5].

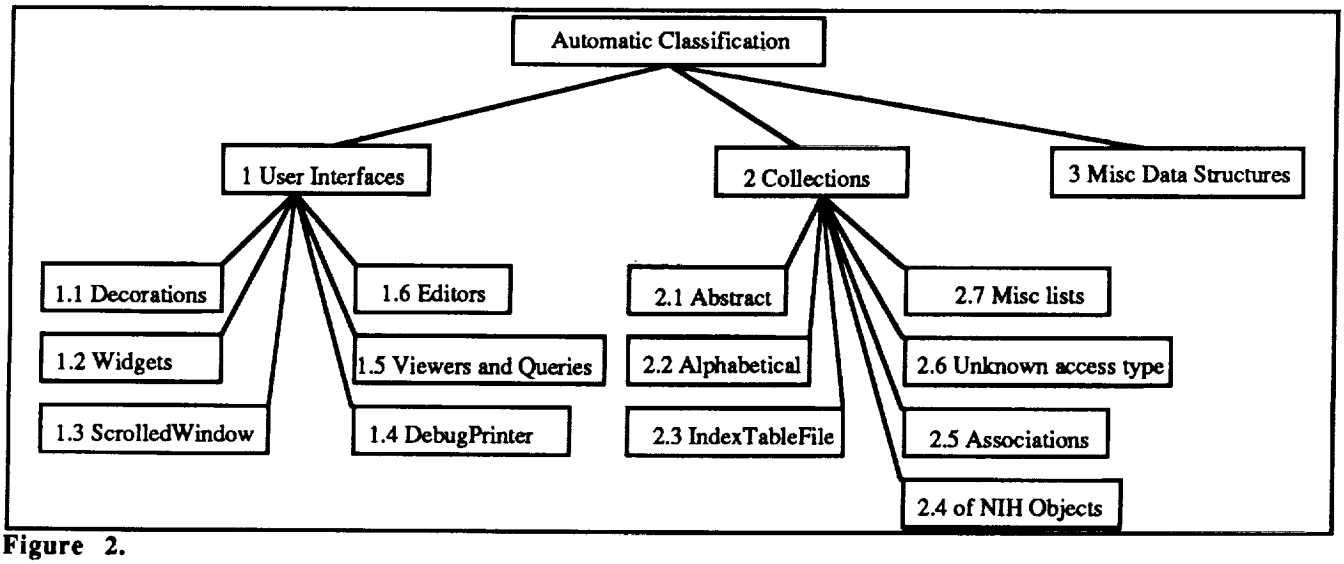


Figure 2.

The top level of the automatically generated classification hierarchy contains three concepts. Examination of the characteristic features reveals that they represent user interfaces, lists, and data structures respectively. These top level concepts are then further divided into sub-concepts that have been labeled in the diagram based on their contents and characteristic features.

4.1 Description of the Cobweb Classification Hierarchy

Concept 1 contains twenty five user interface classes. There are no user interface assets contained in any of the other top level concepts, so the retrieval of user interface components will be reliable at this level. The first sub-concept of concept 1, labeled Decorations, includes two widget classes which perform no function other than to provide visual feedback for a host window. Neither of these widget classes supports user input, and their collocation within this concept seems reasonable. The second sub-concept (1.2) also includes widget classes, but these classes allow user interaction. Next there are two singleton sub-concepts, one for the asset ScrolledWindow (1.3) and one for the asset DebugPrinter (1.4). Although DebugPrinter is a novel class in the library and, as such, deserves its own concept, ScrolledWindow seems subjectively to be a good fit with the Viewers and Query Dialogs concept. The placement of ScrolledWindow in a concept of its own could adversely effect the efficiency of retrieval. The next sub-concept (1.5) represents interaction windows which are not used for editing a file. The final sub-concept of concept 1 represents editors.

Concept 2 contains eighteen *collection* assets; code components that collect multiple instances of some type such as linked lists. There are no collection assets contained in any of the other top level concepts, so the retrieval of collections will be reliable at this level. The concept is further subdivided into sub-concepts of abstract base classes, alphabetically organized lists, lists of NIH Objects, associations, lists with unknown

[5] We found through experimentation that this threshold revealed features which both described the concept and differentiated it from its siblings.

access methods, and a concept for lists which presumably did not fit elsewhere. The clustering of assets is good for each lower level concept with two exceptions. First, NIHDictionary (a sub-concept of 2.4) and the elements of the Associations concept (2.5) both provide associations between a key and a value. A query leading to the associations concept will not turn up NIHDictionary and vice versa. This problem points to a weakness in constructing exclusive categories (categories that do not share elements with their siblings): at lower levels a decision must be made on how to discriminate classes, and that decision may not perfectly represent the relative importance of different features in the class descriptions. In this case, the classification decision appears to have been made based on the types of values collected: pointers to NIH objects for concept 2.4 and untyped pointers for concept 2.5. The impact of these decisions could be reduced through the use of multi-branch search within the classification tree.

Concept 3 contains every asset which is not an interface or a collection. The sub-concepts within concept 3 are not well formed, having few if any characteristic features to distinguish them. Several appropriate clusters seem to be forming, such as iterators and date/time representations, but overall these sub-concepts seem to be in a very nascent state. This could be symptomatic of poor descriptions, or of a lack in regularity within this subdomain.

4.2 Reliability Evaluation of the Cobweb Classification Hierarchy

Retrieval reliability is dependent on the system's ability to appropriately cluster similar assets within the repository. For the purposes of this evaluation we have presumed that the clustering presented in the manual classification is correct, although that organization has gone through multiple revisions and the assessment of its quality is somewhat subjective.

Table 1 shows how the assets within the automatically generated concept hierarchy fall within the manual hierarchy for the *User Interface* category. The column headings represent the sub-categories of *User Interface* within the manual classification. The row headings represent the sub-concepts of *User Interface* for the automatic classification, followed by the total number of assets identified with that concept in parentheses. Numbers within the cells of the table indicate how the assets from a concept fall into the manual categories.

Generated Concepts	Panels	Widgets	Editors	Non user interface
Decorations (2)		2		
Widgets (5)		5		
Viewers (12)	9	1	2	
Editors (4)			4	
Other UI (2)		1		1

Table 1.

Every asset in the *User Interface* category was identified within the *User Interface* concept, so reliability for this concept was 100%. One additional asset was included in the *User Interface* concept which was not included in the *User Interface* category, so the match at the *User Interface* level was 24 + 25 = 96%.

Although the sub-concepts formed within the *User Interface* concept do not match the manual partition of the *User Interface* category, a mapping between them is evident from inspection of the data and from the statistically generated concept descriptions. First, the *Decorations* and *Widgets* concepts represent a partition of the manually generated *Widget* category. Second, the *Viewers* concept is very similar to the manual *Panels* category. Given this mapping, correct classification inside the User Interface Concept is 20 + 25 = 80%.

Table 2 shows how assets within the *Collections* concept fall within the partitions of the *Data Structures* and *Utilities* categories. Within the manual classification most of these assets inhabit multiple categories. These assets are arbitrarily labeled by letters to indicate how a member of a concept falls into the manual categories.

Generated Concepts	Indexes	Files	Lists	Time	String	Debug
Abstract (2)			a, b			
Alphabetical (2)	c	c	d		d	
Linked Lists (3)			e, f, g		g	
Misc Lists (2)			h, i	h	i	
Other (1)	k	k				

Table 2.

Every asset in the *Lists* partition of the *Data Structures* category was identified by the *Collections* concept, so reliability for Lists was 100%. Two assets representative of the *Collections* concept were not part of the manual *Lists* category, so the match between the concept and the category was 8 + 10 = 80%. For completeness it should be noted that the *Associations* and *NIH List* concepts were not part of the manual classification and so were omitted from this discussion.

Table 3 shows how assets within the *Misc Data Structures* concept fall within the partitions of the *Lists* and *Utilities* categories. Some of these assets inhabit multiple categories, so letters are used to indicate how a member of a concept falls into the manual categories.

Generated Concepts	Indexes	Files	Lists	Time	String	Debug
Reporters (2)		a				b
Other (8)	a	a,b,c,d	e	f, g	h	
Iterators (1)			k			
Dates (3)				m, n, o		

Table 3.

We can compute the reliability of the *Misc Data Structures* concept by comparing it to the partitions of the *Data Structures* and *Utilities* categories except for the *Lists* partition of those categories, which should have been accounted for by the *Collections* concept. Given this interpretation the reliability for the *Misc Data Structures* concept was 12 + 14 ≈ 86%.

The last two tables show that the categories which straddle concepts are *Indexes*, *Files*, *Time*, and *String*. Reliability problems with Time and String retrieval cease if the requestor indicates that a collection of these types is desired. Retrieval reliability for assets manually classified within the Indexes and Files categories is not good, and reflects the nascent state of the associated concepts within the automatic classification.

One cause for the divergence between the manually generated and automatically generated hierarchies is a difference in the definition of the classification process. Cobweb creates exclusive concepts (concepts that do not share elements with their siblings). The manual classification generated in [Bewtra and Lide '92] includes assets in multiple categories. Cross-referencing enhances the reliability of retrieval at the expense of efficiency of retrieval. Thompson [Thompson '92] has suggested that a more sophisticated "beam" search in response to queries would enhance the reliability of retrieval with a fixed decrease in retrieval efficiency. This may represent an alternative to non-exclusive catagories. Another possible solution would be to use a classification algorithm that assigns assets with varying probabilities to different categories, such as AutoClass [Cheeseman '88].

4.3 Efficiency Evaluation of the Cobweb Classification Hierarchy

Retrieval efficiency is inversely related to the number of matches returned for a search request, thus the maximum efficiency that could be attained is 1.0. Since the manually constructed classification hierarchy is only two levels deep, efficiency is limited by the average size of a second tier partition. For the assets considered in this experiment the average size of a second tier partition is 6.67 elements, and so the average efficiency would be 1 + 6.67 or approximately 0.15.

Cobweb produces a concept hierarchy whose leaves contain a single exemplar. Thus if the generated concept hierarchy was perfect, the average number of returned cases for a request would be one! Our data for the hierarchy that was actually constructed suggests that this strategy would be too unreliable. If we were to restrict retrieval to second tier concepts we would attain about 80% reliability. The average size of a retrieved set would then be 3.43 elements, and the average efficiency would be approximately 0.29.

What this suggests is that an automatically constructed but manually corrected concept hierarchy would attain higher efficiency with less effort than a purely manually constructed hierarchy, while still retaining the same reliability.

5. Summary

At its upper level the automatic classification performed by ElvisC is of high quality and fulfills our criteria for reliable retrieval. For the first two categories this claim can be extended to lower level subcategories, as can the claim for efficient retrieval. The last category, however, demonstrates the frailty of weak-theory systems: all that the algorithm has to go on are the descriptions given it of the sixty-seven classes, and specifically of the twenty-four assets that fall into this category. Cobweb's strength lies in its ability to exploit the fact that the world is sparsely populated from the set of things that *could* be. The algorithm is not effective until enough cases have been seen and the regularity in how attributes co-exist in objects becomes evident.

The degree of correlation between the manually constructed classification and the automatically generated classification is remarkable considering that the automatic algorithm was unsupervised and so made its own decisions on the kinds of concepts to form and then on how to further partition those concepts. The results show that, with a relatively small number of samples, a good classification can be arrived at without human supervision or embedded knowledge of the domain.

This technique allows the reuse of work products by a large group of participating individuals when a domain theory on how to organize the repository is not available. We hope to continue to accumulate evidence to support these findings, and thereby establish this novel application of concept formation.

References

Arango, G. and Teratsuji, E. (1987) *Notes on the Application of the Cobweb Clustering Function to the Identification of Patterns of Reuse*. Technical Report ASE-RTP-87. Advanced Software Engineering Project, Department of Information and Computer Science, University of California, Irvine, CA.

Basili, V. (1990) Viewing maintenance as reuse-oriented software development. *IEEE Software*, January 1990.

Basili, V. and Rombach, H.D. (1988) *Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment*. Technical Report UMIACS-TR-88-92, Computer Science Department, University of Maryland, College Park, MD. December 1988.

Berlin, L. (1990) When objects collide: experiences with reusing multiple class hierarchies. In *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages, and Applications/European Conference on Object-Oriented Programming*, pages 181-193. ACM Press, October 1990.

Bewtra, M., and Lide, D. (1992) *Code 522 Technology Component Analysis* Technical report to NASA Goddard Space Flight Center. CTA Inc. Rockville, MD.

Boland, D., Booth, E., Green, D., Odt, T. (1991) *Combined Operational Mission Planning and Attitute Support System (COMPASS) Operation Concepts*. NASA/Goddard Space Flight Center, Flight Dynamics Division (Code 550) report number 550-COMPASS-107. May 1991.

Brooks, F. (1987) No silver bullet: essence and accidents of software engineering. *IEEE Computer*, Vol. 20, No. 4. April 1987.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988) A Bayesian classification system. Proceedings of the Fifth International Conference on Machine Learning, pages 54-64. Morgan Kaufman Publishers, Ann Arbor, MI.

Cox, B. Planning the software industrial revolution. *IEEE Software*, Vol. 7, No. 6. November 1990.

Davis, R. and Lenat, D.B. (1982) *Knowledge-Based Systems in Artificial Intelligence* McGraw-Hill Inc.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990) Indexing by latent structure analysis. *Journal of the American Society for Information Sciences*, 41(6), pages 391-407.

D'Ippolito, R. Using models in software engineering. *Association for Computing Machinery*, September 1989, pages 256-261.

Fisher, D. and Pazzani, M. (1991) Models of concept learning. In *Concept Formation: Knowledge and Experience in Unsupervised Learning*, ed. D. Fisher, M. Pazzani, and P. Langley. Morgan Kaufman Publishers, San Mateo, CA.

Fisher, D. (1987) *Knowledge acquisition via incremental conceptual clustering* Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Gibbs, S., Tsichritzis, D., Casais, E., Nierstrasz, O., and Pintado, X. (1990) Class management for software communities. *Communications of the ACM*, Vol. 33, No. 9, pages 90-103. September, 1990.

Goldberg, A. (1984) *Smalltalk-80: The Interactive Programming Environment.* Addison Wesley Publishing Company, Reading MA.

Henderson, S. (1992) *Automated Software Component Classification using ElvisC: Results and Discussion.* Technical report to NASA Goddard Space Flight Center. CTA Inc. Rockville, MD.

Hughes, P. and Luczak, E. (1991) *The Generic Spacecraft Analyst Assistant (GenSAA): A Tool for Automating Spacecraft Monitoring with Expert Systems* 1991 Goddard Congerence on Space Applications of Artificial Intelligence, Greenbelt, Maryland.

Knight, J. (1992) *Issues in the Certification of Reusable Parts.* Technical Report TR-92-14, Department of Computer Science, University of Virginia.

Krone, J. (1988) *The Role of Verification in Software Reusability.* Ph.D. Thesis, Department of Computer and Information Science, Ohio State University. August 1988.

Maarek, Y. and Smadja, F. (1989) Full text indexing based on lexical relations. An application: software libraries. In *Proceedings of SIGIR '89*, pages 198-206, ACM Press, Cambridge MA.

Mark, W., Tyler, S., McGuire, J., and Schlossberg, J. (1992) Commitment -based software development. *IEEE Transactions on Software Engineering*, October 1992.

Meyer, B. (1988) *Object-oriented Software Construction.* Prentice Hall International Series in Computer Science, New York, NY.

McKusick, K., & Thompson, K. (1990). *Cobweb/3: A portable implementation.* Technical Report No. FIA-90-6-18-2. NASA Ames Research Center, Artificial Intelligence Research Branch. Moffett Field, CA.

Moore, J.M. and Bailin, S. (1991) Domain analysis: framework for reuse. In *Domain Analysis and Software Systems Modelling*, ed. R. Prieto-Diaz and G. Arango. IEEE Computer Society Press.

Neighbors, J. (1989) Draco: A method for engineering reusable software systems. In *Software Reusability*, ed. T. Biggerstaff and A. Perlis, Vol. 1, pages 295-319. Addison-Wesley Publishing Company.

Ostertag, E., Hendler, J., Prieto-Diaz, R., and Braun, C. (1992) Computing similarity in a reuse library system: an AI-based approach. *ACM Transactions on Software Engineering and Methodology*, Vol. 1, No. 3. July 1992.

HR (1989) *Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation.* Staff study by the Subcommittee on Investigations and Oversight, Committee on Science, Space and Technology. U.S. House of Representatives. August 3, 1989.

Measday, A. (1991). *Transportable Payload Operations Control Center (TPOCC) Implementation Guide for Release 3.* Technical report to NASA/Goddard Space Flight Center by Computer Sciences Corporation and Integral Systems, Inc. NASA/Goddard report number 511-4SSD/0291.

Prieto-Diaz, R. (1991) Implementing faceted classification for software reuse. *Communications of the ACM*, Vol. 34, No. 5. May 1991.

Prieto-Diaz, R. and Arango, G., ed. (1991) *Domain Analysis and Software Systems Modelling.* IEEE Computer Society Press.

Reich, Y. (1991a) *Building and Improving Design Systems: A Machine Learning Approach.* Ph.D. Thesis, Engineering Design Research Center, Department of Civil Engineering, Carnegie Mellon University, EDRC 02-16-91.

Reich, Y. (1991b) Constructive induction by incremental concept formation. In *Artificial Intelligence and Computer Vision*, ed. Y.A. Feldman and A. Bruckstein. Elsevier Science Publishers.

Reich, Y. and Fenves, S. (1991) The formation and use of abstract concepts in design. In *Concept Formation: Knowledge and Experience in Unsupervised Learning*, ed. D. Fisher, M. Pazzani, and P. Langley. Morgan Kaufman Publishers, San Mateo, CA.

Simos, M. (1991) The growing of an organon: a hybrid knowledge-based technology and methodology for software reuse. In *Domain Analysis and Software Systems Modelling*, ed. R. Prieto-Diaz and G. Arango. IEEE Computer Society Press.

Sperry (1985). *MSOCC Applications Executive (MAE) Software Requirements Specification.* Technical report to NASA/Goddard Space Flight Center. Sperry Corporation, November 1985.

SPC (1991) *Synthesis Guidebook.* Technical Report SPC-91122-MC, Software Productivity Consortium, Herndon, VA.

Thompson, K. (1992) Personal correspondence.

WEC (1979). *Control Center Standard Software Final Report.* Technical report to NASA/Goddard Space Flight Center. Westinghouse Electric Corporation. July 27, 1979.

Visualizing The Semantic Content of Large Text Databases Using Text Maps

N93-25982

Nathan Combs
TASC
Reading, MA 01867
ncombs@tasc.com or (617) 942–2000

Abstract

A methodology for generating text map representations of the semantic content of text databases is presented. Text maps provide a graphical metaphor for conceptualizing and visualizing the contents and data interrelationships of large text databases. Described are a set of experiments conducted against the TIPSTER corpora of Wall Street Journal articles. These experiments provide an introduction to current work in the representation and visualization of documents by way of their semantic content.

Introduction

This paper presents a methodology for deriving text-map representations of large text databases. Text maps are useful graphical metaphors that can aid users in visualizing the semantic contents of large text databases. The text map graphical metaphor relates to Artificial Intelligence (AI) research in two substantive ways: first, text maps are often generated using "neurally inspired" computational paradigms; second, they can be effective tools for relating at a high level with the complex knowledge structures generated by AI systems. It is this capacity as a vehicle for communicating intuitions about large quantities of highly interconnected knowledge that is of primary interest. This kind of capability is seen as relevant to NASA activities in the areas of text processing (Driscoll et al. 1992), information retrieval (Rorvig 1991), and knowledge understanding. Specific applications would include searching and navigating amongst the contents of regulatory document databases as well as technical and scientific document collections. Additionally, text-based retrieval methods can be generalized to other information domains. Thus, for example, databases that link textual information to geographic and image objects can make use of textual features to organize and access these objects (e.g., Carlotto 1992). Using textual information as a key data source can be useful for describing a range of data for two reasons: first, textual information tends to be abstract — which facilitates high level data classification; and second, textual descriptions are expressive — and are thus robust across applications and data types.

Also introduced in this paper are preliminary results of a set of text-map experiments conducted against the TIPSTER corpora of Wall Street Journal articles. These experiments made use of a simple statistical text pre-processor. More relevant to this paper, however, is the follow-on discussion describing how the simple experimental system is currently being extended with a semantic-based text preprocessor.

The discussion in this paper is framed conceptually in terms of two levels of understanding: a micro-scale level of understanding which is concerned with finding the "gist" of the semantic content of individual documents; and a macro-scale level of

understanding which is concerned with integrating the mosaic of individual document interpretations into a larger meaning. Abstracting from the micro scale to the macro scale is the function of such tools such as inference (rule) generators, database knowledge "mining" techniques, and visualization maps. It is the latter technique that is proposed by this paper.

The proposed visualization approach is of general relevance to the NASA community working with "vector-product" information retrieval systems (Rorvig 1991). Furthermore, this work is of specific relevance to NASA projects in text browsing and retrieval such as Kennedy Space Center's QA system (Driscoll et al., 1992) and other work conducted in the Astrophysics Data Facility at Goddard. As a knowledge abstraction technique, text maps generate simple topographic knowledge structures for interpreting the contents of a database. From these structures, users infer associations and similarities amongst database items.

Visualization

There are two challenges confronting text processing systems working with large document databases: how to extract meaning from documents; and how to integrate and represent this extracted meaning. Browsing large text databases whose contents are not fully characterized, or whose contents are subject to change, requires information abstraction tools. Advocated here is an approach that integrates information about the meaning of multiple documents into a single gestalt which can be graphically and intuitively conveyed to a user. The contrast between this approach to information abstraction and other artificial intelligence methods is made in a later section.

With text database understanding, there are two separate but related concerns:

1.) How can the aggregated output of a text preprocessor be concisely presented?

2.) What kind of higher abstraction can be used to express the interrelationships of documents?

One method for representing document classifications is by the RANKED LIST. With the ranked list the documents in the database are linearly ordered according to how well they match a target set of concepts. The more relevant a document is to a target set of concepts, the higher in the list it is positioned. How a document relates with other documents with regard to a specific set of criteria can be communicated by this list. One example of a ranked-list system that allowed users to select database objects whose descriptions best fit a user query is given by Rorvig (1991). With this system, if the retrieved objects do not match the query, a user can extend the search and look at objects which are "like" the best matching objects found in the list (relevance feedback).

One drawback of the ranked list representation is that the concepts that are being searched for need to be known before the search is implemented. In circumstances where the significant concepts or terminology are not well understood, a ranked linear representation can be restrictive: it does not easily communicate how documents differ from a query, and to what extent. Thus, with Rorvig's example, the results of the relevance feedback are not integrated into a single representation which communicates the relationship of the queries with the contents of the database. Instead, what is presented is a series of disparate "snapshots" of the database as it is evaluated against an evolving query.

Documents can also be represented hierarchically using dendrograms, or trees (a product of single-link clustering, for example). Each leaf in the tree denotes a document. The tree depicts how the documents are incrementally aggregated into ever larger groups: links connect documents or groups of documents to their nearest neighbors. Hierarchical structures are interpreted visually by sequentially traversing their component links: relationships are identified by paths through the cluster hierarchy.

An alternative display for hierarchical document structures that de-emphasizes their sequential interpretation has been proposed by Schneiderman (1991) in his work with tree-maps. With tree-maps, database objects are denoted by surface-filled, color-coded rectangles. Rectangles are colored to show the object type, and the rectangle areas indicate how relevant that object is to its type. While tree-maps are easier to grasp visually than a sprawling tree structure, they, like dendrograms, do not easily communicate how arbitrary documents are related.

The text map is proposed here as an alternative to both the hierarchic structure and the ranked list. This representation provides a comprehensive picture of all documents in a database, unlike the ranked list, and is preferred to hierarchical cluster representations because of its intuitive use of the two dimensional viewing surface.

While hierarchic structures can serve a useful role in facilitating database search and retrieval and thus may underlie the data organization of any representation (van Rijsbergen 1976), for many text database comprehension tasks, hierarchical document displays may be counter-intuitive:

1.) hierarchies require sequential interpretation.

2.) hierarchies restrict comparisons between documents.

3.) re-balancing hierarchic structures with new documents can cause dramatic changes to the structure.

A more basic complaint, however, relates to the hierarchic classification methodology itself: because the implicit goal of this method is to partition data into disjoint sets, it cannot easily represent structures derived from statistical distributions (Kohonen 1982). When viewing document distributions, how the parts relate to the distribution carries meaning. It is this connectivity between the documents that is undermined by hierarchic representations: rather than emphasizing how the parts are connected to the whole, what is emphasized is how the whole is decomposed into ever shrinking sets.

In contrast, the text map approach assumes that the interrelationships between documents are significant and representable. It is meant to provide a macro perspective of the database that is intuitive as well as abstract. Because it avoids representing the semantics of the extracted text, it is a visualization method that can be used across a variety of applications and databases.

Examples of vector-based systems that use visualization maps to represent database information analogous to the approach described here include Carlotto (1992) and Chang (1990).

Text maps

Text maps are forwarded here as a visual metaphor for graphically communicating the taxonomy of the contents of large text databases. Using text-maps, documents are classified contextually with associations between documents being implied by proximity. The text visualization (TEXTVIZ) approach assumes a vector representation of the meaning of documents: each vector encodes a set of features which characterize the content of each document. Vector components index individual document features and vector component values denote the pertinence of a feature to a particular document. As will be illustrated later, the actual semantic content of a feature is determined by the preprocessor. Preprocessors that use a semantic model of a domain can generate semantically meaningful features for that domain. Whereas preprocessors that use non-semantic models, e.g. statistical systems, the features correlate with other properties of the document such as word distribution.

Thus, for each document in the database, there exists a corresponding vector description of its content. These vectors, points in vector space, are then projected onto a two-dimensional text-map surface for display.

The TEXTVIZ procedure converts the information extracted about the content of each document into a numeric vector, a signature. Differences in meaning between documents is reflected by differences in their vector patterns. Thus, from a macro perspective, the organization of the database is inferred from observing how the signature patterns vary across all documents.

Ultimately, documents are presented to the user as points on a viewing surface or map; distances between points represent the difference in the estimated meaning of the documents. It is the relative similarity (dissimilarity) of the meaning of documents that is graphically represented by the text map. Thus, there are two levels of abstraction contained by the TEXTVIZ approach: first, meaning is abstracted from text using a text preprocessor; second, descriptions of all documents are aggregated, abstracted, and then displayed graphically. The abstraction process is accomplished by projecting the document signature vector into a two dimensional visual space (x and y coordinates on the text map).

Figure 1 provides a system overview of the text visualization process. Figure 2 is an example of a text map that was generated by a surrogate-coding system developed at TASC (Carlotto 1992). In this map, individual documents are represented by word labels.

Visualization and Artificial Intelligence.

The TEXTVIZ approach intersects disciplines in AI in two ways: first, neural network procedures, including the Kohonen self-organizing map (1984,1982) can be used to generate visualization (text) maps; second, visualization maps can serve as a powerful means for integrating the output of AI semantic reasoning processes. The semantic text processor that is introduced later in this paper serves as one example. Similarly, as a knowledge abstraction tool, the function of visualization maps parallels many database rule induction, data classification, and data clustering methods (Piatetsky-Shapiro and Frawley 1991).

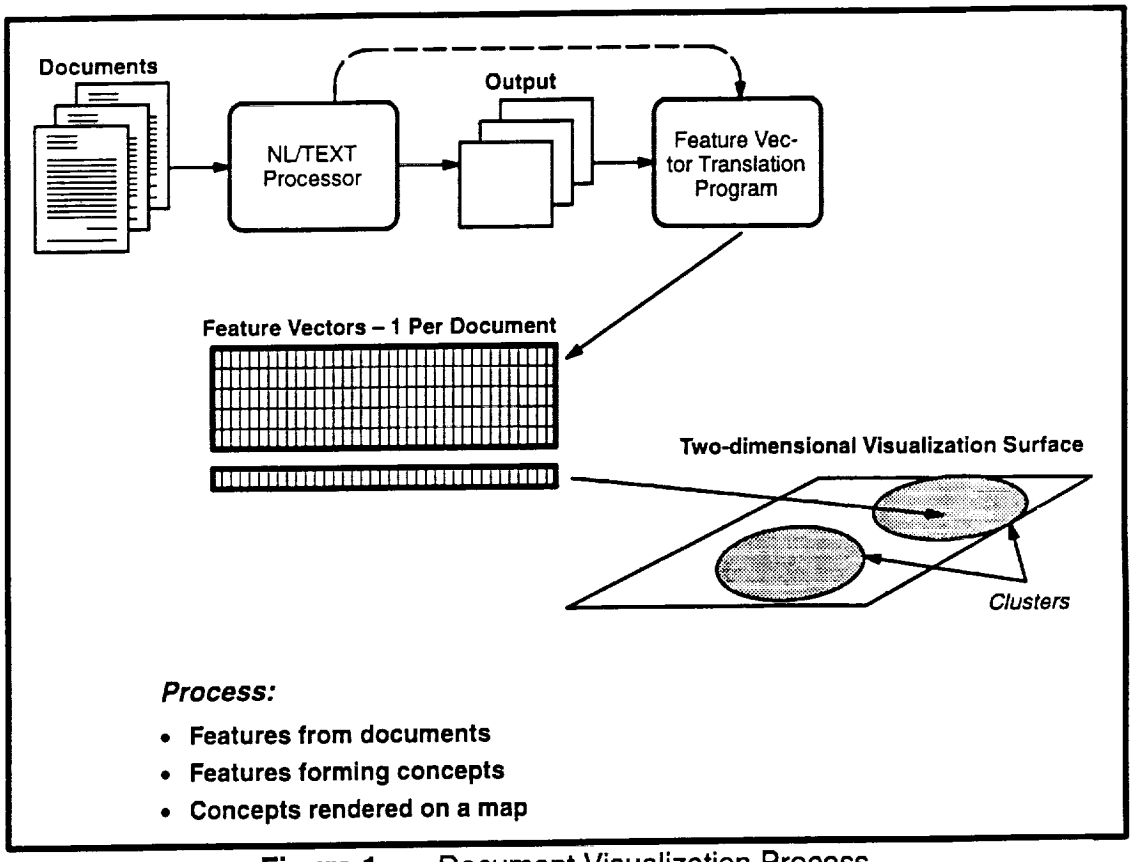


Figure 1. Document Visualization Process

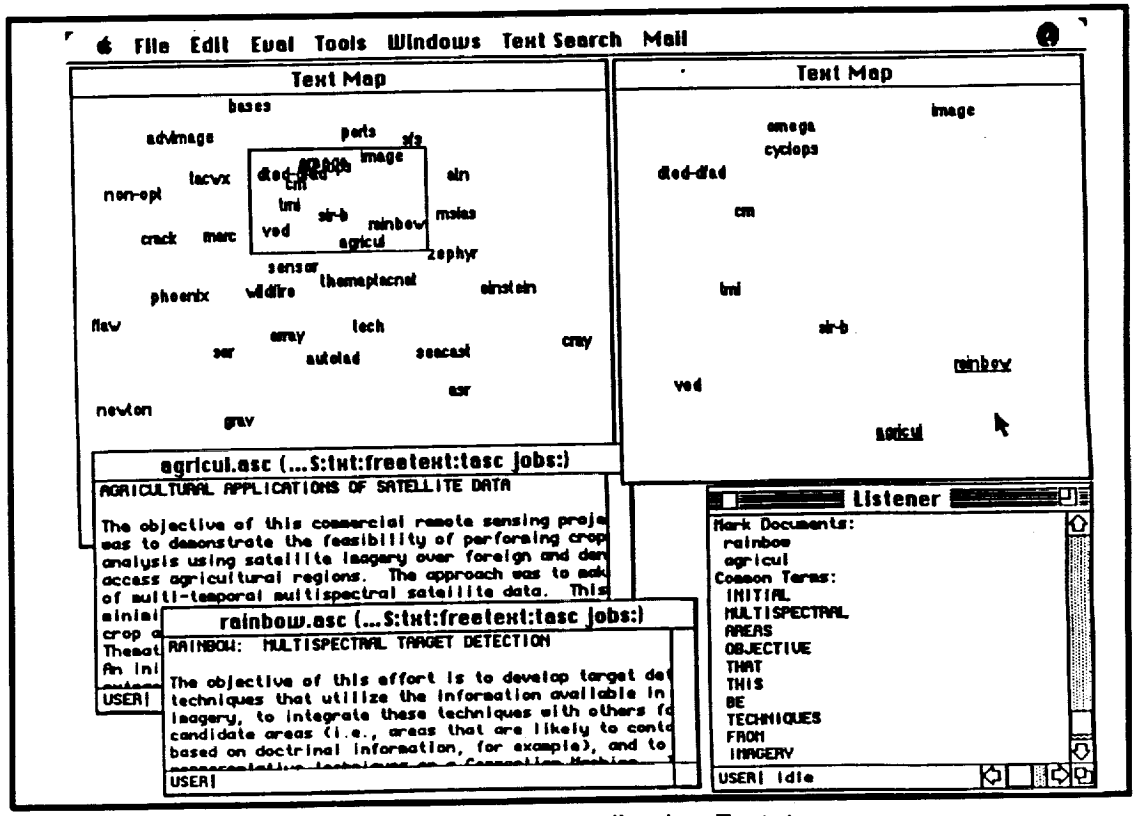


Figure 2. Text Visualization Prototype

Text visualization, however, does differ from many other AI-based database knowledge discovery methods by its abstract representation. Text maps provide a high-level topological portrait of the database that uses spatial distance on the display surface to measure the degree of association between database text objects. This approach is advantageous with large databases of objects where measurements of similarity are reducible to a scalar measure of "distance." When more complex data relationships exist and their description is sought, a visualization map would be less effective because of its inability to express fine-grain dependencies between individual items.

For large text database navigation purposes, a representation that trades fine-grain document contrasts for a cohesive global picture of the database is desirable. There are two reasons for this. First, real-world text processing often requires operation over broad subject domains using unconstrained natural language text. This means that the information extracted from documents may be too coarse to infer exact relationships. Second, the information navigation problem is primarily one of localizing areas of search through a process of iterative user guidance. This process must initially be coarse-grained because most users who browse a database either do not have a concise description of what is being searched for, or an exact understanding of the contents of the database. The need for broad-scoped database navigation is emphasized by databases whose contents are volatile and subject to change.

Early Experiments

In order to speak tangibly of the TEXTVIZ approach, an initial set of experiments using text-maps will be described here. This discussion will introduce the text-map visualization method. Then, in the next section, this discussion will be broadened to include current work. The following initial experiments were conducted using a subset of the Wall Street Journal documents contained by the 1992 NIST TIPSTER corpora (2 gigabytes) of documents.

In these experiments, the information content of documents was crudely estimated using a statistical procedure based on word-frequency counts (Figure 3). The meaning of a document was estimated by examining the frequency profile of significant words that occurred within that document. For these experiments, the set of significant words coincided with the set of words that occurred infrequently in a training corpus of documents. The frequency threshold cut-off varied across experiments and was arbitrarily selected. Low frequency words were used because of an assumption that they were generally more indicative of the meaning of a document than high frequency words. Early work in text processing (e.g., discussion in van Rijsbergen 1979) bears this out.

Once the set of significant words was selected, these then become the set of word features by which the content of documents in the test corpus would be characterized. This approach at estimating the content of a document is analogous to the vector-based score-and-rank systems used by Salton (1971) and Stanfill and Kahle (1986).

Upon completion of analysis of the training corpus of documents, a set of feature words were identified which were then used to analyze a test document set. The results were portrayed on a text map. For most experiments, the training and test document sets were identical. All the documents were analyzed, and for each document, a feature vector

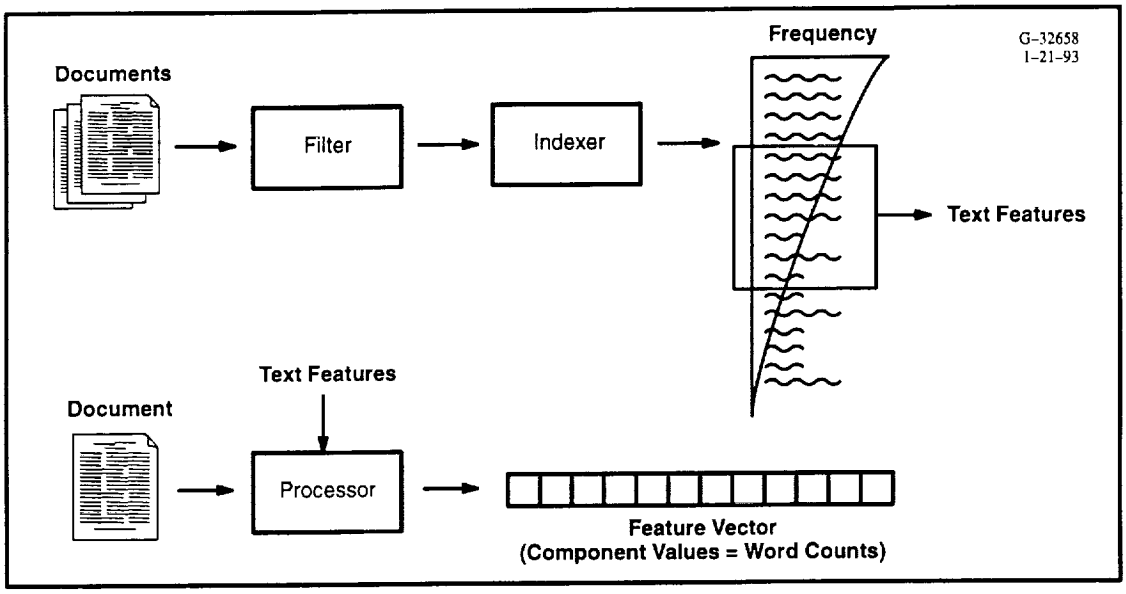


Figure 3. Text Processing: Using Word Frequencies To Estimate Meaning

was constructed. Every feature word indexed a specific component in the feature vector. Component magnitudes encoded the actual number of times a word occurred within a given document.

The procedure for analyzing the information content of a document worked as follows:

1.) An initial vector of length N (= size of feature set) was created for each document. All component values in the vector were set to zero.

2.) Every word in the document was examined; if it were a member of the feature word set, the indexed vector component value was incremented.

3.) Every vector for each document was normalized with respect to the total word count for that document.

After completion of the document analysis phase, the feature vectors were then projected onto a two dimensional visualization map. Documents were positioned on the map to reflect how close in vector space their feature vectors were. Documents of similar feature word profiles, and hence with similar estimated content, were placed close together while documents of dissimilar content were placed farther apart. Feature vectors were projected from vector space onto a two-dimensional visualization surface.

Initially, a self-organizing map (SOM) was used to implement the vector space to map projection. A SOM is an algorithm that simulates a planar "neural" network of interconnected processing units (Kohonen 1982, 1984). These processing units converge to an "accurate" portrait of the database through an adaptive process based on a competitive neural network learning procedure. However, given the large number of features that a feature-word document analysis procedure can generate (some experiments had as many as 12K feature words), a SOM was too costly to use. The size of the connectivity matrix and the number of processors required for high resolution maps restricted its use.

For this reason, an alternative, stochastic steepest-descent approach was used. While it was similar to Sammon's non-linear mapping algorithm (1969), it also made use of a stochastic search regime analogous to a simulated annealing procedure (Kirkpatrick 1987). An overview of the algorithm is as follows:

1.) Choose an initial random map configuration.

2.) Calculate the error-per-map configuration by measuring the deviation of the relative point distances on the display map from the relative distances of the vectors in vector space.

3.) Effect random perturbations to the map configuration; seek to minimize the error iteratively by using gradient descent. To minimize the possibility of local minima traps, a global "annealing" regime was in place that allowed the system to sometimes accept non-optimal changes (a monotonically decreasing probability of acceptance).

The product of these experiments was a set of text-maps such as the one given by Figure 4. By visually analyzing these text-maps and relating their topographical distribution with their interpreted differences in content, it was clear that there were several deficiencies with this approach.

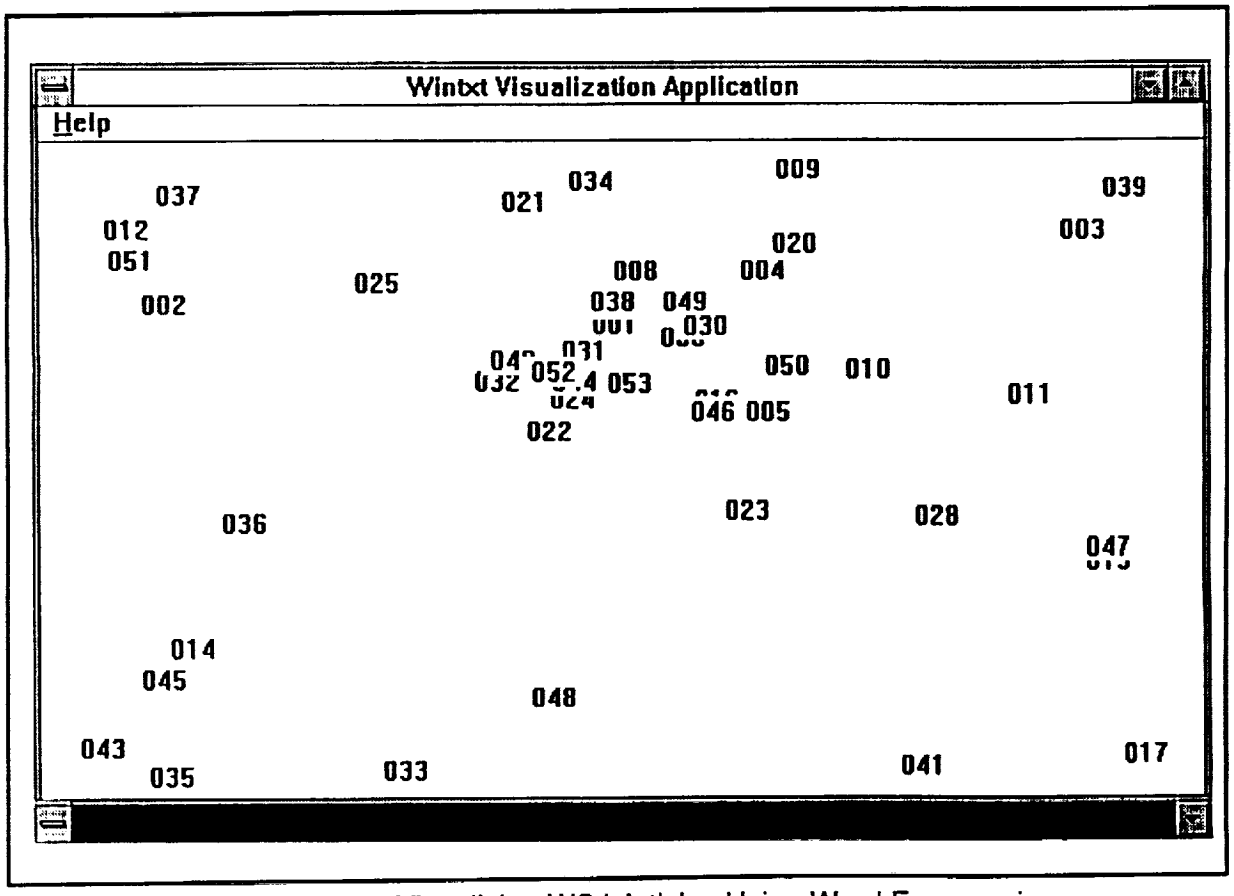


Figure 4. Visualizing WSJ Articles Using Word Frequencies

First, it was apparent that feature word profiles were at best weak predictors of the content of documents. This would seem to be especially true for large text sets. Other experiments suggested that larger linguistic units such as phrases would be better predictors of document content.

Second, it was difficult to predict which frequency ranges were best for selecting feature words from an arbitrary training corpus. Within training corpora whose documents were narrowly focused upon a few distinct subjects, idealized feature words tended to occur more frequently than they would in wide-ranging document corpora. Similarly, technical documents also tended to exaggerate the frequencies of idealized feature words.

Thus, document corpora can bias word frequency distributions in ways that are unpredictable in advance. The problem was one of predicting distribution biases without knowing anything about the actual semantic content of the documents.

And finally, this approach to generating text features did not scale-up well to larger, more discriminating applications. In other words, expansion of the feature word set to include more content words did not necessarily make the system more robust (better able to recognize a subject), and less brittle (cover more subjects). Feature word sets can be expanded by accepting more words from the training corpus (broadening the frequency range) or through dictionary or synonym expansions. The problem with word expansions, however, is that while it does improve the system's ability to recognize content words, it also introduces substantially more "noise words" into the system. The introduction of noise words distracts from meaningful comparisons between documents. Strategies for minimizing the number of "noise words" such as stemming (to remove redundant inflected forms of words), and filtering of low content words such as determiners and prepositions do exist: while they do help, they do not solve the problem.

Visualization by Semantic Content

As was suggested by the earlier experiments, more predictive text features would be needed if documents were to be accurately characterized. The current section will describe an approach that seeks to estimate the meaning of documents from the semantic content of recognized words and phrases found in a document. The described approach is undergoing implementation at TASC and will be tested in 1993 against the TIPSTER corpus of documents. This system will introduce key concepts about how a varied distribution of semantic content information from a document can be hierarchically integrated into a single feature vector description which in turn is represented on a text map.

The algorithm described here is similar to S. Gallant's work (1991, 1992) with several major modifications. It is hypothesized that these modifications will improve document discriminations by providing mechanisms to identify larger linguistic units, to better localize meaning within documents, and to provide a more accurate representation of multiple subjects within documents. These mechanisms are designed to operate in concert, to enhance intra-document content discrimination and representation. This is important when trying to represent the content of a large document without "blurring" together the subjects contained within that document.

Words, in the described model, are defined by vectors of semantic features. Each feature indexes a vector component and the component magnitude encodes the correlation of that particular word to a semantic class. Syntactic features are also included. Representing the content of a document in terms of a vector of features is analogous to vector-based linguistic models of word-sense and semantic discrimination (e.g., Miikkulainen and Dyer 1991, McClelland and Kawamoto 1986).

Thus, for example, BELGIUM might be defined by the following features (and respective correlations):

BELGIUM ->
+Nation(10)
+European-Economic-Community(3)
+North-Atlantic-Treaty-Organization(3)
+Policy:Agricultural-subsidies(4)
+Lang:french(5)
+Lang:flemish(5)
+Brussels(8)
+Geo:SAmerica(0)
+Geo:Europe(8)
+Geo:NAmerica(0)
+Geo:Asia(0)
+Noun(10)

Note that while this example illustrates a single definition of BELGIUM, multiple definitions per word may exist. The feature set, as well as specific correlation values for individual definitions, are hand-generated.

Figure 5 provides a simple overview of the system. Level (1) processing involves extracting from the text stream literal string phrases (mostly proper nouns) such as "Wall Street Journal" and "Singapore Airlines." Words from the text stream that do not match phrase rules are then stemmed (suffixes removed). Phrases and stemmed words, or "tokens," are sent on for processing at level (2). At this level, tokens are looked up in a dictionary; each dictionary entry consists of a token and a set of corresponding feature vector definitions (multiple definitions are possible). These vectors can be weighted.

If a text stream token matches a dictionary entry, all defining feature vectors associated with the matching dictionary entry are then aggregated into an ordered list of vectors, or vector stream. Because a word can have multiple definitions, a method of consolidating meaning within document regions is required. The approach adopted at level (3) is to partition the vector stream into bins which correspond to "chunks" of text from a document. The text regions can be of arbitrary size and will generally be delimited by text breaks such as paragraph boundaries. Thus, the size of the vector stream bins will co-vary with the size of the text regions. The size of text regions will be user/application defined, and will depend upon the granularity of the meaning that is sought from the document.

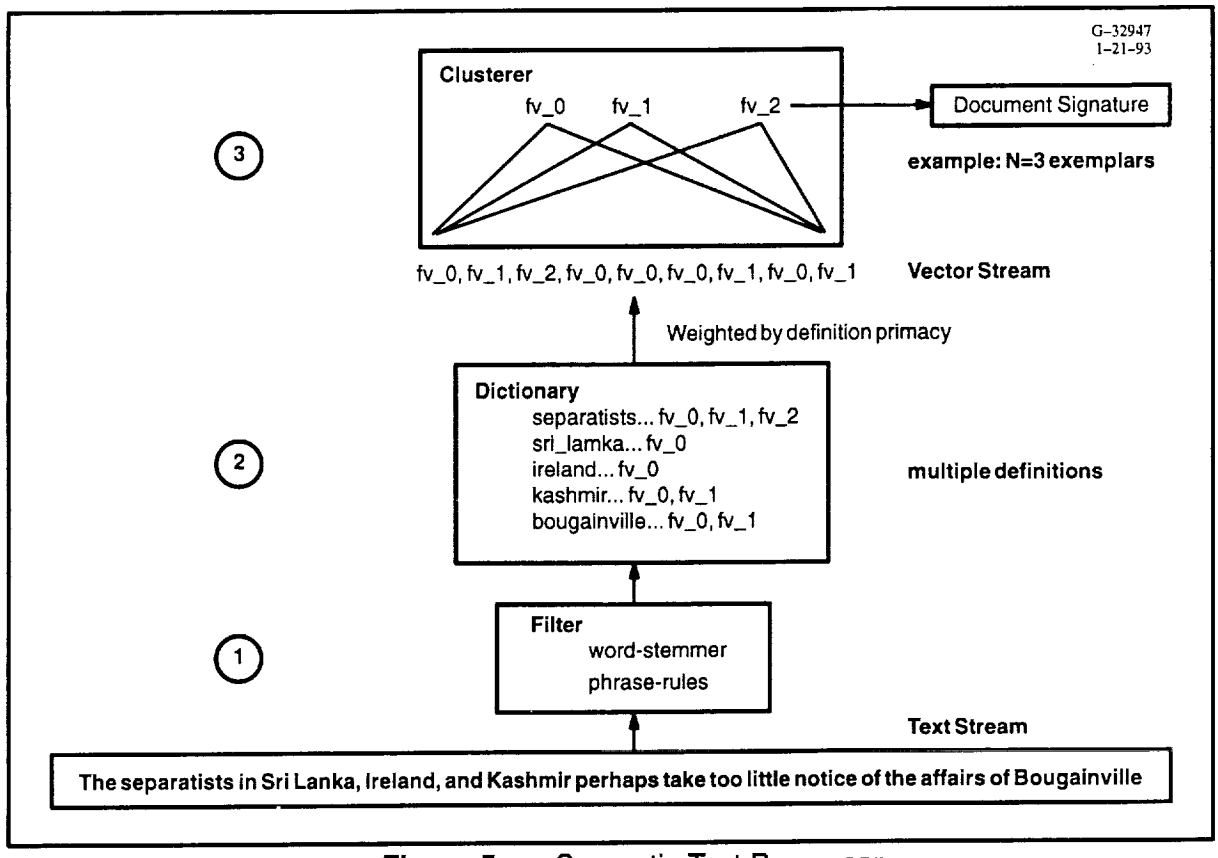


Figure 5. Semantic Text Processor

Within the context of a vector stream bin, a consensus of meaning is sought. It is through a process of consensus building that alternative, inconsistent definitions are pruned. The objective is then to select exemplar definitions (feature vectors) to signify the content of each vector bin. This exemplar will serve as the consensus content vector representing a text region.

The exemplar selection process, carried out for every text region in a document, is an abstraction process that uses context to prune alternative definitions as well as limit the background noise generated by a large population of definitions. The number of definitions can be magnified when working with large text bins and/or working with large dictionaries. In this way, abstraction is used to preserve the integrity of the extracted content information by restricting noise as well as reducing ambiguity.

The exemplar selection process uses a stochastic gradient descent clustering technique that is similar to the algorithm used to generate the text maps (described earlier). Definitions may be weighted to reflect a priori assumptions about which definitions are most significant (e.g., primary definitions are more important than secondary and tertiary definitions). Features within definition vectors may also be used to modulate the exemplar selection process. For example, if syntactic features were included in token definitions, and if it were assumed that certain syntactic categories were more predictive of the meaning of a document than others, then selected syntactic category information (e.g., "+Noun") could be weighted to bias the selection of candidate exemplars.

As stated, the effect of this clustering procedure is to discard extraneous interpretations of tokens as well as to reinforce a consensus in meaning. This consensus is codified by the selection of a set of exemplars used to represent the document. The manner in which a consensus of meaning is reinforced is analogous to Gallant's (1991) ideas on using the local context around a word to disambiguate individual words. In contrast, the approach described here uses a "regional" context, consisting of all words within a text region, to modulate choice of exemplars. A regional approach is generally faster to compute and also serves as a more general mechanism of information abstraction.

The product of level (3) processing is a set of exemplar vectors, one per text region, that are used to symbolize the content of the document. Additional vector abstraction may also be used to post-process the level (3) output in order to consolidate meaning across text regions, i.e. eliminate redundancies in the exemplar list. The approach described here advocates preserving distinct representations for unique regions of text. Thus, the described approach differs conceptually from Gallant's (1992) strategy of vector addition. With this approach, the complete set of exemplar vectors is used to denote the contents of a document. This prevents a document signature from being "blurred" by background feature "noise." Similarly, it provides a means for localizing intra-document content searches and is hypothesized to be of significant value when working with long documents.

Conclusions

As illustrated, text maps can abstractly render general semantic relationships among the contents of large text databases. While this assumes the existence of a semantic model with which to analyze the database text objects, it would not explicitly require such a model for visualization. By separating the semantic interpretation process from the visualization process, users can conceptualize and navigate large complex databases at a high level. Text maps thus provide a graphical and spatial metaphor for reasoning about the contents of large text databases.

Described in this paper is an approach for hierarchically integrating the semantic content of a spatial text stream. Aside from specific relevance to NASA interests in text and document retrieval (Driscoll 1992), such an approach may have broader implications for information management and database knowledge extraction. For example, semantic comparisons between text and other spatially structured data types, such as imagery, could be pertinent.

Furthermore, TASC is investigating how text information can be generalized to other information domains. For example, textually-derived information has been used to augment information from geographic as well as image sources (Carlotto 1992). This work underscores the versatility of the visualization map graphic metaphor, as well as suggests a conceptual interface design for integrating information of diverse types.

References

Carlotto, M (1992, March). "A Text-Based Geographic Information System." TASC white paper.

Chang, S (1990). "Visual Reasoning for Information Retrieval from Very Large Databases." Journal of Visual Languages and Computing, 1. pp. 41–58.

Driscoll, J., J. Lautenschlager, M. Zhao (1992). "The QA System." Preprint of the Proceedings of the Text Retrieval Conference (TREC). Rockville, MD. November 4–6, 1992.

Gallant, S. I. (1992). "HNC's MatchPlus System." Preprint of the Proceedings of the Text Retrieval Conference (TREC). Rockville, MD. November 4–6, 1992.

Gallant, S. I. (1991). "A Practical Approach for Representing Context And for Performing Word Sense Disambiguation Using Neural Networks. Neural Computation 3(3). 293–309.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P (1983). Science, 220. pp. 671–680.

Kohonen T (1984). Self-organization and associative memory. Berlin: Springer-Verlag.

Kohonen T (1982, Oct 19–22). "Clustering, Taxonomy, and Topological Maps of Patterns." Proceedings of the 6th International Conference on Pattern Recognition.

Piatetsky-Shapiro, G., and W.J. Frawley (1991). Knowledge Discovery in Databases. Menlo Park: AAAI Press/ MIT Press.

McClelland, J.L., and Kawamoto, A.H (1986). "Mechanisms of sentence processing: Assigning roles to constituents." In J.L. McClelland, and D.E. Rumelhart (Eds). Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations. Cambridge, MA: MIT Press.

Miikkulainen, R., and M. Dyer (1991). "Natural Language Processing with Modular PDP Networks and Distributed Lexicon." Cognitive Science 15. pp. 343–401.

Rijsbergen, C.J (1979). Information Retrieval. Butterworths: Boston.

Rorvig, M. E (1991, Dec 3–5). "A Vector-Product Information Retrieval System Adapted to Heterogeneous, Distributed, Computing Environments." Proceedings of Technology 2001, NASA conference.

Salton (1971). The SMART Retrieval System — Experiment in Automatic Document Processing. Prentice-Hall: Englewood Cliffs, NJ.

Sammon, J (1969, May). "A nonlinear mapping algorithm for data structure analysis" IEEE Transactions on Computers, C–18(5).

Schneiderman, B (1991, August). "Visual User Interfaces for Information Exploration." Department of Computer Sciences Technical Report No. 2748.

Stanfill, C., and Kahle, B (1986, December). "Parallel Free-Text Search on the Connection Machine System." Communications of the ACM, Vol 29,12.

In Search of Meta-knowledge

N93-25983

Antonio M. Lopez, Jr., Ph.D.
Mathematical Sciences
Loyola University Box 51
New Orleans, LA 70118
(504) 865-2657
e-mail: lopez@loynovm.bitnet

Abstract

Development of an Intelligent Information System (IIS) involves application of numerous artificial intelligence (AI) paradigms and advanced technologies. The National Aeronautics and Space Administration (NASA) is interested in an IIS that can automatically collect, classify, store and retrieve data, as well as develop, manipulate and restructure knowledge regarding the data and its application (Campbell et al., 1987, p.3). This interest stems in part from a NASA initiative in support of the interagency Global Change Research program. NASA's space data problems are so large and varied that scientific researchers will find it almost impossible to access the most suitable information from a software system if meta-information (metadata and meta-knowledge) is not embedded in that system. Even if more, faster, larger hardware is used, new innovative software systems will be required to organize, link, maintain, and properly archive the Earth Observing System (EOS) data that is to be stored and distributed by the EOS Data and Information System (EOSDIS) (Dozier, 1990). Although efforts are being made to specify the metadata that will be used in EOSDIS, meta-knowledge specification issues are not clear. With the expectation that EOSDIS might evolve into an IIS, this paper presents certain ideas on the concept of meta-knowledge and demonstrates how meta-knowledge might be represented in a pixel classification problem.

Introduction

There is no single view of what constitutes an IIS nor how to apply AI techniques to develop such a system (Goyal, 1989; Kerschberg, 1990). However, some researchers (Kaula & Ngwenyama, 1990) envision an IIS as evolving from a large number of independently developed systems that communicate and cooperate by passing messages (data, knowledge, and information). These independently developed systems will have evolved using various software paradigms including different AI paradigms such as object-oriented or logic-oriented ones. In addition, the use of neural networks or genetic algorithms to solve very domain specific problems will be supported by advanced technologies tailored for the independently developed system. Each of these independently developed systems will have their own assumptions, constraints, and goals. Yet, they will be "partners in a bigger scheme of things." In this development, there is no global schema. At times, one system will be called upon to pass portions of its knowledge to another system and, likewise, acquire knowledge from their communicating partners as the need arises.

Given the many and varied Earth science systems that have been independently developed by NASA to this point in time and the EOS project that will collect more data than ever collected before, EOSDIS seems ideally positioned to evolve into an IIS. EOSDIS will be responsible for the storage and distribution of large volumes of data that will support scientific research into the global change problem domain. The EOSDIS Information Management System (IMS) will provide the software tools to search, locate, select, and order data archived at Distributed Active

Archive Centers (DAACs). The IMS will manage a set of metadata that includes among other items, directory, catalog, and inventory level information, summary statistics, algorithm descriptions, mission information, and user profiles (McDonald & Blake, 1991). The current stage of development (Version 0) attempts to integrate and expand data management capabilities being used now by different Earth science disciplines. As the EOSDIS IMS evolves, challenges will exist in the specification of the scope of the system and in dealing with the many uncertainties found in the end-user community.

At NASA's Goddard Space Flight Center the Intelligent Data Management (IDM) project team is conducting research into the information and data management needs of Earth and space missions that will produce terabyte-sized spatial databases that cannot be effectively managed using present data management and mass storage technologies (Campbell & Cromp, 1990). This basic research may have an impact on the evolution of EOSDIS IMS. The IDM project team has proposed among many other techniques the use of semantic data modeling to organize object-oriented databases, thereby extending the mass storage model. To test this approach, they have developed an Intelligent Information Fusion System (IIFS) prototype. The IIFS employs several key AI concepts and methodologies such as object/frame representations, multiple inheritance, and rule-based decision making. Applications of AI throughout the IIFS attempt to remove from the end-user (novice to expert) the need to understand the various complexities and nuances of the system and of the particular problem domain. However, the IDM project team has recognized that future science research will require even more comprehensive pre-existing knowledge about the data granules, problem domains, and end-users of the system (Cromp et al., 1992). In short, more meta-information is needed.

Meta-information: Metadata and Meta-knowledge

Meta-information is the underpinning of any IIS. It is the information about the information stored within the system that allows the system to be perceived as intelligent. To take a page from Kidder's book (1981), "meta-information is the soul of an IIS." The IDM project team (Campbell & Cromp, 1990) describes meta-information as incorporating into an IMS knowledge about the structure (syntax) of and the relationships (semantics) between data components, and the hidden questions behind a user's query and the assumptions behind the system's response to that query (pragmatics). The pattern of evolution that this research into meta-information is taking is classical (Lenat & Guha, 1990). The metadata research and development addresses the factual knowledge or the zero-order correction. The research into object-oriented data management with related semantic data modeling holds promise for handling heuristic knowledge or first-order correction. The second-order correction is meta-knowledge. Meta-information is both metadata and meta-knowledge where the metadata is mostly syntax, the meta-knowledge is mostly pragmatics, and both share in the semantics between the data components and the current status of information in the system.

Generally speaking, the metadata for an IIS standardizes what data describes the information resource, and it formalizes policies by specifying what data must be maintained as the system is developed and used (March & Kim, 1989). Intelligent metadata management is a key ingredient in the performance of an IIS (Kaula & Ngwenyama, 1990). In addition, the performance of an IIS can be improved by supplying it with meta-knowledge. Meta-knowledge comes in many forms, but two general categories seem to encompass much of what is considered to be meta-knowledge. First, there is meta-knowledge that guides the user of the IIS to the "best" rules to apply, that is, strategies that will focus quickly on the relevant group of rules to be used on a particular problem (e.g., browsing and searching). This category contains knowledge regarding knowledge permanency, priorities of knowledge, and knowledge on how to resolve conflicting knowledge from different sources. For example, in this category, meta-knowledge

on an ozone data set would include knowledge about a derived data set obtained by a researcher (Is this an interim processing file with additional work forthcoming?), its level of reliability (Was the pixel classification work done for thoroughness or expedience?), its relative importance with respect to other derived data sets (How does this derived data set match the profile (level of expertise, desire for detail, etc.) of the researcher making the request?), and an evaluation of the performance of the cognitive processor (novice to expert) who developed it. Second, there is meta-knowledge that oversees the IIS. This category contains knowledge regarding the ability to explain system responses, to detect inconsistencies, and to restructure system knowledge. For example, Earth scientists will want to know why the system is responding the way that it is for a particular query. What is its justification? Is it the opinion of an established expert whose knowledge has been captured? Not only is this meta-knowledge, but the act of extracting the domain specific knowledge from the expert, coding it, and putting it into the system itself is also meta-knowledge (Cromp, 1990).

In the IIFS, the metadata for the object-oriented data management with related semantic data modeling has evolved into a knowledge-base with objects and relationships between objects being explicitly declared (Campbell et al., 1991). The meta-knowledge too has been recognized and dealt with explicitly as the "pre-existing" knowledge about the problem domain, the sensor device, and the interpretation of the sensor's measurements (Campbell et al., 1989). However, with the increased research that will naturally follow EOS, it is imperative that newly acquired knowledge (new meta-knowledge) be ingested and available to all in the scientific community (Short, Jr., 1991).

In the design and development of an IIS, the automation of meta-knowledge is essential. An IIS must recognize the limitations of its knowledge and gain new knowledge by interacting with the users that it is serving. To this end, meta-knowledge must be represented in a language that is high-level and robust yet has the appropriate primitives to integrate multi-paradigm software systems.

A Knowledge Representation Language for Meta-knowledge

Zarri (1990) proposed a "conceptual" knowledge representation language suited to the construction and use of intelligent information retrieval systems. This conceptual knowledge representation language exploits the organizational strength found in definitional hierarchies and the power realized in a theorem-prover with a unification algorithm. The components of the language are organized around a semantic predicate ("has", "produces", etc.) that identifies the basic type of situation to be described. The semantic predicates are frame-like in structure with "arguments" (objects) and "roles" (slots). The choice of semantic predicates is pragmatic and depends on the architecture of the system and on the problem domain, in particular the arguments and the roles of those arguments in an application. Roles can be categorized as descriptive (such as: SUBJECT, OBJECT, SOURCE, DESTINATION, etc.), binding (such as: COORDINATION, SPECIFICATION, ALTERNATIVE, ASSOCIATION, etc.), and causal (such as: CAUSE, MOTIVATION, CONFER, GOAL, etc.). As a conceptual unit, the semantic predicate can be further characterized by "determiners" (attributes), for example, location and temporality. Figure 1 is an example of how the conceptual knowledge representation language might be applied to remote sensing domain knowledge. It is a predicative conceptual unit (a predicative occurrence) having a semantic predicate "created_using", arguments such as "data_set_ssc150" and "CAMS", and determiners. The importance of this work is that it provides a conceptual base from which to study the inclusion of meta-knowledge into an IIS. Both the binding and the causal roles can be very useful toward this end; they can allow control strategies to be explicitly defined. An implementation of the proposed knowledge representation language would be a compromise between object-oriented and logic-oriented paradigms.

created_using	SUBJECT:	data _set_ssc150
	OBJECT:	CAMS
	SOURCE:	flight_7824
	SPECIFICATION:	mission_request_8182
	ALTERNATIVE:	data_set_ssc278
	COORDINATION:	[5_percent_cloud_cover, 30000_feet_altitude]
	MOTIVATION:	deforestation_study
	[location: Mexico_Guatemala_border]	
	[date: 24_july_1990]	

Figure 1. An example of a predicative occurrence.

The knowledge representation language described above allows the system designer to declare data, metadata, and meta-knowledge without knowing the details of its implementation. A preprocessor could then be used to produce an effective and efficient implementation of the design. Such a preprocessor could be either a meta-interpreter (Sterling & Beer, 1989) or a translator (Console & Rossi, 1989). The latter approach is being taken for several reasons. First, the knowledge representation language lends itself to this method. Second, Zaniolo (1984) demonstrated that object-oriented programming can be embedded in a logic programming language (PROLOG). Furthermore, today, the integration of object-oriented and logic-oriented paradigms is a robust and productive area of research (McCabe, 1992). Finally, since logic programming has already been used in metadata specification to make designs of semantic networks and frames into executable code that can be queried (Lopez and Saacks, 1992), it seems only natural to extend its use via a translator to implement the knowledge representation language.

FROG (**F**rames in PROLO**G**) is a logic programming language that combines frames, production rules, and PROLOG (Console & Rossi, 1989). In FROG each frame can contain either slots or production rules (with various kinds of inference strategies). Descriptive meta-knowledge on the relationships that exist between frames can be embedded in various kinds of links supported by FROG. Trigger links stipulate conditions under which a frame will be activated. Specialization links structure the hierarchy of frames. Associational links connect highly correlated frames. Alternative links suggest other possible hypotheses of solution to be considered when a frame cannot be instantiated. In addition to these links, FROG frames have knowledge components that can either be local production systems or prototypical descriptions. Control knowledge is vested in a "superframe," which is the top most frame in the frame hierarchy as stipulated by the specialization links.

Many of the concepts and ideas expressed in Zarri's conceptual knowledge representation language seem to have been implemented in FROG. In particular, the binding links of SPECIFICATION, ASSOCIATION, and ALTERNATIVE seem to match directly with the FROG links of specialization, associational, and alternative. The superframe allows the explicit specification of the control strategy and a separation from the knowledge components of the frames. The knowledge components allow the system designers to embed even more meta-knowledge in the form of prototypical descriptions or production systems. The knowledge-base itself is an object-oriented structure.

An Application

The classification of pixels in a data set obtained from aerial or satellite images is a difficult and time-consuming process. Rules used to classify regions in remotely sensed images are not universal truths. Human experts have developed heuristic knowledge that allows them to focus on those classification features that help refine the initial analysis of the image that might have been done by unsupervised training algorithms. In using unsupervised training, the data analyst specifies some parameters that the algorithm can use to determine statistical patterns that are inherent in the data. This is useful only if the classes that are produced can be manually interpreted. The interpretation depends on the expertise of the data analyst because the classes do not necessarily correspond directly to meaningful classifications such as water, crops, manmade objects, etc. The process involves the ingredients of data, metadata, and meta-knowledge, and can be used as a testbed for research ideas involved in the development of IIS.

During FY92, a small study was done at Stennis Space Center on a knowledge-based pixel classification approach using PROLOG as the vehicle to investigate the relationship between low and high resolution feature identification in Calibrated Airborne Multispectral Scanner (CAMS) data sets (Lopez et al., 1992). The goal was to be able to use knowledge in various forms to construct a system with the potential of changing the means by which it characterizes a given class of pixels (structuring and restructuring knowledge). Knowledge-based methods when used with statistical classifiers tend to improve the accuracy of the overall classification of pixels in an image (Short, Jr., 1991). Rules for image classification for this study were developed on the basis of the expert data analyst's knowledge of the numerical values produced by the statistical (maximum likelihood classification) unsupervised training. Knowledge obtained from this study is used below to demonstrate some of the constructs of FROG.

If a pixel class is to be identified as water, it will usually exhibit a low near-infrared reflectance. An expert data analyst's own interpretation of this previous statement might be that the mean in the red channel of the class is less than 40 and the near-infrared mean is less than 30. If this is realized by a pixel class it will "trigger" further investigation into whether or not the pixel class is indeed water. There are also both necessary and sufficient conditions for a pixel class to be water but if the pixel class meets the necessary conditions, then it does not have to meet the sufficient conditions to be interpreted as water. Furthermore, there can always be supplemental knowledge that can support the interpretation. This is particularly important when uncertainty factors are added to the "knowledge components". The constructs of FROG are used below to explicitly embed the meta-knowledge that has been discussed. Uncertainty factors have not been incorporated into this example.

```
frame_control(water_class,activation) :-
    knowledge_component(water_class,trigger),
    (       (knowledge_component(water_class,necessary);
               knowledge_component(water_class,sufficient)   ) +
    knowledge_component(water_class,supplementary),
    frame_control(water_class,specialization).

knowledge_component(water_class,trigger) :-
    slot(water_class).

slot(water_class) :-
    conditions(water_class).
```

```
conditions(water_class) :-
    implies(water_class,red_channel_mean_less_than_40),
    implies(water_class,near_infrared_channel_mean_less_than_30).

knowledge_component(water_class,necessary) :-
    slot(implies(water_class,red_to_green_ratio_is_0.4),
    slot(implies(water_class,red_to_near_infrared_is_0.1).

knowledge_component(water_class,sufficient) :-
    slot(implies(water_class,minimum_spectral_distance_from_water_classes));
    slot(implies(water_class,above_diagonal_and_left_in_green_near_infrared_plot)).

knowledge_component(water_class,supplementary) :-
    slot(contextual_information).

frame_control(water_class,specialization) :-
    frame_control(clear_water_class,activation);
    frame_control(muddy_water_class,activation).
```

The activation of the water_class frame succeeds if the trigger knowledge component can be instantiated and either the necessary or the sufficient knowledge component instantiated. As in standard PROLOG coding, the comma is used for the connective "and," and the semicolon is used for the connective "or." The plus symbol in FROG is an additive evidence combination operator and, if certainty factors were being used, would increase the certainty that the pixel class was water if the supplemental knowledge component was instantiated. This operator allows two knowledge components with knowledge from different sources leading to the same conclusion to be combined. Finally, a subframe is invoked for specialization.

Conclusion and Future Research Direction

In the past, NASA has just provided data to researchers and done little to capture into its archives the knowledge derived from the researcher's use of the data. If the acquired knowledge is to be unified and made available to the entire scientific community, then any future IIS will have to rely more heavily on meta-information. In particular, meta-knowledge will have to be recognized and explicitly coded into such systems. To support this effort, more research needs to be done on the application of Zarri's conceptual knowledge representation language to space systems such as EOSDIS. Hand in hand with this effort is the research that is needed in implementation languages. A logic programming language such as FROG holds great promise. However, it is safe to say that the search for meta-knowledge in IIS is just beginning.

References

Campbell, W., & Cromp, R. (1990). Evolution of an Intelligent Information Fusion System. *Photogrammetric Engineering and Remote Sensing*, 56(6), 867-870.

Campbell, W., Hill, S. & Cromp, R. (1989). Automatic labeling and characterization of objects using artificial neural networks. *Telematics and Informatics*, 6(3/4), 259-271.

Campbell, W. Roelofs, L. & Short, Jr., N. (1987). The development of a prototype intelligent user interface system for NASA's scientific database systems. *NASA Technical Memorandum 87821*, Greenbelt, MD.

Campbell, W., Short, Jr. N., Roelofs, L. & Dorfman, E. (1991). Using semantic data modeling techniques to organize an object-oriented database for extending the mass storage model. *Proceedings of the 42nd Congress of the International Astronautical Federation*, Montreal, Canada.

Console, L. and Rossi, G. (1989). Using PROLOG for building FROG, a hybrid knowledge representation system. *New Generation Computing*, 6(4), 361-388.

Cromp, R. (1990). Knowledge acquisition for spatial reasoning about satellite imagery. *Proceedings of the 4th International Symposium on Data Handing*, Zurich, Switzerland.

Cromp, R., Campbell, W., & Short, Jr., N. (1992). An Intelligent Information Fusion System for handling the archiving and querying of terabyte-sized spatial databases. To appear in the *American Institute of Physics Conference Proceedings.*

Dozier, J. (1990). Looking ahead to EOS: The Earth Observing System. *Computers in Physics*, 4(3), 248-259.

Goyal, P. (1989). Intelligent information systems: The concept of an intelligent document. *Information Systems*, 14(4), 351-358.

Kaula, R. & Ngwenyama, O. (1990). An approach to open intelligent information systems. *Information Systems*, 15(4), 489-496.

Kerschberg, L. (1990). Expert database systems: Knowledge/data management environments for intelligent information systems. *Information Systems*, 15(1), 151-160.

Kidder, T. (1981). *The soul of a new machine.* Boston, MA: Little Brown.

Lenat, D. & Guha, R. (1990). *Building large knowledge-based systems: Representation and inference in the CYC project.* Reading, MA: Addison-Wesley.

Lopez, A., Junkin, B. & L. McGregor. (1992). An investigation into the use of artificial intelligence in the classification of remotely sensed data. Under review.

Lopez, A. & Saacks, M. (1992). Logic programming and metadata specification. *Proceedings of the Goddard Conference on Space Applications of Artificial Intelligence*, Greenbelt, MD, 245-253.

March, S., & Kim, Y. (1989). Information resource management: A metadata perspective. *Journal of Management Information Systems*, 5, 3, 5-17.

McCabe, F. (1992). *Logic and Objects.* New York, NY: Prentice Hall.

McDonald, K., & Blake, D. (1991). Information management challenges of the EOS Data and Information System. *Proceedings Technical Papers ACSM-ASPRS Annual Conference*, Baltimore, MD, 258-267.

Short, Jr. N. (1991). A real-time expert system and neural network for the classification of remotely sensed data. *Proceedings Technical Papers ACSM-ASPRS Annual Conference,* Baltimore, MD, 406-418.

Sterling, L., & Beer, R. (1989). Metainterpreters for expert system construction. *Journal of Logic Programming*, 6(1/2), 163-178.

Zaniolo, C. (1984). Objected-oriented programming in PROLOG. *Proceedings of the IEEE International Symposium on Logic Programming*, Atlantic City, NJ, 265-270.

Zarri, G. (1990). A knowledge representation language for large knowledge bases and "intelligent" information retrieval systems. *Information Processing and Management*, 26(3), 349-370.

The StarView Intelligent Query Mechanism

R. D. Semmel
The Johns Hopkins University Applied Physics Laboratory
Laurel, MD 20723
rds@aplcen.apl.jhu.edu

D. P. Silberberg
The Space Telescope Science Institute
Baltimore, MD 21218
davids@stsci.edu

Abstract

The StarView interface is being developed to facilitate the retrieval of scientific and engineering data produced by the Hubble Space Telescope. While predefined screens in the interface can be used to specify many common requests, ad hoc requests require a dynamic query formulation capability. Unfortunately, logical level knowledge is too sparse to support this capability. In particular, essential formulation knowledge is lost when the domain of interest is mapped to a set of database relation schemas. Thus, a system known as QUICK has been developed that uses conceptual design knowledge to facilitate query formulation. By heuristically determining strongly associated objects at the conceptual level, QUICK is able to formulate semantically reasonable queries in response to high-level requests that specify only attributes of interest. Moreover, by exploiting constraint knowledge in the conceptual design, QUICK assures that queries are formulated quickly and will execute efficiently.

1. Introduction

The Space Telescope Data Archive and Distribution System (ST-DADS) is the repository for scientific and engineering data produced by the Hubble Space Telescope and ground system. Data sets recording the scientific results of six different astronomical instruments and more than twenty different sets of engineering data are archived onto optical disk at a rate of approximately one terabyte per year. Thousands of people with diverse interests, located around the world, are concerned with different aspects of the archived data. In particular, interests are shared among scientific investigators, guaranteed time observers, general observers, instrument calibration scientists, archival researchers, and project engineers.

ST-DADS is a computer cluster composed of archive, catalog, and host-computer subsystems. The archive subsystem is devoted to archiving, managing, and retrieving data sets. The catalog subsystem, or ST-DADS Catalog, is a descriptive database of the archived data. The host computers are devoted to managing user access to the ST-DADS system. StarView is the user interface software that facilitates access to ST-DADS. It is designed to operate on the ST-DADS host computers, allowing Telnet and dial-in access. It also can operate in client/server mode in which the client runs on a workstation and the server runs on the host computers.

Currently, the StarView user interface is a set of screens that allows users to browse the ST-DADS Catalog. Each screen is composed of a set of data fields appropriate for a specific search. Search qualifications are entered in screen fields and serve as selection criteria. When a request for data is submitted, StarView constructs the corresponding Structured Query Language (SQL) query and transmits it to the ST-DADS Catalog. The query is processed and the results are returned to the StarView screens; the returned records describe archived data sets. The user marks appropriate records

to indicate to StarView those data sets that are to be retrieved. StarView bundles the data set retrieval requests and transmits them to ST-DADS, which, in turn, returns the requested data via the Internet or other storage media.

As user interests vary significantly, it is not possible to predetermine the criteria for data selection. Furthermore, the ST-DADS Catalog currently consists of approximately 1500 attributes distributed among more than 40 relations, and grows at the rate of approximately 100 megabytes per year. As a result, developing an interface for ad hoc requests using traditional approaches has not been possible. Thus, while StarView provides more than forty screens with the most common sets of fields, these screens cannot be used to formulate queries in response to the many requests that have not been anticipated. Unfortunately, enabling users to build screens is not feasible, as typical users do not possess the expertise or knowledge of the database design to ensure that semantically meaningful results would be produced. Moreover, providing screens for the complete set of possible requests is not feasible because of the combinatorial nature associated with the many possible selection criteria involving multiple attributes distributed among multiple relations.

To simplify interface construction, a knowledge-based approach has been used that enables StarView to present a universal relation interface to users. This interface presents all database attributes as residing in a single relation, thus eliminating the need for explicit knowledge concerning the structural complexity of the underlying database design [Leymann 1989; Maier et al. 1984]. However, the interface must be "smart" about how various relations can be associated via relational joins, and it must ensure that semantically reasonable queries are generated in response to high-level requests that specify only attributes of interest.

StarView has been adapted to exploit the universal relation approach by allowing users to select attributes from an ad hoc query screen. Specifically, the complete list of ST-DADS attributes is displayed and the user selects those that correspond to his request. StarView then places the selected attributes on a temporary screen that is functionally equivalent to a defined screen. However, the temporary screen does not specify predefined joins. Instead, StarView passes the request to an intelligent query system known as QUICK (for "QUICK is a Universal Interface with Conceptual Knowledge") that formulates the query based on conceptual design knowledge and passes the query back to StarView for further processing.

In this paper, an overview of QUICK is presented. In the next section, semantic data modeling is discussed and the need for knowledge beyond the relational level is justified. In particular, an extended Entity-Relationship model is described that enhances the basic Entity-Relationship model with selected knowledge representation constructs, and a portion of the current design of ST-DADS using the extended constructs is shown. Then, in Section 3, the notion of contexts is introduced as a means for segmenting an EER conceptual schema into overlapping subgraphs of strongly associated objects that facilitate inference of valid relational joins. In Section 4, it is demonstrated how contexts can be used to automate query formulation and thus facilitate the construction of high-level and intelligent interfaces. Finally, the current status of development is described and some current research topics are discussed.

2. Semantic Data Modeling

Designing a database as complex as ST-DADS requires many person-years of effort. Unfortunately, the relational model, with its elegant, but simple, notions of relations and attributes, is too sparse a representation to use for direct modeling of complex domains. Instead, higher level representations typically are used that enable designers to communicate effectively with users as well as abstract an application domain to an appropriate level. These richer representations enable designers to model entities or objects in a world directly, and provide constructs for specifying explicit relationships or associations among entities [Hull and King 1987; Peckham and Maryanski 1988]. In contrast, the relational model does not distinguish between entities and relationships, requiring instead that each

abstraction be cast as a relation. Consequently, knowledge that was explicit at the conceptual level becomes implicit at the logical level. Moreover, knowledge that could be used to facilitate query formulation is lost as the world of interest is mapped to a set of relations (see Figure 1).

Exacerbating the loss of knowledge is the usual practice of relegating the conceptual schema to a minor role once the logical design has been completed. For instance, it is not uncommon for the conceptual schema to be used for initial documentation only. Once the logical schema has been created, changes often are made directly to it instead of to the conceptual schema, thus reducing the validity of the higher level representation. Moreover, users typically are not given access to the higher level conceptual schema, having to rely instead upon the sparse logical schema to formulate queries. Yet, the conceptual schema contains knowledge that can be used effectively for query formulation, and thus can serve as a knowledge base for an intelligent interface.

2.1. The Entity-Relationship Model

The Entity-Relationship (ER) model is the prominent semantic data model used for conceptual design. From a knowledge representation standpoint, the ER model resembles a restricted associationist scheme. Diagrammatically, entity types are represented by rectangles, and relationship types are represented by diamonds. As an example, consider Figure 2, where the entity type ARCHIVE-DATA-SET-ALL is related to the entity type OBSERVATION via the relationship type ADS-FROM-OBS. The ARCHIVE-DATA-SET-ALL entity type corresponds to one of the core relations in ST-DADS and contains general information about the class of data, generation time, and the name of the archive. Similarly, OBSERVATION corresponds to a core relation and contains information about the predicted and actual observations such as target position, magnitude, and duration of the exposure.

The two pairs of numbers associated with ADS-FROM-OBS indicate that there is a many-to-one relationship from ARCHIVE-DATA-SET-ALL to OBSERVATION. Specifically, a pair (MIN,MAX) indicates the minimum and maximum number of times a particular entity can participate in a set of relationship instances. Thus, the notation facilitates the specification of both cardinality ratio constraints (i.e., many-to-many, many-to-one, and one-to-one) and participation constraints (i.e., total and partial). Hence, an ARCHIVE-DATA-SET-ALL entity can be related to at most one OBSERVATION entity, but need not be related to any (i.e.,

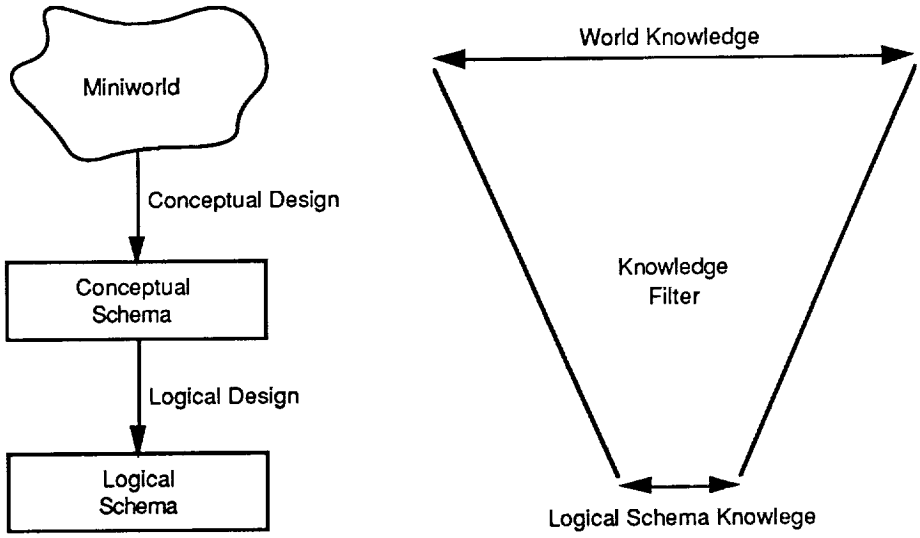


Figure 1. Knowledge lost during design.

participation is optional). On the other hand, an OBSERVATION entity must be associated with at least one ARCHIVE-DATA-SET-ALL entity (i.e., participation is total), and can be related to many.

As also shown in Figure 2, entity types (as well as some relationship types) have attributes, which are similar to the slots associated with frames, though attributes tend to have a restricted set of facets. Diagrammatically, an attribute is represented as a labeled oval connected via an edge to the ER object it characterizes. The set of attributes that can be used to uniquely identify a real-world object is known as the identifier, or key, and is shown underlined. (Note that only a few of the attributes associated with ARCHIVE-DATA-SET-ALL and OBSERVATION have been shown. Furthermore, the attribute names have been simplified to avoid confusion. As attributes do not play a central role in the discussion that follows, they will not be shown in subsequent diagrams.)

Given the simple ER specification of Figure 2, it is straightforward to create a corresponding set of relation schemas. First, entity types and their attributes are mapped to relation schemas. Then, the key attributes of the relation schema corresponding to the one-side entity type (i.e., OBSERVATION) are added as a foreign key to the relation schema corresponding to the many-side entity type (i.e., ARCHIVE-DATA-SET-ALL). The foreign key thus serves as the representation of the relationship type ARCHIVE-DATA-SET-ALL at the logical level.

The resultant relation schemas are as follows (note that logical level objects will use underscores instead of hyphens in names):

```
ARCHIVE_DATA_SET_ALL(Name,
   Archive_Class,Generation_Date,
   Access_Time,...,Update_Time,
   Program_ID,Obsnum,Obset_ID)

OBSERVATION(Program_ID,Obsnum,
   Obset_ID,Actual_Duration,...,
   Broad_Category)
```

As illustrated by the above relations, formulating even simple queries can be difficult at the logical level. For example, given a request for a list of PROGRAM-IDS, a formulator must know to use OBSERVATION instead of ARCHIVE_DATA_SET_ALL. While a heuristic could be used to favor the relation in which the attribute is an element of the key, the heuristic fails when more sophisticated knowledge representation constructs are employed. For example, a generalization lattice maps to a set of relation schemas, each of which has the same key. As a result, a request to list the key will be ambiguous at the logical level; however, as described below, this ambiguity is resolved easily at the conceptual level.

2.2. An Extended ER Model

The original ST-DADS conceptual schema used only the basic ER modeling constructs described above [Loral Aerosys 1992]. Though richer than the corresponding logical schema,

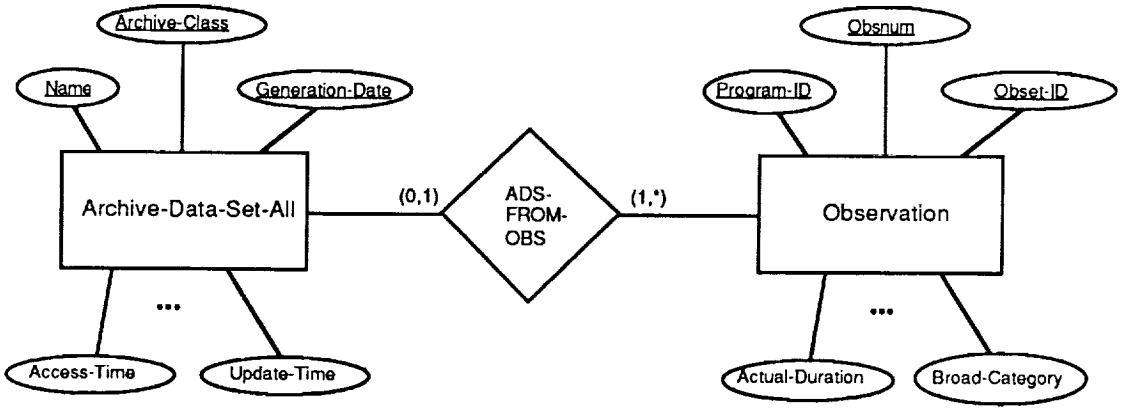


Figure 2. Simple ER objects.

the original conceptual schema did not accurately represent the ST-DADS domain. For, example the schema did not represent the fact that ARCHIVE-DATA-SET-ALL is a direct superclass of six subclasses. Instead, because generalization was not supported, simple relationships were used to associate the six subclasses to ARCHIVE-DATA-SET-ALL. However, this made it impossible to infer the hierarchical associations that actually existed. As a result, identifier attributes that should have been inherited from ARCHIVE-DATA-SET-ALL had to be explicitly replicated in the six subclasses, thus resulting in duplication of attributes and an inability to infer the primary ER object with which an attribute was associated. Furthermore, the hierarchical associations are disjoint, which means that an ARCHIVE-DATA-SET-ALL entity can participate as an entity in at most one subclass. Because the conceptual schema lacked this knowledge, a query formulator might incorrectly infer that the disjoint subclasses could be related via natural joins at the relational level.

As a result of the representational inadequacy of the basic ER model, an extended ER (EER) model has been adapted and augmented with constructs that are needed to model more complex domains [Batini et al. 1992; Elmasri and Navathe 1989; Teory 1986]. Figure 3 illustrates a portion (approximately half) of the current ST-DADS EER conceptual schema. Some of the more significant constructs are described below. In keeping with the basic ER model, entity types and basic relationship types are represented as in Figure 2.

In addition to basic entity types, weak entity types are supported in the QUICK EER model. (Weak entity types actually were defined in Chen's original model [Chen 1976]; however, many automated design tools do not support them.) Weak entity types lack a complete set of identifier attributes and, thus, a weak entity can be identified only with respect to an owner entity. For example, a DATA-SET-COMMENT entity, which provides a comment about a data set in the ST-DADS archive, can be identified only in the context of a specific ARCHIVE-DATA-SET-ALL entity. Thus, at the logical level, the key of the DATA_SET_COMMENT relation schema is the key of the ARCHIVE_DATA_SET_ALL relation schema concatenated with the partial identifier attributes from the DATA-SET-COMMENT entity type (i.e., USER-ID and COMMENT-TIME). Diagrammatically, a weak entity type is indicated by a double-edge rectangle connected to its identifying owner via an identifying relationship type, which is represented by a double-edge diamond.

Without the weak entity type abstraction, the identifier attributes of ARCHIVE-DATA-SET-ALL would have to be duplicated in the conceptual schema. With the abstraction, identifier attributes are associated only with their primary entity types. The remaining identifier attributes are, in a sense, inherited. (Note that attribute duplication cannot be avoided in the logical schema, as common attributes serve as associational links among relations.) Moreover, as illustrated in Figure 3, weak entity types can be defined in terms of other weak entity types. Thus, a set of DATA-SET-COMMENT-LINE entities makes sense only in the context of a particular DATA-SET-COMMENT entity. As before, identifier attributes are inherited, this time directly from DATA-SET-COMMENT and indirectly from ARCHIVE-DATA-SET-ALL.

Note that the weak entity type inheritance described above is different from the classical notion of inheritance in A.I. or object-oriented systems [Booch 1991; Rich and Knight 1991]. That is, a DATA-SET-COMMENT entity is not an ARCHIVE-DATA-SET-ALL entity; rather, a set of DATA-SET-COMMENT entities exists to support an ARCHIVE-DATA-SET-ALL entity, and thus is identified in the context of an ARCHIVE-DATA-SET-ALL entity. Consequently, only the identifier of the owner entity type is inherited as opposed to the complete set of attributes of the owner.

To deal with conventional inheritance, the generalization abstraction is provided, though the abstraction has been extended to deal with more complex types of inheritance as well. Diagrammatically, generalization is indicated by a labeled circle with an arrow directed toward the parent class. For example, Figure 3 illustrates that ARCHIVE-DATA-SET-ALL has six subclasses. The circle is labeled with a D,

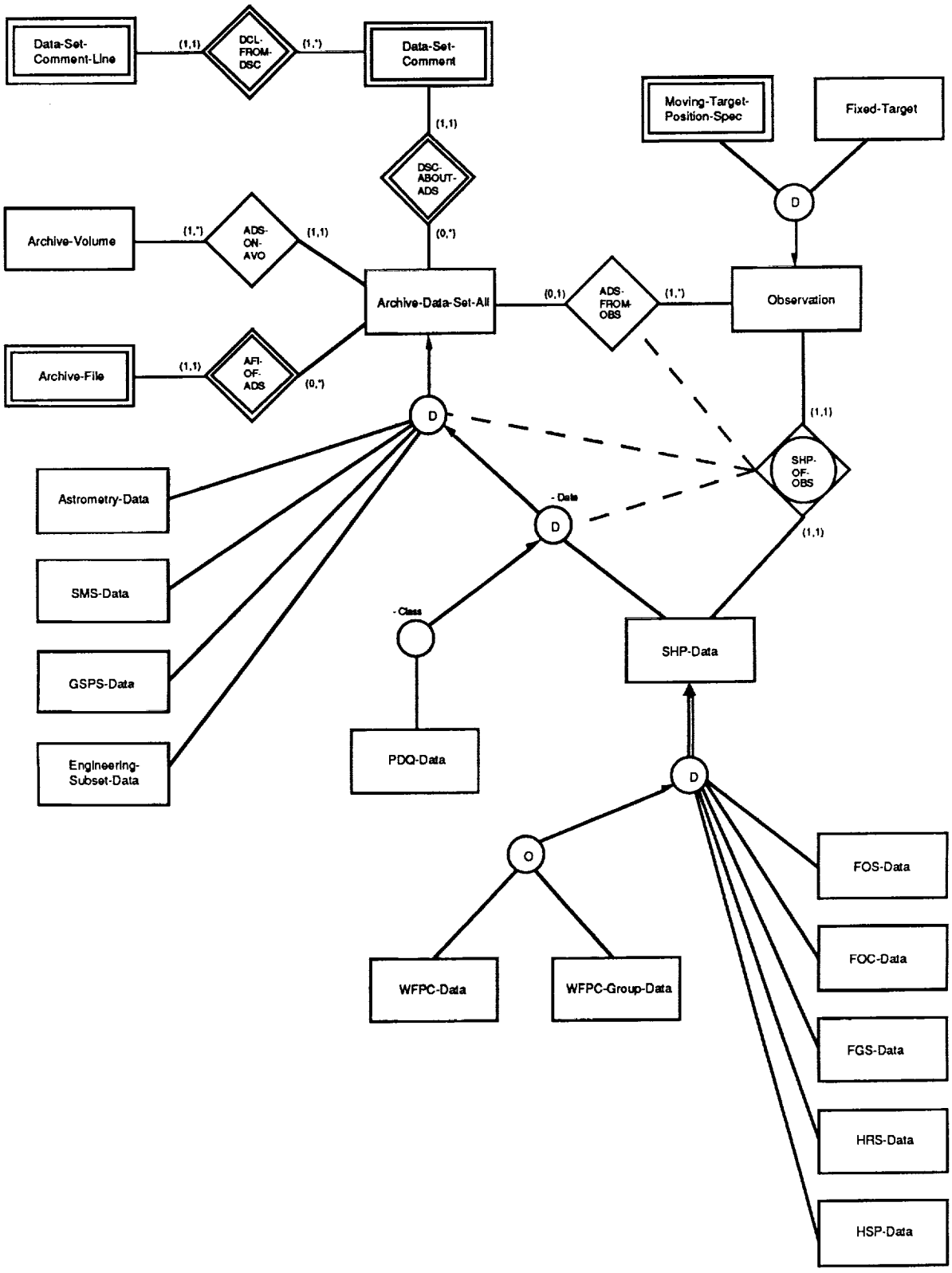


Figure 3. Portion of ST-DADS EER design.

C-4

indicating that the association is disjoint; therefore, an ARCHIVE-DATA-SET-ALL entity may participate in at most one subclass.

The four subclasses in the left portion of the diagram are direct subclasses, and thus inherit the complete set of attributes of ARCHIVE-DATA-SET-ALL. However, the two subclasses in the right portion of the diagram (i.e., SHP-DATA, a standard header packet containing telemetry values and spacecraft operation data, and PDQ-DATA, which contains product data quality data) are different. Consider, for example, the generalization type between ARCHIVE-DATA-SET-ALL and SHP-DATA. The identifiers are identical except that the DATE attribute exists in ARCHIVE-DATA-SET-ALL, but not in SHP-DATA. On cursory inspection, one might mistakenly classify ARCHIVE-DATA-SET-ALL as a weak entity type of SHP-DATA. However, the existence of an ARCHIVE-DATA-SET-ALL entity does not depend on an SHP-DATA entity. Rather, SHP-DATA is semantically a subclass of ARCHIVE-DATA-SET-ALL. In fact, the effect of removing DATE from the generalization type is to allow an SHP-DATA entity to be a child of multiple ARCHIVE-DATA-SET-ALL entities. The association of PDQ-DATA to ARCHIVE-DATA-SET-ALL is similar, except that only the NAME attribute is significant. Diagrammatically, such inheritance associations are represented as generalization types annotated with the identifier attributes that will not be inherited. If a circle is not labeled, as is the case with the PDQ-DATA generalization type, a simple subset association is indicated.

SHP-DATA has seven subclasses. However, in this case, each SHP-DATA entity must be associated with a subclass entity, as indicated by the double-edge arrow to SHP-DATA. This mandatory participation constraint stands in contrast to ARCHIVE-DATA-SET-ALL entities, which are not required to have corresponding subclass entities, as indicated by the single arrow to ARCHIVE-DATA-SET-ALL. The five subclasses on the right are direct subclasses, inheriting the identifier from SHP-DATA (and, therefore, from ARCHIVE-DATA-SET-ALL). Similarly, the two subclasses on the left (i.e., WFPC-DATA and WFPC-GROUP-DATA) correspond to an overlapping set of entities, as indicated by the o label on the generalization type circle. From a query formulation standpoint, this indicates that the relations corresponding to WFPC-DATA and WFPC-GROUP-DATA can be joined, though neither can be joined with relations corresponding to the other five subclasses of SHP-DATA. From a modeling standpoint, this representation was a concession resulting from the limitations of the underlying database management system. That is, conceptually, there is one WFPC entity type corresponding to the wide-field planetary camera; however, the huge amount of data produced by this camera requires that the relation corresponding to the conceptual WFPC entity set be split into two relations. To preserve mapping integrity from the conceptual level to the logical level, the conceptual notion of WFPC was split into two entity types at the EER level.

The disjoint association shown with OBSERVATION illustrates another facet of EER inheritance. Specifically, an OBSERVATION entity type can be classified as a fixed target or a moving target (or, strangely enough, as neither a fixed target nor a moving target as indicated by the single-edge arrow to OBSERVATION). The relationship of FIXED-TARGET to OBSERVATION is representative of the conventional inheritance association described previously. However, the relationship of MOVING-TARGET-POSITION-SPEC to OBSERVATION is different. In particular, a set of MOVING-TARGET-POSITION-SPEC entities corresponds to a single OBSERVATION, with the additional identifier attributes corresponding to date and time. Thus, a MOVING-TARGET-POSITION-SPEC is similar to a weak entity type, but, in this case, each MOVING-TARGET-POSITION-SPEC entity is an OBSERVATION entity. Furthermore, because of the disjoint generalization type, no MOVING-TARGET-POSITION-SPEC entity can exist as a FIXED-TARGET entity.

The final abstraction to be discussed is the optimization relationship type, as illustrated by SHP-OF-OBS in Figure 3. Optimization relationship types provide a means for representing efficiency decisions that have been made at the logical level to enhance

performance. Specifically, it sometimes is the case that joins of selected relations are needed on a regular basis. For example, in ST-DADS, OBSERVATION often is joined with ARCHIVE_DATA_SET_ALL, which in turn is joined with SHP_DATA. Often, ARCHIVE_DATA_SET_ALL is needed in the join only to associate OBSERVATION with SHP_DATA. As this sequence of joins is expensive, ST-DADS designers decided to improve performance by establishing a direct link from OBSERVATION to SHP-DATA. For requests involving only OBSERVATION and SHP-DATA, the direct link is used. If ARCHIVE-DATA-SET-ALL also were involved, then the longer join sequence would be used. Thus, SHP-OF-OBS serves as a direct link that informs a query formulator of the semantic equivalence of the path from OBSERVATION to SHP-DATA to the longer path from OBSERVATION through ARCHIVE-DATA-SET-ALL to SHP-DATA.

Note that if SHP-OF-OBS were represented as a simple relationship type, then a query formulator would have to infer that two different join paths existed that would result in semantically distinct results. On the other hand, specifying SHP-OF-OBS as an optimization relationship type ensures that the system can infer that the paths are equivalent, and thus serves to constrain the set of relations that will be used in the final query.

3. Contexts

From the portion of the ST-DADS EER conceptual schema shown in Figure 3, 21 relation schemas would be produced in the logical schema. However, even assuming that these 21 relation schemas constituted the complete logical schema would not enable a user employing only logical schema knowledge to query the ST-DADS system in a straightforward manner. For example, because of weak entity type and generalization type inheritance, the attribute NAME from ARCHIVE-DATA-SET-ALL appears in 17 relation schemas. Thus, if a request were made to list NAMES matching a specific pattern, a formulator would have to select some subset of the 17 relations. However, at the conceptual level, NAME occurs only in ARCHIVE-DATA-SET-ALL; consequently, a query formulator could infer immediately that only the relation corresponding to the entity type ARCHIVE-DATA-SET-ALL is needed.

In addition to attribute duplication, a formulator must be familiar with the rationale underlying the logical design to determine what is and, just as important, what is not a reasonable natural join. For example, from a purely syntactic standpoint, it would appear that any relation corresponding to a subclass of ARCHIVE-DATA-SET-ALL could be joined with any other subclass of ARCHIVE-DATA-SET-ALL because of the inherited, and, therefore, shared key attributes. However, as described in the previous section, these subclasses are disjoint and should not be joined. Once again, such knowledge is explicit in the conceptual schema.

While the conceptual schema is knowledge-rich, it is not necessarily appropriate for direct use as a database interface. For instance, the large number of attributes (i.e., several hundred) associated with many ST-DADS entity types precludes the use of graphical query languages that have been developed for various EER models [Czejdo et al. 1990; Zhang and Mendelzon 1983]. Moreover, the different views of the world held by different users, even at the conceptual level, makes it difficult to create a single, agreed-upon conceptual schema. Thus, it is better to shield the users from the conceptual schema as well as the logical schema.

By presenting a universal relation interface, users can conceive of the database as being structured as a single table of information. In turn, high-level requests must be mapped onto the underlying conceptual schema and subsequently translated into a query at the logical level. For this process to work, the underlying conceptual schema must satisfy two requirements:

1. It must be rich enough to support multiple views of the world.

2. It must map to the logical level in a straightforward way.

Given the many person-years of effort it takes to create a conceptual schema, it is reasonable to assume that Requirement 1 will be satisfied. If it is not, then, as with any knowledge acquisition task, additional effort must be expended in refining the schema. Requirement 2 is satisfied inherently by the QUICK EER model. Specifically, the abstractions in the QUICK EER model map directly to relations at the logical level in a straightforward manner. Furthermore, designers may employ various flags to control the mappings, thus ensuring that an appropriate logical schema will be produced from the conceptual schema. For example, the conceptual schema of Figure 3 was reverse-engineered from an existing logical schema. As the logical schema could not be changed, it was imperative that the conceptual schema map directly to it. By setting appropriate flags (e.g., allowing foreign key null values for selected relationship types), the desired mapping was realized without having to contrive the conceptual schema.

As there are certain join paths that are ruled out by the structures in a conceptual schema (e.g., by a disjoint generalization type), it makes sense to segment the schema into overlapping subgraphs of strongly associated objects. These subgraphs will be referred to as contexts [Semmel 1992], as they implicitly define what relations can be joined in a semantically reasonable way and, given a sufficiently rich conceptual schema, should correspond to classes of reasonable requests.

A context is maximal in the sense that no other object in the ER graph can be added to it without undermining the strong association criterion. However, determining what objects are strongly associated is not well-defined; instead, heuristics must be employed when determining strong association, as the existence of a path in the EER graph does not necessarily imply that a strong association exists among the objects in the path. Fortunately, there is a criterion for strong association at the logical level that can be abstracted to the conceptual level and used for automatic context generation. The criterion is based on a theorem in relational database theory that says if the intersection of the attributes of two relations multidetermines (or, by implication, functionally determines) one of the relations, then the two relations can be joined in a lossless manner [Korth and Silberschatz 1991].

At the conceptual level, the lossless join rule enables contexts to be defined inductively. Intuitively, a relationship type and its participating entity types form a context. This is justified by the fact that a relationship type conceptually contains the identifiers of each participating entity type; thus, a relationship type functionally determines each of its participating entity types. Similarly, an overlapping generalization type and its parent and children form a context, as each relation corresponding to an entity can be joined in a lossless manner. Finally, a disjoint generalization type with n children form n contexts, each consisting of the generalization type, the parent, and one of the n children.

Inductively, a context can be extended to include a relationship type and its participating entity types if a many-to-one or one-to-one cardinality ratio constraint exists from the context and no cycle is introduced in the extended context. This is justified by the fact that a localized functional dependency can be inferred from the entity type in the context to the other participating entity types of the relationship type. Similarly, a context can be extended to include a generalization type and one or more of its children. If the generalization type is disjoint, then for each child, a new context is formed consisting of the old context, the generalization type, and the child. If the generalization type is overlapping, then the original context is extended to include the generalization type and all of its children. As before, cycles are avoided, as joining a relation with itself at the logical level is superfluous.

The final way to extend a context is based on the notion of an articulation point. That is, if the removal of a relationship type disconnects the EER conceptual schema, then the relationship type and its participating entity types are included in the context. The justification for the disconnection rule is based on the conversion of an EER conceptual schema to a hypergraph and inferring multivalued dependencies when the hypergraph becomes

disconnected after removing a hyperedge corresponding to the relationship type [Ullman 1989].

Note that optimization relationship types are not considered when contexts are created. Rather, they are included as part of any context that has the bypassed association types specified by the optimization relationship type. Thus, in Figure 3, any context that included the three association types connected by dashed lines to SHP-OF-OBS would include SHP-OF-OBS as well. It is the role of the query formulator to use its knowledge of optimization relationship types to determine which set of joins is appropriate when the final query is constructed.

In the portion of the ST-DADS EER conceptual schema shown in Figure 3, there are 22 contexts. To see this, note that 11 contexts can be derived from the subclasses of ARCHIVE-DATA-SET-ALL. Four of these contexts are derived from the four subclasses on the left side of the diagram (i.e., ASTROMETRY-DATA, SMS-DATA, GSPS-DATA, and ENGINEERING-SUBSET-DATA). One of the contexts is derived from PDQ-DATA. Finally, six of the contexts are derived from SHP-DATA (i.e., one from the overlapping WFPC entity types and five from the remaining five subclasses of SHP-DATA). Then, for each of the 11 contexts, two new contexts are formed as the OBSERVATION generalization type is extended through to reach MOVING-TARGET-POSITION-SPEC and FIXED-TARGET. Thus, 22 contexts are produced. Of these 22 contexts, eight contain 15 EER objects, two contain 17 EER objects, ten contain 18 EER objects, and two contain 20 EER objects.

There are two more points worth mentioning with respect to contexts. First, in the worst case, the time it takes to generate contexts is an exponential function of the number of association types (i.e., relationship types and generalization types) in the EER conceptual schema. However, real-world designs tend to exhibit constraints that can be exploited to make automatic generation tractable (e.g., acyclic extensions can be initially pruned and then reintroduced once at the end of the context generation process). Exploiting these constraints for the complete ST-DADS EER conceptual schema enables contexts to be generated in less than one minute on a Lisp-based prototype of QUICK running on a Sun SPARCStation 2. As contexts need be generated only when the conceptual schema is created or modified, this performance is acceptable. The second point is that as contexts are generated heuristically, the contexts produced may not be consistent with designer expectations. In this case, the automatically generated set can be used as a starting point, and the set of contexts can be handcrafted. However, to this point in time, there has been no need for such handcrafting.

4. Automated Query Formulation

Given a set of contexts, query formulation is straightforward. First, the attributes in the high-level request are determined. Then, each context that covers the set of requested attributes is found. To ensure that needless joins are not performed, the found contexts are iteratively pruned of leaves until all leaves cover requested attributes. As a result of pruning, duplicate contexts may be introduced. As duplicate contexts are superfluous, they are eliminated. Then, the natural join orders of the remaining contexts are found. Finding these orders is straightforward, as the edges in the remaining context subgraphs identify valid natural join paths. The EER objects in each ordered context then are mapped to their underlying relation schemas. As some EER objects are represented by the same relation schema (e.g., in Figure 2, ADS-FROM-OBS and ARCHIVE-DATA-SET-ALL both map to the ARCHIVE_DATA_SET_ALL relation schema), duplicate relation schemas are eliminated. Then, subqueries are formulated and the union of the subqueries is returned as the final query.

Note that only when the conceptual schema contains cycles will more than one context be involved in the generation of the final query. This follows from the fact that only one path can exist between any two nodes in an acyclic graph, and, therefore, only one subtree will connect some set of nodes. The fact that there are multiple contexts does not affect this property, as context pruning will result in identical subtrees that, in turn, will be

eliminated. As optimization relationship types do not participate in the generation of contexts, no cycles exist in the portion of the ST-DADS conceptual schema shown in Figure 3.

To clarify the query formulation process, consider the following request:

> For all observations made by the faint object spectrograph between January 1, 1992 and February 1, 1992, display the target's description, right ascension, declination, proper motion and red shift, the instrument's detector and position angle of the aperture, and the relevant archived dataset names and comments about the datasets.

Recall that the StarView interface enables users to select attributes and qualify them on an ad hoc query screen. After the request specification is complete, it is translated into a form that QUICK can process. The language used by QUICK is referred to as USQL, as it resembles SQL, but assumes the existence of a universal relation. Thus, StarView passes the following USQL request to QUICK:

```
Select  target-descrip,
        ra,
        dec,
        ra-proper-motion,
        redshift,
        detector,
        pa-aper,
        data-set-name,
        line-text
Where   start-time >=
         "Jan 1, 1992"  And
        stop-time <=
         "Feb 1, 1992"  And
        instrume = "FOS"
```

QUICK begins processing by finding the applicable contexts. In this case, one context applies, as FOS-DATA is needed to cover DETECTOR, and FIXED-TARGET is needed to cover RA-PROPER-MOTION and REDSHIFT. As DATA-SET-COMMENT-LINE is needed to cover LINE-TEXT, only ARCHIVE-VOLUME, ADS-ON-AVO, ARCHIVE-FILE, and AFI-OF-ADS are pruned from the found context. Next, a natural join order is found for the conceptual schema objects of the pruned context:

```
(FIXED-TARGET,OBSERVATION-GT,
 OBSERVATION,ADS-FROM-OBS,
 ARCHIVE-DATA-SET-ALL,DSC-ABOUT-
 ADS,DATA-SET-COMMENT,DCL-FROM-
 DSC,DATA-SET-COMMENT-LINE,
 ARCHIVE-DATA-SET-ALL-GT,
 ARCHIVE-DATA-SET-ALL-SHP-GT,SHP-
 DATA,SHP-GT,FOS-DATA)
```

As will be discussed in more detail in the next example, QUICK semantically optimizes queries to ensure that the number of joins in the final query is minimized. For this example, QUICK recognizes that no attributes are requested from DATA-SET-COMMENT, which is used only to connect DATA-SET-COMMENT-LINE to ARCHIVE-DATA-SET-ALL. As DATA-SET-COMMENT-LINE inherits (via a weak entity type association) the identifier attributes of DATA-SET-COMMENT, the entity type DATA-SET-COMMENT and its identifying relationship type can be eliminated and replaced by a virtual link from ARCHIVE-DATA-SET-ALL to DCL-FROM-DSC. After mapping the remaining conceptual objects to the logical level, the following sequence of relation schemas is produced:

```
(FIXED_TARGET,OBSERVATION,
 ARCHIVE_DATA_SET_ALL,
 DATA_SET_COMMENT_LINE,
 SHP_DATA,FOS_DATA)
```

QUICK now exploits its knowledge of attribute mappings from the conceptual level to the logical level as well as its knowledge of natural join associations among relations to produce the final query. Specific to ST-DADS is the fact that designers provided distinct relation schema prefixes to all attributes, thus giving the impression that distinct conceptual attributes exist at the logical level. QUICK compensates for this by preserving attribute uniqueness at the conceptual level, recording prefix information, and inserting appropriate prefixes only when the final query is generated. However, QUICK also takes advantage of the fact that the logical-level attributes are syntactically unique to avoid relation

qualification. Here, then, is the final query generated in response to the request:

```
SELECT
        obs_target_descrip,
        obs_ra,
        obs_dec,
        fit_ra_proper_motion,
        fit_redshift,
        fos_detector,
        shp_pa_aper,
        ads_data_set_name,
        dcl_line_text
FROM
        fixed_target,
        observation,
        archive_data_set_all,
        data_set_comment_line,
        shp_data,
        fos_data
WHERE
        obs_start_time >=
          "Jan 1, 1992"  AND
        obs_stop_time <=
          "Feb 1, 1992"  AND
        shp_instrume =
          "FOS"  AND
        fit_program_id =
          obs_program_id  AND
        fit_obset_id =
          obs_obset_id  AND
        fit_obsnum =
          obs_obsnum  AND
        obs_program_id =
          ads_program_id  AND
        obs_obset_id =
          ads_obset_id  AND
        obs_obsnum =
          ads_obsnum  AND
        ads_archive_class =
          dcl_archive_class  AND
        ads_data_set_name =
          dcl_data_set_name  AND
        ads_generation_date =
          dcl_generation_date AND
        ads_archive_class =
          shp_archive_class  AND
        ads_data_set_name =
          shp_data_set_name  AND
        shp_archive_class =
          fos_archive_class  AND
        shp_data_set_name =
          fos_data_set_name
```

The complexity of the above query clearly demonstrates the need for an automated query formulation capability. Moreover, it takes only one to two seconds to generate the query, and the query is optimal with respect to the number of joins required. With relations as large as those in ST-DADS, join optimality is critical. However, achieving join optimality can be troublesome when code is generated from a high-level request and only logical level knowledge is available. Consequently, QUICK semantically optimizes final contexts via heuristic pruning to ensure that generated queries use the minimum number of relations and, thereby, require the minimum number of joins.

To gain an appreciation of QUICK's semantic query optimization capabilities, consider the following request:

```
List the data set comments for
astrometry data collected
during February 1993.
```

In USQL, the query can be expressed as follows:

```
Select  line-text,
        target-name,
        data-set-name
Where   generation-date >=
         "Feb 1, 1993"  And
        generation-date <
         "Mar 1, 1993"
```

As in the example above, first all contexts that cover the requested attributes are found. In this case, two contexts apply, both of which include ASTROMETRY-DATA, and one of which includes MOVING-TARGET-POSITION-SPEC and the other of which includes FIXED-TARGET. However, after pruning and duplicate elimination only one context remains:

```
(ASTROMETRY-DATA,ARCHIVE-DATA-
 SET-ALL-GT,ARCHIVE-DATA-SET-
 ALL,DSC-ABOUT-ADS,DATA-SET-
 COMMENT,DCL-FROM-DSC,DATA-SET-
 COMMENT)
```

At this point the conceptual schema objects are placed in natural join order and mapped to

relation schemas:

```
(ASTROMETRY_DATA, ARCHIVE_
 DATA_SET_ALL,DATA_SET_COMMENT,
 DATA_SET_COMMENT)
```

Without semantic query optimization, the following query is generated:

```
SELECT
        dcl_line_text,
        ast_target_name,
        ast_data_set_name
FROM
        astrometry_data,
        archive_data_set_all,
        data_set_comment,
        data_set_comment_line
WHERE
        ads_generation_date >=
         "Feb 1, 1993"  AND
        ads_generation_date <
         "Mar 1, 1993"  AND
        ast_data_set_name =
          ads_data_set_name  AND
        ast_archive_class =
          ads_archive_class  AND
        ast_generation_date =
          ads_generation_date AND
        ads_data_set_name =
          dsc_data_set_name  AND
        ads_archive_class =
          dsc_archive_class  AND
        ads_generation_date =
          dsc_generation_date AND
        dsc_data_set_name =
          dcl_data_set_name  AND
        dsc_archive_class =
          dcl_archive_class  AND
        dsc_generation_date =
          dcl_generation_date AND
        dsc_comment_time =
          dcl_comment_time  AND
        dsc_user_id =
          dcl_user_id
```

The above query produces a semantically valid result; however, the query can be improved. To see how, first note that the identifier of ARCHIVE-DATA-SET-ALL is inherited by ASTROMETRY-DATA, by DATA-SET-COMMENT, and (indirectly) by DATA-SET-COMMENT-LINE. Furthermore, the existence of an identifier value in any of the inheriting entity types implies that the value exists in an entity in ARCHIVE-DATA-SET-ALL. Similarly, the existence of an identifier value in DATA-SET-COMMENT implies that the value exists in an entity in DATA-SET-COMMENT-LINE. Thus, DATA-SET-COMMENT and ARCHIVE-DATA-SET-ALL serve only to associate ASTROMETRY-DATA with DATA-SET-COMMENT-LINE. That is, the conceptual level interpretation indicates that the join of the relations corresponding to ASTROMETRY-DATA and DATA-SET-COMMENT-LINE is equivalent to the join of the original four relations. Furthermore, as the attribute GENERATION-DATE was requested in the USQL WHERE clause, and is an identifier attribute of ARCHIVE-DATA-SET-ALL that is inherited, it can be derived from either ASTROMETRY-DATA or DATA-SET-COMMENT-LINE. Here, then, is the final query:

```
SELECT
        dcl_line_text,
        ast_target_name,
        ast_data_set_name
FROM
        astrometry_data,
        data_set_comment_line
WHERE
        ast_generation_date >=
         "Feb 1, 1993"  AND
        ast_generation_date <
         "Mar 1, 1993"  AND
        ast_data_set_name =
          dcl_data_set_name  AND
        ast_archive_class =
          dcl_archive_class  AND
        ast_generation_date =
          dcl_generation_date
```

There are several points worth noting about the above examples. First, the queries demonstrate the need for an interface that simplifies request specification. In this regard, USQL is a step in the right direction, but a higher level interface like StarView still is needed. Second, the use of conceptual knowledge ensures that semantically reasonable queries will be generated from high-level requests. Third, the queries are generated quickly, which facilitates user interaction with the interface; in fact, even with full optimization, queries typically are generated in less than two

seconds. Finally, the queries generally will execute as efficiently as handcrafted queries produced by experts.

5. Summary and Conclusions

The complexity of the ST-DADS logical schema prohibits most StarView users from generating semantically valid queries that correspond to ad hoc requests. However, by exploiting conceptual design knowledge, methods have been developed for automating query formulation and enabling a user to perceive the database as a universal relation. These methods have been incorporated in QUICK and rely on an extended ER model that enhances the current ST-DADS conceptual schema, thus ensuring that the conceptual schema will play a central role throughout the database life cycle.

By exploiting the notion of contexts, QUICK ensures that generated queries will be semantically reasonable. Moreover, the queries will be generated quickly and execute efficiently. The capabilities of QUICK also enable StarView designers to focus on interface issues instead of conceptual design issues. For example, user modeling can be addressed in the framework of a universal relation schema, thus simplifying the task of interface construction. Consequently, interface designers need not develop a deep understanding of the conceptual schema (which is described by a 900-page database design document [Loral Aerosys 1992]).

The first release of StarView is scheduled for May of 1993. Currently, QUICK is being integrated into StarView and will exist as a module in the May release. QUICK generates SQL queries for all of the screens provided by StarView as well as for the ad hoc field sets selected through the universal relation interface.

Current research efforts are focused in three areas. First, contexts are being extended to support arbitrary predicates. Such predicates could be used, for example, to restrict context access or to disambiguate among multiple contexts that apply to a request. Second, richer request structures are being explored. For example, to generate queries of arbitrary complexity (e.g., queries requiring the cartesian product of two relations), tuple variables are necessary. However, it is not clear how tuple variables can be presented to a user by the StarView interface in an intuitive manner. Furthermore, inferring tuple variables is a difficult problem for which no solution is apparent. Finally, some consideration is being given to alternative conceptual and logical models, which would facilitate the use of QUICK with object-oriented databases.

6. References

Batini, C., Ceri, S., and Navathe, S. B. 1992. *Conceptual Database Design: An Entity-Relationship Approach.* Benjamin/Cummings, Redwood City, CA.

Booch, G. 1991. *Object-Oriented Design with Applications.* Benjamin/Cummings, Redwood City, CA.

Chen, P. P. 1976. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems 1*, 1, 9-36.

Czejdo, B., Elmasri, R., Rusinkiewicz, M., and Embley, D. W. 1990. A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. *Computer 23*, 3, 26-36.

Elmasri, R., and Navathe, S. B. 1989. *Fundamentals of Database Systems.* Addison-Wesley, Reading, MA.

Hull, R., and King, R. 1987. Semantic Database Modeling: Survey, Applications, and Research Issues. *ACM Computing Surveys 19*, 3, 201-260.

Korth, H. F., and Silberschatz, A. 1991. *Database System Concepts*, 2nd ed. McGraw-Hill, New York.

Leymann, F. 1989. A Survey of the Universal Relation Model. *Data & Knowledge Engineering 4*, 4, 305-320.

Loral Aerosys. 1992. *ST DADS Database Design Specification*, Build 2, Revision B.

Maier, D., Ullman, J. D., and Vardi, M. Y. 1984. On the Foundations of the Universal Relation Model. *ACM Transactions on Database Systems 9*, 2, 283-308.

Peckham, J., and Maryanski, F. 1988. Semantic Data Models. *ACM Computing Surveys 20*, 3, 153-189.

Rich, E., and Knight, K. 1991. *Artificial Intelligence*. McGraw Hill, New York.

Semmel, R. D. 1992. "Discovering Context in a Conceptual Schema," in *Proceedings of the First International Conference on Information and Knowledge Management* (Baltimore, Nov. 8-11), Yesha, Y., ed., International Society of Mini and Microcomputers - ISMM, pp. 222-230.

Teory, T. J., Yang, D., and Fry, J. P. 1986. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys 18*, 2, 197-222.

Ullman, J. D. 1989. *Principles of Database and Knowledge-Base Systems*, Vol. 2. Computer Science Press, Rockville, MD.

Zhang, Z., and Mendelzon, A. O. 1983. A Graphical Query Language for Entity-Relationship Databases. In *Proceedings of the 3rd International Conference on Entity-Relationship Approach* (Anaheim, CA, Oct. 5-7), pp. 441-448.

Call for Papers

NASA 1994

Goddard Conference on Space Applications of Artificial Intelligence

May 1994
NASA Goddard Space Flight Center
Greenbelt, Maryland

The Ninth 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
- Simulation-based reasoning
- 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, 1993 along with the author's name, affiliation, address and telephone number. Notification of tentative acceptance will be given by September 16, 1993. Papers should be no longer than 15 pages and must be submitted in camera-ready form for final acceptance by November 16, 1993.

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

1994 Goddard Conference on Space Applications of Artificial Intelligence
May 1994 — NASA/GSFC, Greenbelt, MD

❑ ***Abstracts due:*** Sept. 1, 1993
❑ ***Papers due:*** Nov. 16, 1993
❑ ***Further info:*** (301) 286-3150
❑ ***FAX:*** (301) 286-4627

❑ ***Send abstracts to:***
Mike Moore
NASA/GSFC Code 522.1
Greenbelt, MD 20771
moore@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 1993	Conference Publication

4. TITLE AND SUBTITLE
1993 Goddard Conference on Space Applications of Artificial Intelligence

5. FUNDING NUMBERS
530

6. AUTHOR(S)
Carl F. Hostetter, Editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Mission Operations & Data Systems Directorate
NASA/GSFC, Code 500
Greenbelt, Md 20771

8. PERFORMING ORGANIZATION REPORT NUMBER
93B00040
Code 530

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546

10. SPONSORING MONITORING AGENCY REPORT NUMBER
NASA CP-3200

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 63

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This publication comprises the papers presented at the 1993 Goddard Conference on Space Applications of Artificial Intelligence held at the NASA/Goddard Space Flight Center, Greenbelt, Md on May 10 - 13, 1993. 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
300

16. PRICE CODE
A13

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL