FINAL REPORT
INTELLIGENT MULTI-SENSOR INTEGRATION

Principal Investigators

Richard A. Volz
Ramesh Jain
Terry Weymouth

August 3, 1989

Grant number NAG 2-350
and
JPL Grant Number 958086

INTELLIGENT MULTI-SENSOR INTEGRATION

Richard A. Volz, Ramesh Jain, Terry Weymouth

Grant number NAG 2–350
and
JPL Grant number 958086

1 Introduction

Growth in the intelligence of space systems requires the use and integration of data from multiple sensors. Work on this project is directed toward the development of generic tools for extracting and integrating information obtained from multiple sources. The work addresses the full spectrum of issues ranging from data acquisition, to characterization of sensor data, to adaptive systems for utilizing the data. In particular, there are three major aspects to the project, multi-sensor processing, an adaptive approach to object recognition, and distributed sensor system integration.

2 Hyper-Pyramids—*Jain*

In a complex mult-robot multi-sensor system, each robot and sensor will have an *ego-world* model. This model will contain an *ego-centered* description of the world. Knowledge about the goals, current states, and strategies of other robots and sensors will also be a part of the ego-world model. As more information is acquired, the model will be updated. Based on the ego-world model, each module will decide what it can do to improve its model and to help other modules in accompalishing their task. Based on this reasoning, appropriate information will be communicated to other modules.

At the most detailed level, one may require volumetric representation of a scene in the world model. This representation will be independent of individual sensor locations at a particular time instant. This exhaustive representation may be used to store characteristics, such as color, compliance, and temperature, of the entity at that location in the space.

In this hierarchical model, each sensor has its own representations, but communicates with the world model using 3-D spatial information. The lowest level of representation should contain different properties at points in space. For this level, a volumetric representation is needed. The highest level should contain information about objects, their properties, and relations among the objects. This symbolic level should be related to the exhausive volumetric level. In between these two extremes there should be several resolutions. This representation is shown in Figure 1. In the next section we discuss how hyperpyramids can be used to represent lower levels in the world representation.

2.1 The Representation

We would like to define a space-efficient data structure which can represent property values defined at various points in a 3-D workspace in such a way that connected regions of similar property values, called *3-D property segments*, can be efficiently accessed and processed with respect to a given semantics. This semantics should be one in which 3-D regions are represented in a hierarchical manner in terms of subregions, the leaves being these 3-D property segments. Such a representation of semantically meaningful regions will lend itself to the identification of these regions with real world objects in that properties of objects and relationships among objects can easily be computed from similar computations on their subobjects, these computations being realized on their corresponding regions in the data structure.

Such a data structure has been formulated. Particularized to a single property, it resembles pyramids [Ant82,CiD84,GrJ86] and octrees [JaT80]. Like octrees, when a region of space is reached for which the given property has a uniform value, it is not further subdivided. Like relinkable pyramids, a given node has a set of possible fathers. We refer to this factored data structure as a *relinkable octree*.

Our unfactored data structure, which is called a *hyper-pyramid*, can be considered to be a collection of relinkable octrees, one for each represented property. It consists of a *backbone*, which is a standard octree of processing element (p.e.) nodes. Each backbone node has a non-empty set of associated property nodes. The backbone nodes hold information that is common to each of its associated properties, such as its coordinates. For a given property, the set of property nodes forms a relinkable octree. The details of these data structures are given in [JGr87].

Our approach allows the definition of multiple properties, each property having one of the 4 datatypes *integer*, *real*, *string*, or *greylevel*. Each property has values defined over the workspace, which is divided into $R \times R \times R$ voxels, for R the resolution of the workspace. This resolution may be changed at any time.

Property values may be input in 2 ways:

1. The entire workspace may have values specified. This consists of an input file containing R^3 values. Any previous values of the property are destroyed.

2. A particular value may be specified for a particular plane, row, and column. This approach may be used by various sensors to input their values. Note that by redefining the resolution, entire regions of constant values may be input in a single step. For example, suppose a $4 \times 4 \times 4$ workspace has been input via Method 1 above. Changing the resolution to 2 and specifying a value for $plane = 1$, $row = 1$, and $column = 1$ will then input this value for a $2 \times 2 \times 2$ set of voxels.

A given property's values may also be segmented as previously described based on a user defined notion of closeness of values. This performs relinking so that each 3-D property segment corresponds to a subtree whose leaves are the actual voxels belonging to this segment and whose root is as close to the hyper-pyramid root as possible. The notion of hidden segments are also supported so that each segment may be accessed as efficiently as possible. Our present approach to segmentation is more powerful than our previous one [GrJ86] for the following reasons:

1. Our underlying data structure is octree-based rather than pyramid-based, resulting in savings in space utilization.

2. There are no disabled nodes. Nodes which cannot be relinked cause extra nodes in the data structure. These extra nodes are minimized by a more powerful relinking algorithm. This algorithm causes fewer nodes which would have become disabled in our previous approach.

3. Our algorithm goes through only as many passes as it needs to for the segmentation.

4. Any relinking done after a change in property values is restricted to a small neighborhood surrounding each of the changes. Thus, incremental changes in the workspace result in an incremental amount of effort spent in relinking.

3 Current Work: Oct-Trees

We are currently working on the development of algorithms for the implementation and manipulation of the world model data structure itself, and the development of new techniques for recovering depth information from the environment using grey-scale imagery. Our efforts with the world model have primarily involved the development of algorithms that construct and manipulate oct-trees and similar structures, as this is a logical first step towards the development of full-scale hyper-pyramids. Some algorithms, however, have been implemented for the input of different properties and relinking for segmentation based on properties for the hyper-pyramidal representation. The work done on depth recovery has focused on the MCSO (moving camera, stationary objects) scenario, where camera motion is precisely known, and is concentrated on a technique that does not rely explicitly on the solution of the correspondence problem.

3.1 The Intersection Algorithm

Currently, the primary focus of this portion of our research is on implementing algorithms to build Oct-Tree representations for an unknown static domain. This is done in a manner similar to [SrA87] by acquiring information from different observation points and intersecting this information with that in the current world model. Initially, the algorithm assumes the entire space to be non-navigable or *FULL*. Then, as new information is acquired, *FULL* areas are "cut-off" and replaced by navigable (*VOID*) space.

Input to the algorithm consits of a camera location and orientation, and a set of three-dimensional points produced by some depth recovery procedure. The algorithm uses this information to create a *convex-hull* of all the points received and assumes the space in the scope of the camera between the camera and the closest side of the convex-hull is *VOID*. Next, an intersection procedure similar to that in [SrA87] is employed. New *VOID* nodes are added to the Oct-Tree (if appropriate) and the whole structure is updated. Our procedure differs from that in [SrA87] in that:

- The focus of our efforts is to correctly locate the navigable paths in the domain while they are interested in describing a single object whose location is known.

- The input to our system is three-dimensional information.

Obviously, the performance of such an algorithm is strongly linked to the value of the information it receives as its input.

3.2 Preliminary Results

We have simulated the algorithm in an artificial domain of size $64 \times 64 \times 64$ which contained several simple polyhedral objects. The algorithm was supplied with information "obtained" from 25 sensor locations within the domain, with the world model being updated after each "observation". Figures 2 through 5 show the status of the world model after 1, 10, 20 and 25 iterations. Our results show that all convex features in the scene are correctly identified by the algorithm. However, concave features (i.e. corners of a room) are harder for the algorithm to locate (as would be expected). It should be noted that the algorithm did not produce any *false positive* errors (i.e. the space indicated as *VOID* was always empty).

3.3 Future Efforts

For sake of simplicity, our initial experiments have involved a static domain. Future versions of this algorithm will, however, not be restricted in such a manner. A primary difference between the current algorithm and one capable of dealing with a dynamic environment will be the addition of a routine to deal with conflicts between sensory data and information in the current world model. Such a procedure would identify data due to new or moving objects as such, and update the model accordingly. Future versions will also be extended to introduce some form of uncertainty measure into our data structure.

We are also working on the implementation of representations which are more object oriented (hyper-pyramids). We intend to implement an algorithm similar to that discussed above on the hyper-pyramid (H-P) structure. Our long-term aim is to try and assimilate H-P relinking within the Oct-Tree structure so one may reap the benefits of the H-P's object-oriented representation, while keeping the uniformity of the Oct-Tree structure.

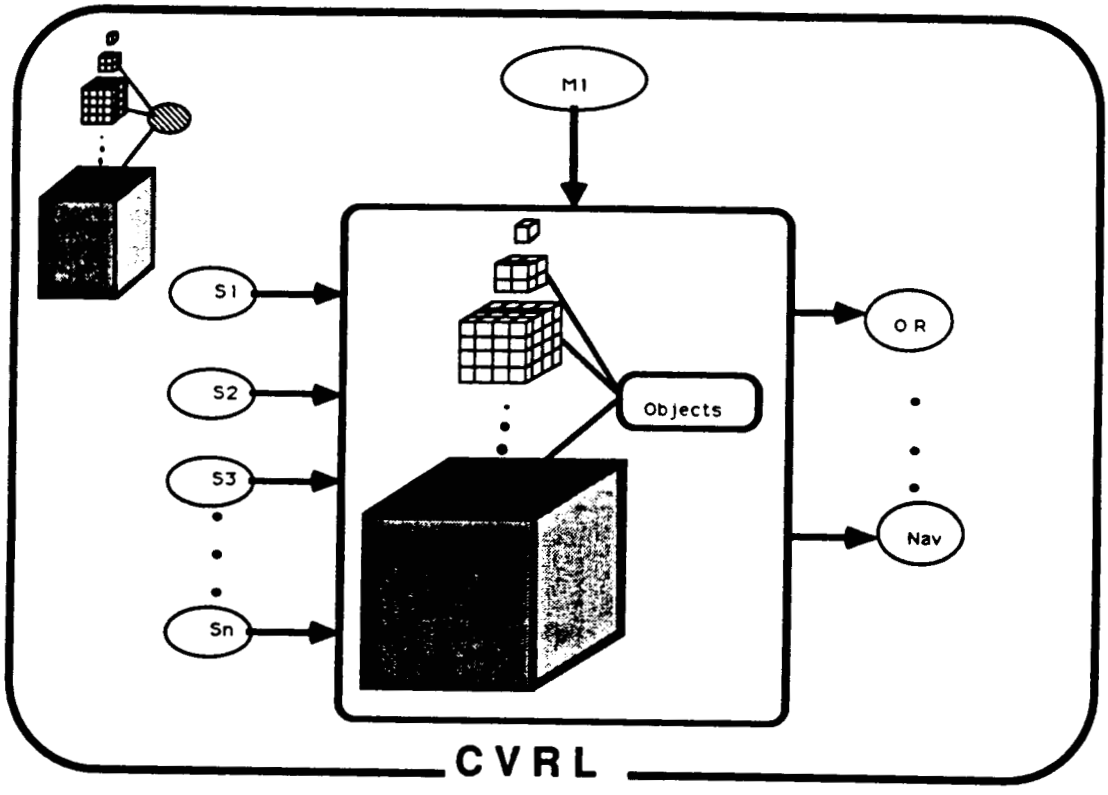

Figure 1. The Hyper-Pyramid at several resolutions.

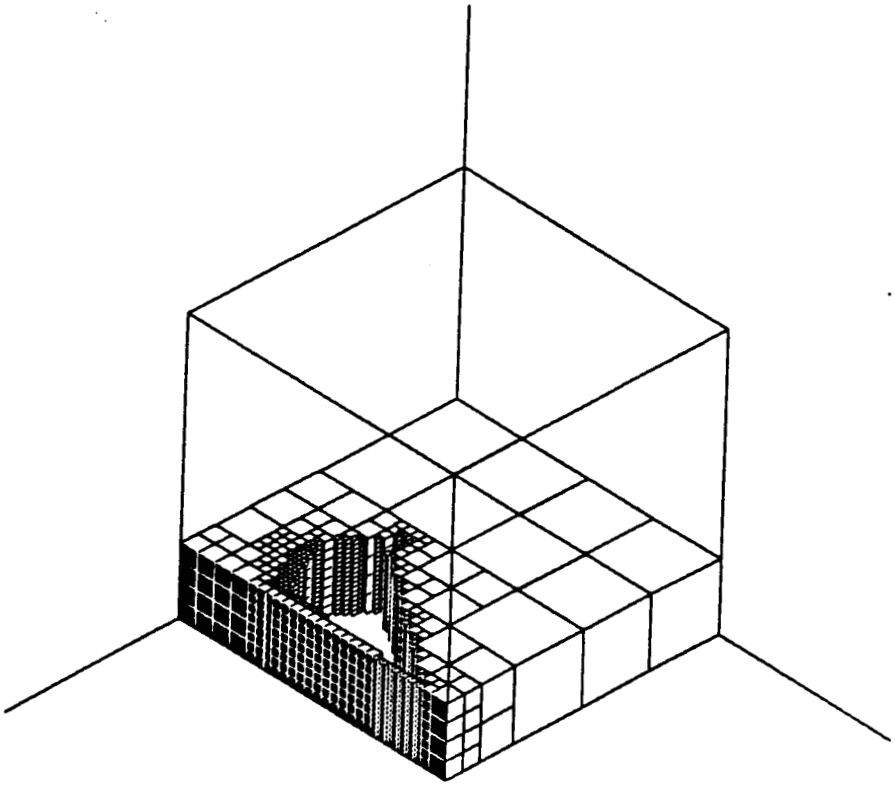

Figure 2. The world model after 1 iteration of the intersection algorithm.

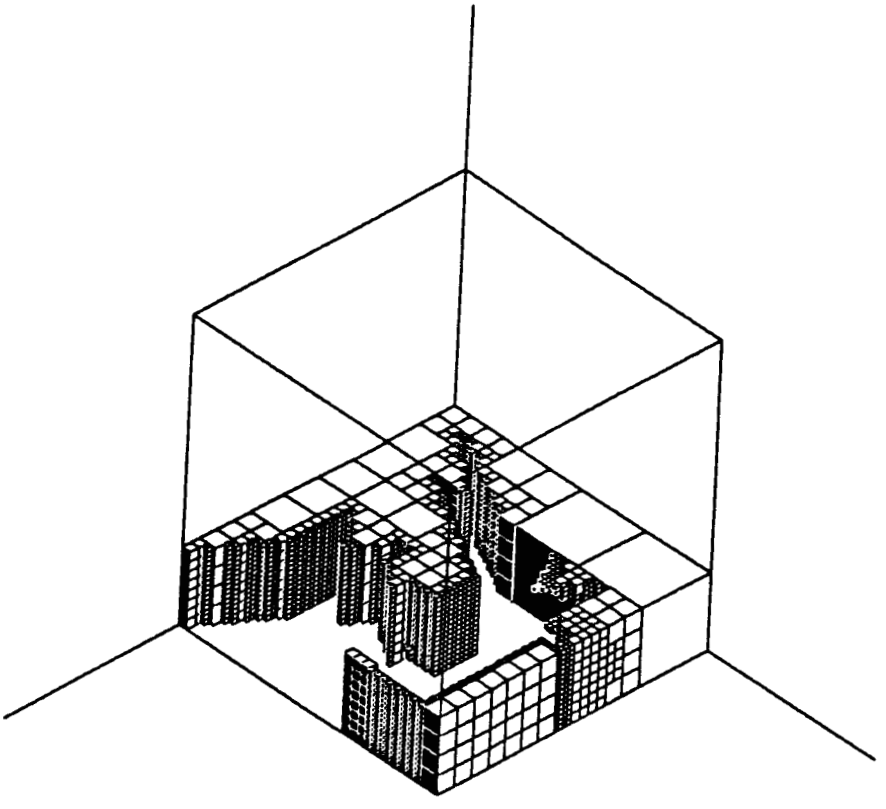

Figure 3. The world model after 10 iterations of the intersection algorithm.

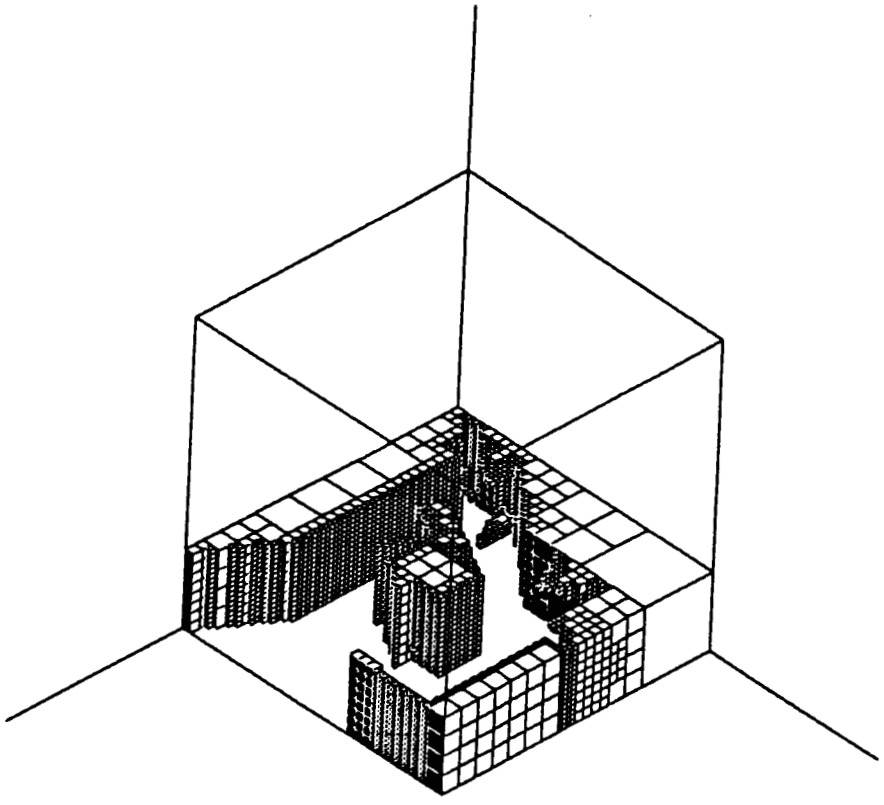

Figure 4. The world model after 20 iterations of the intersection algorithm.

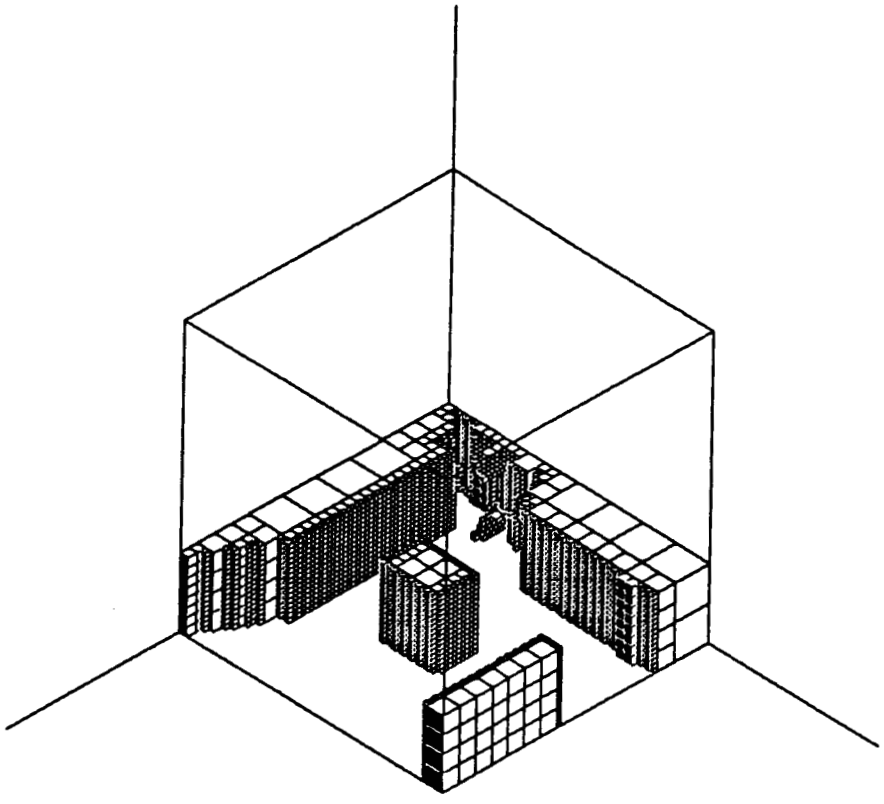

Figure 5. The world model after 25 iterations of the intersection algorithm. The true locations of the obstacles are shaded.

4 Stereo For Navagation—*Weymouth*

Stereo is a desirable means of getting accurate depth information at a distance. However, algorithms for stereo must cope with problems due to noise and the process of searching for correspondance. Because of missing data and mismatched data, a single pair of images is not a reliable source of depth information. One solution is to integrate the information from several image pairs. By incrementally refining estimates of the depth as the camera changes position, we can build a description of the scene which, in turn, can be used as feedback to improve the extraction of depth information.

We have developed a stereo algorithm based on correlation matching that is especially suited to to being integrated into a proposed feedback loop. We have begun experiments with the integration of depth information over a sequence into a single environmental map.

In the typical stereo camera arrangement, two identical cameras with identical fixed-focal-length lenses are mounted so that their image planes are coplanar, having parallel y and z axes and collinear x axes, as shown in Figure 1, viewed from above. Depth can be determined from disparity, which is the measure of the horizontal displacement that an object feature (such as an edge) undergoes between images (as the images were overlaid). Specifically, if the column position of the feature in the left image as measured from the left side of the image is c_l and in the right image is c_r, then the disparity is $d = c_l - c_r$. Note: under the assumptions given, the disparity is always positive.

The relation between disparity and depth is

$$z = \frac{ef}{d}$$

where z is the depth, e is the separation between cameras, f is the focal length of the cameras, and d is the disparity. A derivation for this relation can be developed along the following lines. Consider the distance between two lines, one connecting the focal point of the left camera with a distant point and the other connecting the focal point of the right camera with that point (Figure 1). Let this distance, D, be measured parallel to the image plane of the two cameras. At the focal points $z = 0$ and $D = e$. At the image plane $z = f$ and $D = e + c_r - c_l = e - d$. Since D is linear with respect to z, a general linear expression for D as a function of z that satisfies the first two conditions is

$$D = \frac{1}{f}(z)(e - d) + \frac{1}{f}(f - z)e.$$

Setting $D = 0$ we have that $(z)(e - d) + (f - z)(e) = 0$ or $fe = zd$ then

$$z = \frac{ef}{d}$$

as was desired.

Two observations follow from this relationship between disparity and depth. These can also be seen in the geometry of the cameras. First, for a fixed camera arrangement depth is inversely proportional to disparity; the closer an object gets to the cameras the larger the disparity will be. This has practical implications, it is desirable for the disparities to be small because in solving the correspondence problem there is a searching process involved. The larger the disparities are,

the more often there will be incorrect matches. Thus, objects of interest should be far enough away to result in small disparities. This goal is balanced by the fact that the depth of further objects is less accurately determined than the depth of closer objects, making it desirable to have objects of interest closer. Second, for a given value of disparity, depth is proportional to the constant determined by the product of the focal length, f, and the separation between cameras, e. When this constant is larger (the cameras are further separated, cameras with longer focal length are used, or both), then the accuracy of depth determination increases. This is the basis for our decision to use wide base-line stereo. The consequence of this choice is the need for some method to deal with the potentially large search range needed to discover a correspondence match.

4.1 A Stereo Algorithm: Approximation and Refinement

In our current work we are developing portions of a system which will construct a description of object surfaces over time from a sequence of stereo image pairs (Figure 2). This paper presents the framework of that system and some preliminary experiments within that framework. In our current research we are focusing on two aspects of the overall system. The development of a stereo algorithm that incorporates feedback and the development of a means for assimilation of depth estimates from stereo into a consistent description of depth events in the scene (e.g. points upon scene surfaces).

In our proposed system three processes interact: depth estimation from stereo pairs, the generation of a consistent and current map of surface points in the environment, and the estimation of the camera position. Stereo pairs and the current estimation of surface depth for all the objects in the environment are the inputs to a process, Depth Estimation, which produces an estimate of the current depth to every visible surface in the scene. The camera model and stereo depth points are the input to a process that maintains and updates an estimate of the position of surface points in the environment, Generation of Environmental Map. Finally, the depth estimations from the current frame, and the current estimation of surface depth from the environmental map and any additional position information are the inputs to the process that updates the camera model (Camera Model).

The interacting processes communicate through three sets of parameters: the estimate of camera position, the computation of depth points, and the environmental map. With any two of these sets of information, we can compute the third. However, in general, we have only an estimate of all three. The algorithms we are developing will use the relations among these estimates to incrementally improve each one. For a small change in position, the overall depth map will not change too much and the estimates of camera motion are reasonably accurate; this information guides the computation of depth for the new frame pair. The depth information from the new pair and the estimates of camera position allow the updating of the environmental depth map. Closing the circle, the process of matching the depth information from one stereo pair to environmental map contributes the estimate of the camera position.

With each successive frame pair, guided by the current environmental map, the depth estimation algorithm is able to produce a depth map. Assuming that the system is not in a "start up" state, most of the new points in the depth map should fit the surfaces in the current description.

In this case they can be used to refine those descriptions. Each surface "claims" some subset of the depth points from the depth map, by virtue of the fact that those points are within a certain distance of the surface described by the surface patch. With those points the surface patch description is updated: the points are used to refine the surface fit and an error measure of that fit is updated. This assimilation of incoming points continues until a threshold on the error is exceeded. When that threshold is acceded then a different surface patch (or patches) must be generated to account for the data. This method of surface patch growing has been used successfully in segmenting depth images [Besl87]. When the amount of accumulated error exceeds a threshold, then the current patch no longer adequately describes the data and the description of that data must be accommodated to the new data.

There are two classes of problems with this approach: the character of depth value given by current stereo algorithms and the problem of perceptual groping associated with the construction and maintenance of the environmental map. Stereo algorithms tend to generate sparsely spaced depth values where the spacing of the depth values depends on the type of features used. For algorithms based on correlation of discontinuities the values tend to be clustered at the edge of surfaces. To overcome this, we are investigating stereo algorithms which are area based; this paper presents an early version of one algorithm in that investigation. We also believe that the surface information from the environmental map can be used to interpolate missing values, but this remains to be investigated. This perceptual grouping problem have mostly to do with false starts and ill-defined grouping. We are currently investigating the use of a blackboard architecture to allow the opportunistic pursuit of multiple plausible partial solutions [Nii86]. There is also some evidence that a blackboard architecture might be well suited to the tracking characteristics of this type of problem [Wey87a,Wey87b].

We are guided in our design of these modules by the principle of least committment [Mar82]. Each process functions in generally the same way. The approximation produced by a given process is expressed as a parametric representation appropriate to that process. Thus the result of the stereo matching are expressed in a camera centered depth map, the environmental map is expressed as a volumetric representation with clusters of points linked to a surface description, and the camera parameters are represented as transformation and rotation changes between camera positions. These approximations are refined as additional data becomes available. As long as a consistent set of parameters explain the incoming data, those parameter values are used. An example of this approach, used to track moving objects over a long sequence, is illustrated in the work of Haynes [Hay86].

4.2 Current Work: Depth for Navagation

Currently, we are developing an area-based stereo algorithm. Area-based techniques result in a dense disparity map making them very desirable. But these techniques suffer from a number of limitations, among them Medioni and Nevatia [Med85] note the following:

- They require the presence of detectable texture within each window; therefore, they tend to fail in featureless or repetitive texture environment.
- They tend to be confused by the presence of a surface discontinuity in a correlation window.

- They are sensitive to absolute intensity, contrast and illumination.
- They get confused in rapidly changing depth fields (e.q. vegetation)

The absence of feature, or the lack of unique texture, can be avoided if a large window size is used for the matching. However, a large correlation window size is more likely to fail at the locations in which significant occlusion is occurring. This tension between large and small windows exists in many vision algorithms. The common approach to this problem has been the use of pyramids, i.e. a coarse-to-fine strategy. Many stereo disparity algorithms have used this approach ([Gla83] and [Ana87] among others), both to resolve the window size problem and to reduce the amount of computation. But in some images, textures and features tend to disappear in the smoothing process which is the integral part of the coarse-to-fine strategies. This might lead to wrong matches at the coarse resolution level of the pyramid. Therefore, although variable size windows are necessary in an area-based approach, using pyramids might not be the appropriate method for achieving this capability. A solution for overcoming this problem is to start with large areas and proceed to smaller areas when necessary, but to carry out the matching computation at the finest resolution only. This is the method we used in an initial algorithm for stereo matching in our system. We start with large, non-overlapping image patches in one image, and find the best match in the second image. If the measure by which we determine the quality of our match does not pass a certain test then the patch is divided into four equally sized patches and the search for best match is carried out for all four pieces independently.

4.3 Preliminary Results

The algorithm, although very simple, gives fairly good results for the type of domain we are interested in. It has been tested on a number of stereo-pairs picked from the image-pair sequence taken from an indoor scene[Wey88]. A summary of the algorithm is shown below:

1. Divide the image into large square patches.
2. For each patch do the following:
   (a) Using intensity correlation, find the best match for this patch in the other image.
   (b) If correlation of the best match is below a threshold, then divide the patch into four patches of equal size and and for each patch do the following: If the correlation of the current match is below a threshold, repeat 2(a) and 2(b) recursively; otherwise the current match is the best match.

For our experiments we have obtained stereo-pair image sequences of an indoor scene. Additional experiments are planned, including outdoor scene data. The indoor stereo-pairs were taken by a 50-cm base-line two-camera system that produced images which between them there is only horizontal displacement. The system moved in an straight line for a total of seven meters and the stereo-pairs were recorded at 10 cm intervals. The first eight images of the left view of the image-sequence are shown in Figure 3; each frame is (480 $\times$ 480 pixels). A perspective display of the indoor scene, is shown in Figure 5 where the position and height of all the objects are depicted as a functional surface.

The stereo matching algorithm was tested on the sequence. The first and eight pairs from the sequence are shown in Figure 4. The disparity computation was carried out starting with 64×64 non-overlapping blocks down to a block size of 4×4. The matching criteria used was a normalized correlation with a threshold of 99.5%. The disparity maps of the first and eighth image-pair are shown in Figure 6. The quality of the matching can be better observed by reconstructing the left view from the right image based on the computed disparities (i.e warping the right image). The result for this type of registration is shown in Figures 7. Using the disparity measurements and the camera parameters the $x - y - z$ coordinates of all points were computed in each view. The three dimensional space was quantized into columns of size $10 \times 10\ cm^2$ in the $x - z$ domain with the center of coordinate at the location of camera in the first view. All the points in 3-D space were therefore assigned to the appropriate bin based on their $x - z$ coordinates, and the average height of all points in each bin is determined. A top view of this environmental map can be generated by displaying the bucket counts as a binary image – thresholding those buckets with few contributions. A perspective display can be generated by creating surfaces that have the average height associates with the buckets above a threshold. The top view and the perspective display of the result for the first and eight image-pair is shown in Figure 8.

Although the method appears very similar to multi-level pyramid approach it has some distinct properties which are not found in such approaches. One, as mentioned before, is the fact that non-prominent forms or textures will not disappear, since no or very little pre-smoothing of the image is done. In the pyramid approach the images are smoothed while here large structures that have relatively similar 2D projections in both left and right images can be matched with less ambiguity. Another distinct difference between this method and the pyramid approach is that, if a good match is found for a large piece of the image, the disparity found is accurate to the pixel in the image of highest resolution and there is no need to break the piece into smaller areas. This is not true in the coarse-to-fine strategies. The disparity found at the coarse resolution in a pyramid approach is not as accurate and must be refined by projection and remeasurement at each of the levels of finer resolution; furthermore, the refinements are only loosely tied to the context of the matches at the higher levels of resolution. Thus, if a surface does not have fine grain texture, the finer resolution matches are without data and smoothing or some other error correcting mechanism is necessary.

4.4 Future Work

We propose to develop as part of a system for navagation a sub-system in which the interrelations of the three modules compute a depth map by integrating the data from several frame pairs of a sequence. Such a sub-system will require the simultaneous development of several interacting modules. We propose to use a *blackboard* system for this development. Blackboard systems are inherently flexible and modular, making them ideal for the design of systems where the development involves some degree of *experimental programming* [Nii86]. They are also an ideal medium for the integration of diverse modules, especially when the data communicated among those modules is well known in advance.

Each stage of development in this project will be accompanied by experimental trials on

real and laboratory data (such as the data shown in Figure 5). Each stage in the algorithm development will have a related goal in an experimental plan. The interaction of the design with real data is the single most important source of new ideas [Wey87a]. We have presented a description of our framework and a report on our early results.

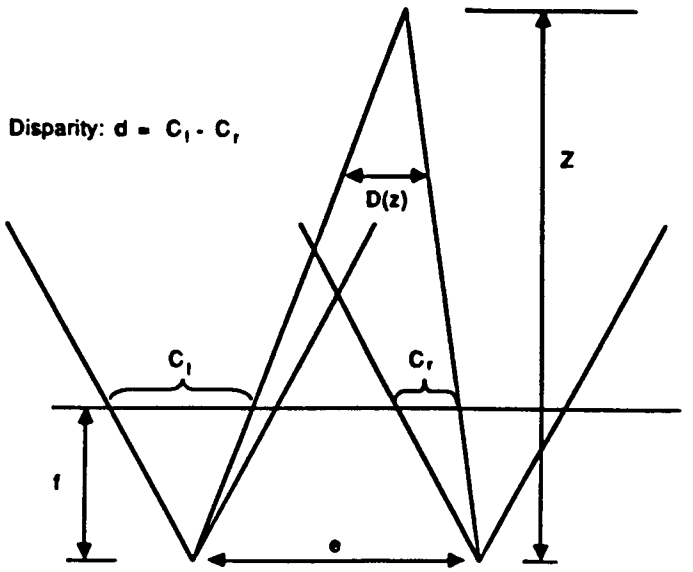

Figure 1: The geometry of the stereo pair of images can be used to compute depth from disparity.

Figures

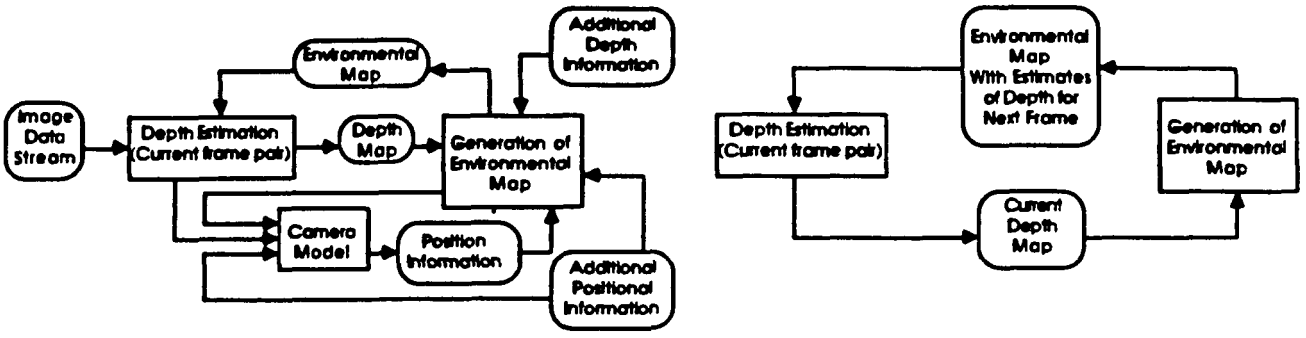

Figure 2: In the system, Each process supplies approximations that guide the other processes. These approximations, together with the data, are combined to produce a better and more timely estimation of parameters. The refined approximations, in turn, assist in better approximations of actual values in the data.

5 Distributed Sensor System Integration—*Volz*

5.1 Problem Discussion

Distributed sensors and distributed computing will be the cornerstone of most future NASA systems. However, the distributed nature of such systems adds immensely to an already complex real-time situation. It is the management of the complexity of the large scale, real-time, embedded data acquisition and artificial intelligence systmes based on distributed computing that is our goal in this part of the project.

Digital communication networks are obviously necessary, but represent only the beginnings of a solution. The principal tools needed are software development tools, and central to software development is the language used. Language is the level at which the operation of distributed systems and algorithms are made explicit, and is thus the natural level at which to view systems as an integrated whole. It is also the level at which programming-in-the-large support is first made evident, the level at which most automated error checking occurs, and the level that currently has the greatest impact upon software costs.

Software tools for distributed systems must deal with both diverse hardware and the use of existing software written in a wide variety of languages. They must also incorporate the best techniques developed in software engineering over the past two decades and extend these concepts effectively for use in distributed systems[1] It is our hypothesis that extending these concepts

[1]Among the key concepts are: 1) data encapsulation and hiding, 2) abstract data types, 3) modularization of programs, 4) separate compilation (of both modules and specifications), 5) concurrency mechanisms at the language level, and 6) extensive compile time error checking.

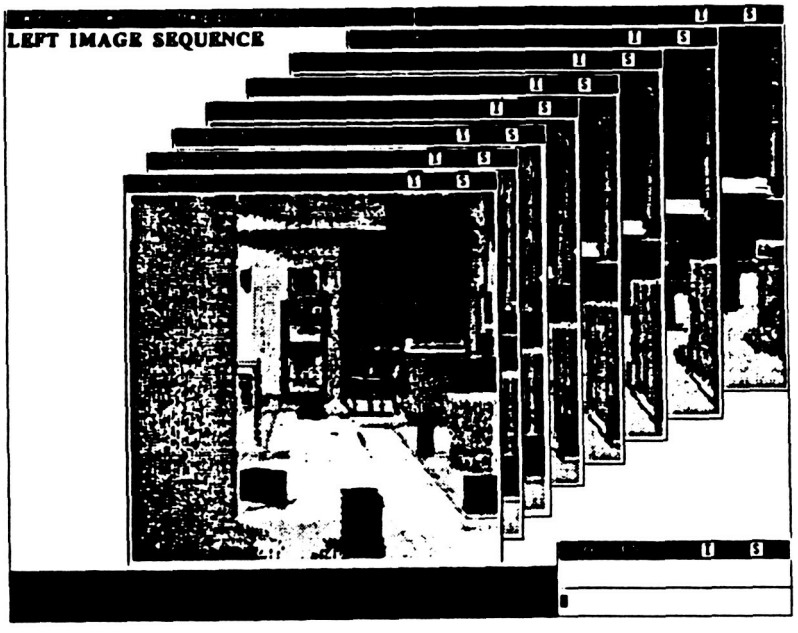

Figure 3: This figures shows the first eight frames of the left image of the sequence of stereo-pair images being used in our test and development.

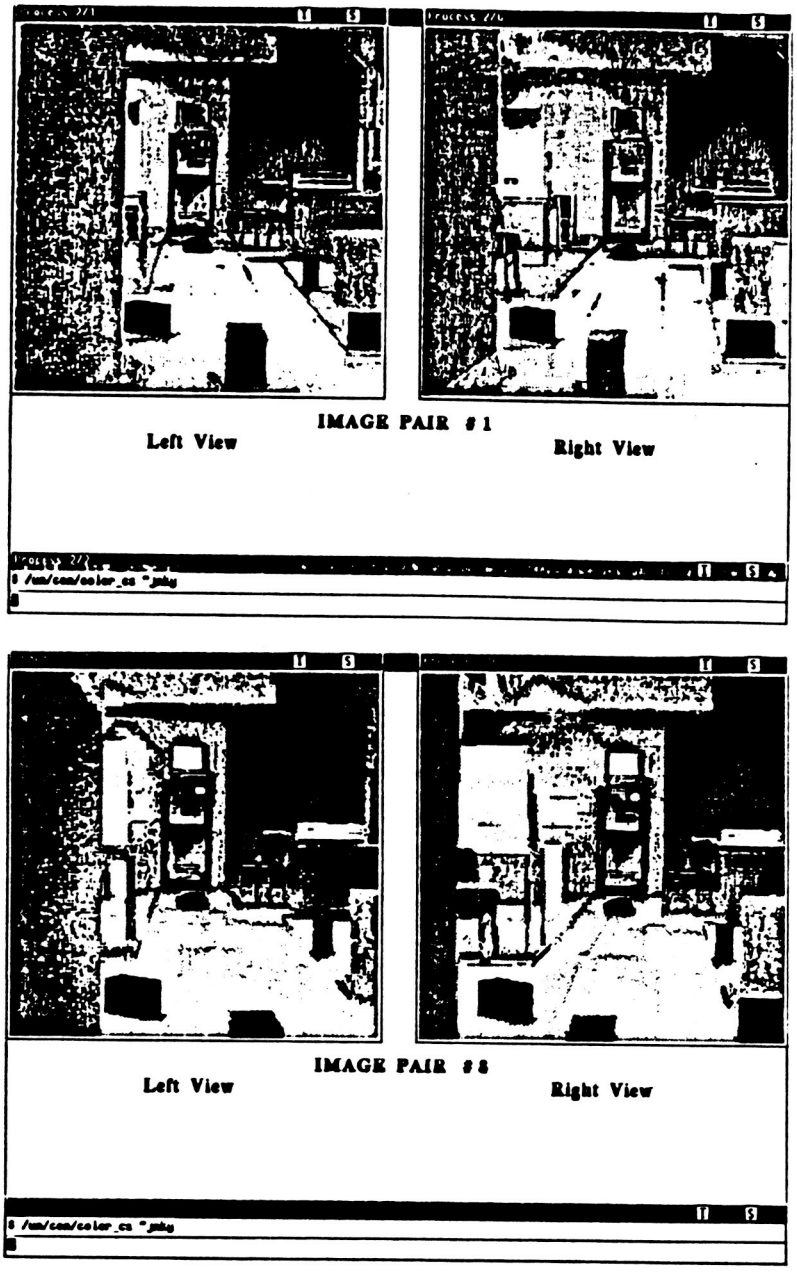

Figure 4: The first and eight image-pair of the sequence; selected to illustrate the effect of the stereo matching algorithm.

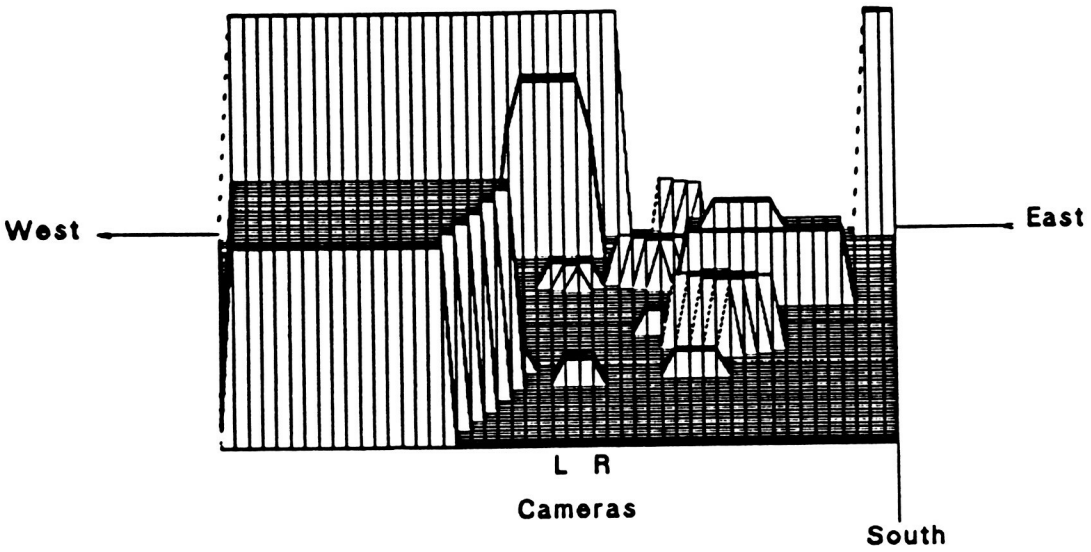

Figure 5: A perspective view of the objects in our lab set up; generated by making a functional surface of the maximum height of each object. The bottom view shows the scene from roughly the same perspective as the image.

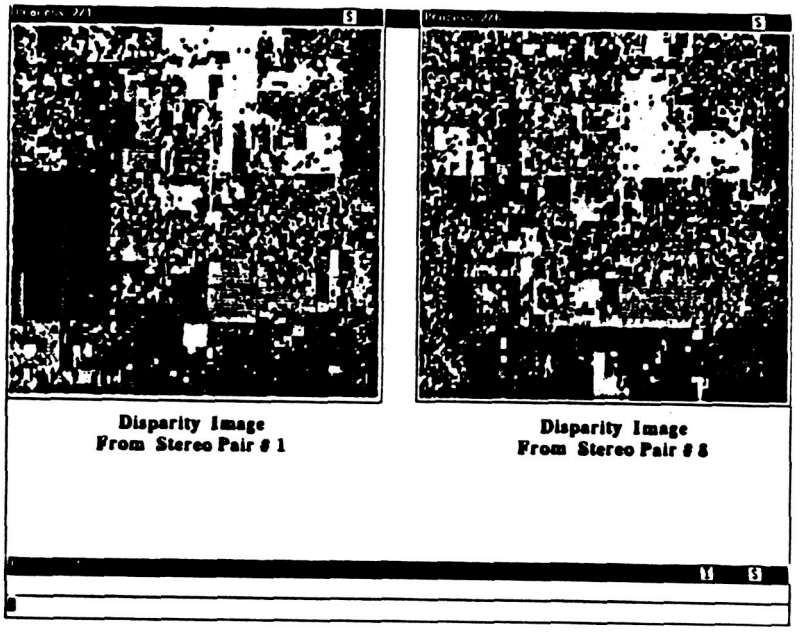

Figure 6: The depth images (from stereo disparity) for the first and eight image-pair. Depth is encoded as brightness. The black areas are those for which no disparity could be computed.

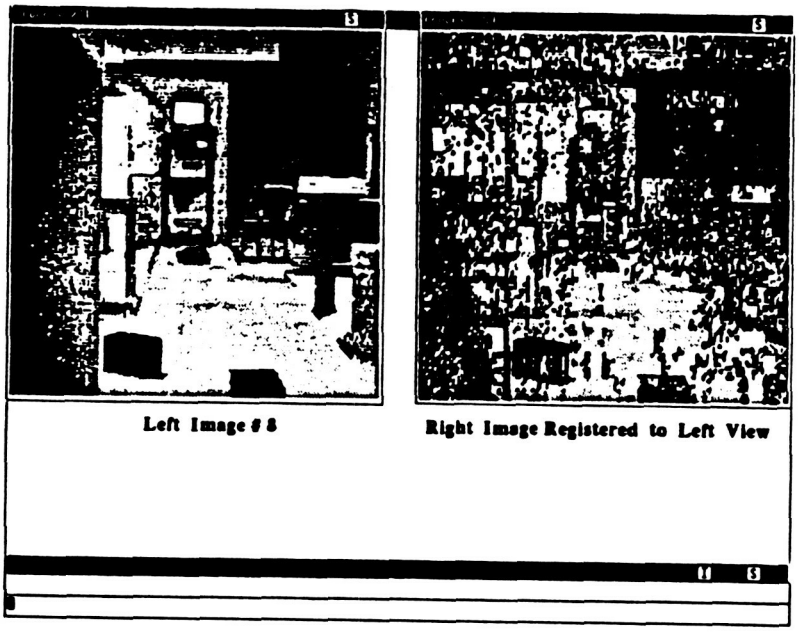

Figure 7: The disparity values can be used to compute a warped image. The right image is mapped to a warped image by displacing the pixels of the right image by the computed disparity values; it is shown here with the corresponding left image for comparison.

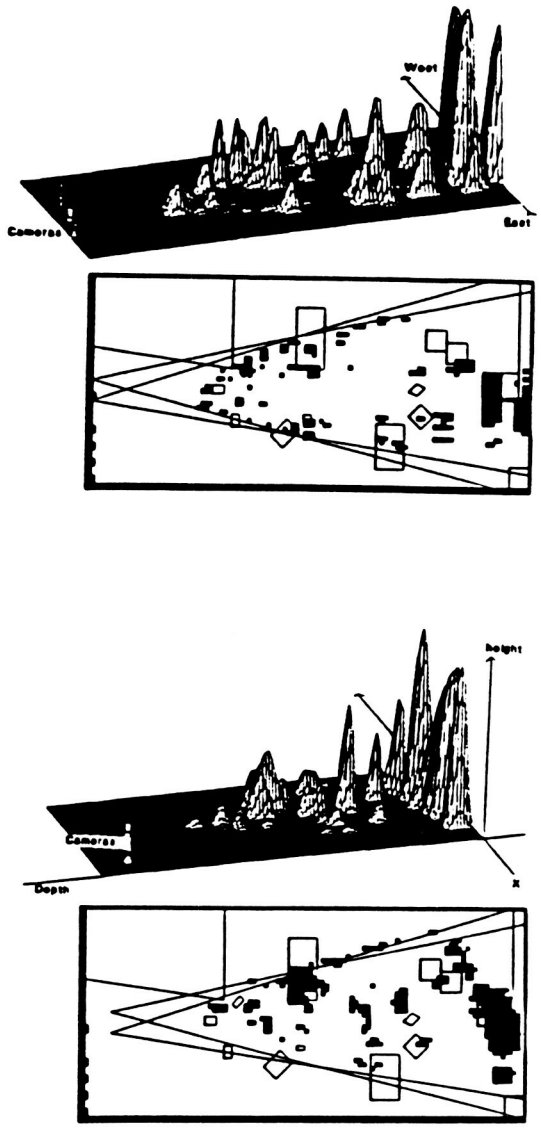

Figure 8: The environmental map can be projected to the floor plan to compare discovered surfaces with known object placement: (a) for the first frame and (c) for the second. Alternatively, a perspective view of the depth information can be generated by computing a functional surface from the average height information in the environmental map.

to permit distributed execution is a critical step in addressing the distributed computing software problem. Two obvious benefits of so doing are the extension of compile time error checking across machine boundaries and allowing the programmer to use normal language mechanisms for expressing parallel (concurrent) operation without having to invent new application level communication protocols. (Note that computer communication networks do *not* adequately address the applications level protocol issue.) These two advantages alone should greatly improve the problem of developing software for distributed systems.

Summary of Previous Status

The implementation of a system to support distributed program execution for real-time applications requires the solution to a substantial number of problems. Previously, we have discussed the following basic issues:

- Problem analysis—the basic issues in distributed program execution, [VMB87],
- Timing mechanisms—basic software timing mechanisms amongst distributed tasks, [VoM87a], and new instruction level mechanisms for simplifying timing implementation, [VoM87b],
- Performance evaluation—specifically oriented toward the real-time performance of emitted code, [CDV86],
- Experimental implementation techniques, [VMN85,VKT87], and
- Application of these ideas to manufacturing software, [NaV87].

The capabilities of the translation system at the time of the last report included the distribution of library packages and subprograms with remote access available to:

- both simple and complex data objects. Record and array objects can be nested arbitrarily deeply within one another and include pointers to remote objects.
- subprograms
- declared task objects (no timed or conditional calls across processor boundaries).

The distribution is accomplished statically among a set of homogeneous processors. There is also a Unix make-file like capability to simplify use of the translation system. Tests have been successfully completed with up to three VAX processors cooperating on the execution of a single program.

Progress During the Past Year

During this past year, we have concentrated on the continued development of techniques for our second generation translator, and porting it to a network of IBM PC/AT computers.

Translator Status and Issues

During the past year, we have dealt with the following issues in the translation system:

- Distribution of task objects created from task types,
- Instantiation of generics on distributed processors,

- Renaming,
- Modifying the underlying mailbox system to use interrupts, and
- revising the type handling mechanism to be more efficient and accept the more general initialization and default specifications.

Presently, the system allows task objects created from task types (in the specification of a library package) to be placed on any site in the system. The handling of generics is nearly completed. And, renaming works for all of the more common situations. Those remaining are logically straightforward, but require substantial effort to implement.

Generics and task types raised basically the same problem: One does not know at the time they are submitted for compilation which object references are local and which are remote. While it would be possible to utilize a general addressing mechanism that checks upon each reference whether it is local or remote, the associated overhead would be prohibitive for those that are, indeed, local. The problem is most easily seen with an example. Consider the following (recall that we use a **pragma** called SITE to specify the location on which each library unit is to execute.):

```
pragma SITE (2);
package A is
    task type T is
        entry E(..);
    end;
      ⋮
end A;

with B;
package body A is
    I: INTEGER;
    task body T is
          ⋮
    begin
          ⋮
        B.P;
          ⋮
        I := .. ;
          ⋮
    end T;
         ⋮
end A;
    ⋮
pragma SITE(1);
```

```
with A;
procedure M is
T1: A.T;
   :
begin
   :
end;
```

The task object T1 created on site 1 (in procedure M) references procedure P in package B. Suppose, now, that procedure M is submitted for compilation after package body B. Then the site of the procedure using task type T is unknown. Hence, it is unknown whether package B is on the same site as the procedure that will use T or not. Thus, it cannot be known at compile time whether the reference B.P is local or remote. Worse, since the task type T may be used by several different types, the reference may be local on one and remote on others. Further, package B may include yet other packages or subprograms. Of course, there can be many such packages referenced by the body of A.

Obviously, then, multiple versions of the task body are required, unless one is willing to accept the overhead of generalized addressing. Still, one does not necessarily know at the time that the body of A is submitted for compilation how to compile task body T. The answer is deferred compilation of copies of the task body T. Each time a unit is compiled that creates an object of type T, a check is made to see if an instance of T has yet been created for the site upon which the unit will reside; if not, a new version of T is created for that site. Fortunately, the complexity of the number of copies of task body T that must be generated is linear in the number of sites in the system.

The above example exposes another issue in the implementation of distributed task objects from task type. The hidden variable I, in the body of package A, becomes a shared variable across the network. Moreover, the remote accesses to it from the various task objects created on the network are hidden. That is, the programmer does not necessarily know that they are there. This latter fact may or may not be important, but if the programmer is developing a real-time system, the timing difference between local and remote accesses may well be important. There is no resolution to these issues within the current language definition.

There is no particular difficulty in handling renaming clauses. However, due to the large number of ways in which renaming can be utilized, the modifications to the translator required to handle all of them are extensive and have little to do with distributed execution. Since our primary intent is learn more about distributed execution, we have chosen to handle the more common cases and ignore (at least for the time being) the rest.

The initial versions of the underlying Ada compiler available to us did not support interrupts. Thus, our initial version of the mailbox used a polling scheme for communication. Shortly before the beginning of the project year we received an upgraded compiler that does support interrupts. A revised system using the interrupts was created. This reduced our communication times somewhat, though not as much as we had anticipated. Benchmark tests were developed to study the communication times. It was found that the Ethernet message times were much longer than expected, being on the order of 30 ms, roundtrip. This completely swamps any overhead

introduced by our translation system, and remote rendezvous times are thus also on the order of 30 ms.

A more significant problem with interrupts arose due to the implementation of the interrupts. The Ada reference manual allows two forms of interrupt handling, those that queue pending interrupts (and correspond to normal task entry calls), and those that do not queue the interrupts (corresponding to conditional entry calls). Unfortunately, the underlying compiler implementation with which we were working used the latter implementation, and from time to time, interrupts, and hence messages, are lost. This problem is serious, but not one of the translation system. Rather, it is one of the underlying network and compiler used. We thus, do not consider it further here.

A final problem that arose during the year relates to initialization of variables or default variables in records. An example will clarify the situation.

```
pragma SITE(1);
with A;
with B;
procedure M is
    X: A.R;
    ..
begin
..
    B(..);
end;
```

```
pragma SITE(2);
with A;
procedure B is
    Y: A.R;
    ..
begin
    ..
end;
```

```
pragma SITE(3);
with C;
package A is
    type R is
        record
        U: INTEGER := C.F(..);
        ..
    end record;
    ..
end A;
```

Suppose that the function C.F(..) has side effects. Then the order in which the calls are made can affect the value returned. On a single processor the elaboration order would be C, A, B, M. However, in the distributed system, this elaboration order need not necessarily occur unless extra controls are developed and the wrong values could be placed in the variables. This suggests strongly that initializing things using functions having side effects is very poor and should be eliminated. Moreover, in our particular implementation, we replicate the types part of distributed packages on the sites that use it. That exacerbates the problem considerably. We have found that it is possible to get around this difficulty, but it involves the creation of a number of additional support packages and extra variables we call *shadow* variables. The operational overhead is not high, but the complexity of the translation system is increased substantially. A better solution, we believe, is to modify the language to prohibit initialization of variables or defaults using functions with side effects. Henceforth, we shall assume such a limitation in our implementation.

Through the generosity of the Verdix Corporation, the distributed translation system was demonstrated at the Ada Expo held in Boston, Mass. in December of 1987. Two different demonstrations were given. The first involved calculating and displaying Mantelbrot sets. An Ada program was written using concurrent tasks to perform these calculations and display a map of the complex values computed. This was displayed in a window on a single Sun computer. At the same time, the program was distributed across two Sun computers, using our SITE pragma, and displayed in another window. The difference in speed was very evident, as expected. In the second demonstration, a mobile vehicle with a television camera mounted on it was set up

to follow a track around the end of the booth. The vision processing from the camera was done on one Sun computer, while the vehicle processing was done on another.

Port to PC

A second major activity during the year was to port the Distributed Ada system to a network of IBM PC/AT's. In this case the Alsys compiler was used. Although the port was successfully completed, several major difficulties arose, and there are limitations (temporary we believe) that make the system less useful than at first expected. We had originally planned to utilize the network of distributed Ada PC's for implementation of a tele-autonomous system. At present the tele-autonomous system, while implemented in Ada, is done with separate programs running on the different machines.

The first difficulty that arose, and one that had nothing to do with the translation system was the interface to the network. The software to drive the Ethernet boards is written in C. Since both Ada and C do resource management, conflicts arise and getting the two work together is much more than just a matter of matching calling sequences. Moreover, although the Alysis compiler provides an INTEFACE pragma for C, it is necessarily implementation dependent, and unless the particular C interfaced to is the same one as used for the Ethernet board drivers, one still does not have a match. It took six months to resolve this problem.

The choice of C as the implementation language for Ethernet board drivers has been made by nearly all of the Ethernet vendors. This decision has consequences that probably were not envisioned (and would not be present if assembly language had been used). It makes it *very* difficult to have any language other than C use the Ethernet in anything other than superficial ways. Moreover, since the C libraries often make use of the underlying operating system, it can make the Ethernet boards useless for embedded systems.

The limitations to our implementation are twofold. First, and more serious, we use the interface to the Ethernet in sophisticated ways, and have uncovered bugs in both the Ethernet board drivers and the Ada compiler's interface to C. Until these are fixed by the vendors (not yet done) we can only operate the network using the polled version of our mailbox system. Second, due to a limitation of Ada programs to the first 640 K of memory in the PC/AT, our translation system can only handle very small programs (since it, itself, takes up most of the memory and leave little working storage). This second problem is not severe, however, because it is easy to cross translate from the Sun.

Future Directions

The work described above will continue after the conclusion of this project. A new project with NASA in the Distributed Ada area will extend this work. We expect to complete the implementation of generics in the near future and to begin studying the implications of Distributed Ada implementations for tightly coupled architectures mixed with loosely coupled architectures. As noted in our paper [VMB87], the system architecture is one of the basic dimensions affecting a distributed execution system. Thus, a number of different issues must be considered in the translator design.

References

[Ana87] Anandan, P., "A Unified Perspective on Computational Techniques for the Measurement of Visual Motion," *Proc. Image Understanding Workshop*, pp. 719-732, Feb 1987.

[Aya87] Ayache, N., and Faugeras, O.D., "Building a Consistent 3D Representation of a Mobile Robot Environment by Combining Multiple Stereo Views," *Proc. 15th Int. Joint Conf. Artificial Intelligence*, vol. IJCAI-87, pp. 808–810, September 1987.

[Bha86] Bharwani, S., Riseman, E., and Hanson, A., "Refinement of Environmental Depth Maps Over Multiple Frames," *Proc. Workshop on Motion: Representation and Analysis*, Charlston, South Carolina, pp. 73-78, May 7-9 1986.

[Bak81] Baker, H. H., and Binford, T. O., "Depth from Edges and Intensity Based Stereo," *Proc. of the International Joint Conf. on Artificial Intelligence*, Vancouver, B. C., pp. 631-636, August, 1981.

[Bar80] Barnard, S. T., and Thompson W. B., "Disparity Analysis in Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 4, pp. 333-340, July, 1980.

[Besl87] Besl, P., and Jain, R., "Segmentation Through Variable Order Surface Fitting," to appear IEEE PAMI.

[Bol85] Bolles, R.C., and Baker, H.H., "Epipolar-Plane Image Analysis: A Technique For Analyzing Motion Sequences," *Proc. of the Third Workshop on Computer Vision: Representation and Control*, Bellair, Michigan, pp. 168-178, Oct. 1985.

[Hay86] Haynes, S. M. and Jain, R., "Event Detection and Correspondence," *Optical Engineering*, Vol. 25, No. 3, March, 1986.

[Gla83] Glazer, F., Reynolds G., and Anandan P.,"Scene Matching by Hierarchical Correlation," *IEEE CVPR Conference*, pp. 432-441, June 1983.

[Gri83] Grimson, W. E. L.,"Surface Consistency Constraints in Vision," *Computer Vision, Graphics, and Image Processing*, Vol. 22,No. 1, pp. 29-69, April, 1983.

[Mar82] Marr, D., Vision, W H Freeman, Chapter 1, 1982.

[Mar79] Marr, D., and Poggio, T., "A Computational Theory of Human Stereo Vision," *Proc. of the Royal Society of London B*, Vol. 211, pp. 151-180, 1979.

[Med85] Medioni, G., and Nevatia, R., "Segment-Based Stereo Matching," *Computer Vision, Graphics, and Image Processing* , Vol. 31, pp. 2-18, 1985.

[Nii86] Nii, H. P., "Blackboard Systems (parts I and II)," *The AI Magazine*, Vol. 7, Nos. 2 and 3, pp. 38-53 and pp. 82-106, 1986.

[Nis83] Nishihara, H. K., "PRISM: A Practical Real-Time Imaging Stereo Matcher," *Proc. SPIE Cambridge Sym. on Optical and Electrical-Optical Engineering*, Cambridge, Mass., November, 1983.

[Oht83] Ohta, Y., and Kanade, T.,"Stereo by Intra- and Interscanline Search using Dynamic Programming," *CMU-CS-83-162*, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, October, 1983.

[Wax86] Waxman, A.M., and Duncan, J.H., "Binocular Image Flows," *Proc. Workshop on Motion: Representation and Analysis*, Charlston, South Carolina, pp. 31-38, May 7-9 1986.

[Wey87a] Weymouth, Terry E., and Amini, Amir, "A Framework for Knowledge-Based Computer Vision: Using a Blackboard Architecture," In Proceedings of VISION-87, Detroit, June 1987.

[Wey87b] Weymouth, Terry E., "Incremental Inference: Spatial Reasoning Within a Blackboard Architecture," Workshop in Spatial Reasoning, St. Charles, Ill., William-Kaufman, 1987.

[Wey88] Weymouth, Terry E. and Saied Moezzi, "Wide Base-Line Dynamic Stereo: approximation and refinement" in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), Ann Arbor, Michigan, Junk 5-9, 1988.

[Ant82] H. J. Antoniss, "Image segmentation in pyramids", *Computer Graphics and Image Processing*, vol. 19, pp. 367-383, 1982.

[CiD84] J.M. Chibulskis and C.R. Dyer, " An analysis of node linking in overlapped pyramids", *IEEE Trans. on Systems, Man, and Cybernetics*, vol SMC-14, pp. 424-436, 1984.

[GrJ86] W.I. Grosky and R. Jain, "Pyramid-based approach to segmentation applied to region matching", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp.639-650, 1986.

[JGr87] R. Jain and W. I. Grosky, "Hyper-Pyramids for Integration of Spatial Information", *Proceedings of the 1987 Workshop on Spatial Reasoning and Multi-Sensor Fusion* (1987) p. 72-79

[JaT80] C.L. Jackins and S.L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects", *Computer Graphics and Image Processing*, vol. 14, pp.249-270, 1980.

[SrA87] Sanjay K. Srivastava and Narendra Ahuja, "An Algorithm for Generating Octtrees from Object Silhouettes in Perspective Views", *Proceedings IEEE Computer Society Workshop on Computer Vision* (1987) pp. 363-365

[VoM87a] R. A. Volz, T. N. Mudge, "Instruction Level Timing Mechanisms for Accurate Real-Time Task Scheduling," *Proc. of the IEEE 1986 Real-Time Symp. and Special Issue IEEE Trans. on Computer*, vol. C-36, no. 8, August 1987.

[CDV86] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, T. Schultz, "Toward real-time performance benchmarks for Ada," *Communication of the ACM*, vol. 29, no. 8, pp. 760-778, August 1986.

[VMN85] R.A. Volz, T.N. Mudge, A.W. Naylor, J.H. Mayer, "Some problems in distributing real-time ada programs across machines," *Ada in use, Proc. of the 1985 Int'l Ada Conf.*, pp. 72-84, May 1985.

[VKT87] R. Volz, P. Krishnan, R. Theriault, "An approach to distributed execution of Ada programs," *1987 IEEE International Symp. on Intelligent Control*, Philadelphia, January 20, 1987.

[NaV87] A. W. Naylor, R. A. Volz, "Design of Integrated Manufacturing System Control Software," *IEEE Trans. Systems Manufacturing and Cybernetics*, vol. SMC-17, no. 11, Nov/Dec 1987.

[NaV87a] A.W. Naylor, R.A. Volz, "Software for integrated manufacturing systems, Part I and II," *Space Operations Automation and Robotics 1987 Conference*, Houston, TX, August 1987.

[VMB87] R.A. Volz, T.N. Mudge, G.D. Buzzard, P. Krishnan, "Translation and Execution of Distributed Ada Programs: Is It Still Ada?," *Special Issue on Ada of the IEEE Transactions on Software Engineering*, 1987.

[VoM87b] R.A. Volz, T.N. Mudge, "Timing Issues in the Distributed Execution of Ada Programs," *IEEE Transactions on Computers*, vol. C-36, no. 4, April 1987.