ETL-0572

AD-A238 570

Vision-Based Navigation and Parallel Computing
Second Annual Report

Larry S. Davis
Daniel DeMenthon
Thor Bestul

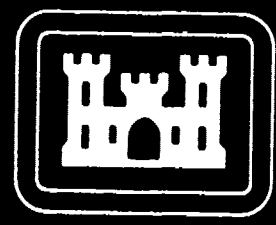
University of Maryland
Center for Automation Research
College Park, MD 20742-3411

August 1990

91-05649

Prepared for:
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209-2308

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvcir, Virginia 22060-5546



E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1990	Second Annual May 89 - May 90

4. TITLE AND SUBTITLE
Vision-Based Navigation and Parallel Computing
Second Annual Report

5. FUNDING NUMBERS
DACA76-88-C-0008

6. AUTHOR(S)
Larry S. Davis Thor Bestul
Daniel DeMenthon

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Maryland
Center for Automation Research
College Park, MD 20742-3411

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Engineer Topographic Laboratories
Fort Belvoir, VA 22060-5546
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209-2308

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ETL-0572

11. SUPPLEMENTARY NOTES
Previous report in series:
ETL-0548 Vision-Based Navigation and Parallel Computing
First Annual Report August 1989 AD-A214 481

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report summarizes the research performed under Contract DACA76-88-C-0008 during the period May 1989 - May 1990, the first year of the contract period. The focus of the research program is visual navigation, with an emphasis on the use of massively parallel algorithms to support basic navigational tasks in vision and planning.

The first section describes research performed on a project called RAMBO. (RAMBO is an acronym for Robot Acting on Moving BOdies). The project attempts to develop and integrate Connection Machine algorithms for low-level vision, intermediate level vision and visual planning to allow a mobile robot to pursue (in simulation) a moving three dimensional target through space in order to maintain visual contact with points on the surface of the target. The next section describes our past year's work on cross-country navigation. We first describe massively parallel algorithms for route planning in digital terrain maps. Then we describe our research on the problem of filling in range shadows. We discuss why classical interpolation methods are not appropriate for this problem, and present new methods. The last section presents brief descriptions of other research projects whose results were reported under this contract during the past year.

14. SUBJECT TERMS
autonomous navigation computer vision parallel processing
artificial intelligence search

15. NUMBER OF PAGES
51

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

Contents

List of Figures

PREFACE

This report describes work performed under Contract DACA76-88-C-0008 by the Center for Automation Research, University of Maryland, College Park, Maryland for the US Army Engineer Topographic Laboratories (ETL), Fort Belvoir, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia. The Contracting Officer's Representative at ETL is Ms. Rosalene Holecheck. The DARPA points of contact are Dr. Erik Mettala and Dr. Rand Waltzman.

1. Introduction

In this report we summarize the research we have performed under Contract DACA76-88-C-0008 during the period May 1989-May 1990, the first year of the contract period. The focus of our research program is visual navigation, with an emphasis on the use of massively parallel algorithms to support basic navigational tasks in vision and planning. Our research is motivated by general problems relating to visual surveillance and pursuit of targets by a mobile robot in a ground environment.

The first section of the report describes the research we have performed on a project we call RAMBO. RAMBO is an acronym for Robot Acting on Moving BOdies, and is a project in which we are attempting to develop and integrate Connection Machine algorithms for low-level vision, intermediate level vision and visual planning to allow a mobile robot to pursue (in simulation) a moving three dimensional target through space in order to maintain visual contact with points on the surface of the target. The RAMBO project has led us to consider many fundamental problems in mapping vision and planning algorithms onto massively parallel machines, as well as develop new vision algorithms for basic problems such as pose estimation, motion estimation and motion planning.

We have traditionally been concerned with applications in ground navigation. As a contributor to the ALV program during 1985-89 we developed a navigation system that was able to navigate a laboratory robot over a terrain board containing a simple road network. These algorithms were also brought to Martin Marietta and used to navigate the ALV over the Denver test track. Recent interest in ground navigation has focused more on cross country navigation than road following, and we have begun to consider problems in ground navigation in our research program. In Section 3 we describe research performed during the past year on topics fundamental to cross country navigation. We first describe massively parallel algorithms for route planning in digital terrain maps. What distinguishes this from previous route planning research in terrain is the incorporation of models for the presence of adversaries in the environment, and route planning criteria reflecting the need to move without being seen by these targets, yet being able to monitor their positions from time to time during navigation. This project was initiated in January 1990, and so our results should be regarded as preliminary. The second part of Section 3 describes our research on the problem of filling in range shadows. We discuss why classical interpolation methods are not appropriate for this problem, and present new methods for filling in range shadows.

While the research described in Sections 2 and 3 forms the greater part of the work conducted under support of the contract, there were many other projects whose results were reported under this contract during the past year. Some of these were the Ph. D. dissertations of students, and others were research projects conducted by visitors to the Laboratory that we felt best fit into our research in navigation. In Section 4 we present brief descriptions of these research projects.

2. Dynamic Visual Surveillance (RAMBO)

2.1. Overview

This section describes our progress in the area of visual surveillance, in the context of the project we call RAMBO—Robot Acting on Moving BOdies. RAMBO is a project designed to explore issues in high level vision and visual planning for an autonomous robot tasked to perform some visual surveillance activity on a moving object. We have developed algorithms for object pose estimation, object motion estimation, and visual planning. Many of these algorithms are massively parallel algorithms that have been implemented on a Connection Machine CM2. We describe an overall system architecture for the types of visual surveillance tasks that RAMBO is to perform, and then provide an overview of our research in vision and planning.

Specifically, we assume that the environment contains an object of known three dimensional structure. On the surface of this object are targets for visual surveillance. The object is in motion, and a general model of its motion is known. RAMBO's goal is to develop and modify a specific model for the motion of the target, and to use this motion model to construct a sequence of trajectories that will allow RAMBO to sight some set or sequence of the visual surveillance targets. We assume that the object will not change its motion in an effort to evade RAMBO. Computer simulations allow us to consider a wide variety of situations in which many parameters of the system (e.g., accuracy with which RAMBO can control its own motion, accuracy of various image analysis procedures, etc.) can be controlled and evaluated. They also allow us to bypass many difficult intrinsic and extrinsic calibration problems that would have to be solved in order to conduct meaningful quantitative experiments using real robots.

The vision and planning algorithms being developed as part of the RAMBO project are massively parallel algorithms designed to operate on the Connection Machine. In fact, another important goal of our research is to better understand how to integrate visual computations at many levels of analysis on massively parallel machines.

RAMBO's architecture has been heavily influenced by our experience with the Autonomous Land Vehicle project[4, 5, 6]; see also Crowley [7]. As part of that project, we developed a system for visual navigation of a ground vehicle over roads and road networks. The organization of that system had the following important characteristics:

1. It operated in one of two control modes. Ordinarily, the system operated in the feedforward mode of processing. Here, each control cycle included three stages of analysis:

   a) Prediction, in which the current road model (constructed on the basis of previously analyzed images) and an estimate of the vehicle's location in the model were used to generate predictions concerning the image locations of key road features (e.g., road boundaries and markings).

   b) Item verification, in which specialized image processing algorithms were employed to verify these predictions. These algorithms were applied to small search windows generated by the prediction stage.

c) Extension, where the road model was extended out towards the horizon using new information provided by the current image. If the verification stage failed, then the system switched to a bootstrap mode of processing, in which the vehicle would come to a standstill and perform a more global and time consuming analysis of its current image in an attempt to relocate itself relative to the road. This bootstrap mode of processing was also used to initialize the vehicle's road model.

2. Our system navigated on the basis of a three dimensional reconstruction of relevant parts of its environments. Its motions were planned and executed in a three dimensional coordinate system. This was, to a large extent, dictated by the relatively long time required to execute a single control loop of the system (this time was, of course, dominated by image processing), and a system with a substantially higher processing rate might have been able to perform road navigation solely on the basis of, say, visual directions to road boundaries (see, e.g., Dickmanns' series of papers [8, 9]).

RAMBO's organization is very similar to that of our road navigation system (Figure 1). In a feedforward mode of processing RAMBO first predicts, on the basis of its current motion model for the target and its own egomotion model, which target features (polyhedron vertices) should be visible in the image, and where they should be located. In fact, unlike the ALV system which acquired an image as soon as a control cycle was completed, RAMBO also needs to reason about when to acquire its next image, since it might not be possible for RAMBO to move along a trajectory from which a sufficient set of target features would always be visible. RAMBO also uses an estimate of errors in its motion models to choose appropriate size search windows for the verification process. These search windows are the byproduct of a Kalman filter employed to estimate target motion and extrapolate target pose.

In the verification phase of processing, we have to perform local searches of the prediction windows for the target features. If an insufficient number of such features were detected, then we would regard the analysis as having failed, and would move away from the target to a safe location from which we could attempt to reinitialize our model of its motion using a bootstrap-like processing. Ordinarily, the locations of the detected vertices are used by a Connection Machine pose estimation algorithm to estimate the location and orientation of the target in space.

The motion model for the target is then verified and extended by a Kalman filtering algorithm that takes the recent history of pose estimates and develops an updated model for the relative motion of the target. Based on this new model of target motion, RAMBO's current trajectory is modified (if necessary) by algorithms that find minimal trajectories to target locations that orbit the visual surveillance points. A next viewing position is then chosen based on potential visibility of target features and the estimates of errors in the motion models.

2.2. Parallel Vision and Planning Techniques for RAMBO

While massively parallel computers, such as the MPP [10], DAP [11], and Connection Machine [12], are ideally suited for low level vision operations such as convolution, image arithmetic and image morphology, machines without the rich interconnection structure of the Connection Machine are at a distinct disadvantage in supporting intermediate level vision processing. There are two main reasons for this:

- In many intermediate level vision algorithms data must be combined from image regions that are physically separated. So, even an operation as simple as the Hough transform for line detection is cumbersome on a mesh connected machine because the *votes* from points lying along any line must be collected, eventually, at a single processor in the machine.

- In other intermediate level vision algorithms, operations are performed not on the image array itself, but on more general data structures. So, for example, to apply our pose estimation algorithm we must develop a representation of the image that is structured as a set of triples of edge junctions detected in the image by a corner detector. The junctions forming any triple may correspond to image points that are widely separated in the image.

The additional interprocessor connections included in the Connection Machine's hypercube are critical to overcoming these problems. First, the diameter of the hypercube (i.e., the greatest number of wires one must traverse in moving information from one processor to another) is only the logarithm of the number of processors in the machine, as opposed to the square root of the number of processors in a mesh. This makes operations such as the Hough transform very efficient, and is also critical for efficient implementation of more general grouping operations.

Second, the hypercube connections support logarithmic implementations of fundamental parallel algorithms, such as grid permutations and scan operations. These operations form the basis for most of the grouping operations that RAMBO performs.

An interesting technical problem arises in the feedforward mode of processing. Here, we would like to employ focus-of-attention vision algorithms similar to those used in our road following system [4]. This will generally involve determining relatively small image windows in which specific object features are predicted to appear, and then applying specialized image processing algorithms that are "tuned" to the expected photometric and geometric properties of the predicted features.

This can be accomplished in a straightforward way on the Connection Machine by simply selecting the processors associated with the window(s) to be processed, and applying the image processing operations to those selected processors. The remaining processors would be idle during this stage of computation. The disadvantage of this approach is that a large part of the Connection Machine would not be utilized. Similar problems occurred with earlier commercial machine vision systems, which required 1/30 second to apply a basic operation to a frame even if only a portion of the frame were of interest. Current systems have so-called *region-of-interest* operators, which essentially involve some additional addressing hardware

that pipeline only a window of the full frame through the processing hardware. With such region-of-interest hardware, the time that it takes to apply a basic operation to a window is proportional to the number of pixels in the window, as opposed to being fixed.

We would like to achieve a similar economy of scale on cellular machines like the Connection Machine. There are two approaches that we can identify that could potentially lead to more effective utilization of the massive parallelism available in today's cellular computers:

- Partition the machine into clusters of processors, and assign each cluster to a single pixel in the window to be processed. This was the approach proposed in Bestul and Davis [14], where we described a so-called *fat pyramid* model for image processing on hypercube connected machines. Rushton [15] contains a proposal for the architecture of a cellular machine that would support this partitioning in hardware, and the software Connection Machine implementations presented in Ziavras and Davis [16] certainly indicate that any efficient instantiation of this concept would require extensive hardware support. Note that the DAP computer [10] supports a limited type of hardware reconfigurability.

- Replicate the image window as many times as will fit into the available processor set. This is the approach employed by Davis and Narayanan [17] to support basic image processing operations including histogramming, table lookups, convolutions and morphological operations. In [14] we describe an implementation of replicated images on the Connection Machine that is much more practical as a software solution to the problem than partitioning.

Replication can be used to address the efficient analysis of "small" instances of important data structures other than images—for example chains (which ordinarily arise from boundary extraction processes) and graphs (representing the geometric and topological relationships between elements in some segmentation of an image). Generally, the following four problems must be addressed in developing replicated data analysis for any specific data structure:

- Embedding - An embedding of the data structure into the underlying interconnection network of the machine must be identified that simultaneously optimizes two proximity criteria. First, within a single copy of the data structure, nodes that are nearby in the data structure should be mapped into processors that are nearby in the interconnection network. Second, between copies of the data structure, processors that represent corresponding nodes (e.g., pixels with the same original image address) should be nearby in the interconnection network. For images, this is easily accomplished using Gray coding methods in hypercube connected networks (see Davis and Narayanan [17]).

  The rationale for the first criterion is that most operations on the data structures involve movement of information between nodes in the data structure, with movement between proximate nodes more prevalent than movement between distant nodes. The rationale for the second criterion is that partial results from the copies must eventually be combined to generate a representation for the complete computation. This ordinarily involves collecting information from corresponding nodes in all copies, usually with a logarithmic merging process; this process is expedited by keeping such corresponding nodes proximate in the network.

- Distribution - Given a goal embedding, algorithms must be developed to generate the copies from the original instance of the data structure (e.g., given an arbitrary $s \times s$ window of an image, we might need to generate 16 copies of the window). Such distribution algorithms are communication intensive, and must be designed to minimize interference in the interconnection network as the copies are distributed to their destinations. Since we might want to process several replicated structures simultaneously, the distribution algorithms should, additionally, consider the problem of multiple instances and partition the machine among the instances.

- Decomposition - Methods must be developed for decomposing a computation into pieces that can be performed in parallel on the copies of the data structure. This would ordinarily be straightforward if the underlying machine architecture were MIMD. However, cellular machines are SIMD machines, and provide us with very limited flexibility in decomposing operations into pieces that can be simultaneously computed on the copies. The Connection Machine, for example, provides the capability for each processor to access a different location in an array stored in its local memory at any time. However, this location will have to be a simple function of the copy and location numbers of each node in the replicated structure, since it is unfeasible to preload the memory of the machine with a problem and data structure dependent addressing pattern. The Connection Machine also supports arbitrary interprocessor communication patterns. In replicated image processing, for example, the copy number is used to determine a relative address for the decomposition of a convolution operation. So, for example, all pixels in copy one might communicate with their north neighbor; all pixels in copy two with their northeast neighbor, etc.

- Collection - Once the projections of the operation are completed within each copy, the partial results must be collected into a final result. This ordinarily involves a logarithmic merging operation. The algorithms must be designed so that at the end of the collection stage each copy has a complete result, as opposed to one copy being designated as the master and the remaining copies having only partial results. The reason for this is that it is likely that we will want to perform subsequent basic operations on the replicated structure, and these will proceed most efficiently if all of the copies have complete rather than partial results.

We are currently developing replicated data structures and processing algorithms for two other common image data structures—chains and graphs. Chains (linear structures) arise naturally from edge detection and border following. Generally, edges comprise only a small percentage of the pixels in an image (2%–5%). Therefore, if an image has been processed and its important edges and boundaries *marked*, then processing them using conventional SIMD algorithms would be very wasteful, since most of the processors would be idle. Similarly, segmentations of an image are naturally represented using planar graphs. These graphs will have far fewer nodes than there are pixels in the image, so they can also profitably be analyzed using replicated data analysis algorithms.

2.3. Present Implementation of Vision Algorithms

We briefly describe the current status of RAMBO's vision algorithms. It should be noted that these algorithms have not been integrated with the planning algorithms described in the next section into a complete system.

2.3.1. Feature Detection

RAMBO's target is a simple, convex polyhedral object. The background is kept visually simple to simplify the low level processing. The image is first operated on using a sequence of conventional image processing algorithms—a local edge detection operator is first applied, the magnitude of the edge detector is thresholded and the binary image is thinned so that all edges are part of simple chains or are vertices where such chains meet. RAMBO's pose and motion estimation algorithms are based on vertex matching, so the polyhedron vertices are detected in one of two ways. We have developed a simple corner detection algorithm that breaks the neighborhood of each edge pixel into a small number of sectors and identifies as corners those points that:

a) have edges in two or more non-collinear sectors, and

b) are local maxima with respect to a simple measure of curvature.

Alternatively, corners can be detected by a sequence of Hough transform, prediction of corners from pairs of clusters in the Hough transform, and verification through some additional image processing. The latter approach is more sensitive and accurate in its location of corners, but the former is much faster.

The feature points detected in an image are grouped into triples (the *image triangles*). Each image triangle can be described by one of its vertices (the *reference vertex*), the length of the two adjacent sides, and the angle between them (the *reference angle*). These adjacent sides do not necessarily have to correspond to actual edges in the image, and all distinct triples of points could be considered, with each triple of points producing three such image triangles. However, if the feature points are vertices, it is useful to only consider image triangles in which the adjacent edges of the reference vertex are actual edges, and to only match these image triangles to object triangles with similar characteristics. This increases the proportion of good matches over the total number of possible matches.

2.3.2. Initial Pose Calculations

The detected triangles are used to estimate the instantaneous six degree-of-freedom pose of the target. Our Connection Machine pose estimation algorithm is based on matching triangles of image corners against triples of target junctions (Linnainmaa, Harwood and Davis [18]). For perspective imaging, two possible pose estimates for an object can be recovered from a fourth degree equation determined by the correspondence between an image triangle and a triangle of the object. However, when RAMBO analyzes its first image, it does not have any *a priori* knowledge of the correct correspondence between the detected image triangles and the actual triangles of the object. In this case, the system uses all the possible combinations of target triangles and image triangles and clusters the pose estimates for these combinations.

Approximated pose of a known triangle from its image Solving a fourth degree equation for each image triangle to object triangle correspondence as proposed in [18], and picking the two correct triangles among the eight triangles given by the quartic solutions is time-consuming. Also, the fourth degree equation is sensitive to roundoff error for certain image parameters. For these reasons, we have developed an approximate perspective projection, *orthoperspective*, which provides simpler computations and reasonably accurate results. The idea is to project the figure onto a plane drawn through one of the vertices of the figure and perpendicular to the line of sight of this vertex (other approximations of this type have used planes parallel to the image plane instead). Then the image of the figure is approximated as the image of this projection (Figure 2).

Notice that if we rotate the camera around the center of projection to bring the optical axis to the line of sight of the vertex we considered, the image plane becomes parallel to the plane on which we performed the orthogonal projection (this type of camera rotation is called a *standard rotation* by Kanatani [33]). After this camera rotation orthoperspective is simply a scaled orthographic projection (also called *weak perspective*). Therefore, orthoperspective is equivalent to the following sequence of operations:

1. A virtual rotation of the camera around the center of projection to make a chosen line of sight coincide with the optical axis.

2. A scaled orthographic projection of the recente .mage

3. The inverse camera rotation to bring back the c. to its original position.

The camera rotation removes the dependence of the construction on the offset of the world object from the optical axis. For this reason orthoperspective is a better approximation to perspective than scaled orthographic projection for image elements which are not centered in the image plane.

With the orthoperspective approximation, finding the pose of a triangle from its image when the shape of the triangle is known reduces to solving a second degree equation. Details are given in DeMenthon and Davis [19].

Before the Connection Machine had its floating point coprocessors, we solved these equations beforehand and built lookup tables for each triangle of the model. The present implementation solves these equations as needed, on the assumption that routing communications to perform table lookups might be slower than direct computations. Comparisons are difficult because the new implementation is written in C-Paris whereas the lookup table version was in *Lisp.

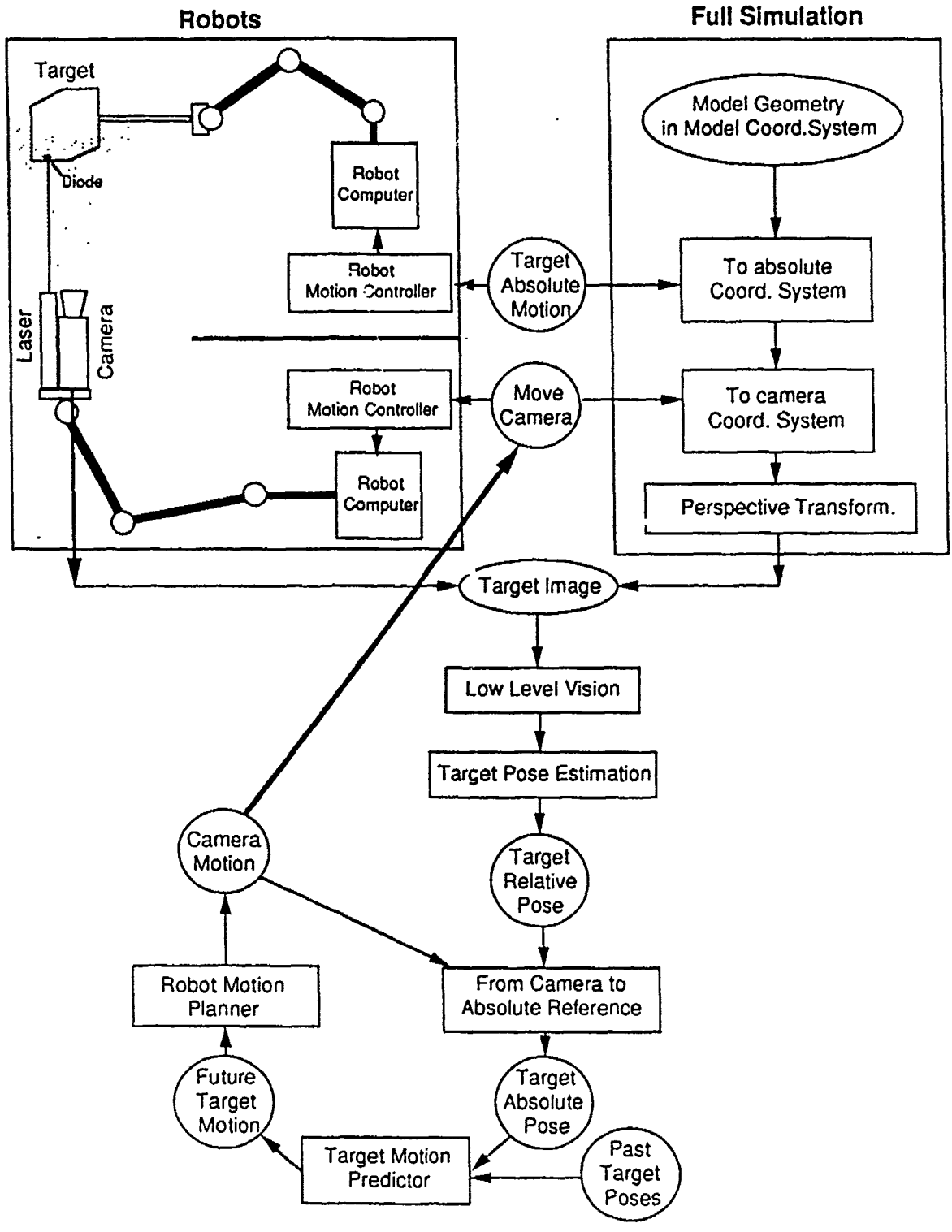


Figure 1: Vision-based control loop for a robot acting on a moving body.

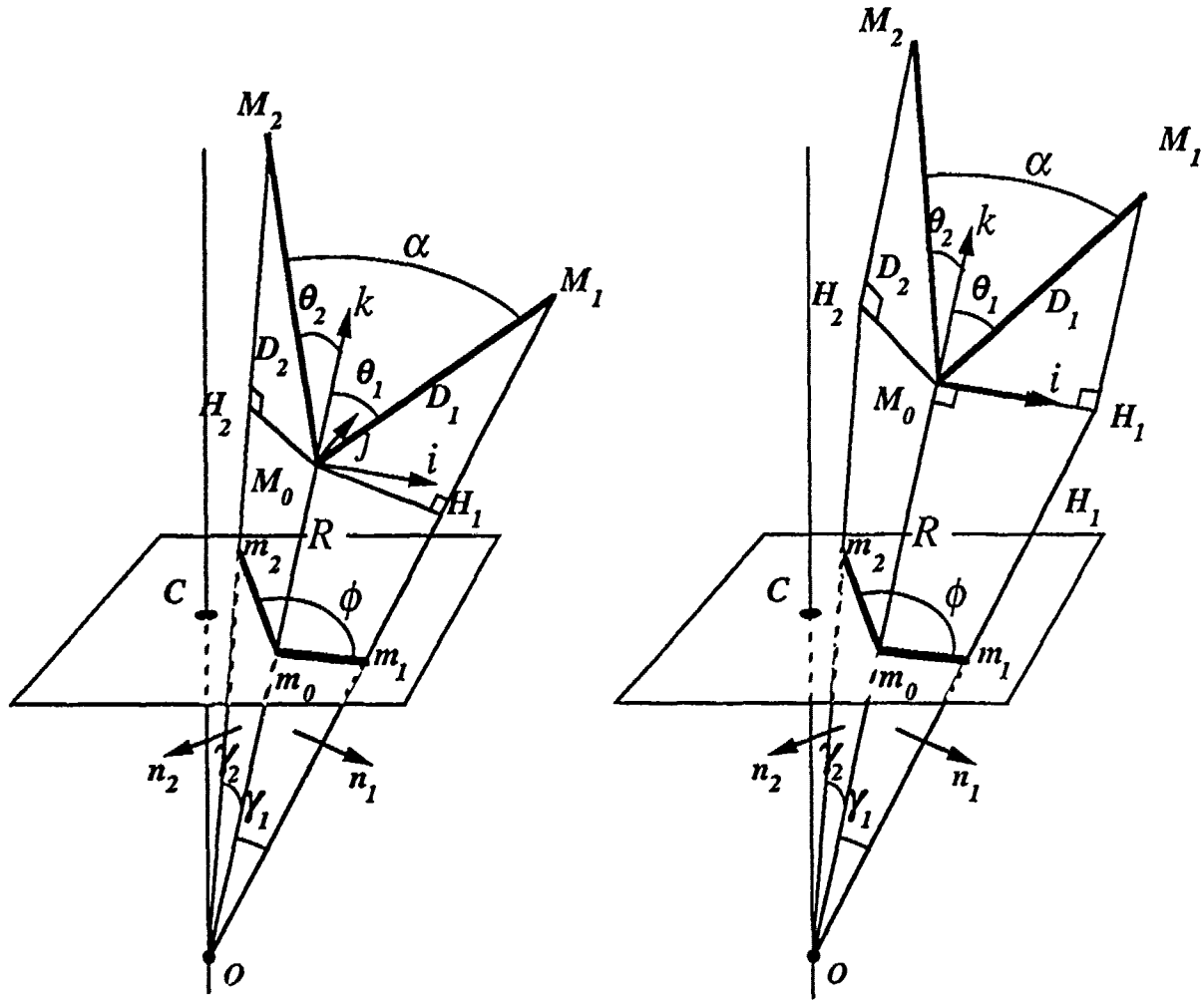


Figure 2: Exact perspective and orthoperspective for a triplet of points.

Clustering of the pose parameters Pose parameters for the object are obtained for each possible combination of image triangle and object triangle. The combinations which correspond to correct matches will have produce similar object pose parameters, and will thus be clustered in parameter space. Our present clustering algorithm uses a virtual processor set with one processor for each data point in the pose space. It also uses a set of virtual processors to represent an orthogonal projection grid, and projections of the six dimensional pose space are computed using several axes. A parallel read-back scheme provides a voting mechanism by which high-density regions in a cloud of points can be identified. This new clustering method produces robust results, with few misidentifications of pose parameters under a variety of pose conditions.

Verifying the pose estimate of the object This pose estimation algorithm cannot make much use of expectations of the target's pose that would be available during the feedforward mode of processing. We have therefore designed and implemented a Connection Machine hypothesize-and-test pose estimation algorithm:

Once we have generated expectations of target poses, we project visible model points with their local features onto the image where they are compared with the image points. The comparison is based on the locations of the image and projected points, on the number of adjacent edges and on their angles. If the projection of the target in the hypothesized pose passes these tests, its pose estimate is refined. We find the rotation R and translation T which minimize the sum of the squares of distances between the actual image features and the projections of the features of the object in its hypothesized position. We reach a minimum by applying Newton's method, starting from the estimated R_0 and T_0.

But even if all the correspondences between model and image points are correct, the optimized pose can be inaccurate, due to noise in the image. Theoretically, we could calculate the exact probability distributions of the pose parameters assuming, for example, bivariate normal location error. This would require, however, the solution of fourth degree equations, with complex analytic functions as a result.

We prefer to calculate an estimate of the reliability of the pose parameters by applying intermediary results of the pose estimate optimization. We perform one Newton iteration assuming bivariate normal noise added to each image of the model point. As a result we obtain normal distributions that estimate the true probability distributions of the pose parameters. Details are given in Pehkonen, Harwood and Davis [20].

2.3.3. Feedforward Tracking and Pose Calculation by Kalman Filtering

The object pose could be estimated independently for every image frame in the sequence by the techniques presented in the previous paragraphs, and used to compute the motion. However, pose estimation starting from raw image data is a computationally expensive procedure that one would prefer to avoid. This is possible once satisfactory initial estimates of the motion parameters have been been obtained as described above. Then the future poses and the image locations of the features that correspond to known points of the model can be predicted, greatly reducing the computational cost. Based on the known correspondences, straightforward pose and motion determination methods can be used.

The motion parameters may change with time and the measurements obtained from the images are not infinitely accurate due to discretization. Knowledge of these uncertainties must be incorporated into the estimates in order to judge the chances for successful task execution.

Kalman filtering has been shown to be a powerful tool in similar problems, e.g., in tracking objects in monocular image sequences [22] and in robot navigation [8, 9]. In our case the camera is in the hand of a moving robot and an accurate geometrical model of the target object is available, while its motion model contains uncertainties.

Uncertainty regions of feature points When tracking the object feature points in the image plane, it is useful to limit the search to regions where they are most likely to occur. By assuming the measurement function $F(*)$ linear, the uncertainty windows of predicted feature points can be approximated using the covariance matrix $V_{k|k-1}$ modeling the prediction of the measurement uncertainty

$$V_{k|k-1} = J_F(u_{k|k-1})Q_{k|k-1}J_F^T(u_{k|k-1})$$

where $Q_{k|k-1}$ is the estimation covariance matrix of the Kalman filter and $J_F(u_{k|k-1})$ is the Jacobian of the measurement function $F(u_{k|k-1})$, or the matrix of first order derivatives of feature point coordinates in the camera frame with respect to the state variables in the motion estimation frame. We have simply used the diagonal elements of this matrix to determine rectangular regions of interest.

The uncertainty windows also offer a simple confusion avoidance mechanism for feature points located close to each other in the image: points whose uncertainty regions intersect should not be used.

Motion model The object tracking performance of a robot depends critically on the match between an *a priori* model of motion and the actual object behavior, unless the system control delay can be made very short. Shorter control delays decrease the relative contributions of unmodeled changes of motion parameters per time unit. This results in smaller Kalman gains, smoothing the variations due to measurement noise. However, image acquisition and mechanical delays determine the minimum control delay of the system that cannot be shortened by computational solutions. Therefore, we need to consider the motion modeling problem in more detail.

When the camera sensor is mounted in the hand of a robot manipulator, the key questions in forming the object motion model are

1. Whether the state vector should be represented in a moving coordinate frame or a global stationary world coordinate system (relative or absolute object motion).

2. What motion parameters to include in the state vector.

Coordinate system If the target object is not able to perform significant accelerations, its motion in a fixed world coordinate frame can be modeled by constant rotation and translation. The possible small accelerations can be taken into account as noise. However, with the camera in the hand of a robot the motion model may not be independent of the mechanical solution. The measurements are done in a camera coordinate system and the transformation to the world frame, or vice versa, requires knowing the robot joint angles at the time of image capture. But the controllers of many off-the-shelf robot manipulators do not support obtaining the angles of different joints simultaneously during motion. This may significantly increase the uncertainties.

A practical compromise is to model the motion in a relatively slowly moving coordinate frame, in which the angle and translation parameters of the camera are simple to determine. For example, the origin of the coordinate system could coincide with the wrist joint of the robot.

Motion parameters If the state vector is represented in a moving coordinate system, modeling the expected relative accelerations as system noise may result in impractically high uncertainties. This also increases the sensitivity to measurement noise.

The uncertainties can be reduced by including the accelerations in the state vector and modeling the possible small changes of acceleration as noise. Unfortunately, a longer state vector increases the computational cost and relative motion uncertainty during the times when accelerations are near zero.

We have investigated two simple compromises that limit the drawbacks of higher dimensionality or higher noise level to the appropriate periods:

1. Variable-dimension state vector: If the estimation errors grow rapidly, the acceleration components are added to the state vector. The switch back to the constant velocity model is performed when the acceleration approaches zero or the errors go below a given threshold.

2. Variable system noise: A constant-velocity motion model is used, but when the estimation errors grow the system noise model components attributed to motion are inflated. When the errors decrease the noise component is returned back to the normal quiescent level.

The exact ranking of these solutions requires knowing the length of the control loop, including the image processing time and robot delays.

Mapping these techniques onto the system equation is straightforward. For simplicity, we have assumed that each motion component is independent and the trajectory of the object in the given coordinate system is straight.

Measurements uncertainties Our principle in dealing with the measurement noise has been to approximate the limits of certainty by the standard deviations of Gaussian distributions. Then straightforward methods can be used for transforming and combining the uncertainties. The feature point locations on the image plane are not arbitrarily accurate

measurements because of discretization. As a result, the feature points in the image may deviate from their exact positions and the same point locations in an image may correspond to a range of poses.

We have considered the $2n$ feature point coordinate values as independent Gaussian distributed random vectors and denote their composite $2n$-by-$2n$ covariance matrix by V_v. By assuming that the standard deviation of the feature point location coordinate error is 1 pixel, V_v becomes an identity matrix.

Note that there is no physical reason why the actual locations of the feature points would cluster around the centers of pixels. However, simulations with uniformly and Gaussian distributed noise showed hardly any difference in the resulting pose and motion estimates.

Pose estimation uncertainty We start from known correspondences, so the pose parameter vector of a particular object is solved from the image feature point locations by the inverse of the measurement function

$$u = F_k^{-1}(v) = G_k(v)$$

If the pose determination function $G_k(*)$ was linear, the distribution of pose parameters, $V_u(u)$, would be Gaussian and could be obtained by

$$V_u(u) = J_G(v) V_v J_G(v)^T \tag{1}$$

where $J_G(v)$ is the Jacobian of $G_k(*)$ or its matrix of first partial derivatives. In the actual non-linear case this provides a local approximation of the pose distribution, hence the arguments u and v.

In this equation, the Jacobian J_G is obtained from the Jacobian of the measurement function:

$$G'(G^{-1}(x)) = \frac{1}{(G^{-1})'(x)}$$

or equivalently

$$J_G(v) = J_F^{+}(u)$$

where matrix pseudoinversion $(^{+})$ is used, because J_F is not a square matrix except when $n = 3$.

Selection of correspondence points The important problem of selecting the feature points used for estimation is now considered. Usually several feature points are available from an image frame, but because of various real-world constraints using only some of them is justified. With the test object that was used *random selection* of the feature points gave good results, and when the actual computational costs of the methods are taken into account, this approach becomes very attractive. Random selection does not favor any particular set of correspondences so this approach works with any number of feature points. The very low computational cost results in high speed and minimum motion modeling uncertainties, helping to offset the possible higher measurement uncertainty.

Note that the performance of the random selection technique depends on the distribution of the feature points on the object. In practice, it may be necessary to make the selection of certain important feature points, such as the ends of prominent extensions, more probable than that of the others.

The method of random feature selection was experimentally compared with three other methods. The first method, the minimum uncertainty method, attempts to choose the features which minimize the covariance matrix. It seems impractical for real time applications because of its computational cost, but can be used as a comparison reference. The second method, the condition method, chooses features that produce well conditioned measurement equations that minimize the sensitivity of the relative error in the system state estimate to the variations in the measurements. The third method, the pose information method, selects features which give the most accurate, or most informative object pose.

Random selection performs surprisingly well against these other methods with our test object. When compared to the minimum uncertainty method, the penalty is almost negligible with the selection of a *single* feature point. With only two points the method outranks both the condition of Jacobian and pose information approaches. The results become worse when more points are selected, however the low computational cost still acts as a significant compensating factor.

Higher sampling rates reduce the motion uncertainties and can in part be achieved by reducing the number of feature points used. Random selection is a promising method for this purpose. In practice, it is necessary to find a trade-off between better measurements and weaker motion model. In our experiments, maximization of the sampling rate has been the most rewarding direction of development.

The mathematical details of the filtering algorithm, along with the results of simulations designed to determine the range of conditions under which the filter might be expected to provide usable motion models, are presented in Silvén [23].

2.4. Visual Planning

We have conceptualized RAMBO's planning activities as occurring at four hierarchical levels of analysis:

- Subtask sequencing - RAMBO is generally tasked to visually sight a set of locations on the surface of the target. If the ordering of subtasks is not specified in the problem description, then one must be chosen during task execution. While it would be possible to identify an *optimal* subtask ordering once some initial target motion model is established, we adopt instead a *greedy* approach since the future motion of the target is only weakly constrained by its current motion, and the cost of solving the combinatorial optimization problem is very high.

- Goal point identification - Once a subtask is chosen, RAMBO must determine a goal point (or small region in space) in the target's coordinate system from which it can sight (i.e., illuminate with its laser pointer) the target surface location. Generally, RAMBO would like to maximize its distance from the target while sighting, but a variety of factors must be considering in choosing a goal point. These include an estimate of

the accuracy of the target motion parameters, RAMBO's accuracy in controlling its own motion, how accurately the laser can be pointed, and the amount of real-time visual monitoring that RAMBO would need to perform in order to guarantee the safe execution of the plan.

- Generation of approach trajectory - once a goal point is chosen RAMBO next computes a trajectory that will take it from its current trajectory to the goal point. This so-called reaching trajectory may either bring RAMBO into zero relative rigid body motion with respect to the target, or to a stationary position at the goal point. It should be chosen so that the target would always be in the field of view if the target motion model were correct.

- Monitoring and failure recovery - RAMBO must continually monitor the target to update its motion model. If the target's motion changes then RAMBO has to decide if a new reaching trajectory should be determined, if a new goal point should be chosen or if the subtask ordering should be changed. It is also possible that the target may leave RAMBO's field of view (or RAMBO may fail to detect the target even though it is within its field of view). In this case RAMBO must initiate a visual search strategy to reacquire the target.

Our current planning system is only able to deal with a small subset of these problems. RAMBO currently makes the simplifying assumption that a complex goal can be decomposed into a sequence of simple subgoals. All subgoal points required for each complex action on a target can be predetermined and stored in a database of actions specific to each target. Each goal point is defined by its pose in the object coordinate system.

As RAMBO progresses from one subgoal to another, it will generally have to change the trajectory along which it moves, so that at some time to we would want RAMBO to launch from its current goal trajectory and to land at some subsequent time. $t_0 + T$, on a new goal trajectory. The duration T is referred to as the reaching duration. Once t_0 and T are chosen, then a reaching trajectory that takes RAMBO from its original goal trajectory to the new goal trajectory can be determined by using, for example, a parametric cubic spline that ensures continuity and smoothness at both takeoff from the original trajectory and landing on the new goal trajectory.

2.5. Conclusions

The RAMBO project, whose long term goal is the development of real time parallel systems for tracking and surveillance of three dimensional objects, has provided us with a framework for studying several new and interesting problems in vision and visual planning. Over and above the design and implementation of Connection Machine algorithms for specific vision and planning tasks, their integration on the Connection Machine reveals a new set of challenging problems, mostly centered around efficient transformations between various image and spatial representations.

3. Ground Navigation

While the dominating problem in visual navigation during the 1980's was road and road network following, the 1990's will see a much greater emphasis on cross country navigation. As part of our research program we are developing new algorithms for vision and planning specifically designed to support applications in cross country navigation. Here we describe two such projects. The first, initiated in mid 1990, involves developing massively parallel algorithms for planning routes for vehicles in natural terrain. Our research differs from most previous research in a variety of ways. One of the most important is that we form plans with respect to models of adversaries moving in the environment. We are also interested in planning route for collections of vehicles that must operate under specific transport protocols. The research described here should be viewed as preliminary, but we feel that the results obtained after even a short time on the project were sufficiently promising that they should be included and given some emphasis in this annual report.

We also describe, more briefly, in this section, research on new interpolation methods for filling in range shadows for range images taken in natural terrain. This research is relevant to local path planning in rough terrain where one would want to make the most informed guess about hidden parts of the environment.

3.1. Planning a Safe Path on a Digital Terrain

3.1.1. Introduction

We now describe our work on planning a safe path in the presence of several moving adversary agents. The algorithms were implemented on the Connection Machine, using a digital terrain map. We first define the problem, then briefly present the algorithm.

3.1.2. Problem definition

Given a digital terrain map and information about a friendly agent and adversary agents moving on the ground, we want to find a path for the friendly agent to a final goal that is hidden from the adversary as it close to points of high visibility, taking advantage of the terrain.

The information available for the friendly agent includes:

- its current position;
- location of the final goal;
- maximal speed of motion:

The information about each adversary agent includes:

- its current position:
- its predicted motion trajectory;

Although we are able to plan a complete path at once, the feasibility of such a path is questionable due to the uncertainty of the prediction about the motion of the adversaries. Thus our approach is to set up a time interval over which we are fairly sure of the motion of the adversary, and make a subplan for that interval to a subgoal. The subgoal is chosen to be closer to the final goal and to have good observability in its vicinity, from which we can observe the activity of the adversaries to update our information about their motion. Then we iterate through this subplan process all the way to the final goal. There are two important time factors for each subplan:

1. subplan interval: the interval of time we are planning for in each subplan;
2. thinking time: the time required to generate a subplan. In our approach, this is a linear function of the product of the number of adversary agents and the length of the time interval we are planning for. We assume that the thinking time is known at the beginning of the subplan.

The criteria for a subgoal are:

1. It is safe at the end of the subplan interval (and preferably safe for a certain period);
2. It must be reachable at the end of the subplan interval through a safe path;
3. It is closer to the goal;
4. It has good observation points in its vicinity.

So the complete path planning contains the following loop:

```
WHILE final goal is not reached
 begin
      Pick a subgoal;
      Plan a path to the subgoal;
      Execute the subplan and update information;
 end.
```

The first two steps in the loop, namely the subplan process, are detailed in the following sections.

3.1.3. Algorithms

This section describes the algorithms in the current implementation. For each subplan, as a first thought, we may want to analyze the situation at the end of the subplan interval, pick a subgoal, and plan a safe path by computing the visibility map at each discrete time step and search through it. However, there are several serious drawbacks to this naive approach: when we pick up a subgoal, there is no guarantee that there is a path reaching that point at

the end of the interval, even if the subgoal is within the mobility constraint. Furthermore, even if such a path exists, we may spend a lot of time to find it or fail to find it.

Considering these factors, we propose a different approach by taking advantage of the power of parallelism. Instead of first analyzing the situation at the end of the interval, we make an incremental computation of the safely-reachable region for each time step, denoted as RS_t, where t is the time index. Initially, t is set to the end of the thinking time, because that is the time at which the friendly agent starts moving. $RS_{end\_of\_think\_time}$ is set to its current location. Then for each time step, we compute RS_t by first expanding RS_{t-1} according to its maximal speed, and then curtailing it by the regions visible from the predicted adversary positions at time t. After iterating this process, t is incremented to the end of the subplan interval, and it is guaranteed that there is a safe path to any point in the region $RS_{end\_of\_interval}$. Then we evaluate each point in this region by the criteria mentioned in the previous section and choose the best point as the subgoal.

With this method for choosing a subgoal, the existence of a safe path to the subgoal is guaranteed, but how can we find this path, especially without an elaborate search? This problem can be thought as a 3-D path planning problem, with the RS's defining a corridor along the time dimension. We want to find a path through this corridor that satisfies the speed constraint of the agent, which is interpreted as the angle between its moving direction and the time axis in the 3-D model. The solution is to build the path backward in time. We claim that for any point, say P, in RS_t, there exists a point Q in RS_{t-1} such that the agent is able to move from Q to P in one time step. This property comes from the way we build up the RS structure.

Thus we start at the subgoal and dilate it according to the speed constraint. The result defines the set of points the agent has to reach at one time step before the end of the subplan interval if we want it to be at the subgoal at the end of the interval. Then we determine the intersection of the set and the RS at that time. The result is a set of points that may be on our path. The property mentioned above guarantees that this set is non-empty. We have to pick one of them according to some goodness measure, and we currently pick the point with the least distance to the straight line between the last path point and the present location of the friendly agent, which tends to minimize the path length. [1] Then from the chosen path point we repeat the dilation-intersection-selection procedure, until the time index is decremented back to the end of the thinking time. Since the current position is the only point in the first RS region, the path must end up at the current location.

The algorithm for the subplan process can be summarized as:

[1] We were using the straight line between the subgoal and the current location, which didn't do very well since once we go astray, we may need extra effort to get back to the straight line.

Algorithm Subplan
Begin

{ Step 1: finding the subgoal by forward dilation }
$RS_{end\_of\_think\_time}$ = the current position of the friendly agent;
For t from the end of the thinking time to the end of the subplan interval
begin
$RS_t = dilate(RS_{t-1})$;
Predict the adversary positions at this moment;
Analyze the visibility from the adversary agents to the vicinity of the friendly agent;
$RS_t = RS_t \cap Non\_visible\_region$;
end ;

Evaluate points in $RS_{end\_of\_interval}$ and pick the best one as the subgoal;

{ Step 2: finding the path by backward dilation }
$PATH_{end\_of\_interval}$ = the position of the subgoal;
For t from the end of the subplan interval back to the end of the thinking time
begin
$PATH_t = dilate(PATH_{t+1})$;
$PATH_t = PATH_t \cap RS_t$;
$PATH_t = best\_point(PATH_t)$;
end ;
return($PATH$);
End

3.1.4. Extensions

This work is still in progress. We are presently investigating path planning issues such as the traversability of the terrain and the requirements for the friendly agent to observe the adversary at several points along the path, to remain close to other friendly agents, and to take risks on limited portions of the path in the absence of safe alternatives.

3.2. Range Shadows

In range images, there are regions behind obstructions that remain occluded (Figure 3). The sizes of these shadows depend on the relative elevation of the range scanner and the height of the obstruction. Many techniques have been developed for surface reconstruction. Most are based on weighted local averaging [3]. There are more elaborate techniques that also incorporate surface slope in local height averaging (for example, see [2]). These techniques often yield an obviously *incorrect* result, in the sense that the elevation of the reconstructed surface is overestimated and could not have been in shadow in the first place. We propose an algorithm that does not yield such incorrect results, while it preserves the statistical properties of the surface. The method is best suited for random stationary surfaces, such as rough terrain.

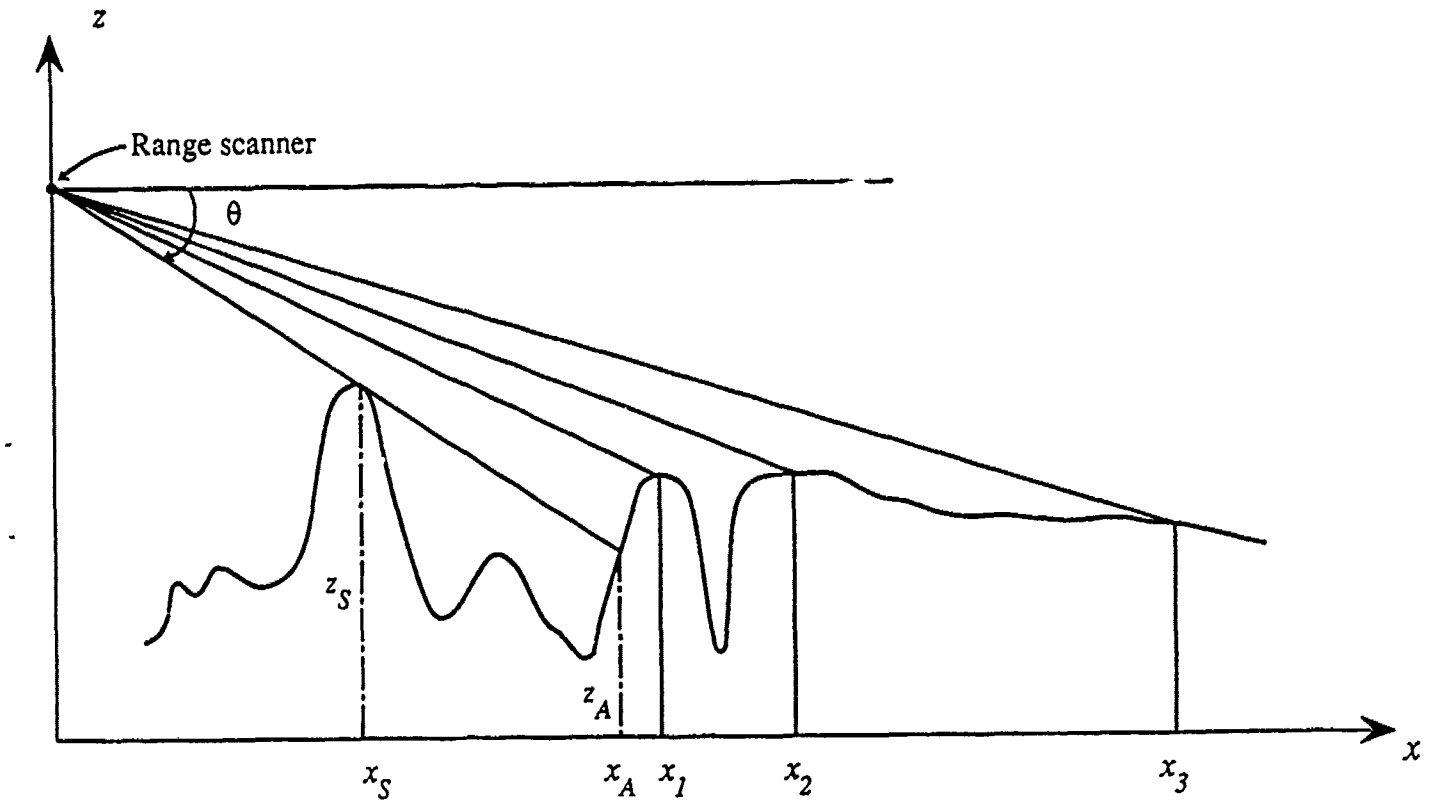


Figure 3: Schematic representation of a 1D surface and its range image shadows.

The algorithm estimates the surface elevation, z, at a given point $\vec{r} = (x, y)$. Regarding the elevation of a point in the range image shadow we have only one solid piece of information, namely, that the surface is below the shadow line, i.e. $z \leq z_S - |\vec{r} - \vec{r}_S|\tan\theta$, where θ is the angle of the shadow line with the x-axis, S is the point casting the shadow—how far below, we cannot know. In the absence of any other information about the depth of the shadow, it is reasonable to expect that the amount by which the surface lies below the shadow line is related to the variance of the surface elevation in the neighborhood. Therefore, as the simplest hypothesis, we take the point to lie *one standard deviation* below the shadow line, that is,

$$z = z_S - |\vec{r} - \vec{r}_S|\tan\theta - \sigma, \tag{2}$$

where σ is the standard deviation obtained from the variance

$$\sigma^2 = \sum_{i=1}^{n} w_i (z_i - \overline{z})^2 \tag{3}$$

In this equation, $\overline{z}$ and w_i are defined by the following expressions

$$\overline{z} = \sum_{i=1}^{n} w_i z_i, \tag{4}$$

where n is the number of nearest neighbors, z_i is the elevation of the ith neighbor, and the weight w_i is inversely proportional to some power of the (Euclidean) distance $|\vec{r} - \vec{r}_i|$ of the

point from its ith neighbor

$$w_i = \frac{1}{N\,|\vec{r} - \vec{r}_i|^{\nu}}, \qquad \text{with} \quad N = \sum_{i=1}^{n} \frac{1}{|\vec{r} - \vec{r}_i|^{\nu}}. \tag{5}$$

N is the normalization factor, and ν is a positive integer which is usually taken to be equal to 2. The number of neighbors, n, is usually between 3 and 8, and is often set equal to 5 [1].

The intuitive justification for this scheme, which we refer to as the lowering method, is that if the surface is smooth (small σ) then the chances are that the shadow will not be very deep. Obviously this scheme is well suited for reconstruction of random stationary surfaces, such as rough terrain, where the local elevation variance is a characteristic property of the surface and is meaningful. Nevertheless, even for reconstruction of surfaces behind obstacles on a flat road the results are remarkably good and definitely better than the weighted averaging technique (Figure 4) [Kamgar-Parsi, Narayanan and Davis, CS-TR-2298].

4. Conclusions and Summary

Our research program has focused on two inter-related goals over the past year:

1. The identification of new representations and algorithms for autonomous navigation that address fundamental problems in both vision and planning.

2. The development and integration of these algorithms on massively parallel computers.

Our research on navigation has focused on problems relating to the detection, pursuit and interaction with a three dimensional model of known geometry. Specific vision problems addressed included parallel algorithms for contour analysis for the purpose of identifying feature points in images of the target, parallel algorithms for instantaneous pose estimation of the target, and Kalman filtering algorithms for developing a model for the motion of the target through space. We are currently developing massively parallel algorithms for trajectory generation that will allow RAMBO to track and interact with the target on the basis of its predicted trajectory through space.

Turning to parallel processing, our research has uncovered several fundamental problems in the effective application of massively parallel computing to vision and planning. First, the importance of both multiresolution image analysis and focus-of-attention vision has led us to consider various methods for efficiently processing small images and other data structures on massively parallel computers. Our research to date has addressed the efficient processing of image arrays and contours. Second, the need to plan effectively in dynamic environments has led us to study trajectory planning algorithms both for RAMBO and for ground vehicles moving over terrain modeled by digital terrain models.

5. Report Abstracts and Summaries

This section presents the abstracts of all the research reports published under the present contract. For the reports related to our research effort on visual navigation, details have been given in the previous sections. In these cases we give a reference to the section in which these details appear. However, over the course of the past year, Ph.D. students and visiting researchers have also written reports which explore many other facets of artificial intelligence, and it was difficult to group the accounts of their work into a few stand-alone sections, because of the lack of common thread between the research directions. In these cases we have chosen to supplement the abstracts of these reports with extended summaries when the abstracts did not cover the whole scope of the research.

5.1. Rand Waltzman, "Geometric Problem Solving by Machine Visualization", CAR-TR-454, CS-TR-2291, July 1989

ABSTRACT: The goal of this research was to investigate automatic non-logical reasoning techniques based on principles of visualization in geometric domains and to develop a methodology for implementing these techniques in computer programs. At the heart of the methodology that I developed is the idea of a representation formalism for geometric objects that is both isomorphic and canonical and completely integrates metrical and topological information. Next there is a defined set of problem-solving operations for storing, retrieving, and manipulating objects represented in this formalism. These operations are unified by the fact that they each entail some type of search or construction process that is highly constrained by the structure inherent in the formalism and, *"via isomorphism"* by the geometrical structure of the objects involved. I refer to these operations collectively as machine visualization techniques. Finally, there is a programmable problem-solving system in which various functional components are implemented using these visualization techniques. While the problem-solving system interpreter directs the actions of the system's functional components, the behavior of the interpreter itself is controlled by the programmer through problem-solving heuristics written in a pattern matching language that is a direct extension of the basic representation formalism. This language allows the programmer to express constraints on problem-solving activity directly in terms of the geometrical structure of the problem. Thus, both low level and high level problem-solving activities of the system are highly constrained by problem structure. As a concrete illustration of this methodology, I have implemented a geometric problem-solving system for solving a subclass of three dimensional packing problems and have successfully applied the system to solving two non-trivial three dimensional jigsaw puzzles (formulated as packing problems).

SUMMARY: Our methodology is based on a representation formalism for geometric objects that integrates metric and topologic information. We define problem-solving operations for storing, retrieving and manipulating objects. These operations are constrained by the structure of the formalism and by the geometric structure of the objects. We call these operations *machine visualization* techniques. The behavior of the interpreter which controls the operations can be controlled by the programmer through problem-solving heuristics written in a pattern-matching language. As a demonstration of the power of this approach we solved two non-trivial three dimensional jigsaw puzzles.

Human reasoning can be broadly divided into two categories: logical and non-logical. Until now, research in artificial intelligence has concentrated almost exclusively on logical reasoning. However, human problem-solvers use powerful reasoning techniques other than logic that are rooted in fundamental visualization capabilities. We constructed a model of such reasoning that consists of several components. The essence of this model is a formalism which expresses the geometric constraints of the problem in a way which makes them available to the set of problem-solving operations. The formalism applies specifically to convex or concave polyhedra and is isomorphic to the actual structure of the polyhedra, in the sense that there is a one-to-one mapping between the formal representation and the actual structure of the polyhedra. Consequently, the formalism totally integrates the metric and topologic informations of the objects. Another powerful feature of the representation is that it is intrinsic, i.e. independent of any frame of reference. Therefore the operations can manipulate objects without any consideration of their location or orientation in space.

More specifically, polyhedra are thought of as being composed of polygonal faces joined together by surface edges to form specific dihedral angles. Polygonal faces are circular ordered sequences of edges of specific lengths joined together at particular angles. The ordering of face edges, the signs of the angles at which they are connected, and the signs of the dihedral angles connecting the faces serve to define the orientation of the polyhedral surfaces. This results in a two-layered structural organization that is implemented as a planar graph whose nodes and arcs themselves have structured values (Figure 5). Polyhedra are also thought of as being composed of three dimensional vertices which are joined together by surfaces edges of specified lengths. Vertices are thought of as circular ordered sequences of face angles of specified size joined at particular dihedral angles. Analogous with the face-centered decomposition, the ordering of the face angles and the signs of the dihedral angles connecting them determine the orientation of the polyhedral surfaces. The face centered and the vertex centered decompositions of a given polyhedron are dual to each other in the sense that one can be unambiguously and automatically derived from the other. Therefore the same data structure can be used for both decompositions. This offers many advantages in terms of the problem-solving operations that can be defined.

In practical terms the expression *machine visualization* refers to the collection of operations used to store, retrieve and manipulate the representation described above. For example we discuss the importance of being able to "see" when two different pieces are identical in shape and size. The system does this by comparing the representation of the two polyhedra, exploiting the two layered structural composition of the representation. It takes one of the nodes of the face-centered graph representation of the first polyhedron and checks to see if it matches one of the nodes of the graph representation of the second polyhedron. If no match is found then the two polyhedra are different. However if a match is found the match acts as an *anchor point* from which to do the remaining comparisons. Thus comparing the remaining nodes is no longer a case of comparing circular lists. Each attempted match is a simple list comparison. The two graphs are then jointly traversed using a breadth first search with appropriate second layer comparisons (dihedral angles for arcs and faces for nodes) made at each step.

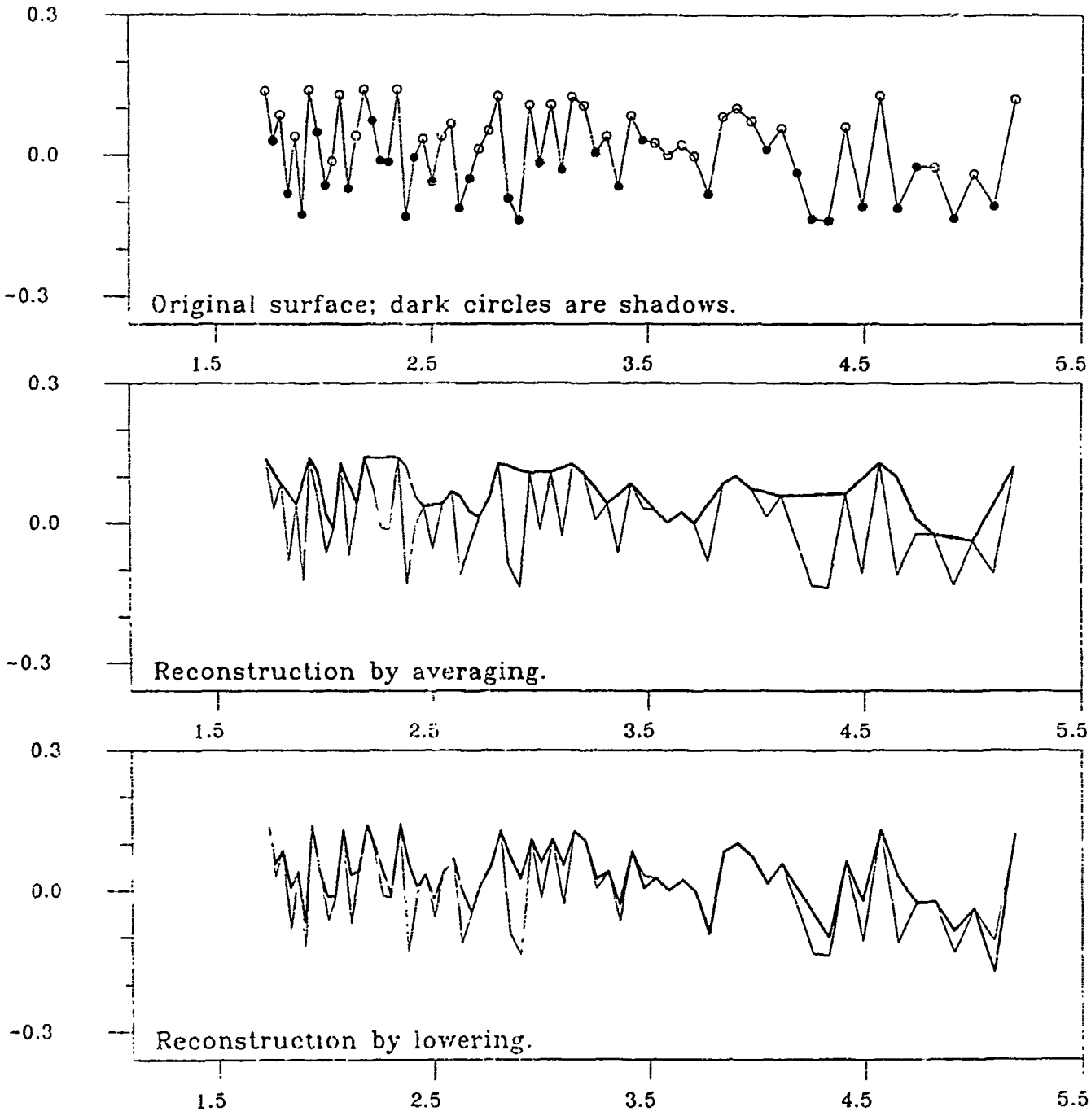


Figure 4: A digitized rough 1D surface and its reconstruction by the local averaging and lowering methods.

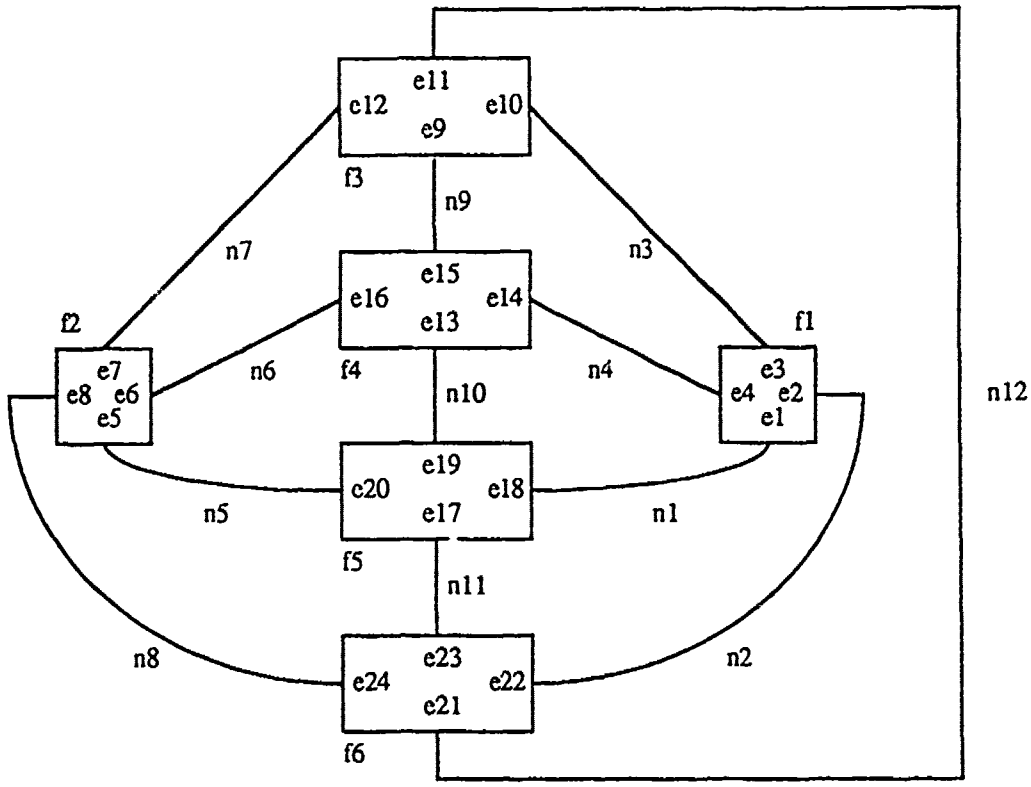


Figure 5: Graph representation for a rectangular parallelepiped.

The method described above is effective when we want to compare only two polyhedra. However to determine whether one polyhedron is identical to any number of other polyhedra we introduce a method that use an additional data structure called an *overlay graph*, that stores structural information regarding a set of polyhedra. The representation is unique. Common pieces of geometric structures share common corresponding parts of the storage data structure. Only local structure information is stored for recognition purposes. Overlay graphs are used to sort a set of polyhedra into groups of identical objects. We begin with an empty overlay graph. For each polyhedron in the set we first check to see if the structural information for that object is contained in the graph. If it is, then we take the name of the polyhedron associated with the structure and add the current object to the corresponding group. Otherwise we add the structural information for the current object to the overlay graph. The same technique is used to store old search states. We create an overlay graph that we use to store the structural information for the containers associated with each state. Then for each new state we determine whether the containers associated with that state appeared in a previous state by checking that overlay graph.

Another important visualization technique is that of *adjoinment*, i.e., putting two pieces together to form a new piece. When two pieces are brought into contact, certain faces remain unaltered while others either disappear entirely because they are completely covered by the faces of the other polyhedron or are modified because they are only partially covered. Because the representation is isomorphic and intrinsic to the actual geometric structure, it is possible to implement adjoinment as a straightforward splicing operation between the

representations of the objects being adjoined independently of location or orientation.

Another operation that is important to problem-solving is recognizing equivalent orientations of a given object. This operation uses a rotational symmetry detection algorithm. Piece placement instructions often have the form: Place piece A in container B so that vertex a of A coincides with vertex b of B. If vertex a' of A is equivalent to a under a rotational symmetry, then looks exactly the same from the viewpoint of a as it does from a'. This means that placing piece A in container B so that a' is coincident with b will result in placing A in an orientation and location equivalent to that resulting from the placement specified in the original instruction. The two-layered structure of the representation admits the use of a generate and test algorithm where both the generate and test parts of the algorithm are simple searches that are highly constrained by the structure of the representation.

Tentative piece placement decisions are heuristically made on the basis of structural features of the pieces to be places as well as the container into which they are to be placed. However the final decision to place a pieces depends on whether it actually fits into the desired location. To answer these question we have introduced a specialized containment algorithm that is based on a spatial localization process. The result of this process is to distribute the surface boundaries of the polyhedra in question into volume sectors so that surface boundary comparisons (checks for intersection) can be done simply and need be done only within each sector. Once containment is established, the piece is placed by subtracting its volume from that of the container. This subtraction process is similar to the adjoinment operation in that it is only a question of splicing together the representations of the piece and the container.

The system performed very well on two jigsaw puzzle examples. The first puzzle had six pieces. The possible number of state computations required to solve the puzzle without making any use of the problem structure is probably more than 27,000. The system was able to solve this puzzle in roughly 10 state computations. The second puzzle was much harder and had 12 pieces. A naive approach would require more than 3×10^29 state computations. The system was able to solve the puzzle in roughly 200 state computations.

This work is perhaps the first demonstration of a problem-solving system that solves non-trivial problems using isomorphic representations and non-logic based reasoning techniques. The pattern matching language for expressing problem-solving heuristics makes the system general and applicable to a broad class of problems. More importantly, it concretely illustrates the potential of a new methodology for constructing reasoning systems.

5.2. E.V. Krishnamurthy, "Semantic Petri Nets—Applications to Multiple Inheritance and Knowledge Acquisition Systems", CAR-TR-447, CS-TR-2258, June 1989.

ABSTRACT: This paper demonstrates the use of inhibitory Semantic Petri Nets (SPIN) for a topological net- k-oriented approach to default reasoning in multiple inheritance systems having exceptio- . The basic principles of this approach are explained and illustrated by examples. The superiority of SPIN over the Touretzky graph model is demonstrated.

SPIN can be extended to include probabilities, priorities, delays and time-outs. Hence it will be extremely useful for applications in semantic information based on connectionist

model and intelligent decision support systems. Also its logical structure permits its implementation in concurrent programming languages such as Ada and Occam. We also briefly explain the relationship between SPIN and temporal logic. Also we indicate its applicability to the cover and differentiate and propose and revise heuristic classification used in knowledge acquisition, diagnostic, and other related expert systems.

SUMMARY: Inhibitory Petri nets can be used for parallel knowledge representation. We have found that they are superior to NETL [29], marker propagation machines [30], and the directed graph model of inheritance system recently described by Touretzky [31].

An inheritance system is a representation system founded on the hierarchical structuring of knowledge. In these inheritance systems, knowledge is organized by first creating abstractions. By abstraction, we mean a collection of properties shared by members of a set. For example, sheep and humans share the properties of being mammals. When abstraction is organized by inclusion relations such as the one described above we can represent this by a directed tree; this tree is known as a "taxonomic hierarchy" or "inheritance hierarchy". The taxonomic hierarchy provides an efficient method of representation and makes search very efficient in knowledge-based systems. When several properties are orthogonal or overlapping, we get the situation of multiple inheritance where we have an abstraction that holds for most members but with certain exceptions. In such complex situations the tree representation is usually inadequate and there is a need for the use of a more general directed graph. We have found that the use of inhibitory Petri nets provides a very clean semantics in this situation and permits efficient parallel representation of knowledge and its retrieval. This is due to the powerful property of inhibitory Petri nets which can represent first order predicate logic efficiently and also generate and recognize context-sensitive and type-0 languages. We have also briefly explored the relationship between SPIN and temporal logic.

Knowledge representation and the associated control strategy play a key role in the design of expert systems. In particular, matching task requirements with the knowledge representation and control strategy is the key issue in knowledge engineering. SPIN is a good step in this direction, since (1) it is based on a very fundamental model of concurrent computation that pervades concurrent computer architectural and language design along with the associated interwoven pattern of data and control flow, and (2) it works on the intention-action mode that can take into account the inhibitory, nondeterministic and probabilistic behaviors which are essential to intuition and creative reasoning in human expert systems.

5.3. E.V. Krishnamurthy, "What Can the Petri Net Model Offer to Neural Network Studies?", CAR-TR-446, CS-TR-2257, June 1989.

ABSTRACT: This paper explores the possible application of a Petri net-like device as a node in neural network models. Such a model *called a Petri neural net* can be of great value in modelling neural processes exhibiting concurrency, asynchrony, intentionality and execution, nondeterministic choices (influenced by environment and learning, and independent of environment), inhibition, livelocks, deadlocks and divergence. This is illustrated by examples of Petri net modelling of multistable phenomena in vision. Specific analysis methods are described to understand the behavior of Petri neural nets. It is shown how properties such as categorization and content addressability exhibited by neural nets respectively correspond

to the problems of reachability and controllability in Petri nets. We also study some properties and limitations of existing neural nets and show how their computing power can be improved by including Petri net-like devices. The concept of Petri neural net can further be extended to include probabilistic concepts and this will lead to adaptive Petri neural nets which can be subjected to maximum entropy analysis. However, this analysis could become computationally intractable and even undecidable.

SUMMARY: Petri nets play a very important role in understanding the behavior of complex asynchronous concurrent and distributed computing systems. However the potentialities of the Petri net and related parallel computational models have not been explored in the study of neural networks and their modelling. We have shown how some of the salient aspects of Petri net and other models can possibly be combined with the existing neural network models to improve the modelling power of the latter and facilitate their simulation in concurrent asynchronous and parallel synchronous machines using concurrent programming languages such as Ada, CSP and Occam 2. We illustrated the use of the Petri net model for multistable phenomena in vision. We suggested using a Petri net-like device as a node in a neural net. We also discussed how Petri neural nets may be analyzed using linear algebraic techniques based on consistency checks. The use of consistency checks in Petri nets, as well as in Hopfield neural nets, was illustrated by examples. This study showed that Hopfield and other related neural nets essentially model proof by refutation in propositional logic by linear algebraic methods. In this sense they are no more powerful than a subclass of Petri nets called "marked graphs" and deterministic context-free production systems. We also dealt with stochastic Petri net-like devices and their learning behavior. In conclusion we indicated how Petri net-like devices can play an important role in the design of neural networks although their analysis could lead to computationally intractable and undecidable issues.

The examples on multistable phenomena in vision clearly demonstrate the potential of Petri neural nets. Presently, neural net models do not deal with asynchrony, concurrency, nondeterminism, intentionality and execution; Petri neural nets would permit inclusion of all these aspects thereby enlarging their scope to hybrid (digital and analog) neural nets. Also existing neural nets essentially model propositional logic via linear algebraic systems which are no more powerful than context-free production systems; unless the neural nets include facilities that can model predicate logic involving binding, unification, substitution and resolution their decision power cannot be increased. However this may result in computationally complex and undecidable issues. The Petri net model can be fitted together with the concept of tensor-product networks. This should enable us to understand coordination and motor learning behavior. Further research in this direction would permit the cross-fertilization of ideas from formal language theory, logic, computability and complexity theory, thereby improving our understanding of the immensely complex domain of neural nets.

5.4. E.V. Krishnamurthy, "Modelling Dynamically Constrained Distributed Programs", CAR-TR-455, CS-TR-2292, July 1989

ABSTRACT: Semantic Petri Net (SPIN) modelling of dynamically constrained distributed programs is discussed. Unlike Leler's constraint graph and Numao's cell-relation model, SPIN can be automated for consistency checking based on T-invariants during the update

propagation. Hence it is useful for the design of robust distributed programs in declarative languages.

SPIN can also model data-structure-resident parallel processes having different granularities.

Further studies on temporal logic based Petri nets are needed to understand dynamically constrained distributed systems.

SUMMARY: Constraint programming is a new paradigm in which the programmer states a set of relations (including exceptions and constraints) among a set of well-defined objects. The system then finds a solution that satisfies these relations together with exceptions and constraints. The constraint programming system has applications in areas such as CAD, CAM, graphics and AI, to mention only a few. Leler [28] defines a general purpose specification language based on a constraint graph model, that allows a user to describe a constrained satisfaction system using rules. However, this approach does not provide a systematic reasoning procedure for answer extraction and consistency checking. Also the constraint graph is not a suitable model when the objects themselves are processes defined on data types among which relations are specified, e.g. concurrent programs arising in robotics, service and protocol specification in asynchronous distributed systems. We have shown how a Semantic Petri Net (SPIN) can model constraints, as well as data-structure-resident processes. Reformulating a constraint program as SPIN rather than as a constraint graph has several advantages: (1) The constraint graph model is a passive network; hence, it cannot express intentions and actions. SPIN, however, provides the intention-action mode that can be converted to guarded Horn clauses for constraint satisfaction. (2) The constraint satisfaction during an update propagation in a Petri net corresponds to the firing of the different transitions; this amounts to computing a non-zero T-invariant for the incidence matrix of the Petri net. This is equivalent to using resolution to prove that a set of clauses contains a contradiction, hence it is an algorithmically superior model in comparison to the constraint graph of Leler for the representation as well as analysis of constraint programs. (3) SPIN can model distributed programs that include data-structure-resident processes, such as queues, arrays, etc.

5.5. Anup Basu and John (Yiannis) Aloimonos, "An Efficient Algorithm for Motion Planning of Multiple Moving Robots", CAR-TR-457, CS-TR-2297, August 1989.

ABSTRACT: The problem of coordinating the motion of multiple robots of the same size translating in the plane is examined. First the complexity of the decision problem is shown to be polynomial. Then the complexity of the problem of obtaining the final configuration from the initial one with the minimum number of moves is determined to be NP-complete. Finally, we present an efficient algorithm for solving the problem with expected time $O(n^2\sqrt{n})$, provided the distance between the initial and final configurations is at most of the same order. This complexity can be further improved since the proposed algorithm is parallelizable.

SUMMARY: We have addressed the problem of efficiently coordinating the motion of multiple objects in a discretized environment when there is sufficient space to guarantee that a solution to the problem exists. Here, as in several previous papers, we considered a special

case of the problem. In particular we considered a rectangular workspace area and all elements as squares. This subcase of the problem has applications in assembly of electronic circuits where the components are laid out on a grid, memory management in distributed systems, mobile robots in a structured workspace.

Consider the problem of 2D memory management. The programs can be thought of as rectangles in a 2D workspace (memory). One may encounter the problem of rearranging the programs so as to collect the free memory space from time to time (this problem is also known as compaction). The problem would then be to devise efficient algorithms to perform this task.

Another problem of interest is rearrangement of goods in an automated warehouse. It is easy to design simple robots which are capable of pushing the boxes (containing goods) around. Goods may be used up over time in a warehouse and replacement usually occurs in bulk. To help in efficiently retrieving items one may be interested in rearranging the warehouse at certain times depending on changing demand patterns. If multiple robots capable of pushing boxes exist then one would be interested in efficient parallel algorithms for warehouse rearrangement.

In some factory environments robots can be made to follow certain paths on a grid. Usually the grids are laid out below the factory floor and each robot is programmed to pick up some specific signals. The problem here is how to efficiently coordinate the movement of many robots in a cluttered environment.

The difficulty in the problem is inherently due to the shortage of free space. The algorithm works as follows. In the first step the original rearrangement problem is divided into smaller problems. These subproblems are solved locally, eliminating the need for priority graphs. Elements are then exchanged across the boundaries. The process of solving subproblems locally and exchanging across boundaries continues until some intermediate final configuration is obtained. Space is rearranged from the intermediate final configuration to obtain the final arrangement. The crucial step in the process is how to divide the original problem into subproblems. For this an iterative scheme and a recursive scheme are described. The recursive procedure is shown to have a better average case performance. We have implemented a distributed and parallel version of our algorithm.

5.6. Behzad Kamgar-Parsi, P.J. Narayanan and Larry S. Davis, " Reconstruction (of Rough Terrain) in Range Image Shadows", CAR-TR-458, CS-TR-2298, August 1989.

ABSTRACT: The common approach to estimating the surface elevation at a given point is based on weighted averaging of the elevations of the neighboring points. This approach often yields incorrect results for surface reconstruction in range image shadows: the elevation of the shadow region is often overestimated such that it could not have been occluded in the first place. In this paper we propose an algorithm that does not yield such incorrect results, and preserves the statistical properties of the surface. The method is best suited for reconstruction of random stationary surfaces, such as rough terrain. It has applications in off-road autonomous land navigation (see Section 3.2. for details).

5.7. Tapio Seppänen, "Speed-up Analysis of Parallel Programs with an Image Analysis Program as a Case Study", CAR-TR-459, CS-TR-2302, DACA76-88-C-0008, August 1989.

ABSTRACT: In this report we preliminarily introduce a method for efficiency analysis of parallel systems. The method is constructed on the concept of critical paths of concurrent programs. Utilizing timing profiles of a program running on the desired target computer, a model explaining the speed-up behavior of the program is constructed. The model will be later used for pointing out those factors that seem to affect the speed-up behavior of the program. An image analysis algorithm is implemented on the Butterfly Parallel Processor (BPP) and the experiment is used as a testbed for the method. The image analysis program performs a gray-level connected components analysis and feature extraction for area-segmented images. A gray-level connected components analysis is first carried out to generate an area-segmented labelled image. Then intra-area and inter-area features are computed for the area segments, including adjacency graphs. Parallel computation is achieved by dividing the input image in equal sized blocks and assigning each block to a different processor of the BPP, according to the principles of data parallelism. An asynchronous slave process is attached to each image block, and an asynchronous master process controls the slaves via synchronizing barrier variables. Mutual exclusion among the slaves is implemented with locking primitives. Both of the main stages of the program contain three steps: first, image blocks are processed locally as long as possible; second, a description of block interfaces is created to be utilized in the third step, in which the partial results are merged.

SUMMARY: The good match of our model to the timing measurements indicates that the essential features of our concurrent program influencing its speedup behavior have been grasped. The program described above includes several parallelizable loops and hence provides a good vehicle for the development of analysis methods. We demonstrate that if a concurrent program can be divided into consecutive, separately parallelizable segments, which are synchronized to each other by appropriate techniques, the speedup analysis of the resulting concurrent system can be carried out by analyzing the segments separately with the proposed critical path analysis approach, and then combining the subresults to achieve the properties of the whole system.

5.8. Behzad Kamgar-Parsi and Behrooz Kamgar-Parsi,"On Problem Solving with Hopfield Neural Networks", CAR-TR-462, CS-TR-2310, DACA76-88-C-0008, August 1989.

ABSTRACT: Hopfield and Tank have shown that neural networks can be used to solve certain computationally hard problems; in particular, they studied the Traveling Salesman Problem (TSP). Based on network simulation results they conclude that analog VLSI neural nets can be promising in solving these problems. Recently, Wilson and Pawley presented the results of their simulations which contradict the original results and cast doubts on the usefulness of neural nets. In this paper we give the results of our simulations that clarify some of the discrepancies. We also investigate the scaling of TSP solutions found by neural nets as the size of the problem increases. Further, we consider the neural net solution of the Clustering Problem, also a computationally hard problem, and discuss the types of problems

that appear to be well suited for a neural net approach.

SUMMARY: In their seminal paper, Hopfield and Tank [24] showed that neural networks can be used to solve computationally hard problems such as the Traveling Salesman Problem (TSP). To investigate how well the network performs they simulated its behavior and found very encouraging results, in that the network frequently finds valid solutions of high quality. Recently, Wilson and Pawley [25] reported the results of their simulations of the Hopfield and Tank solution of the TSP. Their results differ from those presented in [24] in two respects: (i) the number of trials yielding valid solutions is considerably less than that reported in [24]; and (ii) the solutions found by the network are not much better than randomly selected tours.

We have performed simulations of the Hopfield and Tank solution of the TSP (Figure 6). We found that, indeed, the number of times the network succeeds in finding valid solutions is considerably less than that found by Hopfield and Tank. However, we found that when the neural net finds a valid solution it is of remarkably good quality. Further, we showed how the success rate of the network in finding valid solutions can be markedly improved.

We also investigated how the neural net solution of the TSP scales with the size of the problem. The results are not encouraging, in that the scaling is poor. Although the quality of solutions that are found by the network remain good, finding valid solutions becomes increasingly difficult as the size of the problem increases. This suggests that neural nets may not be suitable for solving computationally hard problems. However, there does not appear to be a universal answer to this question, because there are other computationally hard problems, such as clustering, that appear to be well suited for the neural network approach and have good scaling properties [26].

In deciding whether the Hopfield model of neural networks is suitable for solving a computationally hard problem, one has to consider two factors: (i) how good are the solutions; and (ii) what is the success rate for finding valid solutions.

It appears that when the analog network finds a solution its quality is generally very good. A specific solution, of course, depends on the chosen initial state. There is overwhelming empirical evidence (based on Monte Carlo estimates) that deeper minima of the energy function of an analog network have generally larger basins of attraction; thus a randomly selected initial state has a higher chance of falling inside the basin of a deep minimum and finding a very good solution. Hopfield neural nets are complicated dynamical systems, and as yet, there are no theoretical estimates for the sizes of these basins. Therefore, one cannot give the probability of finding the best solution. However, one can claim that analog networks greatly enhance the probability of finding very good solutions. This is true for both problems—TSP and clustering—we have investigated in this work.

The success rate appears to be dependent on the problem. For TSP the success rate is rather modest; in particular, it scales poorly as the size of the problem increases, which suggests that neural nets may not be suitable for solving TSP. For the clustering problem, on the other hand, the success rate is very high, nearly 100%, and its scaling as the number of points increases is excellent. The main difference between these two problems is in their matrix representations of valid solutions. A given tour in the N-city TSP problem is represented by a $N \times N$ permutation matrix with constraints on both the rows and the

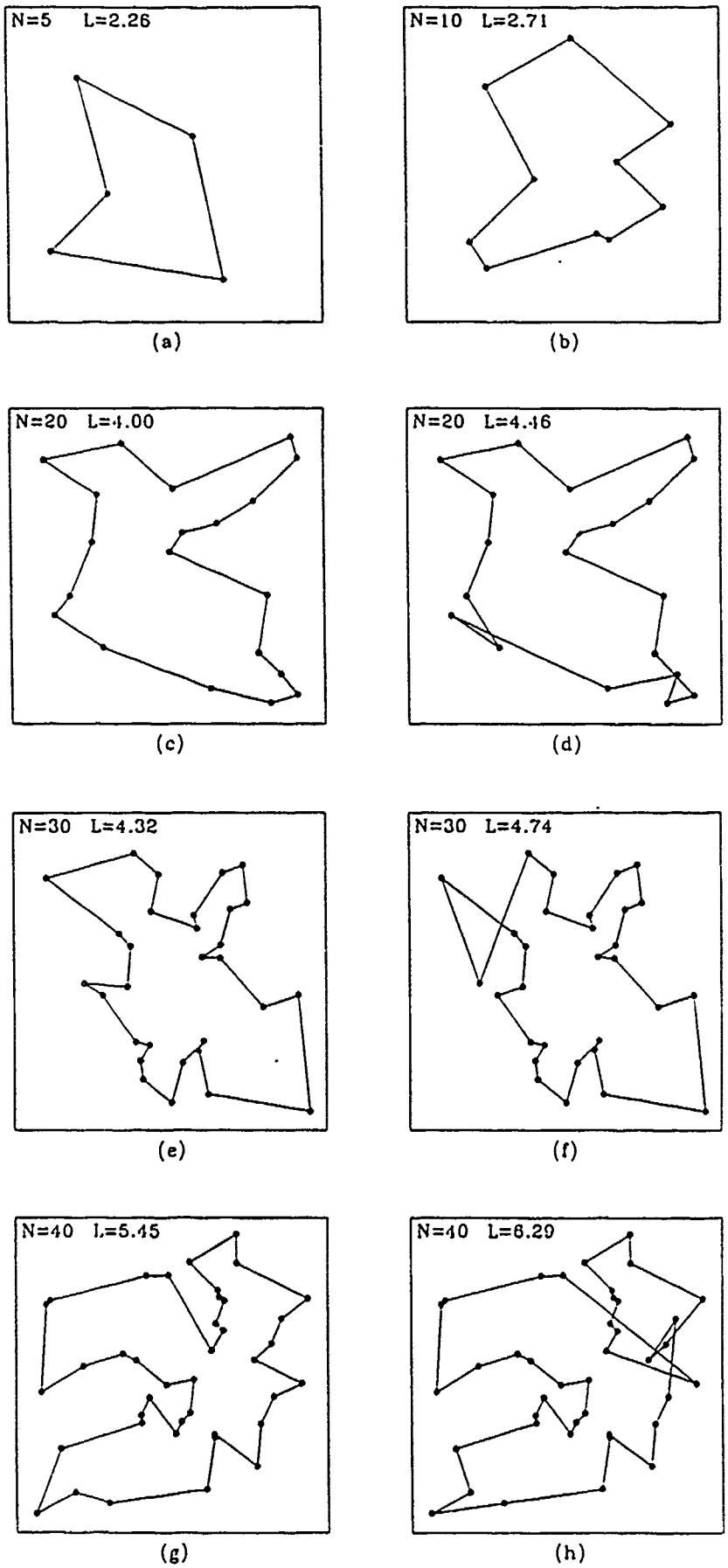


Figure 6: Shortest tour for 5 cities (a) and 10 cities (b). Shortest tours (c, e, g) and best found tours (d, f, h) for 20, 30 and 40 cities.

columns, which is a hard syntax to satisfy. A given partitioning of N points among K clusters in the clustering problem is represented by a $K \times N$ matrix with constraints only on the columns, which is a much easier syntax to satisfy. This appears to be the reason for the much higher success rate of the clustering problem in finding valid solutions. Furthermore, the scaling of the success rate with problem size may be related to the density of "on" neurons in the syntax matrix, i.e. the ratio of 1's to the total number of elements in the syntax matrix. For TSP this ratio is $N/N^2 = 1/N$, which scales inversely with the size of the problem N. For the clustering problem this ratio is $N/KN = 1/K$, which scales inversely with the number of clusters K. Since in most problems K is small and much less than N, increasing the number of points should not adversely affect the scaling, which is confirmed by our results.

Based on these observations, it appears that problems that are suitable for a neural net approach are those whose valid solutions can be represented with few constraints, i.e. easy syntax. From simulations, we find that neural nets converge to solutions very rapidly, within a few τ. The simulations, however, require a great deal of computer time as the network must be updated thousands of times. The real benefit of neural nets, however, may lie in the future when they can be mapped on analog chips. Analog VLSI neural nets are being developed; these devices have very fast processing times in the micro to millisecond range [27], thus making their use in solving suitably chosen computationally hard problems very attractive.

5.9. Sharat Chandran, "Merging in Parallel Computational Geometry", CAR-TR-468, CS-TR-2333, October 1989.

ABSTRACT: In this report we consider problems in parallel computation and computational geometry. The goal is to come up with a solution that is as fast as possible by using a large number of processors, without being too inefficient in the total number of operations performed by the processors. For a problem of size n we will typically employ n processors, and will accept $O(log^n)$ time algorithms as fast. If we perform about the same number of operations as the sequential algorithm, the algorithm will be deemed cost-optimal.

We ıde a cost-optimal parallel solution to the problem of computing the area of a ur . rectangles that runs in logarithmic time. The previously best known parallel algc . ns in $O(n)$ time, and the parallel lower bound for this problem using n processors is $O(\log n)$. We are able to do this by constructing an efficient parallel data structure using the parallel *plane-sweep* technique and pipelined merge sort.

The same technique can be used to solve other problems in VLSI and computer vision cost-optimally. Problems in dynamic computational geometry, for instance, demand a variation on the above scheme. These are rather straightforward to tackle in the uniprocessor version; we introduce the *dynamic merging* technique to solve these problems in the parallel context. The algorithm is cost-optimal and runs fast. Sometimes parallel solutions can be obtained by finding new geometric insights rather than developing new algorithmic techniques. We are able to unify the solution to the problem of computing inscribed and circumscribed approximations to a large sided convex polygon by one such characterization. Previous algorithms for the two problems were different and had more complex correctness proofs. We also

substantially improve the time complexity to $O(\log \log n)$ using $\log \log n$ processors. This is one of the first few examples of a non-trivial parallel algorithm in computational geometry that runs in sublogarithmic time.

An orthogonal issue is that of characterizing parallel algorithms. Theories for sequential computing developed over the years cannot be readily adopted for parallel computation. Cost-optimality is one indication of a good parallel algorithm. We suggest that the number of processors times the square of the time (i.e. pt^2) is another indication. The analogy to the quantity "area-time-squared product" (i.e. at^2) in VLSI is obvious. Our models essentially emphasize to the user of parallel machines that different parallel algorithms may need to be employed depending upon the problem size and multiprocessor size.

5.10. Subbarao Kambhampati, "Flexible Reuse and Modification in Hierarchical Planning: A Validation Structure Based Approach", CAR-TR-469, CS-TR-2334, October 1989

ABSTRACT: Generating plans from scratch is a computationally expensive process. A general framework for minimal modification and reuse of existing plans in new problem situations can significantly improve the efficiency of this process. Although some past planning systems addressed this problem, their methods did not pay adequate attention to the flexibility and minimality of modification.

This report provides an integrated framework for flexible reuse and modification in hierarchical nonlinear planning. The framework, implemented in the PRIAR system, is based on the *validation structure* of the plans. The validation structure is a formal representation of the internal dependencies of the plan. It is annotated on the plan by the planner to make later reuse more effective.

Reuse of a plan is formally characterized as the process of repairing the inconsistencies that arise in its validation structure when it is used in a new problem situation. The repair process is domain-independent and exploits the validation structure to remove any inapplicable parts of the plan and to suggest additional high level refit tasks to be achieved. The planner is then invoked to reduce these refit tasks, and is guided in this process by a heuristic control strategy which utilizes the validation structure of the plan. The heuristic strategy localizes the refitting by minimizing the disturbance to the applicable parts of the plan. The validation structure is also used to judge the utility of reusing the plan in a given new problem situation, forming the basis for plan retrieval.

The flexibility and the efficiency of plan modification in the PRIAR framework are evaluated both through empirical studies and through formal characterization of the reuse processes. The coverage of its modification techniques for hierarchical nonlinear planning is formally characterized in terms of the plan validation structure.

5.11. Daniel DeMenthon and Larry S. Davis,"New Exact and Approximate Solutions of the Three-Point Perspective Problem", CAR-TR-471, CS-TR-2351, November 1989.

ABSTRACT: Model-based pose estimation techniques which match image and model triangles require large numbers of matching operations in real world applications. We show that

by using approximations to perspective, lookup tables can be built for each of the triangles of the models. Weak perspective approximations have been applied previously to this problem; we consider two other perspective approximations, paraperspective and orthoperspective. We obtain analytical expressions which are as simple as those obtained by weak perspective. These approximations have much lower errors for off-center images than weak perspective. The errors of these approximations are evaluated by comparison with exact solutions. The error estimates show the relative combinations of image and triangle characteristics which are likely to generate the largest errors. The corresponding cells of the lookup tables can be flagged, so that object pose calculations would disregard image and model triangle pair combinations when their characteristics correspond to the flagged cells (see Section 2.3.2. for details).

5.12. Zen Chen,"A Flexible Parallel Architecture for Relaxation Labeling Algorithms", CAR-TR-474, CS-TR-2355, November 1989.

ABSTRACT: The design of a flexible parallel architecture for both the discrete relaxation labeling (DRL) algorithm and the probabilistic relaxation labeling (PRL) algorithm is addressed. Through proper space-time arrangement of the computation steps involved, the relaxation labeling processes can be run on a systolic-array like architecture in linear time for each iteration. Thus a high degree of computation parallelism is obtained. The arrays use one-dimensional, one-way communication lines between adjacent PEs and interface with the external environment through only a single I/O port. Because of the hardware simplicity and programmability features of the PEs, the architecture is well suited for VLSI implementation and is flexible enough to execute different relaxation algorithms. An illustrative example of running a region color labeling problem on the proposed architecture is described and a general running procedure is also given. Finally, performance comparison between the proposed architecture and other existing ones is presented in some detail.

SUMMARY: Relaxation labeling algorithms have been applied successfully in a number of fields, for instance, signal restoration and identification analysis, language identification, graph or subgraph homomorphism, image segmentation and pattern recognition, scene interpretation and understanding problems. One of the major issues of the labeling algorithm is that it is computationally intensive and typically requires exponential time with a single-processor architecture. Many researchers have presented multiprocessor approaches and tried to build fast architectures to support relaxation labeling algorithms because the problem can be divided into mutually exclusive subproblems and is well suited for parallel processing. These techniques are often faced with difficulties in task partitioning, scheduling and synchronization for relaxation operations among multiprocessors. Or they may have an appealing run time performance from a theoretical viewpoint, but suffer from physical realization problems such as data communication congestion and I/O routing complexities. Most of these approaches remain in the phase of virtual software. Lately, there is an increasing interest in implementing relaxation operations by using dedicated VLSI hardware architecture that can be built in a modular fashion, for instance, in the form of systolic arrays. Several preferential factors of the systolic approach, such as the feasibility of pipelining, high degree of parallelism, and avoidance of global communication, make the arrays

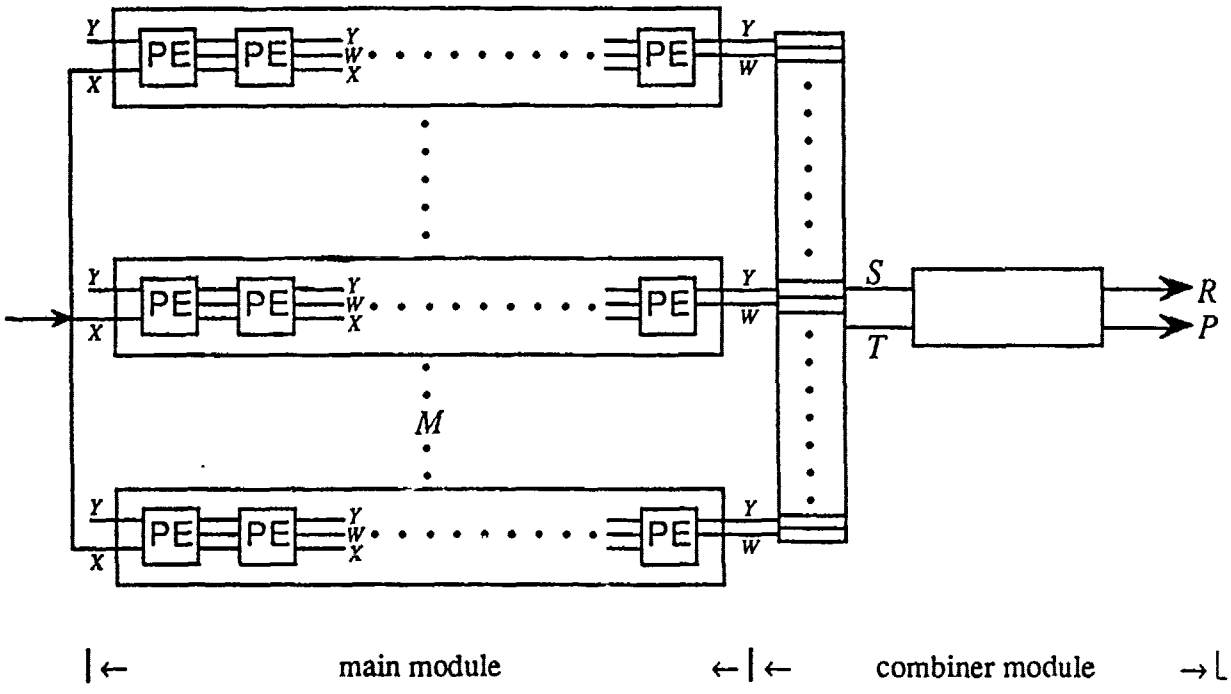


Figure 7: The organization of the proposed architecture.

suitable for VLSI fabrication. The underlying idea behind these methods is the application of suitable transformations to the relaxation algorithms to obtain a representation that can be easily mapped onto their proposed architecture. Thus, different relaxation labeling algorithms give rise to different array designs. However, the function of the resultant systolic array is somewhat restricted simply because the applied architecture is too fixed to cover different applications. In our work, we considered the relaxation labeling algorithms in two different forms: discrete relaxation labeling (DRL) and probabilistic relaxation labeling (PRL). Both of them are solvable on our proposed systolic architecture. The arrays use one-dimensional and one-way communication lines between adjacent PEs and interface with the external environment through a single I/O port (Figure 7). Because of the hardware simplicity and programmability features of the PEs, the architecture is well suited for VLSI implementation and is flexible enough to be adaptable to a number of applications, without compromising both speed and hardware complexity. Moreover, the proposed architecture runs in linear time for each iteration of the labeling process, and is also able to detect the consistency status of DRL as well as the convergence condition of PRL at the hardware level without host involvement. On the other hand, the use of one-way flow between the arrays simplifies the circuit design and the system can be converted to self-timed arrays to avoid the clock skew problem for large labeling processes.

We have introduced a flexible parallel architecture for relaxation labeling algorithms. A transformation scheme is used for parallel computation of the supporting evidence so that a high degree of parallelism is feasible. We simplify the control structure of the relaxation operations to clear away the difficulty of complicated data communication between objects and their neighbors. We preload the compatibility coefficients which will reside in each PE so

that only the object data need to be circulated among the PEs. This speeds up the process. The one-dimensional and one-way layout of systolic arrays is suitable for fault tolerant and self-timed circuit design; thus it increases reliability and avoid the clock skew problem. Our performance evaluations show that the proposed architecture is generally superior to existing systems.

In order to verify the design of the proposed systolic array and the combiner module, a simulation software package called the DAISY system [32] has been used to check the specification. We have finished the logic simulation to check the timing sequence for the architecture operation and the alphanumeric simulation of signals to verify the intermediate processing results. The experiments show that the proposed architecture is working properly. Future research includes the development of the VLSI custom chip and proper modification for covering a wide spectrum of related algorithms.

5.13. Shie-rei Huang and Larry S. Davis, "Speed-Quality Tradeoffs in Heuristic Search", CAR-TR-486, CS-TR-2396, February 1990.

ABSTRACT: In this paper, we study speed-quality tradeoffs in heuristic search both theoretically and empirically. We start by analyzing a heuristic search algorithm which is equivalent to the weighted heuristic search algorithm originally suggested by Pohl. We show that this algorithm, when using a heuristic with "constant relative error", can find an optimal solution with *linear* complexity, as opposed to A^* which has *exponential* complexity. We then prove some theorems to illustrate that the performance of this algorithm is determined by the *relative* accuracy of heuristics which may in some cases *improve* with increasing distance from the goal, in contrast to that of A^* which is determined by the *absolute* accuracy of heuristics which almost always *deteriorate* with increasing distance from the goal. New dynamic weighting schemes which are to some extent opposite to what Pohl had suggested are proposed. Finally a new heuristic search algorithm which attempts to exploit speed-quality tradeoffs is presented. This algorithm was tested on the Flow-Shop Scheduling Problem and detailed experimental results are provided.

SUMMARY: We present a new heuristic search algorithm which exploits speed-quality tradeoffs. It describes the notion of "constant relative error" of a heuristic function, in which the ratio of the difference of the estimated error and the actual error to the estimated error is constant for all nodes. We show that a particular existing heuristic search algorithm can produce an optimal solution in linear time provided that the heuristic function used has constant relative error. We prove various theorems relating the relative accuracy of heuristic functions to the performance of the heuristic search algorithms. The new heuristic search algorithm presented has the property that it computes ever-improving solutions as it progresses to the final optimal solution. Thus it could be used in a speed-quality tradeoff scheme in which sub-optimal solutions could be provided in situations where time was critical, and better solutions could be provided where time constraints were more relaxed but solution quality was important. The new algorithm has the flavor of simulated annealing, in that it begins by quickly finding a solution of unknown quality, and then incrementally improves the solution as computation progresses. However, unlike simulated annealing which relies on random processes to improve the solution quality, the new algorithm relies on the heuristic function itself. The algorithm is based on an evaluation function which is a weighted sum

of the cost found to reach a node and the estimated cost remaining to a goal node, i.e. the heuristic function. However, through successive iterations improving the solution, the values of the weighting factors are adjusted based on the cost of the best solution found so far. With each iteration the adjustment of weighting factors tends to favor the cost-to-node summand over the estimated-remaining-cost summand in the evaluation function. This process is shown to eventually produce the optimal solution to the search problem. Detailed results of experiments with the flow-shop scheduling problem are given.

5.14. Shie-rei Huang and Larry S. Davis, "Sequential and Parallel Heuristic Search under Space Constraints", CAR-TR-497, CS-TR-2438, DACA76-88-C-0008, March 1990.

ABSTRACT: Practical applications of heuristic search algorithms, such as A^*, are often thwarted by memory limitations. The memory limitation of A^* is overcome by the Iterative-Deepening-A^* (IDA^*) algorithm. It has been claimed and is widely believed that IDA^* is asymptotic time optimal. We argue in this paper that this claim is only true for some limited problems such as the 15-Puzzle but is not true generally. A sequential and a parallel heuristic search algorithm which not only overcome memory limitations but also make effective use of available memory to better exploit space-time tradeoffs are then presented. These algorithms has been tested on the Flow-Shop Scheduling Problem and experimental results are provided.

SUMMARY: We analyze an existing, linear-space variation on A*, called IDA*, which is widely believed to be asymptotic time optimal, and presented evidence which contradicts this belief. Through careful argumentation we demonstrate that the asymptotic time optimality of IDA* depends on the existence of a negative correlation between the number of "ties" of values of the evaluation function taken over the set of state-space nodes and the magnitudes of the node values themselves. We argue that this correlation holds for the particular search problems and heuristics discussed by the proponents of IDA* but does not hold for some other interesting search problems and heuristic functions. The existence of this correlation was embodied in a particular assumption of the proponents of IDA*, specifically that the algorithm searches deeper in the search tree with each new iteration. Besides this analytical evidence, we describe experimental evidence of the non asymptotic time optimality of IDA* obtained from experiments with the flow-shop scheduling algorithm.

We then describe and analyze a new tree search algorithm, AD*, which is a hybrid of the A* and IDA* algorithms, and which operates by alternating between two modes: the A* mode is used until a preset limit M on the number of nodes on the OPEN list is reached, and the IDA*-like mode is used where this limit is exceeded. The difference between the IDA*-like mode of AD* and IDA* itself is that with each iteration, after updating its threshold value, IDA* performs a depth-first search from the start node of the search tree, whereas the IDA*-like mode of AD* performs a depth-first search from one of the nodes on the OPEN list. By changing the value of M appropriately, the algorithm can provide a desired space-time tradeoff. We analyze results of experiments performed using AD*. We also analyze a parallel version of AD*, called PAD*, which was based on our own earlier PIA* algorithm, and we obtain experimental results for PAD*.

5.15. Ling Tony Chen and Larry S. Davis, "A Parallel Algorithm for List Ranking Image Curves in O(log$N$) Time." CAR-TR-501, CS-TR-2458, April 1990.

ABSTRACT: This paper describes an algorithm for ranking the pixels on a curve in O(log N) time using a CRCW PRAM model. The algorithm accomplishes this with N^2 processors for an $N \times N$ image. After applying such an algorithm to an image, we are able to move the pixels from a curve into processors having consecutive addresses. This is important on hypercube-connected machines like the Connection Machine because we can subsequently apply many algorithms to the curve (such as piecewise linear approximation algorithms) using powerful segmented scan operations (i.e. parallel prefix operations). The algorithm was implemented on the Connection Machine, and various performance tests were conducted.

5.16. Kari Pehkonen, David Harwood and Larry S. Davis, "Parallel Calculation of 3-D Pose of a Known Object in a Single View." CAR-TR-502, CS-TR-2459, April 1990.

ABSTRACT: This paper describes a selective generate-and-test algorithm for SIMD-parallel calculation of the 3-D pose of a known object from a single perspective view. The algorithm consists of three main stages: object pose estimation, optimization, and reliability analysis. The first stage involves parallel generate-and-test of candidate pose solutions obtained by selectively matching model and image point triples, testing their correspondence by parallel transformation of all visible model points and comparison of their features with the image. Instead of exact algebraic calculations, the three point perspective pose estimation problem is solved numerically. In the second stage, the initial pose estimate is optimized using a least-squares method. In the final stage, the reliability of the optimized pose is estimated using covariance analysis. The simulations showed that the approximate initial pose estimates are sufficiently good to obtain very accurate optimized results. Furthermore, the processing times for an object with 12 vertices were on the order of a few hundreds of milliseconds on a Connection Machine-2 with 512 floating point units (see Section 2.3.2. for details).

5.17. Olli Silvén, "Estimating the Pose and Motion of a Known Object for Real-Time Robotic Tracking." CAR-TR-503, CS-TR-2461, April 1990.

ABSTRACT: This report deals with the problem of estimating the pose and motion of a known object in three dimensions by using an initial estimate and a sequence of monocular images. The camera is assumed to be attached to a robot arm used for vision-controlled tracking of the object. The finite image resolution, robot errors, and changes of object motion during the non-zero delays for image acquisition and analysis introduce uncertainties into the measurements and estimation process. The uncertainties are handled by an extended Kalman filtering technique that produces object pose and motion estimates from feature point measurements in the images. The important problem of selecting the feature points used for estimation is also considered. Usually several feature points are available from an image frame, but because of various real-world constraints using only some of them is justified. We present four methods for selecting a given number of feature points and

compare them experimentally. With the test object that was used random selection gave good results, and when the actual computational costs of the methods are taken into account, this approach becomes very attractive (see Section 2.3.3. for details).

5.18. Enrico Puppo and Larry S. Davis, "Surface Fitting of Range Images using a Directional Approach." CAR-TR-504, CS-TR-2477, May 1990.

ABSTRACT: Range images can be represented as piecewise-smooth $2\frac{1}{2}$D surfaces. The segmentation of a range image requires low-level processing which filters the noise and estimates the surface normal at every pixel. Robust algorithms are needed which are able to perform the above tasks while preserving the image structure (surface discontinuities). Algorithms based on robust statistics often suffer from high computational complexity and low efficiency in the presence of high-variance noise. In the paper we propose a new method for processing images of polyhedra. The method is based on the application of a one-dimensional algorithm to slices of the image taken along several directions. Results obtained through the one-dimensional processing are subsequently integrated to produce two-dimensional results. The algorithm exhibits low complexity, high efficiency and high parallelism. Moreover, the method can be adapted to any kind of noise by modular substitution of the basic one-dimensional algorithm.

References

[1] H. Akima, A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points, *ACM Trans. Mathematical Software*, **4**, 148, 1978.

[2] J.G. Harris, A new approach to surface reconstruction: the coupled depth/slope model, First International Conference on Computer Vision, 277–283, 1987.

[3] R.J. Sampson, Surface II Graphics System, Series on Spatial Analysis, Kansas Geological Survey, 1984.

[4] A. Waxman, J. LeMoigne, L. Davis, B. Srinivasan, T. Kushner, E. Liang and T. Siddalingaiah, A visual navigation system for autonomous land vehicles, *IEEE Transactions on Robotics and Automation*, **2**, 124–142, 1987.

[5] L. Davis, T. Kushner, J. LeMoigne and A. Waxman, Road boundary detection for autonomous vehicle navigation, *Optical Engineering*, **25**, 409–414, 1986.

[6] S. Dickinson and L. Davis, An expert vision system for autonomous land vehicle road following, Computer Vision and Pattern Recognition Conference, 826–831, 1988.

[7] J. Crowley, Coordination of action and perception in a surveillance robot, International Joint Conference on Artificial Intelligence, 793–796, 1987.

[8] E. Dickmanns and V. Graefe, Dynamic monocular machine vision, *Machine Vision Applications*, **1**, 223–240, 1988.

[9] E. Dickmanns and V. Graefe, Applications of dynamic monocular machine vision, *Machine Vision Applications*, **1**, 241–261, 1988.

[10] K. Batcher, Design of a massively parallel processor, *IEEE Transactions on Computers*, **31**, 377-384, 1982.

[11] S. Reddeway, DAP—a distributed array processor, First Annual Symposium on Computer Architecture, 61–65, 1973.

[12] D. Hillis, *The Connection Machine* MIT Press, Cambridge, MA.

[13] J. Little, G. Blelloch and T. Cass, Algorithmic techniques for computer vision on fine grained parallel machines, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, 244–257, 1989.

[14] T. Bestul and L. Davis, On computing histograms of images in $\log n$ time using fat pyramids, University of Maryland Center for Automation Research Technical Report 271, 1987.

[15] A. Rushton, *Reconfigurable Processor Array: A Bit-Sliced Parallel Computer*, MIT Press, Cambridge MA, 1989.

[16] S. Ziavras and L. Davis, Fast addition on the fat pyramid and its simulation on the connection machine, University of Maryland Center for Automation Technical Report 383, 1988.

[17] L. Davis and P. Narayanan, Efficient multiresolution image processing on hypercube connected SIMD machines, University of Maryland Center for Automation Research Technical Report 430, 1989.

[18] S. Linnainmaa, D. Harwood and L. Davis, Pose determination of a three-dimensional object using triangle pairs, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10**, 634–647, 1988.

[19] D. DeMenthon and L. Davis, New exact and approximate solutions of the three-point perspective problem, First European Conference on Computer Vision, 1990.

[20] Kari Pehkonen, David Harwood and Larry S. Davis, "Parallel Calculation of 3-D Pose of a Known Object in a Single View." CAR-TR-502, CS-TR-2459, April 1990.

[21] L. Mathies and S. Shafer, Error modeling in stereo navigation, *IEEE Transactions on Robotics and Automation*, **3**, 239–248, 1987.

[22] J. Wu, R. Rink, E. Caelli, and V. Gourishankar, Recovery of the 3-D location and motion of a rigid object through camera images (an extended Kalman filter), *International Journal of Computer Vision*, **3**, 373–394, 1989.

[23] O. Silvén, "Estimating the Pose and Motion of a Known Object for Real-Time Robotic Tracking." CAR-TR-503, CS-TR-2461, April 1990.

[24] J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141–152, 1985.

[25] G.V. Wilson and G.S. Pawley, On the stability of the Travelling Salesman Problem algorithm of Hopfield and Tank. *Biological Cybernetics*, **57**, 63–70, 1988.

[26] B. Kamgar-Parsi, J.A. Gualtieri, J.E. Devaney, and B. Kamgar-Parsi, Clustering with neural networks, University of Maryland Center for Automation Research Technical Report 417, 1989. Also in Proceedings of the Second Symposium on Massively Parallel Scientific Computation, 31–38, 1988.

[27] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, 1989.

[28] W. Leler, *Constraint Programming Languages*, Addison-Wesley, 1988.

[29] S. Fahlman, Design sketch of a million element NETL machine, National Conference on Artificial Intelligence, 249–252, 1980.

[30] M.R. Quillian, Semantic Memory, in M. Minsky (Ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968.

[31] D.S. Touretzky, *The Mathematics of Inheritance Systems*, Pitman, London, 1986; Morgan Kaufmann, Los Altos, CA, 1986.

[32] Daisy System Corp., *Advanced Simulation*, Students' Workbook, 1986.

[33] K. Kanatani, Constraints on length and angle, *Computer Vision, Graphics and Image Processing*, **41**, 28–42, 1988.