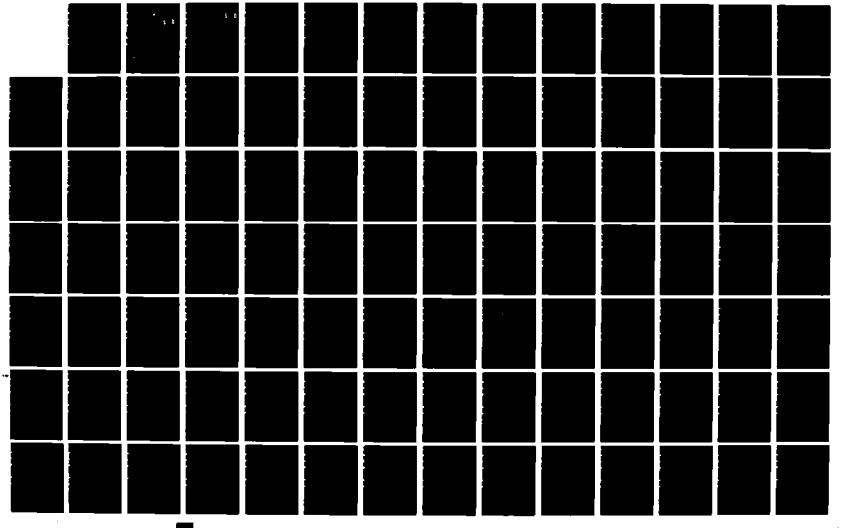
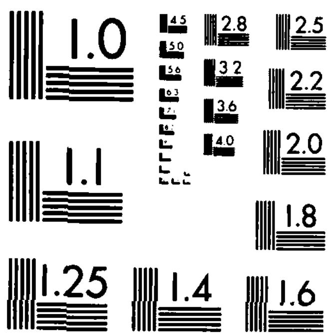
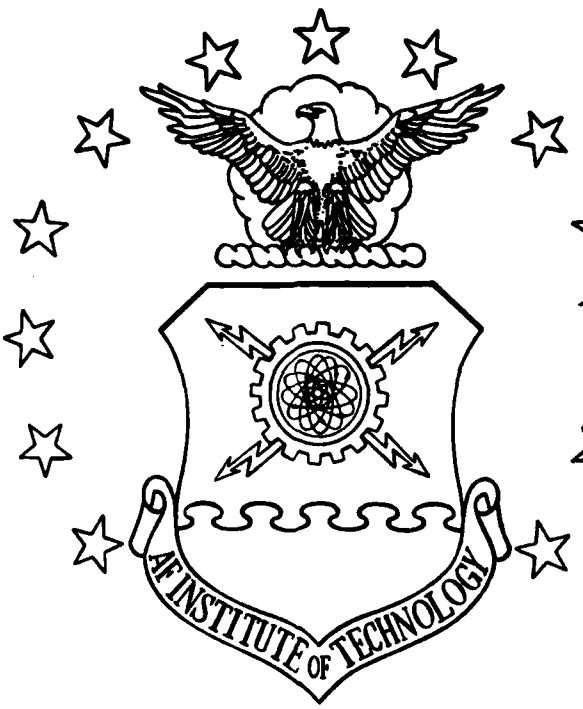
AD-A164 202 A ROBOT VISION SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J R HOLTEN DEC 85 AFIT/DS/ENG/85D-1 1/3

UNCLASSIFIED F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART
)ARDS-1963-A



A ROBOT VISION SYSTEM

DISSERTATION

AFIT/DS/ENG/85D-1 James R. Holten III
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENG/85D-1

A ROBOT VISION SYSTEM

DISSERTATION

AFIT/DS/ENG/85D-1 James R. Holten III
Capt USAF

AFIT/DS/ENG/85D-1

A ROBOT VISION SYSTEM

DISSERTATION

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

James R. Holten III, B.S., B.S., M.S.

Captain, USAF

December 1985

AFIT/DS/ENG/85D-1

A ROBOT VISION SYSTEM

James R. Holten III, B.S., B.S., M.S
Captain, USAF

Approved:

	25 Nov 85
	25 NOV 85
	25 Nov 85
	25 Nov 85

Accepted:

25 Nov 85
Dean, School of Engineering

Preface

This work was motivated by the widespread need for computer vision systems. Interest in autonomous vehicles by the Defense Advanced Research Projects Agency and many private concerns spurred my interest.

This vision approach avoids active illumination of the viewed scene. Through passive stereo television images, the system attempts to model the environment. The system works, but demonstrates the inherent shortcomings of systems based on this approach.

Thanks are due to my committee chairman, Dr. Matthew Kabrisky, committee members, Dr. Dennis W. Quinn, Capt. Steven K. Rogers, and Dr. John Jones for their support for the research, and in the preparation of this report. Thanks also to the numerous individuals who, through many conversations, acted as my sources of inspiration, especially Dr. Kabrisky; my wife, Mona; Capt. Robert Russel; and Capt. Ric Routh.

I would especially like to thank my wife and six children for their tolerance, patience, and support during the writing of this thesis.

James R. Holten III

Table of Contents

List of Figures

List of Tables

Abstract

Techniques are outlined for implementing each aspect of a comprehensive stereo computer vision system. The implemented system uses only ambient light scene illumination, and generates a model of the features in the scene and their 3-dimensional space location. A new, fast, production rule-based method for extracting low-level image features, the Queen Victoria algorithm, is presented. It forms the foundation of the environmental model generation, upon which suggested techniques can add further capabilities. Methods for deriving first and second order statistics of 3-dimensional feature location in the internal model are given. Also a technique of camera position estimation from feature matches between images and the model is given. Stereo disparity is shown to give useful results at relatively short ranges, such as those used in computer aided manufacturing. For Air Force flight line robots or other autonomous vehicles, modelling from a single camera in motion is suggested, using multiple cameras for expanding the field of view and improving the self-testing capability of the system.

I. Introduction

This project includes the design and implementation of a vision-based goal achievement system. The system relies only on ambient light, and through the use of a passive vision system, allows an autonomous roving vehicle to traverse an environment to a goal. The project includes visual processing of stereo optical image sequences into a 3-dimensional model, and the generation of a path through the model. The system outputs the sequence of robot commands necessary to traverse the path, modifying the path as needed during the traversal to compensate for vehicle and environment variations from the model. A unit designed with these characteristics can act as the heart of a general purpose autonomous vehicle transport platform.

This research effort achieved the implementation and evaluation of a stereo vision system, as well as outlining the modelling requirements, and proposing modelling and path-finding algorithms to support the full project. The result is a foundation for further research to develop the autonomous vehicle transport platform. This stereo vision can also be used for modelling the work area at a remote goal location for the platform, so that robot arms or other devices can be visually monitored and controlled to perform remote tasks. The platform based vision system leads to many useful capabilities.

Such an autonomous vehicle transport platform may carry

various payloads. The autonomous vehicle could be a weapons system base, a supply vehicle for hazardous environments, or, by carrying specialized robot arms, the basis for future hazardous environment maintenance robots. The vehicle size, ignoring the payload, can range from just large enough to carry itself, to a size nearing aircraft carrier proportions. It is not unreasonable to consider submarines, spacecraft, aircraft, or automobiles relying on such a system and operated in this way. The uses for such systems are many and varied, and could save lives and money in the future. However, such a vehicle is still barred from use by several technological barriers.

Some of the technological barriers which make current attempts at autonomous vehicles unuseable in most real world applications are as follows: (1) timing: traversing, for instance, a twenty meter course in hours is too slow for practical use; (2) reliability: failure of strategic parts may disable an entire vehicle; (3) robustness: it may be too easy to 'lose' such a vehicle if it is too far from its intended path or the lighting changes abruptly; and (4) accuracy: shooting the wrong target can be disastrous. Each of these barriers can be significant, but they are all cumulative properties of the needed sequence of operations. Each operation itself is plagued by all four barriers.

The sequence of operations required in the visual guidance function of autonomous roving vehicle system can be

broken down into smaller recognized problems which are commonly discussed in current literature:

1. Image registration --locating the same points in multiple images,
2. Building a 3-dimensional model from a set of 2-dimensional images,
3. Internal environmental model representation,
4. Goal identification in the internal model,
5. Goal achievement path searching, and
6. Robot motion command sequence generation.

Each of these problem areas has had extensive study, and to achieve a useable autonomous roving vehicle requires extending and combining all of these research areas.

Once these elementary level problems are solved and combined there are two important additional problems. The autonomous system must be able to not only avoid obstacles, but occasionally recognize them and perform sequences of operations to move the obstacles out of their path. Also, the autonomous vehicle must be able to track objects in the environment which are in motion, and then avoid or intersect the objects, depending on the vehicle's goal or mission. Neither of these are trivial problems, nor can they be solved before the lower level problems listed above have adequate solutions.

Background

The autonomous roving vehicle sub-problems are all current issues in pattern recognition, artificial intelligence, and robotics. Several sub-problems have been solved in each area, with several solutions each, with each solution having its own good and bad aspects. To make the overall system practical, however, the subproblems must not only be solved, but the solutions must all work together. An internal model which works well for goal achievement path finding may not necessarily be easy to build from 2-dimensional stereo images. Such interfaces between subproblems must have efficient cooperative solutions before a practical working system can be built.

Image Registration. The image pair registration problem, as applied here, involves finding subregions of interest in one image, then identifying where each subregion occurs in the other image of the pair. The pair may be successive images from one camera as the vehicle moves, or they may be simultaneous images from separate cameras, forming stereo pairs. Methods for finding the subregions of interest vary substantially, but finding their matches in the second image is most often done by point by point correlation (YAKI 78), (MORA 83), or by feature combination extraction and pattern matching (MEDI 83), (MESS 83) (BAKE 82). Both techniques result in translational mappings of points of interest in one image to their corresponding

location in the second image. Usually region shapes are distorted by occlusion of objects and changes of perspective. If the two images are taken from the same location with only a small offset and viewing the same environment, then there usually is a single mapping, and the entire scene is copied with a fixed translation, with a possible rotation, but very little other distortion. However, the offset of stereo images and changes over time both can result in disparities, and each subregion will have a slightly different mapping from one image to the other. These disparities in stereo pairs can be used to compute distances to the features in the images. Disparities in time sequence images can be used to compute movement and location.

Three Dimensional Models. Using the row-column locations of features in an image plane and a distance value derived from stereo image disparity, a 3-dimensional model of visible surfaces can be generated. Since only the surfaces visible from the angle of the viewer are locatable, the 3-dimensional model must have an 'implied object' storage capability. This is done by storing a map in 2-dimensions , and placing circles (MORA 83) or other polyhedral 'objects' (MONA 84), (LOZA 79) in the map to represent obstacles. These representations are only 2-dimensional, and are inadequate for real terrain, space, or undersea environments. These environments require at

least a 3-dimensional model, with the possible need for a time dimension to model object or vehicle movement.

Several graphics display applications use a 3-dimensional model to represent an object or scene. The model is then projected onto a 2-dimensional plane to obtain a 'view' of the model. These methods use elementary volume elements, called voxels, which are polyhedral solids, to represent the surface regions of the objects (SRIH 82) in the model. How the model is stored greatly affects the time required to build and access its parts. ADAM (HOLT 82), CARTAM (PETE 77), and hyperoctrees (YAU 83) are data structures designed for just such multidimensional models.

If a model is built entirely from externally collected visual data, then how is a goal represented? There is the classic (unsolved) pattern recognition problem of giving the system a model of the object to be found, then letting the system search its environment for a matching object (SRIH 82). Also, there is the idea of putting a map into the system, letting the system find its location on the map, locating the marked goal on the map, then incorporating the needed information into its model. In both cases there must be 'features' represented in the model, allowing the system to register the map model or the input images to the stored model.

Once the internal model is built up enough to indicate the relative locations of the autonomous roving vehicle

system and its goal, then the task of planning a path for the movement from the current location to the desired location may be started. There are several applicable numerical search strategies for finding paths, but their time complexities depend on the model structural complexities and other task characteristics (WINS 77), (DIXO 72).

Path planning can also become either a formal language or graph traversal problem once the model is translated to an applicable form (WINS 77), (NILS 80), (FU 82), (MONA 84), (LOZA 79). Once the path through the model is generated, the path must be converted into commands for robot motion, then given to the robot motion control elements for execution.

There are a number of robotic vehicles which move under microcomputer control. Among these are the HERO-1 by Heathkit, the Carnegie Mellon (CMU) Rover, the Stanford Cart, the Honeywell Corporation's golf cart, and the Martin Marrietta's road following vehicle. The movement of the robot vehicles may be resolved down to a sequence of commands in a robot command language, and thus the autonomous roving vehicle control system need only generate the proper sequence of these commands for the particular vehicle, but the proper command sequence depends first on finding a navigable path.

Generating a sequence of commands and executing them is

a simple task, but not necessarily a reliable one. Better sensor inputs are needed to reduce errors in positioning and in execution of movement commands, therefore vision is necessary. As the command sequence is being performed occasional 'checks' of movement accuracy and model accuracy, as well as updates of previously hidden environmental regions must be made. If picture processing times are too great, then the robot must stop, get new 'pictures', update the motion path, then begin traversing the new path.

The Stanford Cart and the CMU Rover (MORA 83) both implement the full sequence of operations. They both used algorithms designed by Hans Moravec, and thus use similar techniques. Image registration is done by first locating "high interest" areas in images as defined by the algorithm and then doing local statistical correlation of windowed subregions. No 3-dimensional model is built. A 2-dimensional map, useful only for traversing flat surfaces, is built. The map uses circular regions representing and bounding each obstacle. The Stanford Cart's goal is input as a set of coordinates relative to the position of the vehicle, so initial and propagation model accuracies become all important in achieving the final goal position. The goal achievement path is calculated assuming the vehicle has a circular boundary, and thus the calculations are fairly easy. The algorithms need only guarantee the vehicle's circle never intersects an obstacle's circle. Cart motor

and steering commands were then generated to allow 0.75 meter movements between observations. The internal model of cart motion in the computer was not accurate, requiring frequent course and position updates. The performance of the cart was slow, about 15 minutes for each 0.75 meter movement, or about 3 to 5 meters an hour. For their 20 meter courses it often took over 5 hours, and was not always successful. The CMU Rover requires only about a minute of picture processing time before making a one meter movement, but this is still too slow for acceptable vehicle speeds.

Yakimovsky and Cunningham (YAKI 78) built a vehicle for the Jet Propulsion Laboratory called the Robotics Research Vehicle. It used stereo vision to measure distances for manipulating objects with an arm mounted on the vehicle. It used image correlation techniques similar to those later used by Moravec. These projects by Moravec, Yakimovsky, and Cunningham seem to be the state of the art in autonomous vehicle passive light vision systems. They are not adequate for general use because they have the following drawbacks: (1) they are far too slow as currently implemented, (2) they are not reliable either in hardware performance or reaction to environmental complications, (3) they have little or no robustness and get easily 'lost' even within their own defined environment, and (4) their accuracy is questionable because even though they judge distances between objects well, they can not estimate the position of the vehicle

using exclusively visually acquired information.

Problem Statement

A system is to be defined for a generalized autonomous robot vision system, and then within that framework several recognized problems must be solved. The process framework must be coherent, defining sequences and sets of operations necessary to make the robot vision system work, and to allow future improvements. The system should include provisions for developing robust self-correction in case of temporary disruption.

Specific techniques must be specified for each of the several processes required, including

1. Image Registration,
2. Building 3-dimensional models from images,
3. Model manipulation and Update,
4. Goal identification, and
5. Goal path and trajectory planning.

In specifying techniques for solving these, special consideration must be given to

1. Real time processing needs,
2. Reliability,
3. Robustness, and
4. Accuracy.

Scope

This effort restricts itself to defining approaches for

all 5 categories given above, and solving the first three,

1. Image registration,
2. Building 3-dimensional models from images, and
3. Model manipulation and update.

Methods are suggested for possible solutions to goal identification and goal path and trajectory planning.

Special consideration will be given to reliability, robustness, and accuracy, while it will be assumed that given the proper hardware architectures, any working algorithm can be satisfactorily implemented for reasonable response time. This does imply that eventually the emphasis will be on designing a suitable architecture, but not within the scope of this project

Approach

Many approaches to image analysis concentrate on sub-pixel location accuracy for features. This will be considered metric accuracy, and is assumed to be achievable through algorithm refinement. The more significant problem, which is the main concern here, will be call symbolic accuracy, or the matching of correct symbols, or features in the images.

Methods of high symbolic accuracy are derived, and measurement errors are characterized and propagated. This allows high metric accuracy to be achieved through multiple-look observation updates.

A new technique of image feature extraction is derived

and applied. Multiple characteristics of each feature are extracted, and using these multiple characteristics to reduce ambiguities, high symbolic accuracy is achieved. Using the high-confidence feature matches, feature 3-dimensional location estimates and error characterizations allow robust models to be generated. Keeping a symbolic model of points, lines, and surfaces lets later matches to be made to these same features, which enables multiple-look position updates to be computed.

Methods of goal location using feature matching can then be used, so this is not considered further, and only the path and trajectory computations needed to be described for later implementation.

II. Image Feature Extraction Implementation

To create a three-dimensional model requires extracting features from a two-dimensional image which can be (1) unambiguously matched to the corresponding features in the second image of the stereo pair, and (2) can be reduced to a set of corresponding points or pixels in each image and thus to a set of points in three-dimensional space. To do this a scheme for finding and characterizing regions within an image was devised, and the resulting region representations are reduceable to boundary lines and vertices. Thus, the regions, rich in uniquely matchable characteristics, can be unambiguously paired as much as is practical, then the corresponding components of the region boundaries can be unambiguously matched and located in three dimensional space. Of course the spatial patterning of a surface or a pattern of identical objects can still lead to ambiguities, but this can also be true for confusing human vision, and will not be considered a serious problem at this time. Other means must be used to resolve ambiguities of this type.

The Queen Victoria Algorithm was developed as a way to characterize smooth and changing grey level intervals in a single line of pixels across an image. Since single lines of pixels alone are totally inadequate for feature matching, a group of lines organized into the 'Star Filter' are extracted from the image, and to each is applied the Queen

Victoria Algorithm. The result is that a region containing the Star Filter's center can be characterized by size, while other regions large enough to be intersected by the radial arms are identified and located. Each identified region can then have the Star Filter recentered within it to find its horizontal and vertical extent. Once these regions are characterized by horizontal and vertical extents an estimate of a center can be derived, then the regions can be mapped radially or rectangularly to a bit map characterization of their shape. Bit maps are more fully discussed in Appendix B.

The relative positions and characterizations of the subregions in the image are essential for subregion matching in registering sequences of images, and useful in stereo image registering. However, it all depends on using a robust low level feature extractor and region classifier. Thus, the Queen Victoria algorithm is the method chosen here.

The Queen Victoria Algorithm

The image feature extraction is done using various applications of the Queen Victoria Algorithm. The Queen Victoria Algorithm is a set of heuristically derived formal production rules which reduce a single linear array of pixel grey level values to a sequence of elementary image features. When applied to a single 'slice', or a single horizontal line of pixels across a picture it derives the

locations of edge intervals and smooth intervals, as well as grey levels of the intervals. This feature sequence has been used for image reconstruction for 'cartooned' images (HOLT 85), thus allowing low bit rate image transmission, and is currently being applied to geometrical feature extraction for locating and characterizing large regions within an image.

Background. The Queen Victoria Algorithm (HOLT 85) received its name due to its similarity to the heuristically derived rules used for restoration, modernization, and synchronization of a movie and a recorded cylinder (KABR 85). The two were coincidentally recorded during a dedication ceremony where Queen Victoria presided. A museum official, some years later, brought the two together and a project was born to add the sound track to a modern reconstruction of the movie. Nonlinear and non-causal heuristic filtering techniques were used to overcome poor recording and image quality, as well as to synchronize the two. The result was a modern movie with sound track of a famous individual who died before talking movies existed. This algorithm also uses heuristically derived rules, and attempts to recreate the main information content out of the noisy representation of the real world.

Production Rules. The differences between adjacent pixel greylevel values in a linear array are computed to

TABLE I

Mapping Slice Variations to Symbols

Category	Symbol	Meaning
$d_i < -T$	e-	negative going edge
$-T \leq d_i < 0$	g-	negative gradient, possible edge
$0 = d_i$	s	smooth
$0 < d_i \leq T$	g+	positive gradient, possible edge
$T < d_i$	e+	positive going edge

$d_i = x_{i+1} - x_i$ the difference of successive pixels

T the noise threshold

create a measure of edgeness, and then are compared to a noise threshold for conversion to symbols. Other measures of edgeness could have been used here, but this simple form helps test the robustness of the Queen Victoria algorithm. Five comparison categories of the edgeness measure value are converted into five initial symbols as in Table I. Fifty production rules are then applied to the string of symbols until only terminal symbols remain in the string. The resulting string of terminal symbols represents the sequence of edge interval features and smooth interval features in the linear array. An accounting of the locations and grey

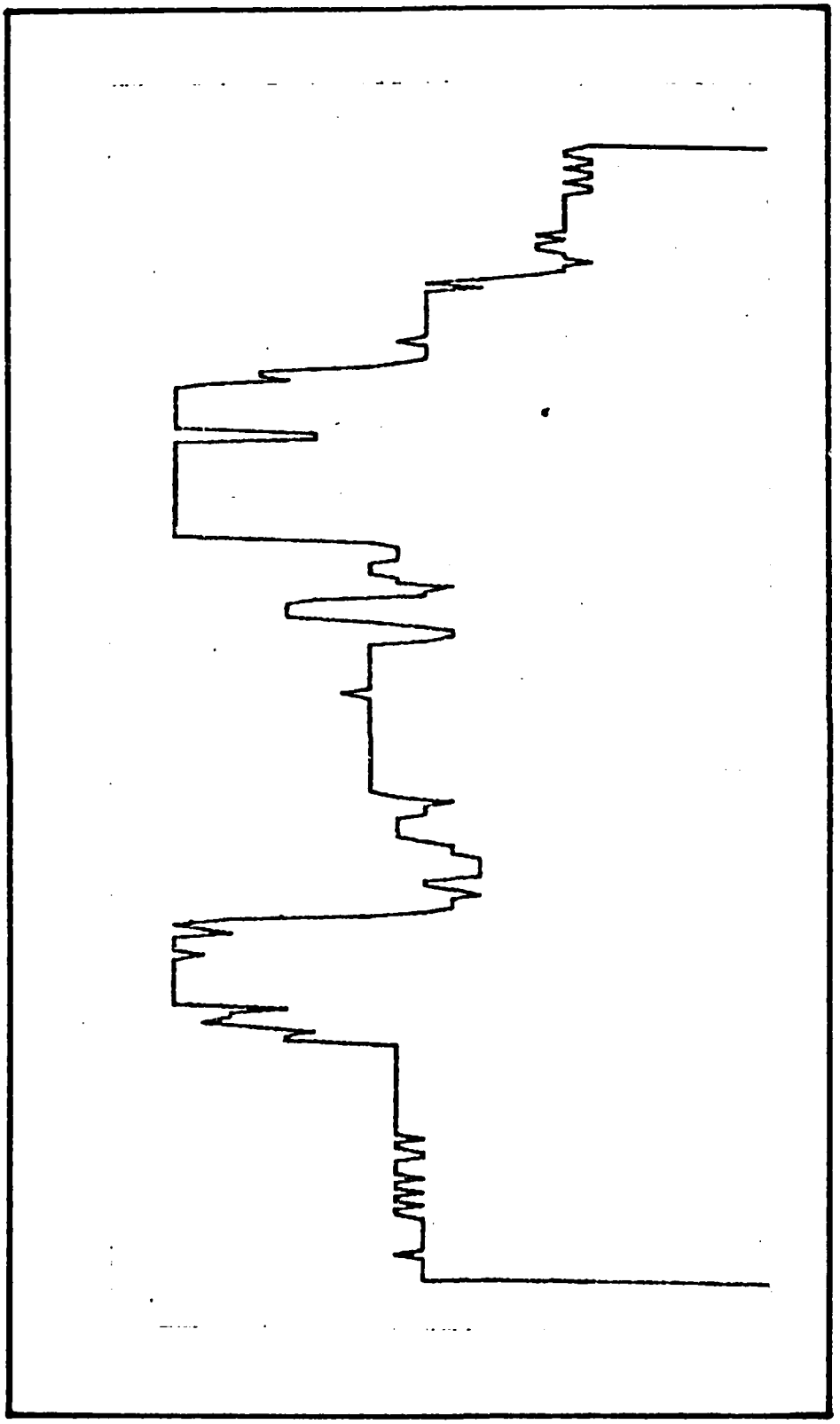


Fig. 2-1 Image Slice Data

Simple productions

-Smooth interval generation

αssβ → αSβ	
αG+sG-β → αG+SG-β	αG-sG+β → αG-SG+β

-Smooth interval continuation

αSsβ → αSβ	αsSβ → αSβ
αSSβ → αSβ	

-Edge interval continuation

αe-β → αG-β	αe+β → αG+β
αG-e-β → αG-β	αG+e+β → αG+β
αG-G-β → αG-β	αG+G+β → αG+β
αG-g-G-β → αG-β	αG+g+G+β → αG+β
αSg-G-β → αSG-β	αSg+G+β → αSG+β
αG-g-Sβ → αG-Sβ	αG+g+Sβ → αG+Sβ
αG-g-G+β → αG-G+β	αG+g+G-β → αG+G-β
αG+g-G+β → αG+G-β	αG-g+G+β → αG-G+β

Noise suppresion productions

-smoothing noise spikes

αsg-g+β → αSβ	αsg+g-β → αSβ
αg-g+sβ → αSβ	αg+g-sβ → αSβ
αSg-g+β → αSβ	αSg+g-β → αSβ
αg-g+Sβ → αSβ	αg+g-Sβ → αSβ
αG+g-Sβ → αG+Sβ	αG-g+Sβ → αG-Sβ
αSg-G+β → αSG+β	αSg+G-β → αSG-β
αG+g-G+β → αG+β	αG-g+G-β → αG-β

-ignoring gradual changes

αsg-sβ → αSβ	αsg+sβ → αSβ
αSg-Sβ → αSβ	αSg+Sβ → αSβ
αSg-sβ → αSβ	αSg+sβ → αSβ
αsg-Sβ → αSβ	αsg+Sβ → αSβ
αG+sG+β → αG+β	αG-sG-β → αG-β

After applying these productions, the resulting string of terminal symbols is a rapid access guide to feature positions and sizes in the original image slice. Using an edge enhancing reconstruction technique, the pixel data in Figure 2-2 was generated.

Usage. For each feature in the string of terminal symbols the location and grey level of each feature change location is recorded. Thus, for matching corresponding features in separate images along epipolar lines as in (BAKE 82) a small number of extracted features in the terminal symbol strings may be matched. These preliminary matchings greatly reduce the number of ambiguities which must be further resolved. These strings of terminal symbols may also be used to reconstruct a low-bit rate caricature of the original scene using various means of feature/edge generation.

The Star Filter Application

When horizontally positioned stereo cameras have colinear raster scan lines between the two images, as in Figure 2-3, the extension of the rasters through both images are called epipolar lines (BAKE 82). All disparity offset, due to the different camera view angles of the scene, lie along these lines, allowing the depth to be obtained from simple triangulation based on pixel locations along those horizontal lines. The problem is how to find the pixels which correspond to the same real scene features in the two images along the epipolar lines. Some of the techniques used in the past have been: convolution over a select window using a window of data from the opposite image (MORA 79), feature extraction and characterization along the epipolar lines using lateral inhibition windows (BAKE 82),

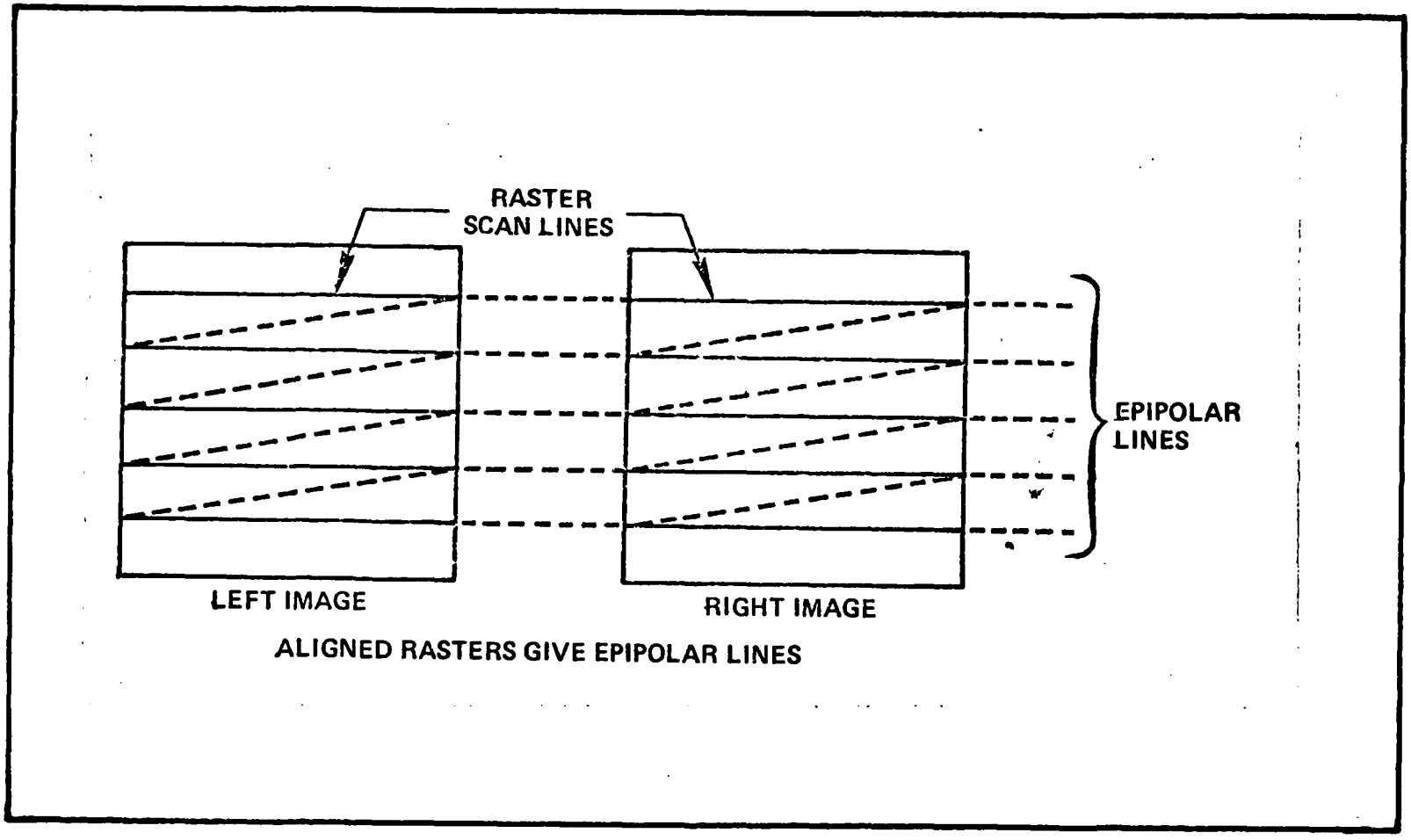


Fig. 2-3 Epipolar Lines

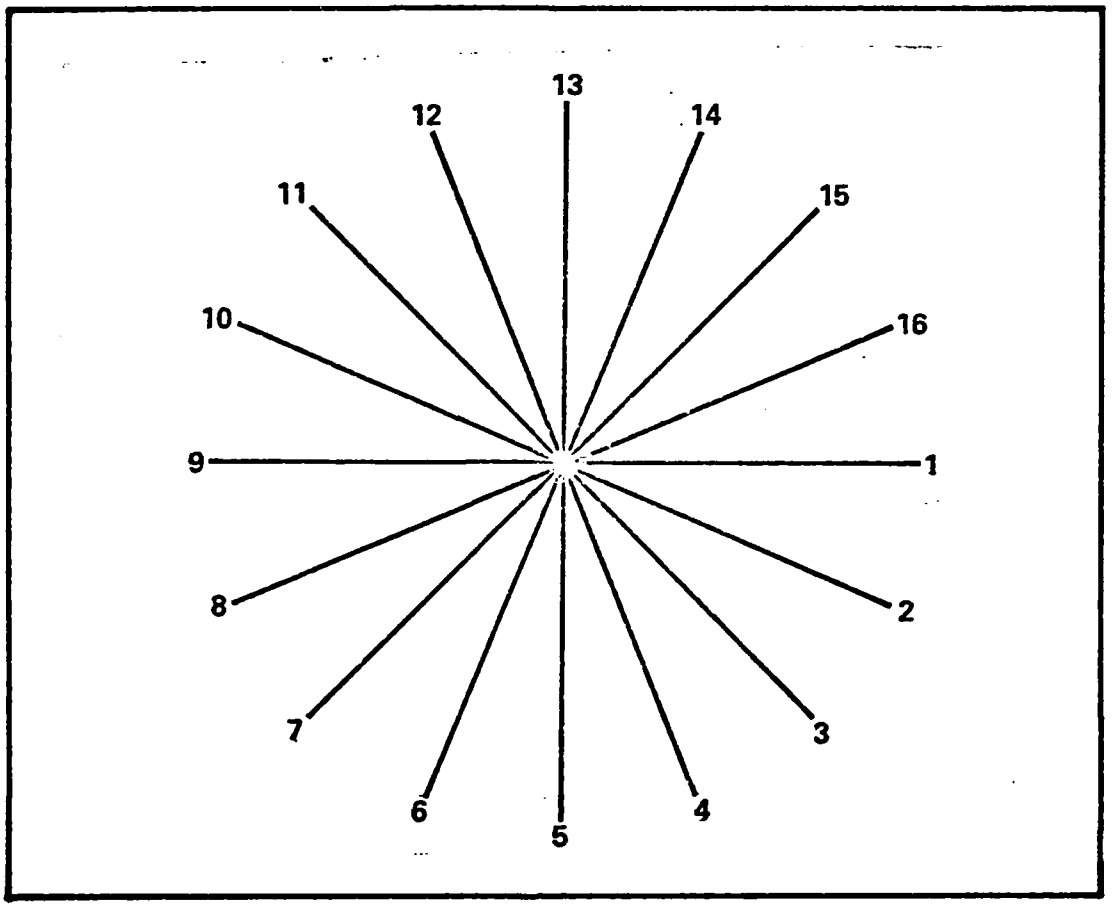


Fig. 2-4. The Star Pattern Filter

the image. This data is then used to resolve the ambiguities which otherwise occur in the matching process.

Only regions which overlap the same epipolar line need be considered as potential matches in a horizontal stereo pair. But after matching regions, the left and right edges of the regions must be matched to compute distance estimates to point locations in the 3-dimensional model of the environment.

Due to its order N (written O(N)) time complexity for an

NxN image, and its ability to 'focus its attention' on areas of particular interest, the star pattern application of the Queen Victoria Algorithm may be implemented on inexpensive microprocessors, allowing low volume distributed processors to handle the vision system in real time.

Image Region Location

How should the system decide which regions to characterize? Two techniques were considered. One technique attempted to characterize features in the image based solely on their size as found by applying the Star pattern. The other technique relied on slices along epipolar lines first, then applied the star pattern to each region found along the slices. First the Star pattern technique is discussed.

Since there is no apriori information about any new image, an arbitrary starting location for the Star Filter was chosen. Once the Star Filter is applied at one location a table of potential subregions is built. The Star Filter is then repositioned to the approximate center of the subregions, and when a satisfactory center is found, the subregion is characterized.

The Star Filter is initially centered in the image, and gives maximum coverage of the entire image for its initial list of subregions. This assures that the subregions directly in front of the camera are characterized. From this central position, the feature lists of the Queen

Victoria algorithm applied along each arm provide the twenty largest intersections with subregions. These are entered into the subregion list as intersections of potential subregions, and do not necessarily form a one-to-one mapping to the subregions in the image. Several arms may intersect the same subregion. Once potential subregions are identified and located, the next step is to characterize them, largest to smallest. To do this, first an approximation of the region center must be found.

To find the center of each subregion based on its intersection by a radial arm, the Star Filter is repositioned to the center of the radial arm interval which was recorded as passing through the subregion. This positioning of the Star Filter causes the subregions's extents left, right, up, and down to be updated, and a better estimate of region center is obtained. This subregion center and the corresponding left, right, top, and bottom extents are then used to characterize the subregion of interest.

The Star pattern alone tended to find the largest regions, but occasionally a long slender region would dominate the list of largest regions because it happened to lie along a radial arm for one of the repositionings of the Star. This hit-or-miss region selection prevented getting the same top 20 regions in the two images. To avoid this problem, slices along epipolar lines were used in the second

approach.

The feaures in the two images which lie along the slices making up an epipolar line must either match features from the opposite image's slice, or be occluded in the other image. If all the features in a slice are characterized, then there is little chance of skipping a significant feature that crosses the line of the slice.

Once a list of features along the epipolar slices is obtained, then these features are then each characterized using the Star pattern or a bit map. Since this includes all the features along the slice, then a higher percentage of valid symbolic feature matches can be guarenteed than in the first method given above. Also, this can be repeated for any number of epipolar positions, allowing a full scan of the image if it seems desireable.

Region Shape Characterization

As long as the camera axes remain horizontal, then subregions can be characterized and compared in successive images without considering rotation. For a scale invariant representation of shape, scaled regions can be mapped as an array of bits, and easily compared for potential matching and registration. If rotations must be considered also, then a size nomalized, centered, radially derived bit map of each surface will work well to find potential matches (GOSH 83). However, since rotation is not considered here, then only a size normalized rectangular bit map is used. Scale

factors for both the horizontal and vertical axes are stored, and the region limits are scaled to a 16 X 16 bit map array. Thus, by letting B be a set of bits, and each bit represent a group of pixels in a scaled rectangle including the subregion the result is a low resolution representation of a scale invariant block. Each bit of the bit map is then set if the corresponding block of the rectangle is 80% filled by the pixels satisfying the grey level criteria of the subregion of interest.

A measure of region shape match can now be generated by logical operation between their bit maps.

B_1 = map of subregion 1

B_2 = map of subregion 2

B_1 XOR B_2 = map of shape differences.

By counting the number of bits in (B_1 XOR B_2) a measure of goodness of match between the two regions is created. Bit maps are discussed more fully in Chapter III and Appendix B.

Summary

Using the Queen Victoria Algorithm gives a feature extractor for edge intervals and smooth intervals along a single line in the image. It is then applied along each radial arm of the Star Filter to characterize the extents of image subregions and to estimate locations of subregions. A table of subregion locations and extents is generated, and the Star Filter is recentered on each subregion to obtain a

best estimate of the region center and allow the subregion's shape to be characterized. The subregion's shape is then characterized by scaling and mapping into a rectangular bit array. Since the bit map is size normalized in both the horizontal and vertical orientations, some of the distortion due to aspect angle has been removed from the subregion matching process. These subregion position centers, scale factors, and shape bit maps are initial necessary steps in image registration and environmental modelling.

III. Three Dimensional Model Generation

The ability to extract and characterize features is necessary to remove the ambiguities normally arising in image registration. Once features can unambiguously be matched, then distance estimates and 3-D locations of the objects giving rise to those features can be estimated using the triangulation technique defined in Appendix A. As with any sensor, it is useful to characterize the error in these location estimates and allow the estimates to be improved later by additional observations. A data base model of characterized surfaces is used to store location and location error estimates, as well as a shape map and the grey level extents. If the modelled surface can be unambiguously matched to newly observed surface data, then these positions and error estimates can be repeatedly improved by later observations.

Feature Characterization and Matching

There are many useful ways of characterizing regions, such as area, height and width, mean grey value, maximum and minimum grey values, visual texture, location, orientation, and shape. All of these are useful, and some are necessary to ensure that only accurate matches occur. There are many ambiguous patterns in the natural and man-made environments, and allowing more ambiguity by using inadequate region characterizations is a sure way to cause ambiguous or

incorrect matches. Preliminary matches can be made between sets of regions or surfaces by using height and width, mean grey values, range of grey values, location, orientation, or surface texture measures, but the most discriminatory characteristic of regions and surfaces is their shape. Goshtasby showed that size normalized bit maps could be used for shape representation (GOSH 83), and that a measure of 'nearness' in shape came from counting the number of bit differences between the two shape representations. The bit map forms are discussed more fully in Appendix B.

Using an unscaled 16x16 bit map representing a rectangular region which includes the region or edge of interest, allows easy comparison of regions and region edges needing only horizontal translations for correspondence matching. Likewise, by centering on a region, a radial bit map can be created which allows rotations of the region to be treated as translations of the map. These two forms of bit maps suit the needs of stereo 2-dimensional region registration as well as 3-dimensional matching of surfaces.

Stereo Image Epipolar Lines. Putting Baker's concepts of epipolar lines (BAKE 82), into a more relevant form, only those regions which subtend common raster lines in the two images need be considered for matching. By eliminating all other regions, the number of comparisons is greatly reduced. Each comparison requires getting approximate matches in grey level, then approximate matches in shape. One property of

epipolar lines is that approximately horizontally positioned cameras cause all horizontal edge pairs to appear in nearly the same epipolar line. Rectangular bit map shape characterizations allow easy shape comparison for matches. Any shape difference between the two views of a single surface will be (1) foreshortening, (2) occlusion by an object or the image edge, or (3) mirror-like reflections. Mirror reflections will not be considered here. Occlusion will either block the entire view giving no match, or will give a partial match which may have no relevant edge shape to match. Partial occlusion may leave enough visible surface for a partial match of one or more edges of the occluded surface. Foreshortening is no problem, because its distortion is partially removed by normalizing the horizontal width to create the rectangular bit map.

Since the only relevant matches are those which give a single pixel position in the epipolar line, then vertical edges are the desired feature to match. By logically combining bit maps, left and right boundary vertical edges can be extracted for comparison.

B = bit map of region

V = (B XOR LSHIFT (B)) = bit map of vertical edges where LSHIFT (B) shifts B left one position.

R = (V AND B) = bit maps of right boundary vertical edges

L = (RSHIFT (V) AND B) = bit map of left boundary vertical edges where RSHIFT (V) shifts V right one position.

The R and L bit maps of matched regions can now be partially matched in spite of partial occlusions. Single edges can be matched to allow estimates of distance to each edge point in the bit map.

These matches are made at the resolution of the bit map, and if better matches are desired for location accuracy then the actual pixels of the regions to be matched should be compared. This can be done by building high resolution bit maps of an area around the edge of each region and matching them at different horizontal shifts until a best offset position is found. A low resolution match obtained using the 16 X 16 bit maps can be used for rough location estimation.

The 'goodness' of a region match becomes a comparison of grey levels, vertical region extents, and shapes. Once a region match is made then the coinciding edges can be matched for location in camera-relative 3-space coordinates.

Occasionally multiple 'good' valued matches will include incorrect matches, and thus still present ambiguities to be overcome. When multiple 'good' matches are found, then comparing the relative geometric position of regions giving ambiguous cases can help to resolve the ambiguities. Once ambiguities are resolved then the region ranges and

locations can be estimated.

Sequential Images and Flow Fields. Stereo camera inputs are static pairs of images, and can only result in static estimates of positions. If sequences of camera images are used, and regions can be matched in successive images, then relative motion between the camera pair and the environment can be estimated also.

Once such a camera-relative surface is located by the binocular disparity techniques above, its shape can be recharacterized using a radial bit map. The radial bit map is dependent on being centered on the region, but it converts rotations to translations and allows easy shape comparisons even when viewed from varying angles. For vertical surfaces always viewed from a floor-following robot, rectangular bit maps are adequate, but any surface which may be viewed from many angles, such as a pattern on the floor, or a vertical panel in space, the radial bit map is essential for easy comparison. Using these radial bit maps, surfaces characterized from successive camera pair positions may be matched to currently observed surfaces, giving the capability of estimating the camera's position relative to the environmental model. By combining the time of image acquisition with the relative motion, the relative velocity can be estimated also.

If the position of a surface in 3-space is known, then by estimating its position relative to the camera pair

location, an estimate of real camera pair position can be made. To match an observed surface to an existing modelled surface may be solved by using the given feature characterizations, thus partially solving the object recognition problem. By matching observed features to previously modelled features position errors can be updated for both the camera pair and the environmental model.

Images to 3-Dimensional Space

Identical stereo cameras horizontally positioned with parallel image axes require only matches along the epipolar lines to estimate range. The range estimate is then combined with the feature's vertical and horizontal offsets from the image center to get the vertical and horizontal distances of the located feature from the reference image axis at that range. By characterizing the error in estimating the feature stereo disparity and the vertical and horizontal offsets then an estimate of location error can be made.

When successive sets of located features can be matched to features in the environment then the camera pair position and orientation can be estimated. From this estimate, updated estimates of the environment feature locations can be generated, giving a progressively better model of the environment. Also, by estimating successive camera pair positions then the camera pair's, and perhaps the robot's, velocities can be estimated, giving sensor inputs in the

form of a position and velocity state vector and covariance estimate to augment a robot navigation system. These equations are derived in Appendix A, and are summarized here.

Feature Range, Offset, and Error Estimation. Estimating the range along the image axis is a simple matter of triangulation. It is derived in Appendix A, and the result is

$$d_p = d_s d_v / (d_L - d_R) \qquad (3\text{-}1)$$

where

d_p = feature range from the reference camera along the image axis.

d_s = the camera seperation

d_v = the distance from the image convergence point to the virtual image plane

d_L = the horizontal offset from the image axis in the left camera image

d_R = the horizontal offset from the image axis in the right camera image

Once the feature range is estimated the horizontal and vertical offsets from the reference camera are given by

$$d_x = d_L d_p / d_s \qquad (3\text{-}2)$$

and

$$d_y = d_D d_p / d_s \qquad (3\text{-}3)$$

where

d_x = feature horizontal offset from the reference camera image axis

d_y = feature vertical offset from the reference camera image axis

d_D = vertical pixel offset from image center

The granularity of the images leads to coarse distance estimate steps for small disparities, but becomes more refined steps for large disparities. These are shown in Table II.

Estimating the error is more difficult. Assuming the maximum horizontal feature location error is e_R in the right image and e_L in the left image, and the maximum vertical position error is e_D, then the 3-dimensional camera relative position errors are given by

$$|e_p| < \frac{d_s d_v (|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_L|)} \qquad (3\text{-}4)$$

$$|e_x| < \frac{(d_L+|e_L|)d_s(|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_L|)} + \frac{d_p|e_D|}{d_v} \qquad (3\text{-}5)$$

$$|e_y| \leq \frac{(d_D+|e_D|)d_s(|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_R|)} + \frac{d_p|e_D|}{d_v} \qquad (3\text{-}6)$$

TABLE II

Pixel Offset to Range Estimates

range estimates for d_v = 180 pixels, d_s = .75 feet*					
offset (pixels)	range (feet)	offset (pixels)	range (feet)	offset (pixels)	range (feet)
0	∞	5	27.00	20	6.750
1	135.0	10	13.50	40	3.375
2	67.5	15	9.00	60	2.250
3	45.0	20	6.75	80	1.688
4	33.8	25	5.40	100	1.350
5	27.0	30	4.50	120	1.125

* The separation units (feet, meters, etc.) are the units of the table entries.

where

e_p = the range error

e_x = the horizontal position error

e_y = the vertical position error

These error terms become meaningless if the disparity error is greater than the observed disparity. Thus points cannot be located accurately unless $(d_L-d_R)<|e_R|+|e_L|$ holds.

Since the estimated error bounds in the image are all one or greater due to picture granularity, then the multiplicative error terms grow rapidly for large image

TABLE III

Range Errors at Different Ranges

disparity for d_v = 180 pixels, d_s = .75 feet*							
actual	error = $\|e_L\|+\|e_R\|$						
(d_L-d_R)	1	2	3	4	5	6	7
2	67.5	---	---	---	---	---	---
3	22.5	90.0	---	---	---	---	---
4	11.3	33.7	101.3	---	---	---	---
5	6.8	18.0	40.5	108.0	---	---	---
6	4.5	11.2	22.5	45.0	112.5	---	---
7	3.2	7.7	14.5	25.7	48.2	115.7	---
8	2.4	5.6	10.1	16.9	28.1	50.6	118.1

* The separation units (feet, meters, etc.) are the units of the table entries.

errors as in Table III. Once these error bounds are converted to error volumes in space then they may be used for position updates only by assuming the location error is characterized by a Gaussian distribution. This may be a reasonable assumption due to the variations in the location qualities due to camera jitter, poor feature location estimates, and varying errors due to image location quantization.

Camera Position Estimation. By matching several features in an observed model whose coordinates are fixed relative the camera, to features in a 3-dimensional environmental model, a one-to-one correspondence can be set up. Obtaining a weighted least-squares best fit of the camera-relative model to the environmental model gives an estimate of the camera's position in the environmental model. Each feature's importance in the least-squares fit is weighted by the reciprocal of its covariance estimates. Low confidence gets less weight, high confidence gets more weight. This is given more fully in Appendix A.

The Symbolic Model

Two dimensional feature matching can be done only by extracting those features which lead to the least ambiguity. These features are the descriptions of the subregions found in the image. Each subregion can be approximated by a planar surface and its boundaries in 3-space, and recharacterized according to its actual shape rather than the perceived shape in the image. Once the actual shape is found its boundaries can be approximated by piecewise linear bounds. If the linear segments are chosen in a manner which prevents ambiguities and allows the corresponding segments to be chosen for similar surfaces, then the vertices can also be unambiguously matched. Once vertex matching is achieved then error volumes can be generated about the vertices, projected along the boundary lines, and extended

over the entire surface. By using a set of symbolic models for these surfaces, lines, and points, this type of environmental model can be created and updated.

The component features of the symbolic model, points, lines, and surfaces, are essential to storing and matching feature characteristics. And once the symbolic model parts are stored and characterized, the estimation of the camera position can be performed. Once the camera position is estimated, feature location estimated means can be improved via a static Kalman filter.

Stochastic Updates

Storing the feature characteristics is essential to matching the features unambiguously. Only after features are matched unambiguously, and the camera position determined, can concepts of position updates be used. An exisiting characterized feature must correspond exactly to a recently observed feature before the old and new can be merged for a better estimate of the feature's location.

Ignoring the error in camera position, which was minimized as given in Appendix A, a static Kalman filter estimator can be applied (MAYB 79) to stochastically improve the location estimate and error estimate for each feature which is matched. The symbolic accuracy of the matches must be exact for those features used in locating the camera, and for the feature location updates. However, as long as the position error can be approximated by a Gaussian

distribution noise function, and a set of covariances can be derived, the positions can be estimated from noisy data. The perfect symbolic match relies on keeping a table of symbolic features, and their descriptive characterizations. To apply the stochastic update requires storing each feature's mean 3-dimensional location and its covariance matrix.

The symbolic table, though necessary for feature matching and updating, is not easy to use for path finding. Since features are not readily accessed by location, then path finding in a symbolic model of surfaces is a combinatorial problem. To avoid this complexity a local volumetric model can be generated from the symbolic surface models for path finding.

Summary

Being able to characterize features so they can be unambiguously matched in the 2-D images is essential to building 3-D models and finding paths through the environment. If these same features can be characterized to the extent that they can be unambiguously matched in 3-D space then the additional capability of model retention and update is possible. To prepare for such future updates two model forms are used, the symbolic model and the volumetric model.

The symbolic model stores each extracted feature and the characteristics which allow it to be unambiguously matched

to new features. It also stores feature mean locations and error estimates. Those symbolic features which occupy the volumes of interest can be represented in a volumetric model as needed for path and trajectory finding.

IV. Path and Trajectory Generation

Using a symbolic model of point, line, and surface, or volume features and searching for paths through three dimensional space is essentially a combinatorial problem (MONA 84). Therefore it is of exponential order (AHO 74). Using a volumetric model can reduce the order of the search, even making it independent of the number of features stored. However, the conversion from symbolic features to the volumetric model is of linear order, because each of the features must be stored into the volume. This means the conversion and the path search, taken as a single path search algorithm starting with a symbolic feature set, is of linear order relative to the number of features stored. This is a far more tractable problem for a real-time system with no apriori knowledge of the number or types of features in the environment.

The volumetric model is regenerated each time the symbolic features are updated or the volume of interest changes. Those symbolic features which occupy the volumes of interest are represented in the volumetric model. The volumetric model is used only for path finding through the modelled environmental volumes.

From the volumetric model comes the set of potential paths, then by applying the models of vehicle motion and cross-section, paths can be eliminated until a workable path trajectory is found. The sequence of vehicle motions

necessary for traversing that trajectory is then stored as the motion command string for the vehicle.

The Volumetric Model

Using the octree, Ruff and Ahuja (Ruff 84) defined a method for finding paths through three dimensional space. The Associative Data Access Method (ADAM) (HOLT 82) is a variation on the octree and can be used similarly. In either there is a hierarchical representation of a space of interest. This space of interest is a cube, and each successively lower level in the octree represents subdivisions of the volume. The number of levels in the hierarchy is determined by the accuracy of representation desired in the model. Each level represents one binary digit of representation. The model need only represent those volumes in the space that are of current interest, and all other volumes are merely left out of the model.

The volumetric model is of interest here because it allows easy searching for connected empty volumes, and the complexity of the search can be adjusted as needed, depending on the accuracy needed to insure a large enough passage cross-section along the path.

Path and Trajectory Generation

The path finding technique of Ruff and Ahuja creates a volumetric model of the empty space, then finds a path through it. To find a path they apply the medial axis

transform to the model of space, obtaining a set of empty region axes, then they perform a connectedness search. Another approach would be to use the hierarchical volume structure to extend region growth to three dimensions. The search starts at the vehicle position and 'grows' empty space areas, giving priority to those voxels at the higher levels which (1) are nearer the goal, and (2) allow the vehicle cross section to pass freely along the volume toward the goal. Once a path is found which joins the current position and the goal, it is tested for minimum cross section to ensure the path can be traversed. Path optimization can then be used to round corners and skim near walls for a reduced-energy trajectory.

In such path optimizing, the system must use models of physically possible vehicle movements. These movements must then be strung together to give a physical vehicle trajectory. Once the movement string is known, then the commands for each of those movements can be generated and stored.

The path planner must retain the ability to eliminate potential paths any time the vehicle motion prevents the trajectory from following the path, or the path cross section does not allow vehicle passage. The restrictions on vehicle motion can be a significant block in developing a robust system. Things which are especially critical are vehicle turning radius, vehicle sharp turn footprints,

vehicle movement directions relative to its shape, and surface-following constraints. The vehicle's modes of movement can dictate the path, and accurate models are needed for useable vhicles in space, underwater, in rough terrain, or on flightlines.

Summary

The volumetric model is regenerated each time the symbolic features are updated or the volume of interest changes. It is used only for path finding through the modelled environmental volumes.

From the volumetric model come the potential paths, then by applying the models of vehicle motion and cross-section, paths are eliminated until a workable path trajectory is found. The sequence of vehicle motions necessary for traversing that trajectory is then stored as the motion command string for the vehicle.

V. Results

A new technique for combining low level features into symbolic forms was created, allowing simple production rules to extract feature characterizations. In addition to this, the groundwork for future development of a robot visual system was laid, with sufficient depth that the effort could be continued by others with a minimal background in many of the disciplines combined herein. Methods were reviewed, and a potential workable combination was presented and tested as much as time allowed. The mathematical derivation of one of the more difficult problems in robot vision is given in Appendix A. The camera location estimation, also known as the camera calibration problem, is converted to a simple equation for ease of implementation.

The use of the camera location equations depends entirely on having perfect symbolic accuracy for some identifiable set of features in the feature extraction and matching algorithms. Thus the major portion of the work was spent on continually improving the quality and variety of independent characterizations which could be applied to surfaces and used for matching them in both 2-dimensional images, and in the 3-dimensional model.

Image Feature Characterization and Registration

The Queen Victoria algorithm, a new technique for combining the low level image features symbolically into

higher level features, is presented. It is then applied to epipolar slices in two images, giving a region cross section for each region intersected by the line. The region cross sections are then used to center the Star Filter for region feature characterization, and to characterize the region edgesusing edge bit maps for accurate edge matching and location in 3-dimensional space. The combination then allows the easy application of either rectangular or radial bit map shape characterization. Taken together these procedures extract a hierarchy of feature characteristics.

Image Geometry. At the top of the hierarchy is the geometric relationship of regions in the images or surfaces in 3-dimensional space. By themselves, these relative positions of symbolic representations apply nicely to the graph representations of web grammars. The complexity of matching algorithms based on geometric position relationships alone is of exponential complexity, so to reduce this complexity the hierarchy of the region's appearance characteristics are matched first to reduce the number of potential matches for each web node.

Region Edge Shape and Grey Level. Since the Queen Victoria Algorithm separates smooth-appearing regions from grey level gradient interval regions, the smooth regions are then matched. However, using such a texture measure as the fractal texture approximation (PENT 84), regions of other

than smooth texture can be represented. Either way, once region extents can be defined, then the region shape can be size normalized and bit mapped for comparison. Using the rectangular bit map and grey level maximum and minimum only for matching criteria, the shape difference and grey level difference between camera views can cause the same region from different views to be mismatched. The cases of shape difference are

1. Occlusion by a foreground obstacle,
2. Occlusion by the image window,
3. Drastic angle of view change, and
4. The appearance of different shape due to reflections.

Also, the cases of grey level difference between cameras can be caused by

1. Different lens f-stop setting, and
2. Reflections on the surfaces.

Reflections are a special case which are not considered here. This greatly simplifies the algorithms, but must be considered in a later refinement of the algorithms. Shadows are not a problem in stereo views, because they actually add high contrast edges on surfaces and help to locate the surface. However, in sequences of images, or later views of previously modelled surfaces, moved shadows and other

lighting changes can make the task of surface matching more difficult.

Partial matches of partial edge shape can be adequate for feature location, and using bit maps the vertical edges of regions are matched. Thus, the next level of the hierarchy of characteristics is boundary shape and position.

Boundaries Into Lines and Points. By matching regions by grey level and partial edge shape, only the boundary lines which correspond in the region matches need be compared for more precise feature matching. These boundaries are easily extracted as horizontal top or bottom, and vertical left or right edges, using bit map shifts and logical operations. These extracted lines can then be matched for their shapes and offsets, giving precise location of the lines in 3-dimensional space, the desired goal of the hierarchy of matches.

Location Errors. Using the hierarchy of feature characteristics, the results progress from potential symbolic accuracy errors, the wrong features matched, to successively better metric accuracy. Once the correct regions are matched, a gross estimate of feature 3-dimensional location can be made, but by more accurately matching particular parts of the regions, such as lines and vertices, more accurate locations of those features may be found. Using edges extracted from non-scaled bit maps, a

high-accuracy estimate of edge and vertex location is computed, as well as its estimated accuracy. The feature point characterizations, locations, and location error estimates are then stored in the 3-dimensional symbolic model.

Three Dimensional Models and Model Updates

The features extracted and characterized should be surfaces, lines, and points. These are to be stored in tables, with a characteristic description of each including mean location estimate and location error estimate. For the surfaces the location should be the surface center for easy matching in the environmental model, and a unit normal vector estimate should be stored as well as a radial shape bit map, the radial scale factor, a list of boundary edge pointers, and a set of grey level extremes. These allow global matching of surfaces, then local matching of boundary lines.

For the boundaries a set of straight line approximations should be stored. Each straight line should be characterized by center point for location, a pointer to the surface it bounds, a pointer to the next line in the boundary, and a pointer to each end point. These allow local matching of the lines relative to the surface, then the even more local matching of the points relative to the lines.

The vertices are to be stored as a location, a

covariance matrix, and a pointer to each of the two boundary lines which meet there. This allows complete links of association between the levels of features associated with a surface. Vertices are the only features which can be located and characterized giving location and error updates using camera relative data, but surfaces are the features which can most easily be symbolically matched unambiguously. Thus the multilevel feature symbolic model is needed at both extremes, and the line models are useful for working between the two.

Continued work is needed to obtain symbolic matches which can be completely reliable to within a useable accuracy. Without the reliability of quality symbolic matching the robot will not be able to find landmarks, estimate its own position, or create a model of any useable quality. Without it the robot will work as if each view were independent of all that has gone before, and the visual sensors will not be much more useful than sonar. The robot would be able to detect and avoid obstacles as long as they were in its field of immediate view, but it would lose track of any obstacles which passed beyond its view.

The Final Implemented Process

In the final implementation two images were taken using the same camera from two different positions. This emulated the two views of a stereo pair and guaranteed identical camera parameters for the two images. The two images were

the static views of the laboratory environment given in Figure 5-1. These two views were both shrunk to half their normal size in each dimension so both could be resident in the OCTEK memory at one time, allowing rapid interaction with both images during the entire processing period.

A slice was taken across each image, and the feature intervals were extracted using the Queen Victoria algorithm. The feature intervals in the two slices were then matched by grey levels, then the resulting ambiguities were resolved using bit map characterizations of the edges of the feature intervals. A sample of two feature interval edge characterizations being compared is given in Figure 5-2.

Using these image feature edge locations the range, horizontal offset, and vertical offset and their errors were computed using equations 3-1 through 3-6.

Matches. The matches obtained using grey levels and edge shape were in two categories, those which were the correct features, and those which had no correct match and so were left with the best match available. Another category which undoubtedly will show up eventually, but was not in this data comes from the "picket fence" problem. Many identical features will be matched, and no proper position correspondence can be made without further information.

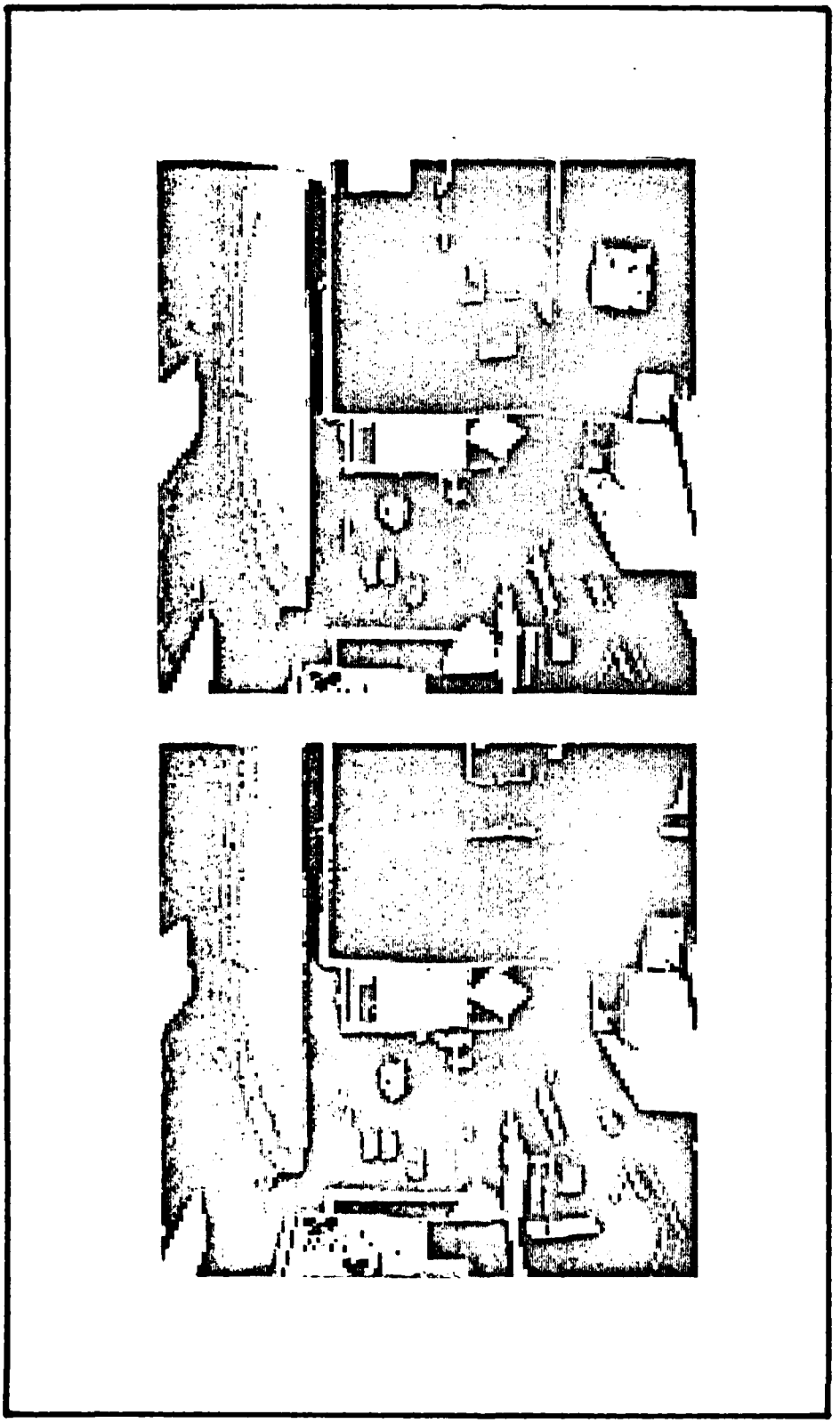


Fig. 5-1 Stereo Images

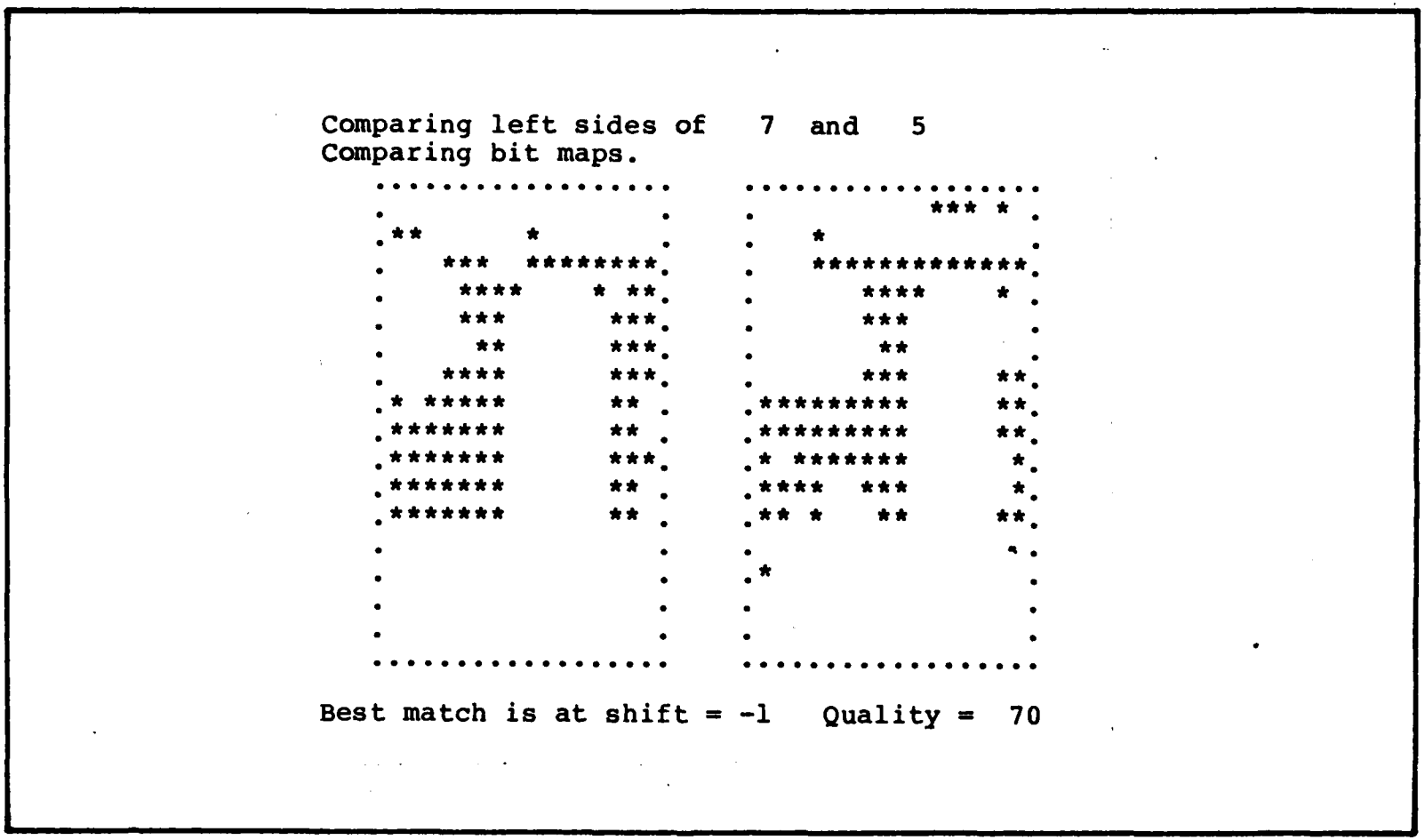


Fig. 5-2 Region Edge Bit Maps

The match categories found indicate a need for a method for automatically determining whether each is a good match, and which matches are the result of a feature which has no match. The "picket fence" problem can only be solved for identical components by matching the first or the last in the sequence, then using a 1-1 matching along the sequences. Even this is not possible if neither both of the first elements nor both of the last elements are visible in the two images.

The matches obtained were high quality symbolic matches and gave a good base for feature edge location estimation. Several sample outputs are listed in Appendix D.

Location Estimation. The pixel location error for feature edges was assumed to be half a pixel width. This gave a disparity error bound of one pixel. For these assumptions the range estimates were fairly accurate for ranges of 15 feet or less. However, from Tables II and III in Chapter 3 any location beyond 15 feet will be given an error bound which can bean the computed location is far from the actual location. This is due to the hardware limitations of pixel granularity and the separation between the camera positions.

As the granularity becomes more coarse the range volume corresponding to a pixel offset and its corresponding range and error becomes larger. Reducing the images by half in each dimension to save processing time caused the location

range error to be doubled at each range increment.

Camera separation can be increased, but increased separation increases the minimum viewing range, and causes greater viewing angle variation between the two cameras. The result is more distortion in the observed features, which can make the matching problem even more difficult.

The maximum abservable range with tolerable error limits can be extended by

1. Using a higher density pixel array (requires A/D converters and more image memory),
2. Using a higher power lens to put less image area into the same pixel array (requires new camera calibration at a different lens power), or
3. Increasing the camera separation.

Several examples of slice features located in 3-dimensional camera relative coordinates are given in Appendix D.

Processing Time. The processing time to get range estimates for the features along a single slice pair was about 10 minutes. This is not a measure of algorithm quality because the major delays were in communications between systems shown in Figure 5-3. The images resided in the memory of the OCTEK image processer, which was a peripheral device on the Data General NOVA minicomputer. Due to lack of memory space on the NOVA the major processing

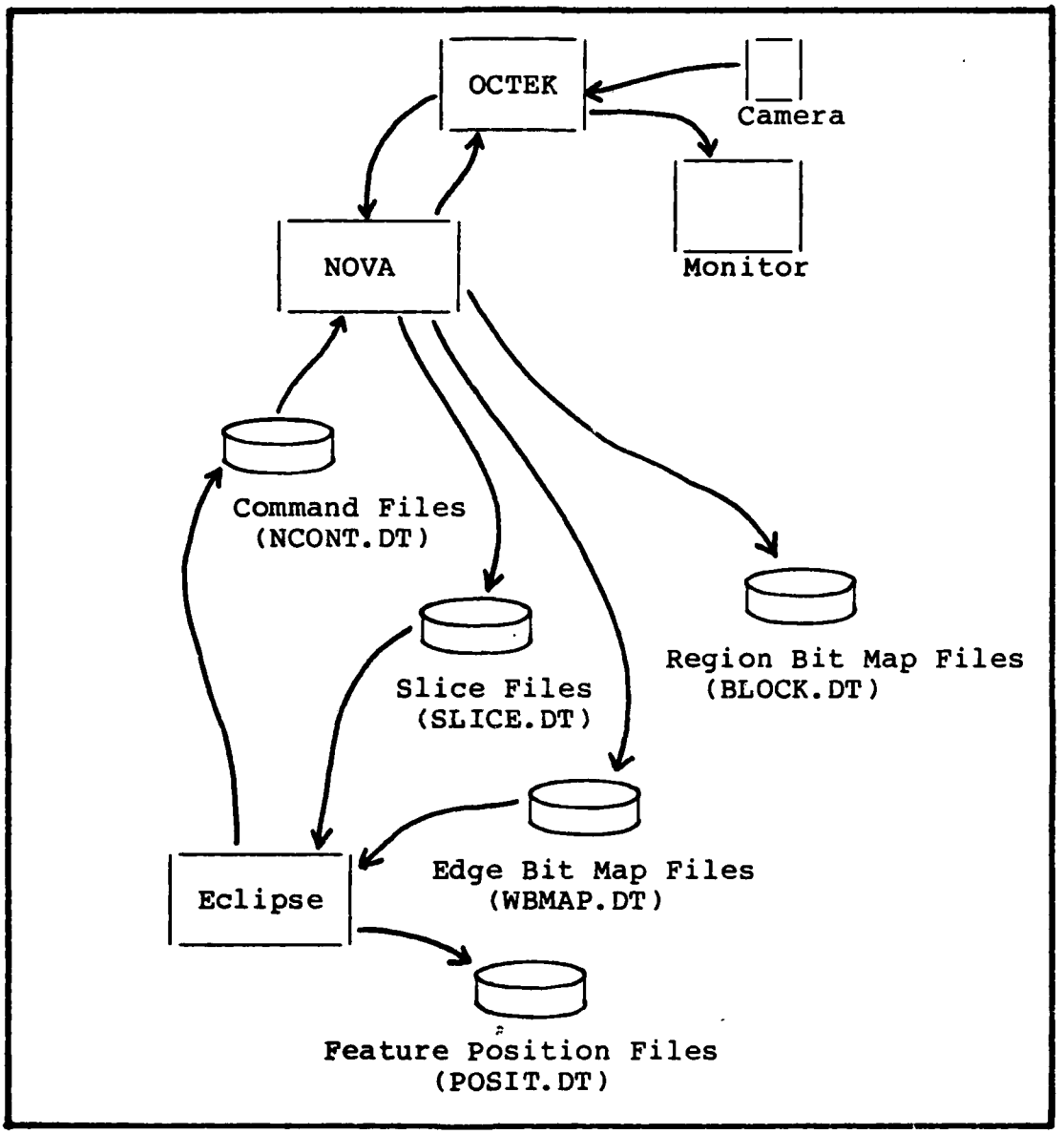


Fig. 5-3 Hardware Communications

was done on a Data General Eclipse, with the system passing all data via files on common disk drives. The major delays were in Eclipse/NOVA communications and NOVA/OCTEK communications.

The processes were all done serially, with the Eclipse commanding the NOVA, then waiting for a response, and the NOVA commanding the OCTEK, then waiting for a response. By using the multiple task priority lists in the NOVA many of the delays in the Eclipse/NOVA communications could be reduced to communications delays, rather than communications and process delays. The NOVA could be working ahead on characterizing the edges into bit maps while the Eclipse is attempting to match feature edges, thus giving parallel oprations in the two systems.

Techniques for getting a major system speed-up fall into two main categories. First, existing improved hardware could be purchased and the software converted, giving quick results. The second choice is to develop a specialized architecture around the algorithms, allowing maximum parallelism and pipelining the processes. Through pipelining the processes and maximizing parallelism, the processing time for a single image may be reduced to a delay of just seconds, and the results for the image frames could be available at the frame rate of 30 images per second. The result would quality processing of each image with all images processed. The processing delay could then be

compensated in the process models interval to the robot control system. If more memory were available the algorithms could be speeded up substantially for either of these choices.

VI. Conclusions and Recommendations

Robot Vision Systems

A robust robot vision system must be derived heuristically using combinations of currently available techniques until a coherent mathematical model of vision and sight can be developed from it. No such complete model exists today, but the simplistic models put forth by many researchers have fallen far short of the desired goals. No system today can travel an arbitrary terrain, avoiding obstacles and picking a path based on visual data and seeking a visually located landmark. A solution which appears to be achieving these goals is to derive a way to recognize some basic parts of the visual scene, then be able to geometrically combine these parts to recognize objects and configurations of objects. Although this is one of the goals of this project, the project has not gotten that far yet. The approach taken here seems reasonable, but the goal is elusive. The vision system foundation is built using old equipment with greatly reduced computing capability, as well as restricted visual processing capability, however the results are promising and useful. This project has fairly overextended the limits of the equipment on hand, and modern replacement equipment was delayed by procurement delays.

The Queen Victoria Algorithm performed remarkably well compared to other techniques of feature extraction, due to its low time complexity and its simplicity of

implementation. It also avoided floating point arithmetic, lending itself to easy implementation on the NOVA, which has no floating point hardware. It also is well suited for future microprocessor implementations in cheap processors making up highly parallel architectures dedicated to the robot vision tasks.

Some aspects of the overall vision system and the particular robot vision tasks do need further research and testing in this environment. These are listed more fully in the next section.

Extensions to Consider

Certain of the task categories have a number of improvement steps. Not all of these steps are necessary, but each of the steps can either enhance the performance of the overall system, or expand its realm of operation. The task categories are the building blocks of the software and hardware organization. They are image preprocessing, feature extraction, modelling, and path planning.

Image Preprocessing. The quality of image obtained using the Dage camera and the OCTEK image digitizer is not good. Sixteen levels of grey gives little dynamic range for automatic brightness or contrast compensation within the digitized range of values. Also the assumption of an undistorted rectangular image limits the processing to a small view angle. Ways to improve these are:

1. Use a wide-angle lens on each camera and a mapping from lens image space to a rectangular planar image.

2. Allow the image processing program to control brightness and contrast

    a. Control iris or image digitizer dc bias for brightness control.

    b. Control video signal amplification around brightness range center for contrast control.

Other techniques of improving image quality and focussing attention on specific local image properties may be useful also. Many of these problems would be less severe if the camera could be panned and tilted under robot control easily, and the succesive image frames could be registered to form an environmental look panorama. Also 256 grey levels of digitized video would be helpful. Colors could be useful too, but they would add a few more dimensions of complication to the algorithms.

Feature Extraction. A number of improvements could be made to the feature extraction process, most of which are concerned with higher level operations. These include separated intervals of matching region parts taken as one region with an occlusion, more symbolic details of shape, greater resolutions, and additional surface characteristics. These are each handled by:

1. Allowing skipping an occluding small central region to scan further out for larger parts of the occluded region to be characterized.
2. Allowing multiple star pattern placements to find moderately straight edges, curves, regions connected via a bottleneck, etc.
3. Allowing masking out larger areas already processed and seeing only more detailed features giving levels of processing for greater detail in the model.
4. Use more levels of grey to allow more threshold changes in the Queen Victoria Algorithm for lower resolution and higher resolution comparisons.
5. Add in code to handle surface texture descriptions.
6. Add in surface curvature estimation techniques.

The resulting characterizations would make the system even more robust, but would greatly increase the amount of processing power required. Extracting these features is useless, however, if improved modelling is not considered.

Modelling. Symbolic models must be greatly improved for reliability of visual judgement by a robot. Even dogs and cats can recognize objects, but to do this some changes

are needed. Accuracy of edge descriptions are necessary and object modelling is needed. These can be handled by:

1. Generate a graph structure of the region boundaries for more accurate updates of edge locations.
2. Build a hierarchical symbolic database to be used for possible 3-D object recognition. The top level is of objects, the next of surfaces, the next of edges, and the bottom of vertices. Parts of objects can be broken out as objects also, giving the capability to describe complex objects in terms of 3-D configurations of vertices, lines, surfaces, and their object components.

This addition would partially solve the object recognition problem.

Path Planning. Planning a path can be done in many different ways, but the three ways suggested here cover the main categories of interest for the basic robot unit. These categories are:

1. Implement a 2-D map for route planning on surfaces, such as the a floor, or the earth's surface.
2. Implement a 3-D map for route planning in 3-D environments such as underwater, in the air, in

space, or in multilevel structures such as mine shafts, caves, or buildings.

3. Build a map registration algorithm to allow a 3-D map of an environment to be projected into a 2-D map, then be registered onto a reference map of the goal location and landmarks.

Building and using a 2-dimensional map directly can be easier, but not necessarily as accurate or robust as building the 3-dimensional model, then getting the 2-dimensional projection over the interval volume through which the robot must pass. Each of these techniques solves a specific problem, but neither solves all the problems encountered in path finding. Each must be modified as needed to solve the problems, and then evaluated for its corresponding worth to each of the applications at hand.

These categories of recommended further investigation all would lead to improved performance over the current system, and most are absolutely necessary to guarantee a system which satisfies the criteria given in chapter I. Follow-on researchers will be able to build on this robot vision system design base.

Final Conclusions

Stereo vision is useless beyond about 15 feet for the camera separation of .75 feet, a picture granularity of 128 pixels width at a virtual image plane distance of 180. The

errors become very large and positions unreliable due to the granularity of the pixels in the image. This can be compensated by either:

a) more pixels per line in the image, or

b) greater camera separation

With the present systems in the signal processing lab the number of pixels per line could be doubled. The images used were shrunk by half in each dimension to allow faster access to both images throughout the process.

Greater camera separation could also be achieved, but the matching process would become less reliable as the differences between what the cameras saw became more extreme. Greater camera separation is only useful for longer range looks, close-up obstacles may be seen in only one image for widespread cameras.

An alternative to stereo cameras and stereo disparity for image processing is a moving single camera. If images are input and processed at the frame rate of 30 per second then the differences between succesive frames would be minimal, and those differences would become the clues to 3-D vision modelling. An estimate of camera motion is needed, as well as a tabulation of sets of differences in successive frames. Some differences will be caused by sets of pixels moving together relative to the reference portions of the image. These sets of differences become views of objects

which are either closer than or further than the reference portion of the image. Set descriptions of shade, shape, and relative location can allow these sets to be matched in various views of a 3-D environment, giving matches and updates for a 3-D model.

Such monocular vision and modelling, duplicated for two cameras, would give a second source of model data for resolving ambiguities, and redundancy for graceful degradation of the system in case øf camera failure.

The current system is useful only for observing object 15 feet or less from the cameras. The close range accuracy may make stereo vision as implemented here more useful for the control of robot arm positioning than for environmental path finding and traversal of paths.

Bibliography

[AHO 74] Aho, A., J. Hopcraft, and J. Ullman. The Design and Analysis of Algorithms. Reading, Massachusetts: Addison-Wesley Publishing Co. (1974).

[BAKE 82] Baker, H. H. and T. O. Binford "A System for Automated Stereo Mapping", Proceedings of the International Society for Photogrammetry and Remote Sensing Commission II, Symposium on Advances in Instrumentation for Processing and Analysis of Photogrammetric and Remotely Sensed Data, Ottawa, Canada (August 1982).

[BENT 79] Bentley, J. L. and J. Friedman. "Data Structures For Range Searching," ACM Computing Surveys, 11: 397-409 (December 1979).

[CLIF 84] Clifford, T. and H. Schnieder. Creating a Mobile Autonomous Research System (MARRS). M.S. Thesis, Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology, (AFIT/GE/ENG/84D-19), (December 1984).

[CROW 84] Crowley, J. L. "A Computational Paradigm for Three Dimensional Scene Analysis", Reprint from the Workshop on Computer Vision: Representation and Control, IEEE Computer Society (May 1984).

[DIXO 72] Dixon, L. Nonlinear Optimisation. Crane, Russak, and Company, New York (1972).

[FU 82] Fu, K. S. Syntactic Pattern Recognition and Applications. Prentice-Hall Inc. Englewood Cliffs, New Jersey (1982).

[GOSH 83] Goshtasby, A. A Symbolically-Assisted Approach to Digital Image Registration with Application in Computer Vision. PhD dissertation, College of Engineering, Michigan State University, TR#83-013, (1983).

[HOLT 82] Holten, J. Associative Data Access Method (ADAM). Master's thesis, Air Force Institute of Technology, Wright Patterson AFB, OH (1982).

[HOLT 85] Holten, J. R., S. K. Rogers, M. Kabrisky, and S. Cross. "Stereo image ranging for an autonomous robot vision system", SPIE Proceedings Vol 579: Intelligent Robots and Computer Vision (1985).

[HORE 80] Horev, M. Picture Correlation Model for Automatic Machine Recognition, Master's thesis, Air Force Institute of Technology, Wright Patterson AFB, OH (1980).

[KABR 85] Kabrisky, M. Personal communication (1985).

[LOZA 79] Lozano-Perez, T. and M. Wesley. "An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles". Communications of the ACM, 22 (10): 560-570 (October 1979).

[MARR 79] Marr, D. and T. Poggio "A computational theory of human stereo vision", Proceedings of the Royal Society of London, B 294, pp. 301-328 (1979).

[MARR 80] Marr, D. and E. Hildreth "Theory of edge detection", Proceedings of the Royal Society of London, B 207, pp. 187-217 (1980).

[MAYB 79] Maybeck, P. Stochastic Models, Estimation, and Control, Volume 1. New York: Academic Press (1979).

[MEDI 83a] Medioni, G. G. Matching Images Using Linear Features. Phd dissertation, ISG Report 103, University of Southern California, August (1983).

[MEDI 83b] Medioni, G. G. and R. Nevatia "Segment-Based Stereo Matching", Proceedings of the Image Understanding Workshop, Defensed Advance Research Projects Agency and Computer Vision and Pattern Recognition Conference of the IEEE computer Society, pp. 128-136 (June 1983).

[MESS 83] Messner, R. A. and H. Szu. "Coordinate Transformation From an Image Plane Directly to an Invariant Feature Space", IEEE Conference on Computer Vision and Pattern Recognition, pp 522-530, (1983).

[MONA 84] Monaghan, G. E. Navigation for an Autonomous Mobil Robot, M.S. Thesis. Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology (AFIT/GE/ENG/84D-47), (December 1984).

[MORA 79] Moravec, H. P. "Visual Mapping by a Robot Rover", Proceedings of the 6th International Joint Conference on Artificial Intelligence, Tokyo, Japan, pp. 598-600 (August 79).

[MORA 83] Moravec, H. P. "The Stanford Cart and the CMU Rover", Proceedings of the IEEE Vol 71, No. 7 pp. 872-884 (July 1983).

[NEVA 84] Nevatia, R. "Image Understanding Research at USC", Proceedings of the Image Understanding Workshop, Defense Advanced Research Projects Agency, pp. 33-41 (October 1984).

[NILS 80] Nilsson, N. Artificial Intelligence. Tioga Publishing Company, Palo Alto California, (1980).

[PENT 84] Pentland, A. P. "Fractal-Based Descriptions of Natural Scenes", IEEE Transactions on Pattern Analysis and Machine Intelligence, V PAMI-6, No. 6, (November 1984).

[OWEN 83] Owen, R. J. Environmental Mapping by a Hero-1 Robot Using Sonar and a Laser Barcode Scanner, M.S. Thesis. Wright-Patterson AFB, OH: School of Engineering, Air Force Institute of Technology (AFIT/GE/EE/83D-52), (December 1983).

[PETE 77] Petersen, S. CARTAM: The Cartesian Access Method for Data Structures with N-Dimensional Keys. Headquarters, Strategic Air Command, Offutt AFB, NB. Reprint of material for his PhD dissertation at California Institute of Technology (1977).

[RUFF 84] Ruff, R., and N. Ahuja "Path Planning in a Three Dimensional Environment", Proceedings of the Seventh International Conference on Pattern Recognition, Vol 1, pp. 188-191, (1984).

[SRIH 82] Srihari, S. N. and J. Tindall. "Multiresolutional Representation of Three-dimensional Objects". IEEE Proceedings of the International Conference on Cybernetics and Society, pp 322-326, (1982).

[STRA 83] Strategic Computing. Defense Advance Projects Research Agency (DARPA), 28 October (1983).

[WINS 77] Winston, P. Artificial Intelligence. Addison-Wesley, Reading Massachusetts, (1977).

[YAKI 78] Yakimovsky, Y. and R. Cunningham. "A System for Extracting Three-dimensional Measurements from a Stereo Pair of T. V. Cameras". Computer Graphics and Image Processing, Vol 7, pp 195-210, (1978).

[YAU 83] Yau, M. and S. Srihari. "A Hierarchical Data Structure for Multidimensional Images". Communications of the ACM, Vol 26, No 7, pp 504-515, July (1983).

APPENDIX A:

Location Estimation

Location Estimation

Estimating the location of features in three dimensional space, based on finding their relative positions in images is basic to modelling the environment based on camera views. Certain assumptions greatly simplify the calculations, yet by modelling measurement errors the simple model can be made robust. Since exactness is beyond the cabability of the sensors, then techniques of estimating locations and their errors are needed, as well as methods of updating those locations and errors through multiple looks and sensor measurement combination.

This derivation assumes that the images are generated by a stereo pair of cameras with parallel image axes. The cameras are horizontally positioned at a fixed separation such that raster scan lines approximately correspond between images, allowing the simplification associated with epipolar lines. Also the cameras are assumed to have less image pixel location distortion than the resolution caused by the sampling of the digitizer. This will not work for wide angle lenses due to the extreme distortion, but is adequate for this application. The cameras are considered to be identical.

To get distance estimates to objects in the images certain properties of the camera pair must be derived or measured. This is the characterization of the cameras.

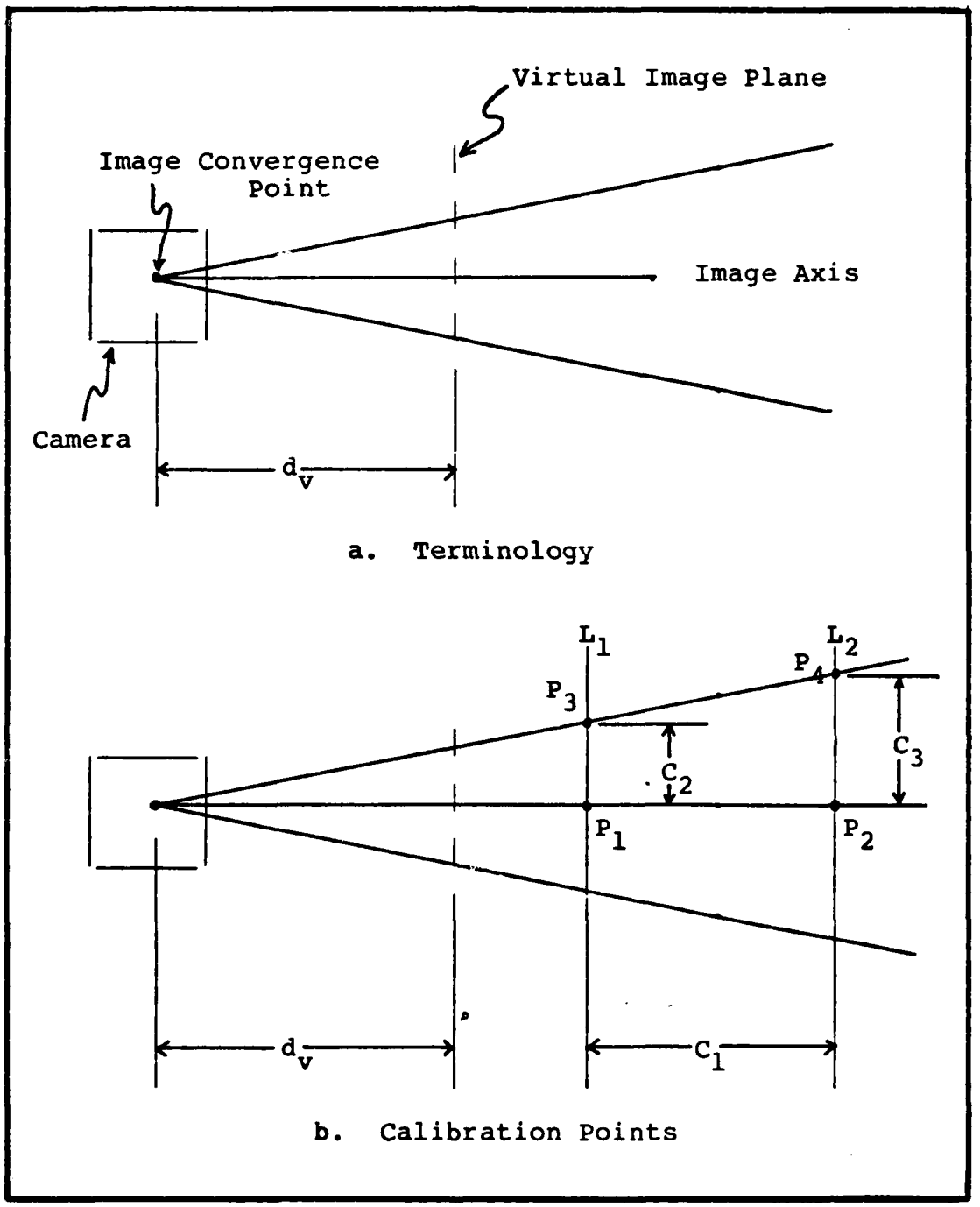


Fig. A-1 Single Camera Characterization

Camera Characterization

The only characteristics needed for the cameras are the distance between their visual axes in applicable units of distance measure, d_s, and a derived distance from the image convergence point to the image plane in pixel widths, d_v. In Figure A-1a a single camera is shown with its image convergence point, image plane, and image axis. In Figure A-1b the same illustration has the calibration measurement points needed to derive d_v.

Two points, P_1 and P_2, on the image axis can be found by causing P_2 to appear in the center pixel in the image, then moving P_1 to occlude the view of P_2. The distance between P_1 and P_2 is then measured as C_1. A horizontal line is then erected perpendicular to the image axis through each of P_1 and P_2. These are L_1 and L_2 as in Figure A-1b. On L_2 a point P_4 is selected such that it appears within the image observed from the camera and the distance C_3 is measured from the image axis to P_4. P_3 is then moved along L_1 until it occludes P_4, then its distance from the image axis, C_2, is measured. The horizontal pixel offset from the image center of the appearance of P_3 and P_4 in the image is then counted and is d_1.

By the properties of similar right triangles the relationship

$$d_1 / d_v = (C_3 - C_2) / C_1 \qquad \text{(A-1)}$$

holds. Solving for d_v gives

$$d_v = d_1 C_1 / (C_3 - C_2) \qquad \text{(A-2)}$$

as long as $C_2 < C_3$, $C_1 \neq 0$, and $d_1 \neq 0$.

A separate d_v could be computed for each camera, and d_v changes with variations in lens power. However, for these derivations it is assumed that the cameras are identical, and the lenses are fixed power. Thus, the camera pair is characterized by d_s, their separation, and d_v, the virtual distance to the image plane.

Camera Relative Coordinates

In camera relative coordinates both image feature point location in 3 dimensions and estimation of the error is straightforward. The mean location is estimated by triangulation, and the error is bounded in the image, then linearly transformed to bounds on the 3 dimensional location estimate.

Location Computation. Assuming the stereo camera pair is positioned so that the raster lines form epipolar lines, then d_s and d_v are used in conjunction with the horizontal pixel offsets to estimate the distance along the reference camera image axis to the perpendicular plane including the feature point. Once the plane distance, d_p, is known then the horizontal offset, d_x, and vertical offset, d_y, can be derived, giving the camera image axis relative feature point

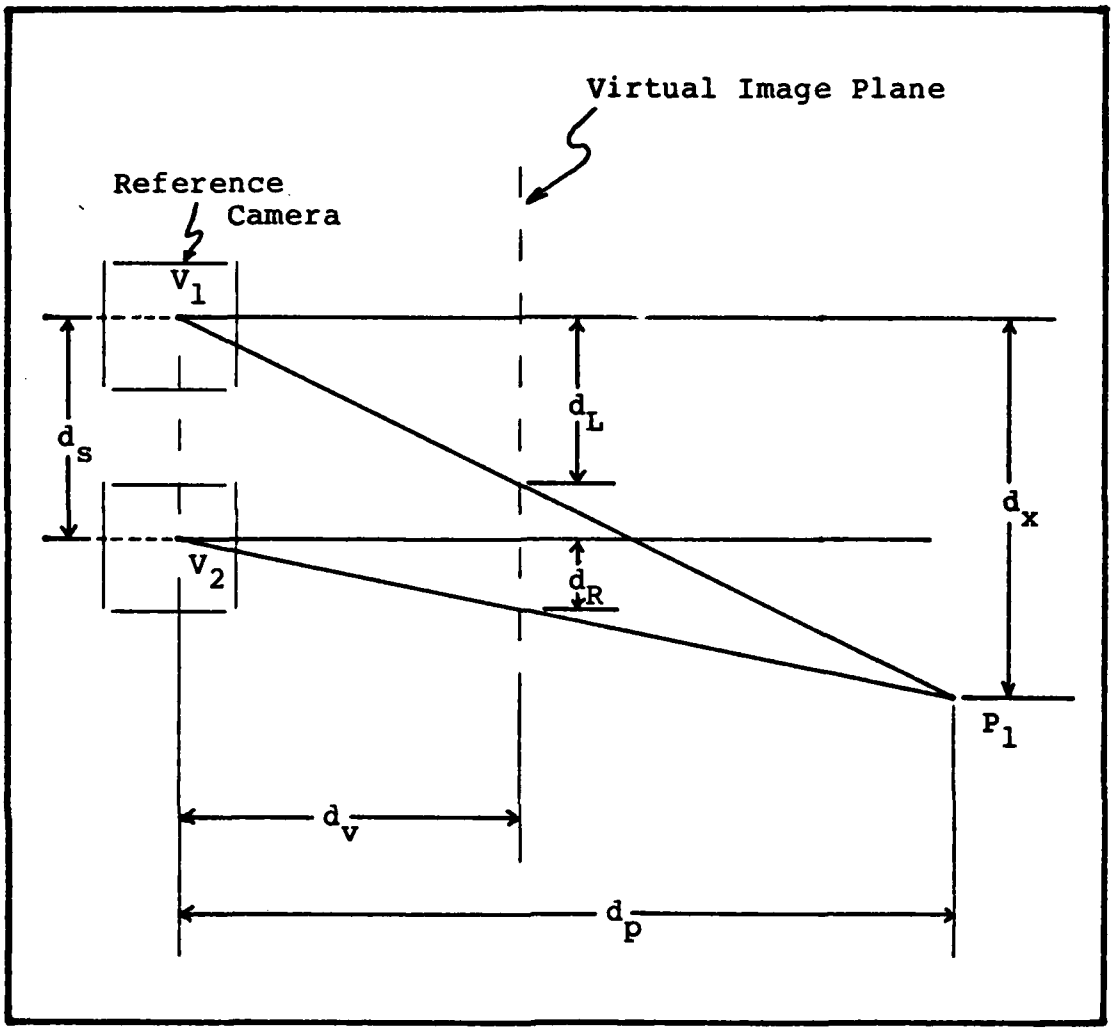


Fig. A-2 Distance and offset Calculation

position as (d_p, d_x, d_y). If the aspect ratio for the camera is not unity, then a vertical calibration value corresponding to d_v must be derived. This derivation assumes the aspect ratio is one.

Figure A-2 shows the image convergence points for a

stereo pair of cameras, and shows the important measurements. Given the horizontal pixel offsets from the image axes in each of the two images, d_L in the left image, and d_R in the right, then the distance to the orthogonal plane including the feature point is given by properties of similar triangles as

$$d_L / d_v = d_x / d_p \qquad \text{(A-3)}$$

and

$$d_R / d_v = (d_x - d_s) / d_p \qquad \text{(A-4)}$$

These can be combined as

$$(d_L - d_R) / d_v = d_s / d_p \qquad \text{(A-5)}$$

giving the distance to the plane as

$$d_p = d_s d_v / (d_L - d_R) \qquad \text{(A-6)}$$

This holds for $d_L > d_R$, $d_v \neq 0$, and $d_s \neq 0$, keeping $0 < d_p < \infty$.

Using Figure A-2 again, or considering equation (A-3) gives

$$d_x = d_L d_p / d_v \qquad \text{(A-7)}$$

and likewise, for digitized images of unity aspect ratio

$$d_y = d_D d_p / d_v \qquad \text{(A-8)}$$

where d_D is the number of vertical pixel widths offset in

the image plane, from the image axis to the feature point pixel.

Error Estimates. These derivations assume that the calibration error is minimal, and not significant relative to feature point position estimation errors. Here the propagation of pixel position errors to physical location error estimates only are considered. The error in position in locating a feature point to a pixel position can be represented by the right image pixel offset error e_R, the left, e_L, and the vertical e_D. From these and equation (A-6) the distance error to the feature point plane, e_p, is given by

$$(d_p+e_p) = d_s\ d_v\ /\ (d_L+e_L-d_R-e_R) \qquad \text{(A-9)}$$

or, subtracting equation (A-6)

$$e_p = \frac{d_s\ d_v}{d_L+e_L-d_R-e_R} - \frac{d_s d_v}{d_L-d_R} \qquad \text{(A-10)}$$

$$= \frac{d_s d_v(d_L-d_R)-d_s d_v(d_L+e_L-d_R-e_R)}{(d_L+e_L-d_R-e_R)\ (d_L-d_R)} \qquad \text{(A-11)}$$

$$= \frac{d_s d_v(e_R-e_L)}{(d_L-d_R)^2+(d_L-d_R)(e_R-e_L)} \qquad \text{(A-12)}$$

Putting bounds on e_p by assuming worst case values of e_R and

e_L gives

$$|e_p| \leq \frac{d_s d_v(|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_L|)} \qquad \text{(A-13)}$$

which only holds for $|e_R|+|e_L|<(d_L-d_R)$. However for cases where the offset error would be greater than or equal to the disparity the position would not be useable, so this is an acceptable assumption.

These results can then be used to derive an offset error estimation using equation (A-7) to get

$$(d_x+e_x) = (d_L+e_L)(d_p+e_p)/\ d_v \qquad \text{(A-14)}$$

Subtracting equation (A-7) leaves

$$e_x = (d_L e_p+d_p e_L+e_p e_L)/d_v \qquad \text{(A-15)}$$

which can be bounded by

$$|e_x| \leq (|d_L|\ |e_p|+d_p|e_L|+|e_p e_L|)/\ d_v \qquad \text{(A-16)}$$

However, using the limits on e_p from equation (A-13) gives

$$|e_x| \leq \frac{(|d_L|+|e_L|)d_s(|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_L|)} + \frac{d_p|e_L|}{d_v} \qquad \text{(A-17)}$$

Likewise, for the vertical offset, and using equations (A-8) and (A-13) gives

$$|e_y| \leq \frac{(d_D+|e_D|)d_s(|e_R|+|e_L|)}{(d_L-d_R)^2-(d_L-d_R)(|e_R|+|e_L|)} + \frac{d_p|e_D|}{d_v} \qquad \text{(A-18)}$$

By propagating these errors to camera-relative position errors, error bounds on the location have been derived. However, the actual errors in camera relative location are more Gaussian in nature, so letting $E_p=\max|e_p|$, $E_x=\max|e_x|$, and $E_y=\max|e_y|$ represent the 2σ multiples of the standard deviations for each dimension for the location, and assuming they are independent gives the point covariance matrix as

$$C = \begin{bmatrix} \sigma_p^2 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_y^2 \end{bmatrix} \qquad \text{(A-19)}$$

where $\sigma_p=E_p/2$, $\sigma_x=E_x/2$, and $\sigma_y=E_y/2$, giving

$$C = \begin{bmatrix} (E_p/2)^2 & 0 & 0 \\ 0 & (E_x/2)^2 & 0 \\ 0 & 0 & (E_y/2)^2 \end{bmatrix} \qquad \text{(A-20)}$$

Environmental Model Relative Coordinates

The position and orientation of the camera at any time may be represented by four vectors, as shown in Figure A-3. The first is the camera position relative to the environmental model coordinate system, P_c, and the other three represent the camera relative coordinate axes. The

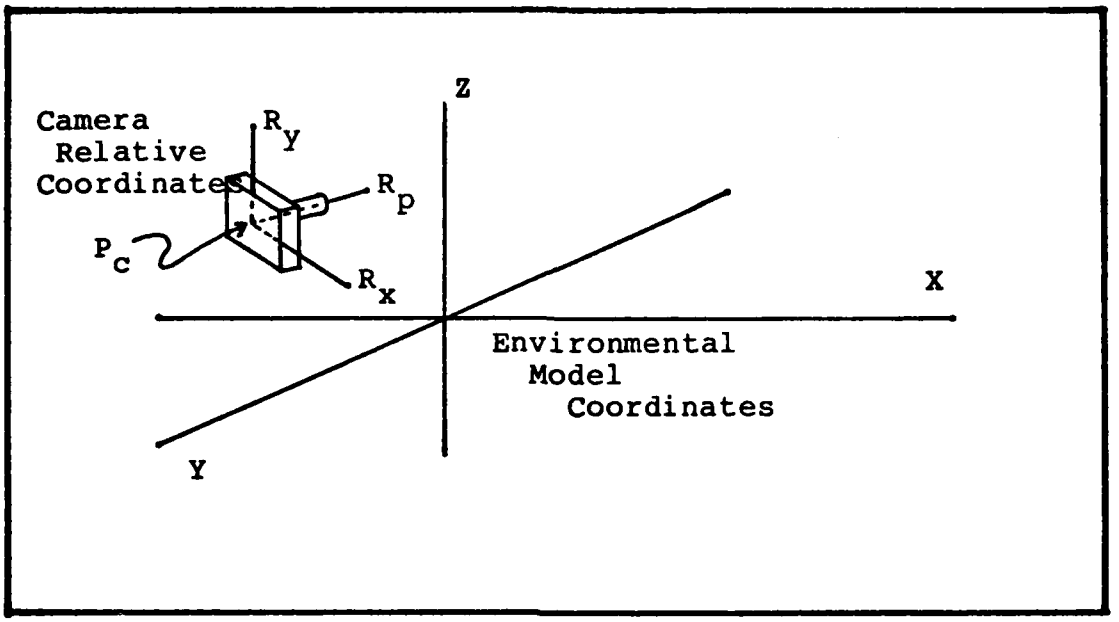


Fig. A-3 Alternate Coordinate Systems

camera relative coordinate axes are each unit vectors, and consist of direction cosines for the camera relative image axis, right horizontal axis, and the upward vertical axis, relative to the environmental coordinate system. These are represented, respectively by R_p, R_x, and R_y. The state vector for the camera is then given by Y, where

$$Y^T = (P_c^T \quad R_p^T \quad R_x^T \quad R_y^T)^T \qquad \text{(A-21)}$$

Each feature position can then be represented by a transformation from the camera state vector to a position in the environmental model. For the ith feature point this is given by

$$P_i = \begin{bmatrix} 1 & 0 & 0 & p_i & 0 & 0 & x_i & 0 & 0 & y_i & 0 & 0 \\ 0 & 1 & 0 & 0 & p_i & 0 & 0 & x_i & 0 & 0 & y_i & 0 \\ 0 & 0 & 1 & 0 & 0 & p_i & 0 & 0 & x_i & 0 & 0 & y_i \end{bmatrix} \quad \text{(A-22)}$$

where

$p_i = d_p$ for the ith feature point,

$x_i = d_x$ for the ith feature point, and

$y_i = d_y$ for the ith feature point.

The transformation is given by

$$P_i Y = X_i \quad \text{(A-23)}$$

where

P_i = the transformation matrix from camera location to feature point location in the environmental coordinates

X_i = the location of feature point i in the environment

By generating a single coordinate transformation from N camera relative feature locations' estimates, letting

$$p^T = (P_1^T \; P_2^T \; \ldots \; P_N^T)^T \quad \text{(A-24)}$$

and expecting their environmental model relative locations to be

$$X^T = (X_1^T \; X_2^T \; \ldots \; X_N^T)^T \quad \text{(A-25)}$$

then the entire transformation of points from camera relative to environmental relative can be written

$$PY = X \tag{A-26}$$

For the purpose of error propagation an augmented form of the covariance matrix in equation (A-19) is generated for each feature point, i=1,2,...,N as

$$C_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma_p^2 I & 0 & 0 \\ 0 & 0 & \sigma_x^2 I & 0 \\ 0 & 0 & 0 & \sigma_y^2 I \end{bmatrix} \tag{A-27}$$

where each of the elements shown is a 3x3 matrix.

Including error in camera position leaves the problem with a set of points with possible position error, being transformed to a coordinate system using a transformation which itself includes error. Thus a model of the conversion from camera relative coordinates to environmental model location coordinates for point i is

$$X_i = (P_i + N_{Pi})\ (Y + N_Y) \tag{A-28}$$

where N_Y is the random noise error in camera position, and N_P is the random noise error in the feature point location in camera relative coordinates. This can be rewitten as

$$X_i = P_iY + P_iN_Y + N_{Pi}Y + N_{Pi}N_Y \tag{A-29}$$

Thus, the contributions of the error terms are multiplicative and may make a substantial contribution to

the final location estimate for each point. A way is needed to bring one of those error terms as near to zero as possible to eliminate the error product. However, N_P depends on the quantized pixel indices of the feature locations, which cannot be any more accurate. By combining multiple sensor inputs, system updates to a vehicle and camera position state model can be made stochastically. These combined inputs and linear model predictions of vehicle motion can give high quality estimates of position. Assuming the camera position and orientation error is identically zero reduces equation (A-29) to

$$X_i = P_i Y + N_{Pi} Y \tag{A-30}$$

and the real-world feature mean locations are given by equation (A-26), while the environmental model-relative point location covariance for point i is given by

$$C_{Pi} = Y^T C_i Y \tag{A-31}$$

$$= \begin{bmatrix} r_{xx}\sigma_x^2 & r_{xy}\sigma_x\sigma_y & r_{xz}\sigma_x\sigma_z \\ r_{xy}\sigma_x\sigma_y & r_{yy}\sigma_y^2 & r_{yz}\sigma_y\sigma_z \\ r_{xz}\sigma_x\sigma_z & r_{yz}\sigma_y\sigma_z & r_{zz}\sigma_z^2 \end{bmatrix} \tag{A-32}$$

These are idealized results, and the only sensor with high enough angular resolution to virtually eliminate camera orientation error is the camera pair. However, getting camera position and orientation from the camera pair is not

a trivial task. It is known as the camera calibration problem.

Camera Location Estimation

Before visual data can be used for camera location estimation some problems must be solved. These are

1. Modelled feature points must be matched reliably to 'seen' feature points,
2. A method of mapping 'seen' feature point locations via camera location and orientation to environmetal model location coordinates must be derived, and
3. The modelled quality of a feature location estimate in the environmental model as well as the camera relative location quality must be taken into account in any 'best fit' strategy.

It is assumed that a technique which has solved the feature matching is used to satisfy item number 1. The other steps depend entirely on this symbolic error being zero. Item number 2 is derived and given as equation (A-26). Only a method of weighing the error contributions of near misses in the several feature points must still be derived.

Since the measure of quality of point location is given by the covariance matrix of each point, then weighing their contributions to a cost function by the inverse of their environmental model relative covariance matrix, given in

equation (A-32), will be used. Let the weighted term for each point be given by

$$D_i = C_{Pi}^{-1} \tag{A-33}$$

and, since the points are independent, the weighted term for N points is

$$D = \begin{bmatrix} D_1 & 0 & \dots & 0 \\ 0 & D_2 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \dots & D_N \end{bmatrix} \tag{A-34}$$

The cost of each point is then given as the square of the difference between the observed location X_i, and the modelled location X_i', weighted by D_i as

$$J_i = (X_i - X_i')^T D_i (X_i - X_i') \tag{A-35}$$

and the total cost as

$$J = (X - X')^T D (X - X') \tag{A-36}$$

Since the search is for the position Y which gives the minimum cost, then

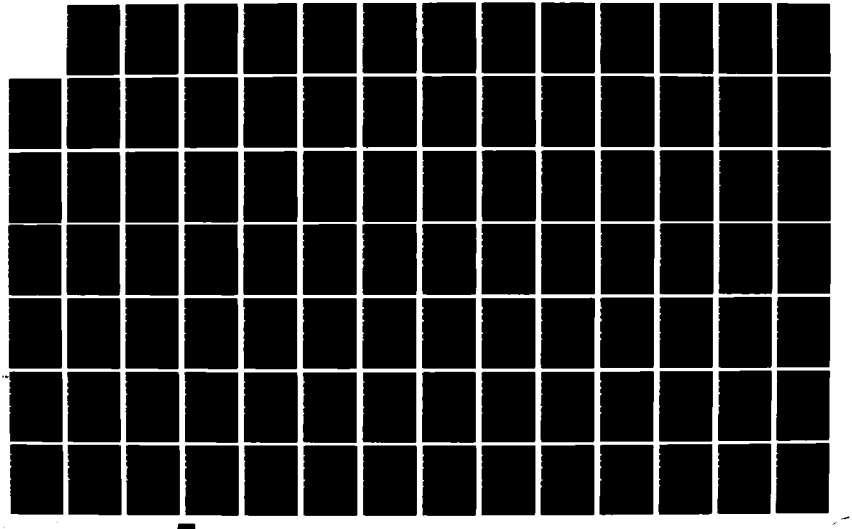
$$J = (PY - X')^T D (PY - X') \tag{A-37}$$

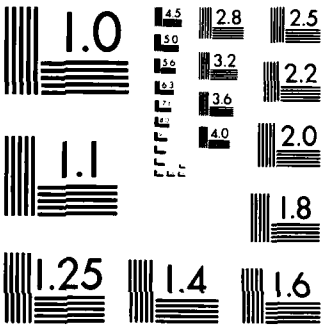
or

$$J = Y^T P^T DPY - Y^T P^T DX' - X'^T DPY + X'^T DX' \tag{A-38}$$

AD-A164 282 A ROBOT VISION SYSTEM(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J R HOLTEN DEC 85 AFIT/DS/ENG/85D-1 2/3

UNCLASSIFIED F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART

But, since D is positive semidefinite, to minimize the cost the value of Y must be found such that $\partial J/\partial Y = 0$. To simplify these calculations equation (A-38) is rewritten in terms of the quadratic cost term, B, and the other cost terms, Z, giving

$$\partial J/\partial Y = \partial B/\partial Y + \partial Z/\partial Y \tag{A-39}$$

where

$$B = Y^T P^T DPY \tag{A-40}$$

$$= Y^T \begin{bmatrix} \sum D_i & \sum p_i D_i & \sum x_i D_i & \sum y_i D_i \\ \sum p_i D_i & \sum p_i^2 D_i & \sum p_i x_i D_i & \sum p_i y_i D_i \\ \sum x_i D_i & \sum p_i x_i D_i & \sum x_i^2 D_i & \sum x_i y_i D_i \\ \sum y_i D_i & \sum p_i y_i D_i & \sum x_i y_i D_i & \sum y_i^2 D_i \end{bmatrix} Y \tag{A-41}$$

giving

$$\partial B/\partial Y = 2(P^T DPY) \tag{A-42}$$

and

$$Z = -Y^T P^T DX' - X'^T DPY + X'^T DX' \tag{A-43}$$

$$= -Y^T \begin{bmatrix} \sum D_i X_i' \\ \sum p_i D_i X_i' \\ \sum x_i D_i X_i' \\ \sum y_i D_i X_i' \end{bmatrix} - \begin{bmatrix} \sum D_i X_i' \\ \sum p_i D_i X_i' \\ \sum x_i D_i X_i' \\ \sum y_i D_i X_i' \end{bmatrix}^T Y$$

$$+ \sum (X_i'^T D_i X_i') \quad (1 + p_i + x_i + y_i) \qquad \text{(A-44)}$$

giving

$$\partial Z/\partial Y = -2 \begin{bmatrix} \sum D_i X_i' \\ \sum p_i D_i X_i' \\ \sum x_i D_i X_i' \\ \sum y_i D_i X_i' \end{bmatrix} \qquad \text{(A-45)}$$

$$= -2 \; P^T DX' \qquad \text{(A-46)}$$

Thus, the minimum is given by the Y value solution to

$$(P^T DP) \; Y = (P^T DX') \qquad \text{(A-47)}$$

a 12 X 12 system of equations in 12 unknowns. If $P^T DP$ is invertible then the solution is given by

$$Y = (P^T DP)^{-1} \; (P^T DX') \qquad \text{(A-48)}$$

and to insure that the solution exists the following criteria must be met

1) at least 3 points are used,

2) at least 3 of the points used must not be colinear, and

3) the points must be perfect symbolic matches.

The error in camera position can then be derived from the covariances of the environmental model feature point location estimates by considering equation (A-48) reorganized to

$$Y = ((P^T DP)^{-1} (P^T D)) X' \qquad \text{(A-49)}$$

as a linear transformation with multiple sensor inputs, and combining these multiple sensor inputs to get the mean camera location. Likewise, the covariance estimate for the camera location estimate is given by

$$C_c = (P^T DP)^{-1} \qquad \text{(A-50)}$$

The greater the number of points and the more accurately positioned the points the better the estimate of camera location, as long as the three criteria given above are satisfied.

APPENDIX B:

Bit Maps

Bit Maps

A bit map is a low resolution binary image representing image areas within the region of interest as 1's and those outside the region as 0's. The result is an approximation of shape for the given region. These shape approximations can be rectangular bit maps or radial bit maps.

The different forms of the bit map have different properties, and the two discussed here have specific properties of interest. The rectangular bit map may be scaled so the region of interest extends from its top to bottom and left to right. This means that horizontal and vertical scaling are independent. The radial bit map is also scaled, but first a center of its extents is found, then the bit map is scaled so the region's greatest extents just reach the edges of the circle of radial arms. Thus both types of bit maps are independent of apparent region size.

Using a fixed, rather than scaled, rectangular bit map allows edges of regions to be easily characterized for matching and locating in 3-dimensional space.

Map Generation

Each of the map types used here map the region of interest into a 16 X 16 array of bits. The bits represent small parts of the total area covered by the map, and if that part contains a higher percentage of pixels within the

region than some threshold, the bit is set to 1. Here the threshold used is 80%. This allows up to 20% "noise" pixels within the region. All the bit areas which contain less than the threshold number of pixels within the region are left at 0. Figure B-1 shows a sample region and its rectangular and radial bit maps.

The rectangular bit map is stored as one column in one word, starting on the left. This allows shifting and comparing by just changing indices for horizontal displacements along the image epipolar lines. The horizontal and vertical scale factors are stored also.

The radial bit map is stored one radial pie shape per word, storing the word with the most ones as the first word, and continuing clockwise around the radial arms. This allows a simple change of index to represent a rotation. The 16 radial arms used in practice are assumed to give great enough resolution that interpolated orientations will not be needed. The scale factor and orientation are stored for the radial bit map.

Both maps have their locations stored also, thus as characteristic features are matched, their relative locations in the image are easily computed.

Map Usage

The purpose of bit maps is to make shape comparison easy, and to give a metric which is relevant to closeness of shapes. Goshtasby (GOSH 83) suggested that using bit maps

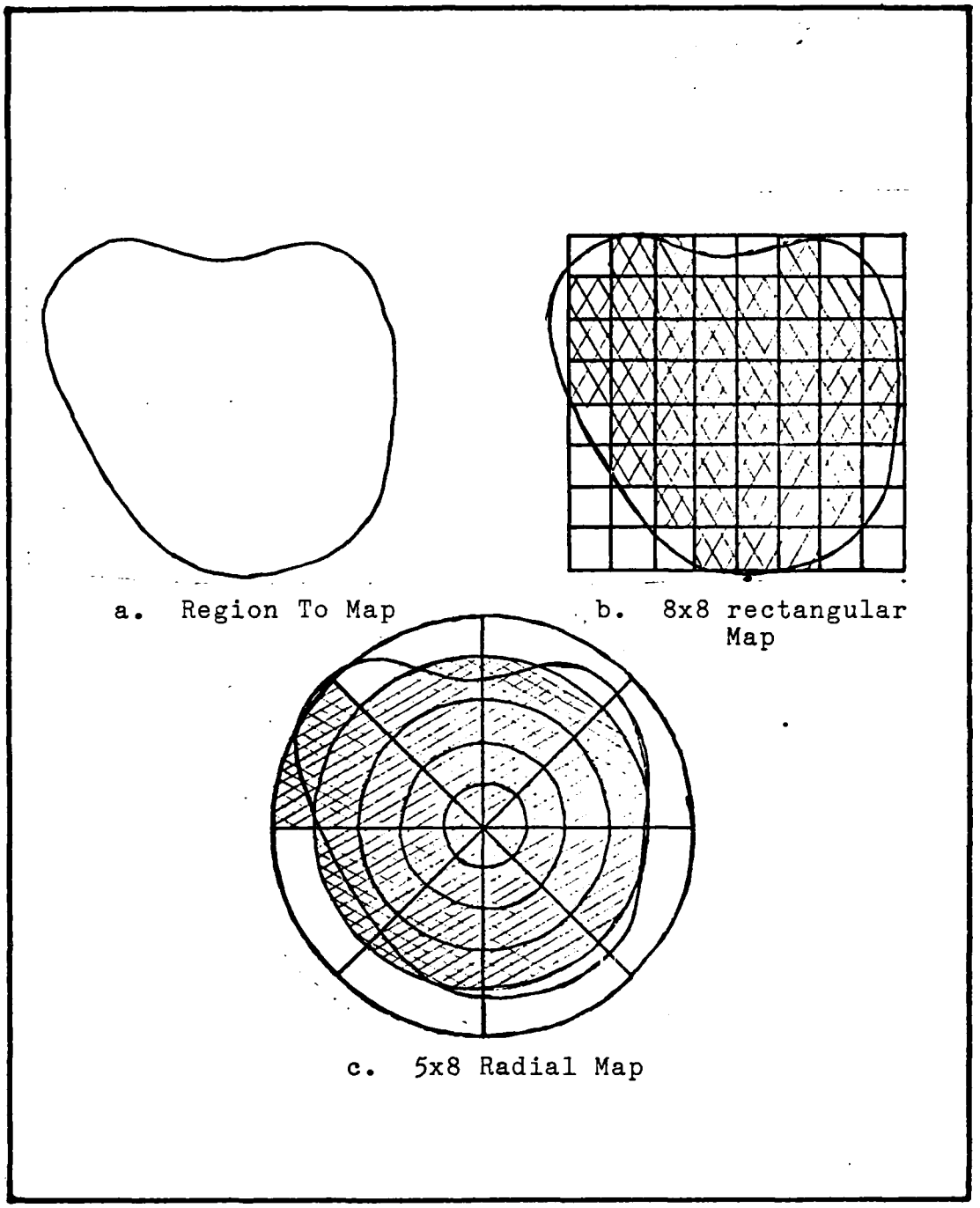


Fig. B-1 Bit Maps

allowed counting the number of bit differences for a metric of shape closeness. This is one use for bit maps of shape, but another is finding edge shapes and their positions relative to the region.

Shape Matching. Since the shapes are stored as arrays of bits, the easiest way to find the number of differences between two is to "exclusive or" their bits. This takes just applying

$$D = A_1 \text{ XOR } A_2 \qquad \text{(B-1)}$$

where A_1 and A_2 are bit maps, and D is a bit map of their differences. Doing this for each word in the map gives only 16 XOR's for an entire map comparison. By counting the number of bits set in B the measure of "closeness of fit" can be obtained.

Rectangular Maps. Using rectangular maps, edges can be found by using a shift of one position before XOR'ing the map to itself.

$$V = A \text{ XOR } LSHIFT(A) \qquad \text{(B-2)}$$

where

A is a rectangular bit map of a region
LSHIFT(A) is A shifted left one bit (both maps are extended left and right with 0's)
V is the resulting bit map of vertical edges.

These edges can then be identified as left or right edges by

$$L = A \text{ AND } RSHIFT(V) \tag{B-3}$$

$$R = A \text{ AND } V \tag{B-4}$$

where

L contains 1's for left edges,

R contains 1's for right edges, and

RSHIFT(V) shifts V right one bit.

Likewise, top and bottom edges can be found by

$$H = A \text{ XOR } USHIFT(A) \tag{B-5}$$

$$T = A \text{ AND } DSHIFT(H) \tag{B-6}$$

$$B = A \text{ AND } H \tag{B-7}$$

where

H is a map of horizontal edges,

USHIFT(A) is A shifted up one bit,

T is a map of top or upper edges of A, and

B is a map of bottom or lower edges of B.

Another useful comparison is given by

$$P = (L \text{ AND } T) \text{ OR } (L \text{ AND } B) \text{ OR } (R \text{ AND } T) \text{ OR } (R \text{ AND } B) \tag{B-8}$$

where P is a map of all vertex points between horizontal and vertical edges, or points occupying edges which have both horizontal and vertical components.

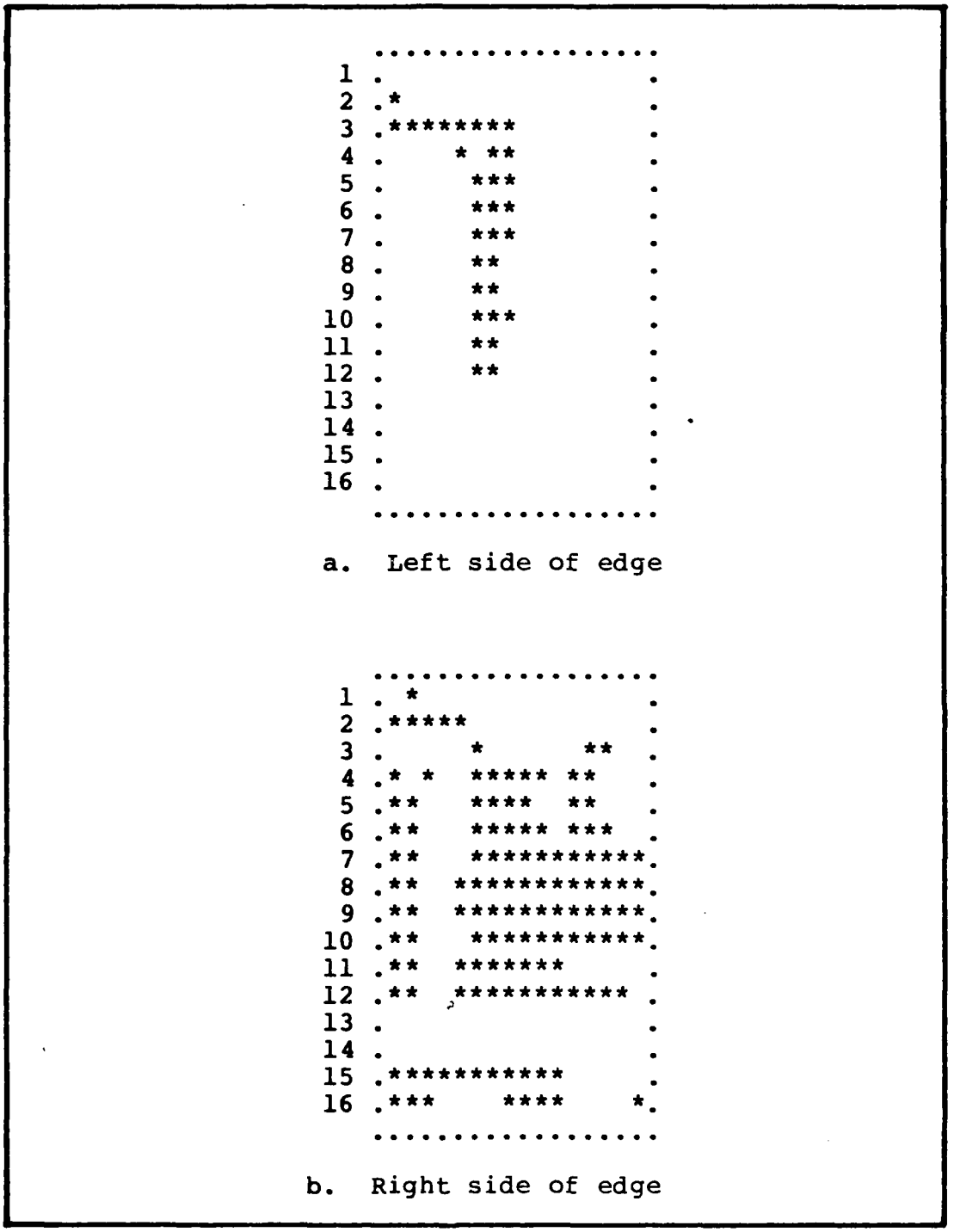
Using rectangular bit maps of regions allows the following:

1. Shape matching of regions,
2. Shape matching of left, right, top, or bottom edges, and
3. Vertex point location.

The number of bits in the map determines the resolution and quality of the matches of represented regions.

Edge maps. In stereo image pairs both cameras see the same features at approximately the same size, if at all. If a feature appears taller to one camera than to the other, then the effect is caused by either occlusion of the feature, or less often, by reflection off the feature. In all the possible cases, if feature edges are to match there will be little or no scale change.

To characterize a feature edge with a vertical component, its shape can be stored as a derived bit map. By placing a 16 X 16 pixel window centered where the edge crosses the epipolar line, the region contents can be bit mapped as in Figure B-2. After mapping the region shape within the window, the edge can be found (left or right) using logical bit map operations defined above. Once the



a. Left side of edge

b. Right side of edge

Fig. B-2 Edge Bit Maps

edge is characterized in this manner it can be used to match edge shapes, and to accurately locate the feature edge in the images for minimizing 3-D location errors.

The edge maps may be compared for multiple horizontal offsets to achieve a "best fit" shape match. This gives a robust technique for compensating for noise errors in feature location.

Radial Maps. Once a radial map is generated, the same operations as performed on rectangular maps may be used to extract characteristics. The main differences are that left and right shifts are now rotations, and as such must wrap around from the last column to the first, and edges are now circumferential or radial straight line allowing curve characterization.

When comparing radial maps, often the observers are at different angles when characterizing the regions, thus rotations may be needed. Thus radial maps may have to be compared in each of the 16 possible orientations, much like a logical convolution, then the "best" shape match may be chosen.

Summary

Rectangular shape bit maps are useful for finding horizontal and vertical lines, as well as matching region edges even under translation. Edge maps can give improved feature edge matching and 3-D location estimation. Radial

maps are useful when rotations may be involved, changing them to bit map translations left or right. Both give a useable metric of shape "closeness" for imperfect fits. Bit maps are versatile, easy to compute, and robust.

APPENDIX C:

Program Use

Program Use

Overall Organization

Figure C-1 shows the interaction between the major software modules of the robot vision system. PRE is a video inputter and preprocessor. It reduces a 256 X 240 image to a 128 X 120 image, by shrinking it, as well as allowing disk saves and retrievals of both size images. The reduced size images allow the robot vision system to retain its field of view, but still use the OCTEK to interact with both images of a stereo pair interchangeably without an appreciable wait. NSTAR is the NOVA half of the runtime robot vision system. It interacts directly with the images, loading images into the OCTEK, and performing various operations on the image as required by ESTAR. ESTAR is the Eclipse half of the runtime robot vision system. ESTAR causes the higher level vision processes to be performed to achieve modelling of the robot environment.

PRE. Through this package the user interactively loads images, preprocesses them, shrinks them, and stores them on disk for further processing by NSTAR. When PRE is initiated it initializes the OCTEK, then displays a menu of user commands. These commands are

1. Load the image from the camera to the OCTEK,
2. Shrink a 256 X 240 image to 128 X 120,
3. Apply the Q-V cartooning algorithm to a

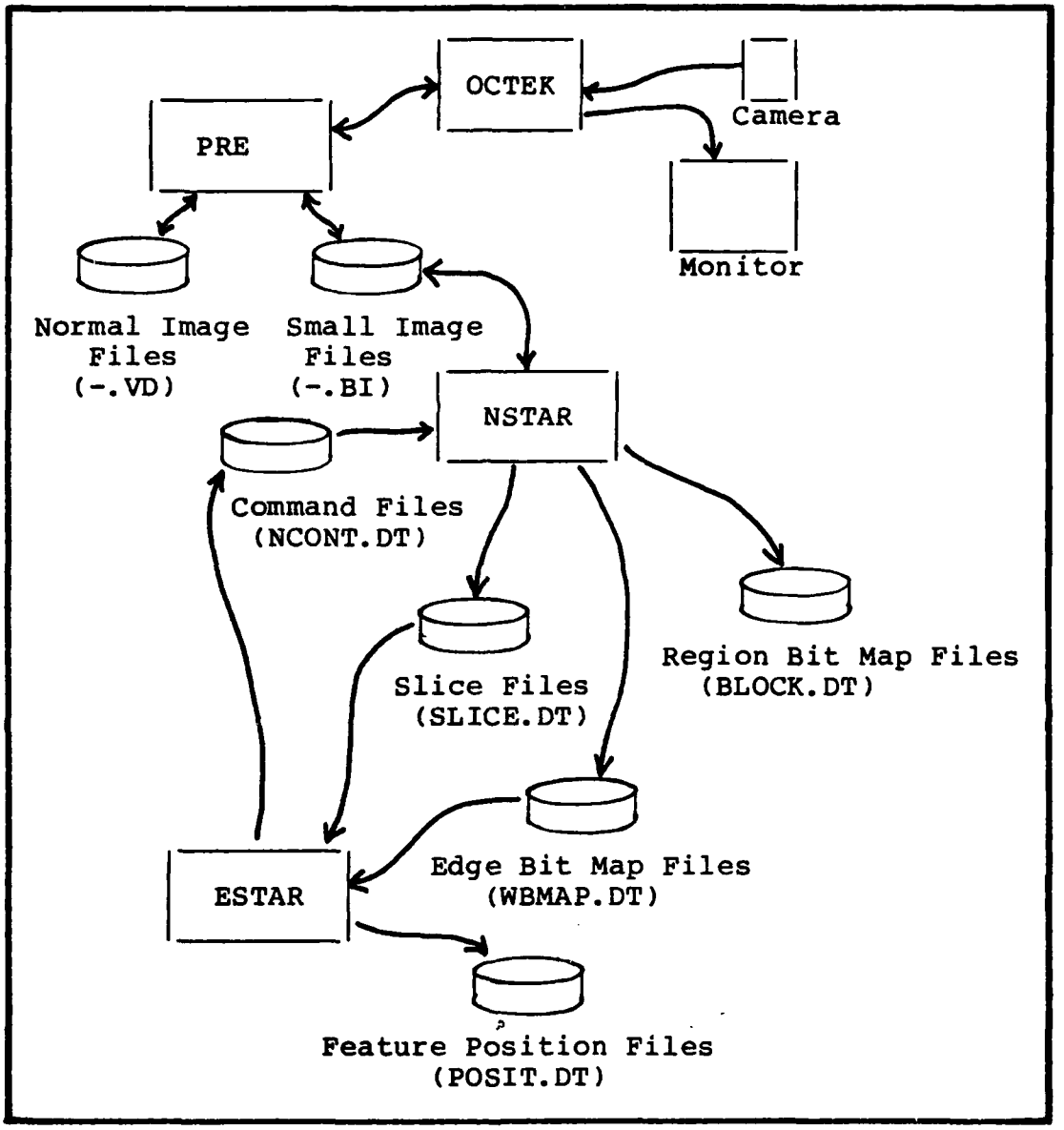


Fig. C-1 Software Communications

128 X 120 image,

4. Load a 128 X 120 image from disk,
5. Store a 128 X 120 image to disk,
6. Load a 256 X 240 image from disk, and
7. Store a 256 X 240 image to disk.

Using commands #1, 6, and 7 PRE allows the user to input and save the normal sized images. This is especially useful in acquiring stereo views and storing them to disk. They can then also be reloaded from the disk files and displayed.

Command #2 shrinks the 256 X 240 image to 128 X 120, clearing the rest of the screen. This image can then be saved using command #5 and reloaded using command #4. The shrunken image is saved in an unpacked format so it can be loaded quickly. The small size is required to retain the full field of view, but allows NSTAR to store both views of the stereo pair in the OCTEK memory for fast interaction with the images on command from ESTAR.

NSTAR. All direct interaction with the image and its pixels is done using NSTAR. It makes calls to the OCTEK library of image manipulation subroutines to perform these low level operations. In support of ESTAR the operations performed by NSTAR are

- Extract horizontal (epipolar) lines of pixel data from both images,

- Extract radial lines around a point in a single image,
- Process linear arrays of pixel data using the Queen Victoria algorithm to get lists of features,
- Find region extents using radial lines and the Queen Victoria algorithm,
- Characterize an arbitrary size region using a 16 X 16 bit map representation, and
- Characterize a region edge by generating a 16 X 16 unscaled bit map of the region along the edge.

These operations are combined to allow ESTAR to command the following

- Obtain a new image pair,
- Get the features along epipolar slices,
- Characterize a region by size, shape bit map, and grey levels,
- Characterize one side of an edge as a bit map, and
- Terminate processing

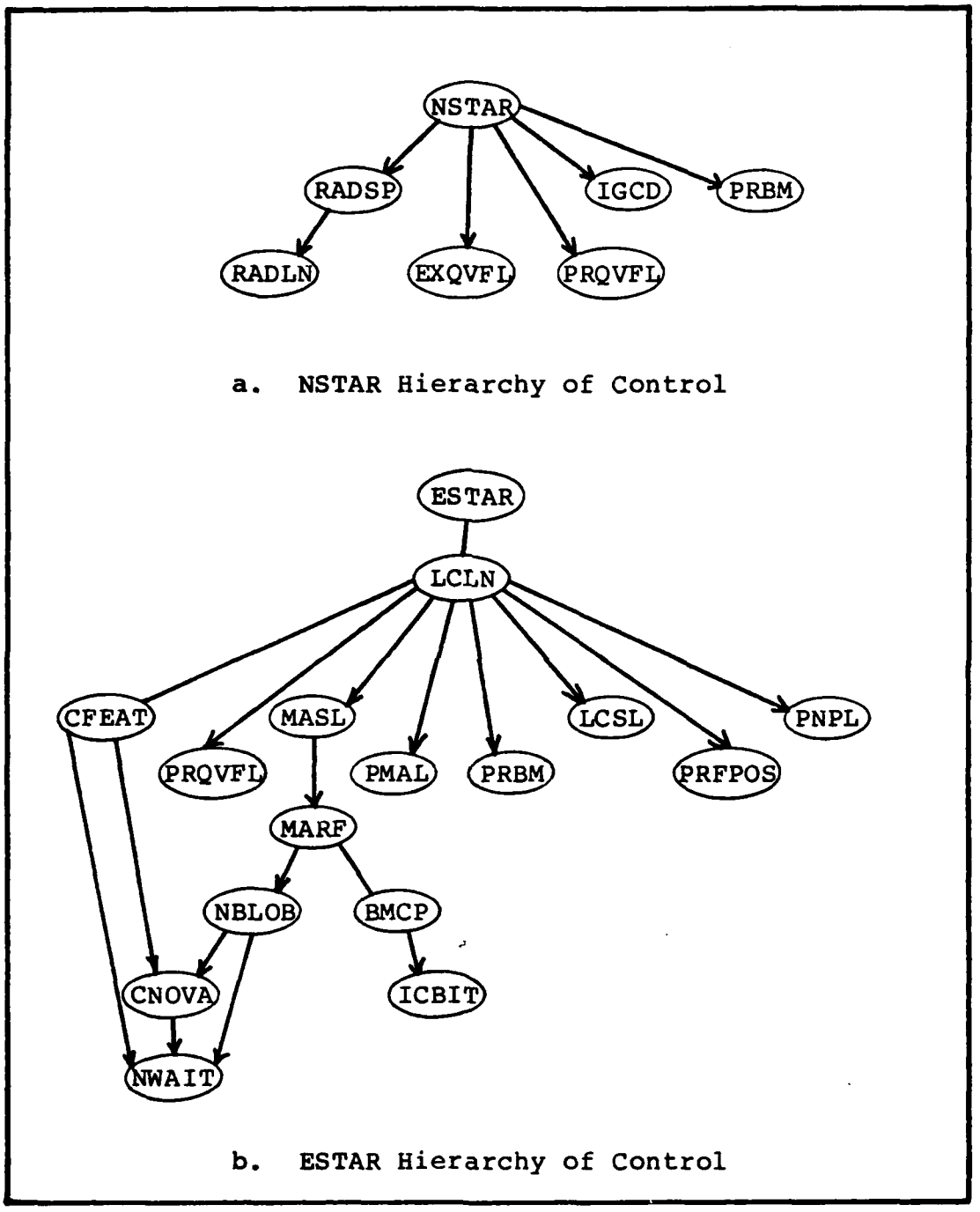
To get any intermediate printouts of data generated by NSTAR the corresponding DEBUG flags must be turned on. A selection of "-1" turns on all the flags and enables all the printouts. The NSTAR activities commanded by ESTAR are displayed on the terminal as NSTAR performs the process. The modules included in NSTAR, and their hierarchy of

control are given in Figure C-2a.

ESTAR. The sequence of operations required to locate features in a single slice across an image is controlled by ESTAR. It causes a single epipolar slice of features to be extracted from the images, then matches the corresponding features from the two images, then generates camera relative location estimates for each of the image features matched. The subroutine LCLN controls the entire sequence, and its input is the vertical location of the slice in the images to be processed. Its output is a file of feature locations and estimated error bounds.

Many of the intermediate results may be printed out by giving ESTAR a set of DEBUG flags. However, some messages are continually displayed during the process to allow the user to follow the progress. The modules included in ESTAR and their hierarchy of control are given in Figure C-2b.

The Collection. Though each of these programs runs separately, as in PRE, or on separate machines, as for NSTAR and ESTAR, all three are necessary to perform the vision tasks. The need for three separate program modules was dictated by the hardware configuration and small available user memory. The combination allows inputting stereo pairs and processing them to feature location lists, but not without some extensive user interaction. The input technique was dictated by the hardware incompatibility



a. NSTAR Hierarchy of Control

b. ESTAR Hierarchy of Control

Fig. C-2 Software Modules and Control

between the OCTEK and the lab's existing stereo pair of cameras.

A discussion of hardware and software usage and requirements follows.

Usage

In preparation, the user must obtain a stereo pair of images, shrink them for use, then save them to their intermediate files using PRE. The user must assure that the camera separation is accurately measured, and that the image axes are parallel. Also special care must be taken to assure the camera characterization is accurate, as described in Appendix A.

Once the shrunk stereo pair is stored in an unpacked form, then NSTAR and ESTAR may be initiated. Since they communicate using "~.DT" named files under the directory "JHDATA" it is good practice to delete all files by such names under the directory before starting.

NSTAR may be initiated by answering with the desired debug flags, often "0"; a code for not saving the current screen, also "0"; and "1" to indicate that the stereo images must be loaded. When the images must be loaded, the system then prompts for the left, then the right image file names. These are commonly stored using the ".BI" extension to distinguish them from the ".VD" packed video files. The differences between these file types are more fully explained in Appendix E. NSTAR then responds that it is

ready for a command file from the Eclipse.

ESTAR may be initiated either before or after NSTAR, as NSTAR merely waits for ESTAR to generate the command file, and ESTAR generates a command file, then waits for NSTAR to generate its response file. Initiating ESTAR requires inputting the camera separation, in whatever units are desired (feet, meters, inches, etc.). These will also be the units of the results. The user must then enter the camera characterization, or the virtual image distance (in pixels). This distance can be calculated using the method in Appendix A, but the Dage camera, with minimal magnification on the zoom lens, the shrunk image gives a distance of about 180 pixels. For an unshrunk image portion it is about 360 pixels. The third entry is the desired combination of DEBUG printout flags. Usually "13" is used as the sum giving the feature list ("1"), the match list ("4"), and the position list ("8"). Also "0" could be used if only the position file is desired, or "-1" could be used if all intermediates including bit maps were desired. The final desired input is the vertical location index. This can be in the range 8 to 112, with 60 as the horizontal center of the images.

ESTAR then writes out the command file initiating processing in NSTAR, and the two programs communicate as necessary until the slices are done. ESTAR then prompts for another slice index after the result file "POSIT.DT" is

written to the directory "JHDATA".

Support Requirements

Support needs for this software can be broken down into several categories. These are hardware configuration, system runtime support software, and system generation and maintenance support software. Each of these categories is discussed separately.

Hardware Configuration. The software is designed to run using a Data General Eclipse, a Data General NOVA, an OCTEK image processor board in the NOVA, and a video camera. A monitor is also useful for viewing the images being processed. The Eclipse and NOVA must have a shared disk drive containing the partition/directory "JHDATA". All communication between the processes on the Eclipse and the NOVA are passed via files under this directory. The NOVA communicates directly with the OCTEK, and controls all direct interaction with the image. This configuration is shown in Figure C-3a.

In Figure C-3b the necessary hardware configuration for software development and maintenance is shown. All the subroutine sources are stored in the shared disk directories "ESTAR", "EPRE", "NSTAR", and "NPRE". These are subdirectories under the partition/directories "EHOLTEN" and "NHOLTEN". The beginning "E" refers to an Eclipse useable directory with links to system routines which run only on

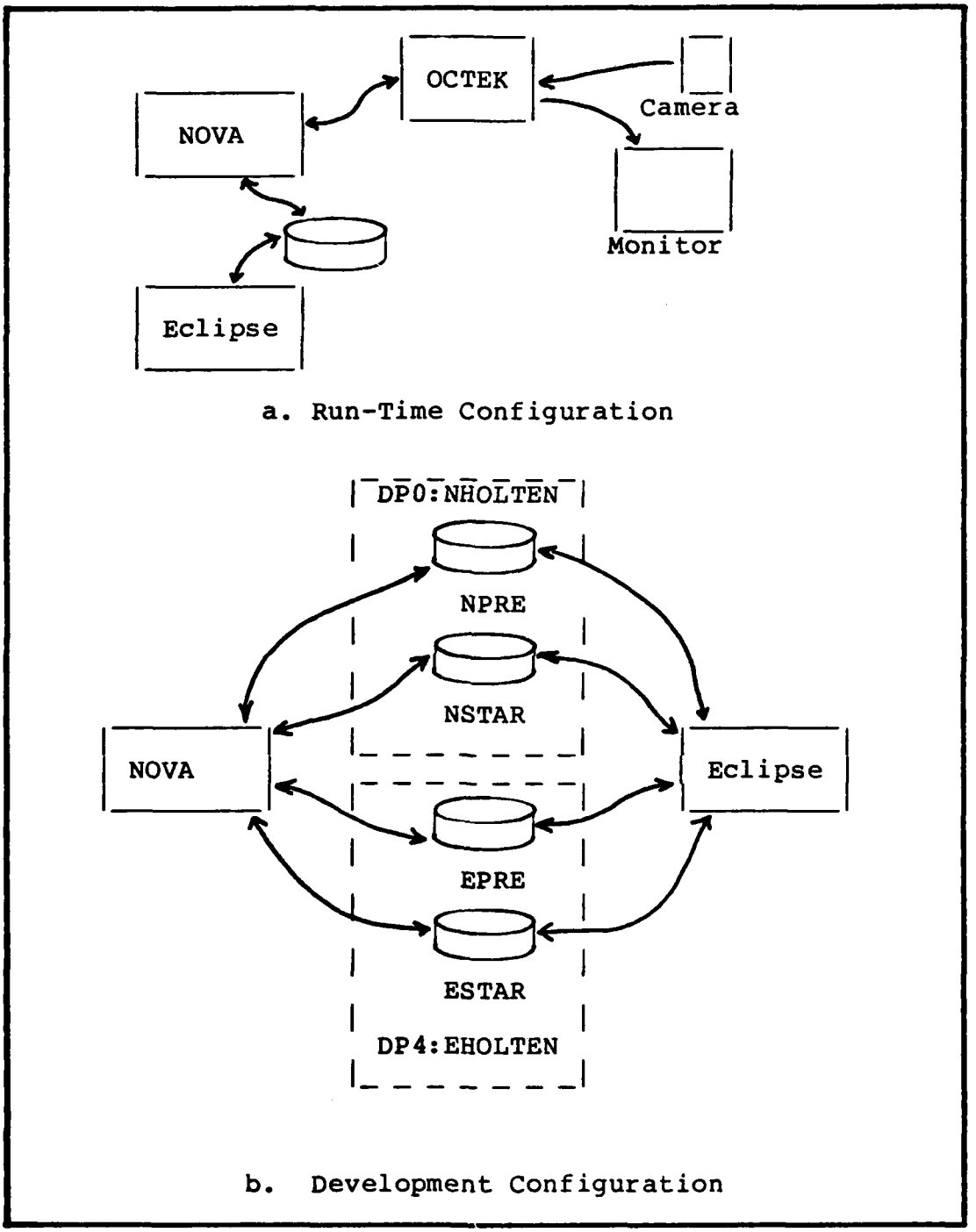


Fig. C-3 Hardware Configuration

the Eclipse, while the beginning "N" refers to NOVA system routine links.

Run-Time Software. From the Eclipse side no runtime support software is needed beyond ESTAR, however the directories "EHOLTEN", "ESTAR", and "JHDATA" must be initialized. From the NOVA, however, not only must "NHOLTEN", "NSTAR", and "JHDATA" be initialized, but NSTAR must have a file or link to the OCTEK directory for access to the file "IACMON.XB" where the binary program modules for the OCTEK processor board resides. The initialization routine of each program which uses the OCTEK copies this file into the OCTEK, then starts its processor executing that code.

Development Software. All source code is stored in both an Eclipse directory and a NOVA directory. As changes are made the source files are backed up to the other system using (from the Eclipse subdirectory "ESTAR")

```
MOVE/V/R  NSTAR  -.FR
```

which moves only the updated source files, and displays a list of those moved.

All the system program links are kept under the partitions "EHOLTEN" and "NHOLTEN", and do not need duplicated under the subdirectories as long as all editing, compiling, and linking is done from those directories.

Suppose LCLN and MASL were to be altered, then the following commands would be used under directory EHOLTEN:

```
EDIT  ESTAR:LCLN.FR
  Edit the text, then quit, saving the file.
EDIT  ESTAR:MASL.FR
  Edit the text, then quit, saving the file.
FORT  ESTAR:(LCLN, MASL)
DIR  ESTAR
MOVE/V/R  NSTAR  -.FR
DIR EHOLTEN
ESTAR:LESTAR
ESTAR:ESTAR
```

The next to last command activates a macro procedure which causes a load module to be built using every relocatable binary module in the directory. The programmer must ensure that no undesired modules have been compiled, and that all the desired ones have been compiled. The procedure assumes that ESTAR is the main module.

Similarly, modules intended to be run on the NOVA may be edited on the Eclipse (since the editor there is much easier to use) then compiled, linked, and run on the NOVA. Suppose EXQVFL and RADSP were to be altered, then the command sequence on the Eclipse under directory EHOLTEN would be

```
EDIT  ESTAR:EXQVFL.FR
  Edit the text, then quit, saving the file.
EDIT  ESTAR:RADSP.FR
  Edit the text, then quit, saving the file.
DIR  ESTAR
MOVE/V/R NSTAR  -.FR
```

On the NOVA the programmer would then continue under

directory NHOLTEN with

```
FORT  NSTAR:(EXQVFL, RADSP)
NSTAR:LNSTAR
NSTAR:NSTAR
```

Note that the references are now to the subdirectory NSTAR rather than ESTAR. Once again the next to last line activates a macro procedure which builds a load module including every relocatable binary in the directory NSTAR. This time, however, the main module is assumed to be NSTAR. This load module is also built allowing external references to be satisfied from the OCTEK library of support routines. Again, the programmer must insure that an undesired relocatable binary is not present, and that all the desired modules have their relocatable binary present.

APPENDIX D:

Sample Results

Sample Results

A sample output is given here. The sample shows the feature list, some bit maps to be compared, the resulting match list, and the resulting position list.

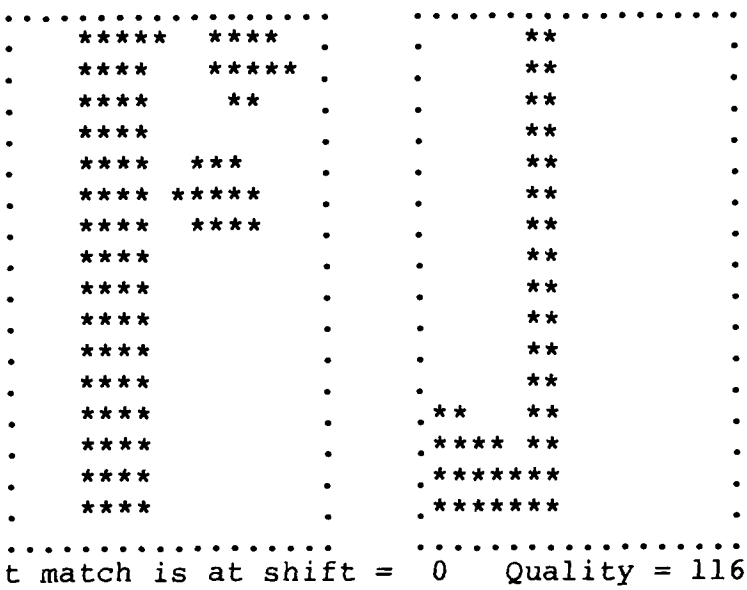
1-Dimensional Slice Feature List
There are 12 features in the slice.

INDEX	START	STOP	GRYL	GRYR
1	4	5	15	9
2	13	14	7	10
3	14	16	10	7
4	17	19	7	15
5	19	25	15	1
6	29	32	1	2
7	34	36	2	5
8	39	42	5	1
9	42	45	1	4
10	46	49	4	4
11	55	61	3	15
12	73	77	15	7

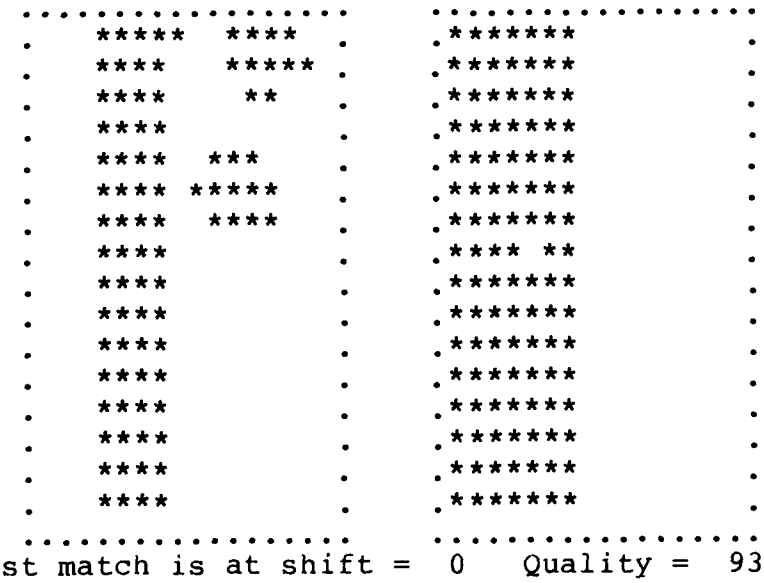
1-Dimensional Slice Feature List
There are 11 features in the slice.

INDEX	START	STOP	GRYL	GRYR
1	6	8	9	7
2	12	14	7	15
3	15	17	15	5
4	21	23	5	1
5	30	33	1	5
6	36	38	5	2
7	39	42	2	4
8	52	56	3	15
9	67	70	15	7
10	110	112	7	4
11	113	115	4	6

```
Comparing left sides of   1  and   3
Comparing bit maps.
```

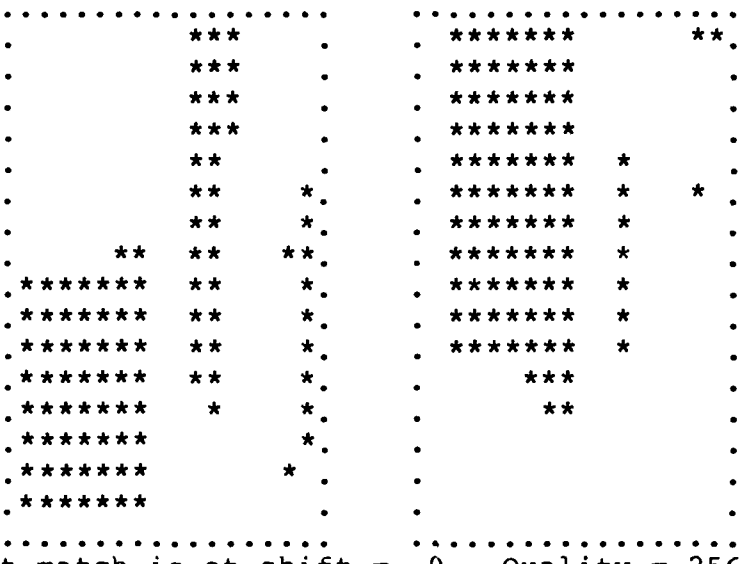


```
Best match is at shift =  0   Quality = 116

Comparing left sides of   1  and   9
Comparing bit maps.
```

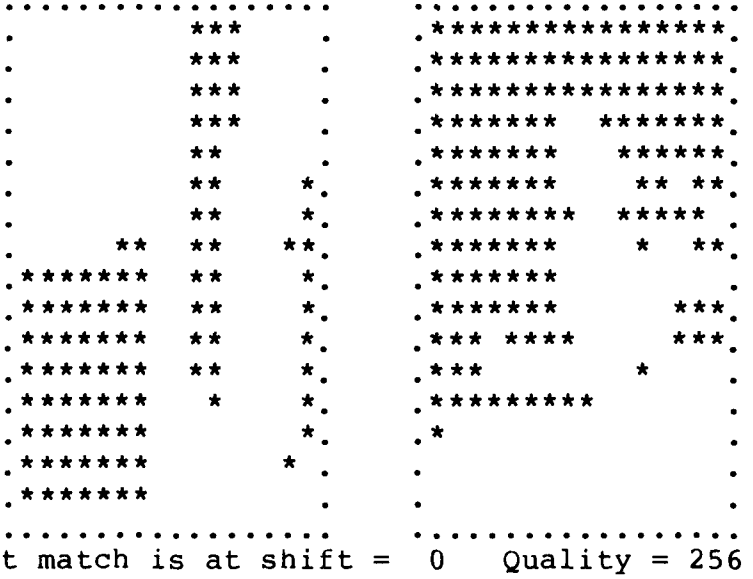


```
Best match is at shift =  0   Quality =  93

Match lists generated.
  INDX=   1   INUML=   0
  KNDX=  51   INUMR=   0
```

```
Comparing left sides of   2  and   2
Comparing bit maps.
```

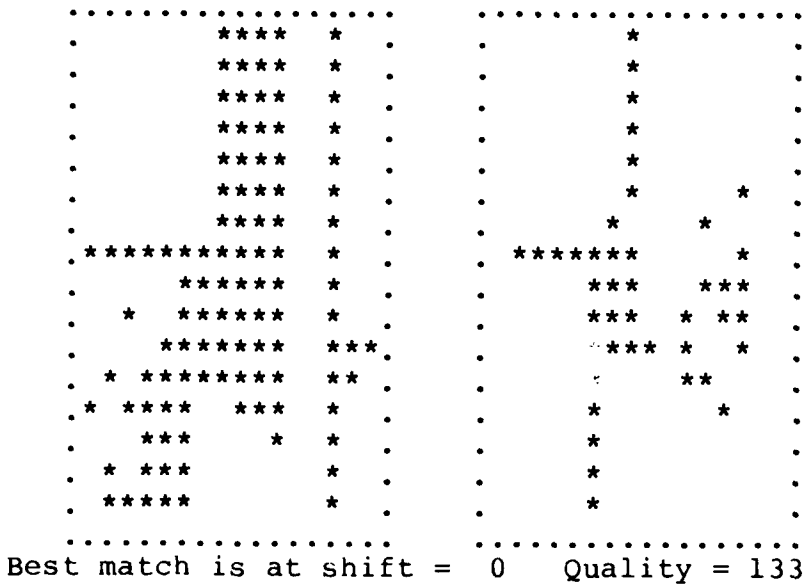


```
Best match is at shift =  0   Quality = 256

Comparing left sides of   2  and  10
Comparing bit maps.
```



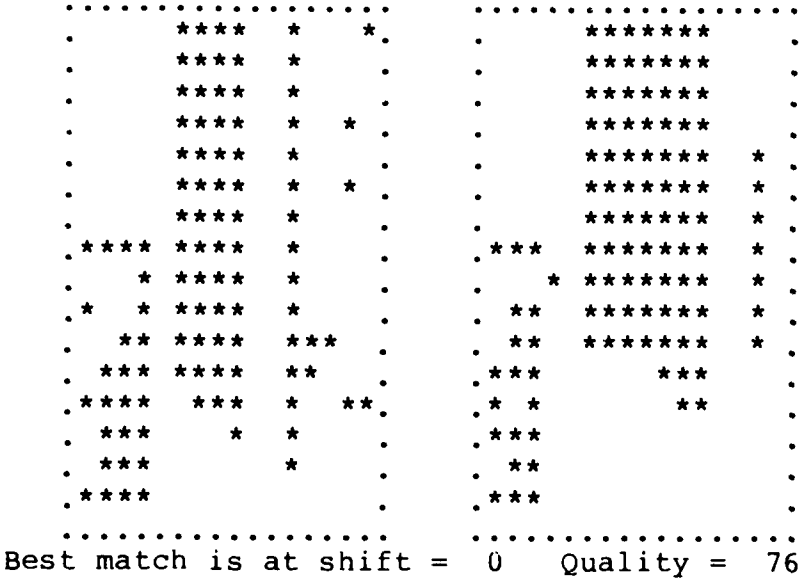
```
Best match is at shift =  0   Quality = 256

Match lists generated.
  INDX=    2    INUML=    1
  KNDX=   52    INUMR=    0
```

Comparing left sides of 3 and 1
Comparing bit maps.



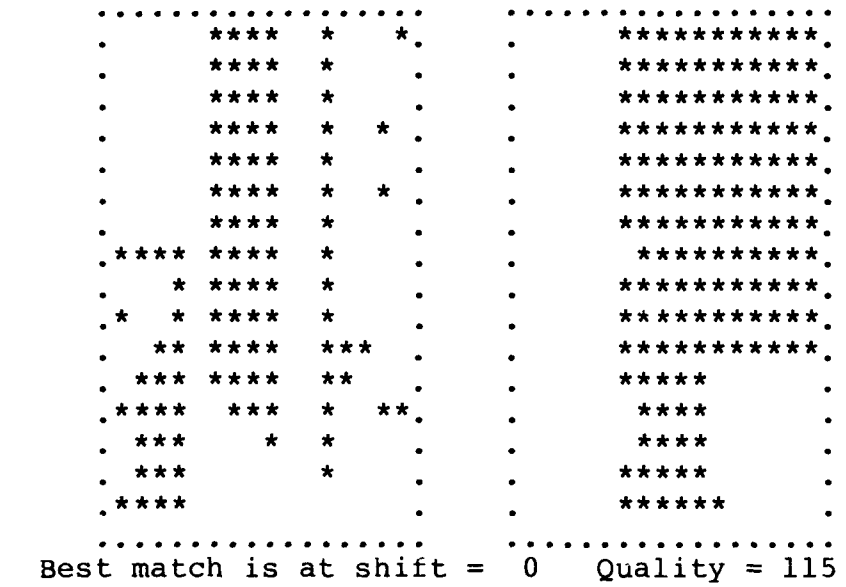
Best match is at shift = 0 Quality = 133

Comparing right sides of 53 and 51
Comparing bit maps.

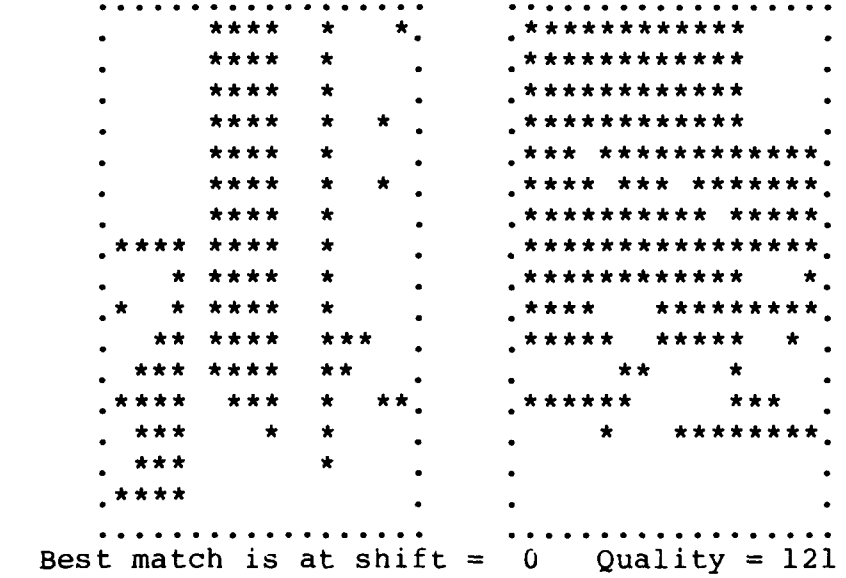


Best match is at shift = 0 Quality = 76

```
Comparing right sides of  53  and  59
Comparing bit maps.
```



```
Best match is at shift =  0   Quality = 115

Comparing right sides of  53  and  61
Comparing bit maps.
```



```
Best match is at shift =  0   Quality = 121

Match lists generated.
  INDX=   3   INUML=   1
  KNDX=  53   INUMR=   1
```

Feature matches, left and right edges. Number of features= 12

IM#1

INDX # OF FEATS	ID	1	2	3	4	5	6	7	8	9	10
1	INDX	---	*** NO MATCHES ****								
	QUAL	---									
0	LOC1	---									
	LOC2	---									
51	INDX	---	*** NO MATCHES ****								
	QUAL	---									
0	LOC1	---									
	LOC2	---									
2	INDX	2									
	QUAL	256									
1	LOC1	13									
	LOC2	12									
52	INDX	---	*** NO MATCHES ****								
	QUAL	---									
0	LOC1	---									
	LOC2	---									
3	INDX	1									
	QUAL	133									
1	LOC1	14									
	LOC2	6									
53	INDX	51									
	QUAL	76									
1	LOC1	16									
	LOC2	8									
4	INDX	2									
	QUAL	127									
1	LOC1	17									
	LOC2	12									
54	INDX	52									
	QUAL	106									
1	LOC1	19									
	LOC2	14									
5	INDX	3									
	QUAL	137									
1	LOC1	19									
	LOC2	15									

```
 55 INDX   54
    QUAL   93
  1 LOC1   25
    LOC2   22
-------------------------------------------------------
  6 INDX   ---   ***  NO MATCHES ****
    QUAL   ---
  0 LOC1   ---
    LOC2   ---

 56 INDX   54
    QUAL  154
  1 LOC1   32
    LOC2   23
-------------------------------------------------------
  7 INDX    5
    QUAL   70
  1 LOC1   34
    LOC2   29

 57 INDX   55   53
    QUAL   68  128
  2 LOC1   36   36
    LOC2   33   17
-------------------------------------------------------
  8 INDX    6    4
    QUAL   58  122
  2 LOC1   39   39
    LOC2   36   22

 58 INDX   56   54
    QUAL  117  134
  2 LOC1   42   42
    LOC2   38   23
-------------------------------------------------------
  9 INDX    7    5
    QUAL  103  164
  2 LOC1   42   42
    LOC2   39   30

 59 INDX   57   55   53
    QUAL  127  133  144
  3 LOC1   45   45   45
    LOC2   42   33   17
-------------------------------------------------------
 10 INDX    4    6
    QUAL  130  140
  2 LOC1   46   46
    LOC2   21   36

 60 INDX   57   55   53
    QUAL  155  159  173
```

3	LOC1	49	49	49
	LOC2	42	34	17
11	INDX	8	7	
	QUAL	27	182	
2	LOC1	55	55	
	LOC2	52	39	
61	INDX	58	52	
	QUAL	1	256	
2	LOC1	61	61	
	LOC2	56	14	
12	INDX	9	3	
	QUAL	13	112	
2	LOC1	73	73	
	LOC2	67	15	
62	INDX	59	51	
	QUAL	19	82	
2	LOC1	77	77	
	LOC2	71	9	

Feature 3-D positions. NP= 12 dv=180.00 ds= 0.750

INDX	dp	dx	dy	ep	ex	ey
1	No data point.					
51	No data point.					
2	No data point.					
52	No data point.					
3	16.875	-4.312	0.000	2.411	0.670	0.054
53	16.875	-4.125	0.000	2.411	0.643	0.054
4	27.000	-6.450	0.000	6.750	1.706	0.094
54	27.000	-6.150	0.000	6.750	1.631	0.094
5	33.750	-7.687	0.000	11.250	2.687	0.125
55	45.000	-8.750	0.000	22.500	4.562	0.187
6	No data point.					
56	15.000	-2.333	0.000	1.875	0.339	0.047
7	27.000	-3.900	0.000	6.750	1.069	0.094
57	45.000	-6.000	0.000	22.500	3.187	0.187
8	45.000	-5.250	0.000	22.500	2.812	0.187
58	33.750	-3.375	0.000	11.250	1.250	0.125
9	45.000	-4.500	0.000	22.500	2.437	0.187
59	45.000	-3.750	0.000	22.500	2.062	0.187
10	5.400	-0.420	0.000	0.225	0.033	0.016
60	19.286	-1.179	0.000	3.214	0.259	0.062
11	45.000	-1.250	0.000	22.500	0.812	0.187
61	27.000	0.150	0.000	6.750	0.131	0.094
12	22.500	1.625	0.000	4.500	0.400	0.075
62	22.500	2.125	0.000	4.500	0.500	0.075

APPENDIX E:

Data Structures

Data Structures

There are two types of data structures which are important here. The first are data structures for each of the files, used in passing commands or data. The second are the arrays used for conveniently passing blocks of useful information to subroutines. These are both defined here.

File Data Structures

There are seven file types manipulated by these programs. These are each described below.

File Name: "-.VD"

Module Usage: PRE reads these files from the disk to be displayed on the OCTEK, and writes them to the disk from the OCTEK. They are Signal Processing Lab standard video files.

Data Form: They contain a 256 X 240 array of pixel values, 4 bits each, packed four to a word.

File Name: "-.BI"

Module Usage: PRE reads these to the OCTEK, and writes them from the OCTEK. The images are generally created by using the PRE option to shrink larger images. This is the form required to input images to NSTAR.

Data Form: These are 128 X 120 arrays of pixels, 4 bits each, one per integer word.

File Name: "NCONT.DT"

Module Usage: This is a command file written by ESTAR and read by NSTAR. It carries the command and data necessary of the next operation required of NSTAR by ESTAR. It is deleted by NSTAR after it is read.

Data Form: Binary, 12 bytes.

Command
- CODE
- IMAGE
- X-center
- Y-center
- Max number
- IDN

where

CODE=
- -1 Terminate
- 1 Characterize current region
- 2 Get a slice of features
- 3 Get a new image set
- 4 Get edge bit map

IMAGE=
- 1 Left image
- 2 Right image

IDN=
- -1 Left side of edge
- 1 Right side of edge

File Name: "SLICE.DT"

Module Usage: This is a data file written by NSTAR and read by ESTAR. After it is read, ESTAR deletes it. It contains the list of features from the horizontal slice NSTAR was directed to process.

Data Form: Binary, 512 bytes.

Feature Buffer
- Dummy (7 times)
- Number of features
- Dummy (2 times)
- Edge feature description (50 times)
- Dummy (46 times)

Edge feature description
- Starting pixel index
- Ending pixel index
- Start grey level
- End grey level

File Name: "WBMAP.DT"

Module Usage: This is a data file written by NSTAR and read by ESTAR. After it is read, ESTAR deletes it. It contains an edge bit map characterization requested by ESTAR and generated by NSTAR.

Data Form: Binary, 34 bytes.

Edge bit map { 1 (constant)
Column bit map (16 times)

Column bit map { Pixel map value (bit) (16 times)

File Name: "BLOCK.DT"

Module Usage: This is a data file written by NSTAR and read by ESTAR. After it is read ESTAR deletes it. It contains a region characterization, including size, location, shape, and grey level extremes. It is generated by NSTAR in response to a command. However, at this time ESTAR generates no such command.

Data Form: Binary, 54 bytes.

Region Characterization {
- ID number
- X scale factor (real)
- Y scale factor (real)
- Center Characterization
- Scaled bit map column (16-times)

Center Characterization {
- X-start index
- X-end index
- Y-start index
- Y-end index
- Minimum grey level
- Maximum grey level

Scaled bit map column { Scaled row entry map (bit) (16 times)

File Name: "POSIT.DT"

Module Usage: This is a data file written by ESTAR. It contains the position and position error estimates for each of the matched features in the current slices. At this time no modules reads this file.

Data Form: Binary, 2560 bytes.

Position File
- Number of points
- Position Entry (100 times)

Position Entry
- Range estimate (real)
- Horizontal offset estimate (real)
- Vertical offset estimate (real)
- Range error estimate (real)
- Horizontal offset error estimate (real)
- Vertical offset error estimate (real)

Internal Data Structures

The internal data structures are used to pass several related data items into and out of subroutines. There are no other uses of data structuring in these programs. The file data structures are in this category, but have already been defined, so only strictly internal structures will be ocnsidered here.

NSTAR Structures--

Variable Name: CA1, CA2

Module Usage: These are direction cosines generated in NSTAR and used to compute radial line locations for various line slopes in RADSP. Each entry is the cosine of the corresponding direction for a single quadrant. Different combinations of the two arrays with different signs determine the quadrant of the star pattern and actual line direction.

Variable Name: ICT

Module Usage: This is the OCTEK device control table. NSTAR must pass it to each OCTEK routine called, but otherwise must not alter it.

Variable Name: IDEBF (NSTAR)

Module Usage: Each bit in this is an activation flag for a corresponding printout of intermediate results.

Variable Name: IDIR

Module Usage: This is used to pass radial line information from NSTAR to RADSP and on to RADLN.

Data Form:

IDIR(I) I=1 Index of last direction processed
=2 X-center index
=3 Y-center index
=4 X-end of line index
=5 Y-end of line index
=6 Number of points in line
=7 Image ID flag, 1=left, 2=right

ESTAR Structures--

Variable Name: IDEBF (ESTAR)

Module Usage: Each bit in this is an activation flag for a corresponding printout of intermediate results.

Variable Name: IMATCH

Module Usage: This is the array of match alternatives listed for each feature in image #1, the reference image. It is used by LCLN, MASL, PMAL, and LCSL to pass the information around.

Data Form:

Match Array { Match list (100 times)

Match List { Match entry (10 times)

Match Entry {
- Image #2 feature index
- Match Quality measure
- Actual pixel location in image #1
- Actual pixel location in image #2

Variable Name: LMATCH

Module Usage: This is an array of data pertaining to the current edge feature side to be compared.

Data Form:

LMATCH(I)

I =	1	Match list index for current feature
	2	Best quality value found for match
	3	X-location index in image #1
	4	X-location index in image #2
	5	Maximum allowable map shift in image #1
	6	Maximum allowable map shift in image #2
	7	Edge side flag, -1=left, +1=right
	8	The vertical location index.

This list is not all-inclusive, but most data structures of any complexity are used in file structures also. These cover all the significant structures of the system.

APPENDIX F:

Program Listings

Program Listings

The Program listings are arranged according to program modules they support. First is PRE, then NSTAR and its support, then ESTAR and its support. The sequence of listings is as follows:

PRE routines--
- PRE.FR

NSTAR routines--
- NSTAR.FR
- EXQVFL.FR, the Queen Victoria algorithm
- RADLN.FR, extracts radial lines
- RADSP.FR, extracts the star pattern from the image
- IGCD.FR, computes the greatest common divisor of two integers
- PRQVFL.FR, prints out the feature lists
- PRBM.FR, prints out a bit map

ESTAR routines--
- ESTAR.FR
- LCLN.FR, locates the features on an epipolar line in 3-space
- CFEAT.FR, creates lists of features along the epipolar line
- PRQVFL.FR, listed under NSTAR
- MASL.FR, creates lists of matches for each feature
- MARF.FR, generates a measure of match quality between two features
- BMCP.FR, compares to edge bit maps
- ICBIT.FR, counts the number of bits set in an integer word
- PMAL.FR, prints the match lists
- PRBM.FR, listed under NSTAR
- LCSL.FR, locates feature match pairs in 3-space
- PRFPOS.FR, prints the list of feature locations
- PNPL.FR, saves the position list on a disk file

```
C
C  Preprocessor for NSTAR.
C       Uses the OCTEK 2000 IMAGE ANALYZER CARD (IAC)
C
C      Written 10 Oct 85 by JAMES R. HOLTEN III
C         Uses original code from OCTEK demo program
C      with substantial revisions for the specific application.
C
        INTEGER ICT(120)
        INTEGER IBUF(1026),IBUF1(513),IBUF2(513)
        INTEGER NFILE(20),VFILE(256)
        COMMON /BUFFER/IBUF
        EQUIVALENCE (IBUF(1),IBUF1(1)),(IBUF(514),IBUF2(1))

        ICMAX=11

        TYPE "<33>E"
        TYPE
        TYPE "Robot Vision Image Preprocessor"
        TYPE "<33>j"

        CALL SINTRO (ICT,63K,IER)
        IF(IER.EQ.1) GO TO 1
        TYPE 'INTRO ECODE:',IER
        STOP "UNABLE TO INITIALIZE"

1       CONTINUE
        CALL OPEN (3,"IACMON.XB",2,IER)
        IF(IER.NE.1) TYPE "WARNING: Unable to access IACMON.XB"
        IF(IER.EQ.1) CALL LXB (ICT,3)

        CALL MPRUN (ICT,0,1)
        CALL GREYSCALE (ICT,1)
        CALL GREYSCALE (ICT,2)

C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C        Menu of Commands
C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
5       CONTINUE
        TYPE "<33>k<33>J"
        TYPE "Image Preprocessor Command List"
        TYPE
        TYPE "Choose a function:"
        TYPE " 1  Video input (Camera) control"
        TYPE " 2  Create condensed image for NSTAR"
        TYPE " 3  Cartoon the image using horizontal and verical QV"
        TYPE " 4  Output the 128x120 condensed image to disk file"
        TYPE " 5  Input a 128x120 condensed image from disk file"
        TYPE " 6  Dummied out"
        TYPE " 7  Dummied out"
        TYPE " 8  Dummied out"
        TYPE " 9  Dummied out"
```

```
      TYPE "10  Input image from disk file to OCTEK"
      TYPE "11  Output image from OCTEK to disk file"
      TYPE "-1  To exit the program"
C
C  THIS IS THE TOP OF THE MAIN INTERACTIVE LOOP
C

7     ACCEPT "Enter function number (0 for help): ",IC
      IF (IC.LT.0) GO TO 999
      IF (IC.EQ.0) GO TO 5
      IF (IC.GT.ICMAX) GO TO 5
      TYPE "<33>k<33>J"

      GOTO (10,20,30,40,50,60,70,80,90,100,110),IC


C ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     Command 1      - Video Input (Camera) Port Control
C ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
10    CONTINUE
      TYPE
      TYPE "Video Input Control"
      TYPE "Camera on (1), Camera off (2)"
11    CONTINUE
      ACCEPT "Video Input command (-1 to exit): ",IC
      IF (IC.EQ.2) GOTO 14
      IF (IC .LT. 0) GOTO 7
      IF (IC .NE. 1) GOTO 10

C
C Camera on performs the following sequence:
C  1) Select interlace display (required for any camera input)
C  2) Select external clock
C  3) Turn on camera
C

      CALL INTLACE (ICT,1)
      CALL SYNC (ICT,1,ILOCK)
      IF (ILOCK.EQ.0) GO TO 19
      CALL VON (ICT,0)
      GO TO 11

C
C After the camera is turned off, the internal clock is
C  selected and the display is made non-interlaced
C
14    CONTINUE
      CALL VOFF (ICT)
15    CONTINUE
      CALL SYNC (ICT,0,ILOCK)
      CALL INTLACE (ICT,0)
      GO TO 7
```

```
19      CONTINUE
        TYPE "** Camera Input is missing (unable to find video sync)"
        GO TO 15

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 2      Condense the 256x240 image to 128x120
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
20      CONTINUE
        TYPE
        TYPE "Condense the current image to 128x120"
        INX=96
        INXL=128
        INYL=120
        INYL1=(INYL/2)-1
        IX=32
        IXL=256
C
C         Loop through the image lines.
        DO 25 INY=0,INYL1
        IY=INY+INY
        IYB=120+IY
        IYT=118-IY
C
C         Input two sets of two lines each, then blank them.
        CALL RVBLK(ICT,IBUF1,IX,IXL,IYB,2)
        CALL RVBLK(ICT,IBUF2,IX,IXL,IYT,2)
        CALL PXFILL(ICT,15,0,320,IYB,2)
        CALL PXFILL(ICT,15,0,320,IYT,2)
C
C         Process the pixels to reduce the image size.
        DO 21 ILOC=0,127
        ILN=ILOC+2
        IL1=ILOC+ILOC+2
        IL11=IL1+1
        IL2=IL1+256
        IL21=IL2+1
C
C         Compute the average of four neighboring points.
        IVAL=IBUF1(IL1)+IBUF1(IL11)+IBUF1(IL2)+IBUF1(IL21)+2
        IBUF1(ILN)=IVAL/4

        IVAL=IBUF2(IL1)+IBUF2(IL11)+IBUF2(IL2)+IBUF2(IL21)+2
        IBUF2(ILN)=IVAL/4

21      CONTINUE

C
C         Write out the two lines.
        INYB=120+INY
        INYT=119-INY
        CALL WVBLK(ICT,IBUF1,INX,INXL,INYB,1)
```

```
      CALL WVBLK(ICT,IBUF2,INX,INXL,INYT,1)
25    CONTINUE

C
C       Put a black box around it.
      IX1=INX-1
      IX2=INX+INXL
      INY=60
      IY1=INY-1
      IY2=INY+INYL
      CALL GVECT(ICT,IX1,IY1,IX1,IY2,0)
      CALL GVECT(ICT,IX1,IY1,IX2,IY1,0)
      CALL GVECT(ICT,IX1,IY2,IX2,IY2,0)
      CALL GVECT(ICT,IX2,IY1,IX2,IY2,0)

      GOTO 7

C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     Command 3        Cartoon the 128x120 condensed image
C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
30    CONTINUE
      TYPE "Cartoon the current image"
      TYPE "   using passes of horizontal and vertical QV processing."



      GOTO 7

C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C     Command 4       Output the current 128x120 image to disk file.
C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
40    CONTINUE
      TYPE "Saving the current 128x120 image to disk."
      TYPE "  Input the desired file name."
      READ(11,1000) NFILE(1)
      CALL OPEN(1,NFILE,3,IER)
      IF (IER .EQ. 1) GOTO 41
        TYPE "  Open error on file.  IER=",IER
        GOTO 49
41    CONTINUE
      IX=96
      IXL=128
      IY=60
      IYL=120
      IYBN=14
      IYI=8

      DO 45 IYP=0,IYBN
      IYB=IYP*4
      IYPOS=8*IYP+IY
      CALL RVBLK(ICT,IBUF,IX,IXL,IYPOS,IYI)
      CALL WRBLK(1,IYB,IBUF(2),4,IER)
```

```
45      CONTINUE

49      CONTINUE
        CALL CLOSE(1,IER)

        GOTO 7

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 5       Input a 128x120 image from disk file.
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
50      CONTINUE
        TYPE "Loading a 128x120 image from disk."
        TYPE "  Input the desired file name."
        READ(11,1000) NFILE(1)
        CALL OPEN(1,NFILE,3,IER)
        IF (IER .EQ. 1) GOTO 51
          TYPE "  Open error on file.  IER=",IER
          GOTO 59
51      CONTINUE
        IX=96
        IXL=128
        IY=60
        IYL=120
        IYBN=14
        IYI=8

        DO 55 IYP=0,IYBN
        IYB=IYP*4
        CALL RDBLK(1,IYB,IBUF(2),4,IER)
        IYPOS=8*IYP+IY
        CALL WVBLK(ICT,IBUF,IX,IXL,IYPOS,IYI)
55      CONTINUE

59      CONTINUE
        CALL CLOSE(1,IER)

        GOTO 7

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 6       DUMMIED OUT
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
60      CONTINUE
        GOTO 7

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 7       DUMMIED OUT
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
70      CONTINUE
        GOTO 7

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 8       DUMMIED OUT
```

```
C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
80      CONTINUE
        GOTO 7

C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C       Command 9       DUMMIED OUT
C cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
90      CONTINUE
        GO TO 7

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C       Command 10      Input from disk to OCTEK
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
100     CONTINUE
        TYPE
        TYPE "  INPUT FROM DISK"
        TYPE
        TYPE " What file name?"
        READ(11,1000) NFILE(1)
        CALL OPEN(1,NFILE,1,IER)
        IF (IER .EQ. 1) GOTO 101
          TYPE " File open error--",IER
          GOTO 106
101     CONTINUE
        CALL PXFILL(ICT,15,0,320,0,240)
        IXP=32
        IXL=256
        IYP=0
        IYL=4
C
C       Read the 64 blocks of the picture, unpacking and saving each
C        The last four blocks are ignored.
C
        DO 104 I=0,59
        CALL RDBLK(1,I,VFILE,1,IER)
        IF (IER .EQ. 1) GOTO 102
          TYPE "Read error--",IER
          GOTO 106
102     CONTINUE
C
C       Repack for output: from (xxxx) to (000x:000x:000x:000x)
C
        DO 103 J=1,256
        K=4*(J-1)+2
        IBUF(K)=ISHFT(VFILE(J),-12) .AND. 15
        IBUF(K+1)=ISHFT(VFILE(J),-8) .AND. 15
        IBUF(K+2)=ISHFT(VFILE(J),-4) .AND. 15
        IBUF(K+3)=VFILE(J) .AND. 15
103     CONTINUE
C
C       Output to OCTEK 2000
C
```

```
        CALL WVBLK(ICT,IBUF,IXP,IXL,IYP,IYL)
        IYP=IYP+IYL
104     CONTINUE
        TYPE "FILE DISPLAYED"
106        CONTINUE
        CALL CLOSE(1,IER)
        IF (IER .EQ. 1) GOTO 108
           TYPE " Error closing file--",IE
108     CONTINUE

        GOTO 7

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Command 11       Output from OCTEK to disk
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
110     CONTINUE
        TYPE
        TYPE "  OUTPUT TO DISK"
        TYPE
        TYPE "What file name? "
        READ(11,1000) NFILE(1)
        CALL OPEN(2,NFILE,3,IER)
        IF (IER .EQ. 1) GOTO 111
          TYPE "Open error on output file--",IER
          GOTO 116
111     CONTINUE
        IXP=32
        IXL=256
        IYP=0
        IYL=4
C        For the 64 blocks of packed video
C
C       The OCTEK has only 240 lines, so the last four blocks
C         will be zeroed out.
C
        DO 115 I=0,63
C
C       Input from the OCTEK 2000
C
        CALL RVBLK(ICT,IBUF,IXP,IXL,IYP,IYL)
        IYP=IYP+IYL
C
C       Repack the data from (000x:000x:000x:000x) to (xxxx)
C
        DO 113 J=1,256
        IF (I .LE. 59)  GOTO 112
          VFILE(J)=0
          GOTO 113
112     CONTINUE
        K=4*(J-1)+2
        ITEMP1=ISHFT(IBUF(K) .AND. 15,12)
        ITEMP2=ISHFT(IBUF(K+1) .AND. 15,8)
```

```
        ITEMP3=ISHFT(IBUF(K+2) .AND. 15,4)
        ITEMP4=IBUF(K+3) .AND. 15
        VFILE(J)=ITEMP1 .OR. ITEMP2 .OR. ITEMP3 .OR. ITEMP4
113     CONTINUE
C
C       Output to the disk file
C
        CALL WRBLK(2,I,VFILE,1,IER)
        IF (IER .EQ. 1) GOTO 115
          TYPE "Write error--",IER
          GOTO 116
115     CONTINUE
        TYPE "THE FILE IS SAVED."
116     CONTINUE
        CALL CLOSE(2,IER)
        IF (IER .EQ. 1) GOTO 118
           TYPE "Error closing file--",IER
118     CONTINUE

        GOTO 7



999     CONTINUE
        CALL OREMOVE (ICT)
1000    FORMAT(S40)
        END
```

```
C
C       NSTAR.FR
C
C         This is the NOVA main program to control the placement of
C       star patterns on the image, and to retrieve the data along
C       the radial lines for later processing on the ECLIPSE.
C
C       DATA STRUCTURES USED
C               INOVAC          Array of NOVA command parameters.
C                       1       Command code as follows:
C                                -1  Terminate
C                                 1   Apply Star to current location and
C                                     bit map the current Block
C                                 2   Get slice feature list
C                                 3   Get new image pair
C                       2       Image selection flag
C                                 1   Left image
C                                 2   Right image
C                       3       X center index
C                       4       Y center index
C                       5       Max ray index
C                       6       Blob ID number
C               IDIR            Array of pixel slice header data
C                       1       direction index (-1 for epipolar slice)
C                       2       X-start coordinate
C                       3       Y-start coordinate
C                       4       X-end coordinate
C                       5       Y-end coordinate
C                       6       Number of points actually stored
C                       7       Image ID (1-left, 2-right)
C               ICNTR           Star center blob characterization
C                       1       leftmost X location
C                       2       rightmost X location
C                       3       uppermost Y location
C                       4       lowermost Y location
C                       5       minimum grey value
C                       6       maximum grey value
C
C
C       ROUTINES USED:
C               System calls (from FORT.LB)
C                       STAT, OPEN, READ BINARY, WRITE BINARY,CLOSE,
C                       RENAM, DELETE
C               OCTEK library calls (from IACF4.LB)
C                       SINTRO, HCTAB, XHAIR, INTLACE, SYNC, VON, VOFF,
C                       RVBLK, OREMOVE
C               RADSP   Retrieves and stores to a file one quadrant of
the
C                       pattern of radial lines.
C
C       HIERARCHY OF CONTROL:
C               NSTAR
```

```
C                IGCD (function)
C                RADSP
C                  RADLN
C                  EXQVFL
C                    ISGN
C                  PRQVFL
C
C
      COMMON /FBUFF/ IFBUF,IDAT
      COMMON /IMAGE/ IXOFF,IXMAX,IYOFF,IYMAX,IMF
      COMMON /DEBUG/ IDEBF
      COMMON /BLOB/ IBLOB
      DIMENSION IBLOB(27)
      DIMENSION IFEAT(50,4),IFBUF(256),ICNTR(6)
      DIMENSION ISTAT(20),ICT(120)
      DIMENSION IDIR(7),IBMAP(16),ICNT(16),ISUM(16)
      DIMENSION CA1(4),CA2(4)
      DIMENSION IMFIL(40),IDAT(1025)
      EQUIVALENCE (IFBUF(1),IDIR(1)),(IFBUF(8),NFEAT),
     A   (IFBUF(9),IFIRST),(IFBUF(10),ILAST),(IFBUF(11),IFEAT(1,1))
      EQUIVALENCE (IBLOB(1),IDN),(IBLOB(2),SFX),(IBLOB(4),SFY),
     A   (IBLOB(6),ICNTR(1)),(IBLOB(12),IBMAP(1))

C
C       This is the Least Common Multiple implicit function
C      definition, using the greatest common divisor function.
      ILCM(IX,IY)=(IX*IY)/IGCD(IX,IY)

C
C       Set a few constants.
      ITHRB=80
      PI=3.1415926
      PID8=PI/8.0
      SQRT2=SQRT(2.0)

      IHMIN=0
      IHMAX=127
      IVMIN=0
      IVMAX=119
C
C       Initialize the radial angles to be used in each quadrant.
      CA1(1)=1
      CA1(2)=COS(PID8)
      CA1(3)=COS(PID8*2.0)
      CA1(4)=COS(PID8*3.0)
      CA2(1)=0
      CA2(2)=CA1(4)
      CA2(3)=CA1(3)
      CA2(4)=CA1(2)

      TYPE "<33>E"
      TYPE "  NOVA half of stereo vision system"
```

```
      TYPE
      TYPE "<33>j"
      TYPE " The debug print options are:"
      TYPE "     1--RADSP direction line values."
      TYPE "     2--Feature lists."
      TYPE "     4--Center blob characteristics."
      TYPE "     8--BLOB bit map."
      TYPE "    16--Edge bit map."
      TYPE
      TYPE "Add up the desired flag values and enter the number."
      ACCEPT "  Input the debug flags. (=0 for no flags.) ",IDEBF
C
C       Initialize the OCTEK as a system device.
      CALL SINTRO(ICT,63K,IER)
      IF (IER .EQ. 1) GOTO 90
      TYPE "  <7> Unable to init the OCTEK. IER=",IER
      STOP

90    CONTINUE
      CALL GREYSCALE(ICT,1)
      CALL GREYSCALE(ICT,2)
      GOTO 800

100   CONTINUE
      CALL HCTAB(ICT,128,255)

      TYPE "   Ready for the ECLIPSE command file."

101   CONTINUE
      CALL STAT("JHDATA:NCONT.DT",ISTAT,IERR)
      IF (IERR .NE. 1) GOTO 101

      TYPE "<33>k<33>J"
      TYPE "         Command file found."
      CALL OPEN(1,"JHDATA:NCONT.DT ",1,IERR)
      READ BINARY (1) ICODE,IMF,IXCTR,IYCTR,IRAD,IDN
      CALL CLOSE(1,IER)
      CALL DELETE("JHDATA:NCONT.DT ")

      IDIR(7)=IMF
      IXOFF=26
      IF (IMF .NE. 1) IXOFF=IXOFF+140
      IXMIN=IHMIN+IXOFF
      IXMAX=IHMAX+IXOFF

      IYOFF=0
      IYMIN=IVMIN+IYOFF
      IYMAX=IVMAX+IYOFF

C
C       Limit the center to the screen.
      IF (IXCTR .LT. IHMIN) IXCTR=IHMIN
```

```
      IF (IXCTR .GT. IHMAX) IXCTR=IHMAX
      IXC=IXCTR+IXOFF
      IF (IYCTR .LT. IVMIN) IYCTR=IVMIN
      IF (IYCTR .GT. IVMAX) IYCTR=IVMAX
      IYC=IYCTR+IYOFF
C
C       Use the code to select the process.
      IF ((ICODE .LT. -1) .OR. (ICODE .GT. 4)) GOTO 100
C
C       Legal command code received.
      IF (ICODE .EQ. -1) GOTO 900
      GOTO (200,600,800,500),ICODE
      GOTO 100

200   CONTINUE
      TYPE "COMMAND= Process star.  IXC=",IXCTR,"  IYC=",IYCTR,
     A     "  Max RAD=",IRAD
      CALL XHAIR(ICT)
      CALL HCTAB(ICT,IXC,IYC)
C
C       Process the star with the current center and radius.
      NP=IRAD
C
C       Initialize the center characterization.
      ICNTR(1)=IHMAX
      ICNTR(2)=IHMIN
      ICNTR(3)=IVMAX
      ICNTR(4)=IVMIN
      ICNTR(5)=15
      ICNTR(6)=0
C
C       Process quadrant #1
C
C       Pack the quadrant data and process it to file.
      IDIR(1)=0
      IDIR(2)=IXCTR
      IDIR(3)=IYCTR
      IDIR(6)=NP
      CALL RADSP(ICT,+1,+1,CA1,CA2,ICNTR)

C
C       Process quadrant #2.
C
C       Pack the quadrant data and process it to file.
      IDIR(1)=4
      IDIR(2)=IXCTR
      IDIR(3)=IYCTR
      IDIR(6)=NP
      CALL RADSP(ICT,-1,+1,CA2,CA1,ICNTR)

C
C       Process quadrant #3.
```

```
C
C         Pack the quadrant data and process it to file.
          IDIR(1)=8
          IDIR(2)=IXCTR
          IDIR(3)=IYCTR
          IDIR(6)=NP
          CALL RADSP(ICT,-1,-1,CA1,CA2,ICNTR)

C
C         Process quadrant #4.
C
C         Pack the quadrant data and process it to file.
          IDIR(1)=12
          IDIR(2)=IXCTR
          IDIR(3)=IYCTR
          IDIR(6)=NP
          CALL RADSP(ICT,+1,-1,CA2,CA1,ICNTR)

          TYPE "  STAR Processed."
          CALL HCTAB(ICT,IXC,255)

          IXS=ICNTR(1)+IXOFF
          IXE=ICNTR(2)+IXOFF
          IYS=ICNTR(3)+IYOFF
          IYE=ICNTR(4)+IYOFF

          TYPE "         Characterize block.  IX=",IXS,IXE,"  IY=",
     A      IYS,IYE
          IF (IDEBF .AND. 4) WRITE(12,1001) (ICNTR(I),I=1,6)
1001      FORMAT(" ---ICNTR =",6I5)

C
C         Process block with current center and size.
C
C         Get size and auto-scale to the screen.
          IF (IXS .LT. IXMIN) IXS=IXMIN
          IF (IXE .GT. IXMAX) IXE=IXMAX
          IXL=IXE-IXS+1

          IF (IYS .LT. IYMIN) IYS=IYMIN
          IF (IYE .GT. IYMAX) IYE=IYMAX
          IYL=IYE-IYS+1

          IF (IYL .GT. 1) GOTO 290
          DO 270 I=1,16
          IBMAP(I)=0
270       CONTINUE
          GOTO 400

290       CONTINUE

          CALL BOCUR(ICT,IXL,IYL)
```

```
        CALL HCTAB(ICT,IXS,IYS)

C
C         Get the block from the screen to a disk file as a bit map.
C       (16X16) bits

        IGMN=ICNTR(5)
        IGMX=ICNTR(6)

        SFX=IXL/16.0
        SFY=IYL/16.0

C
C         Get the number of indices horizontally in the large grid.
        IXDV=ILCM(IXL,16)
C
C         Get the number of large grid indices per image index.
        IXD=IXDV/IXL
C
C         Get the number of large grid indices per bit map index.
        IXB=IXDV/16
C
C         Set up to index through the image positions horizontally.
        IXIP=IXS-1
C
C         Get the number of indices vertically in the large grid.
        IYDV=ILCM(IYL,16)
C
C         Get the number of large grid indices per image index.
        IYD=IYDV/IYL
C
C         Get the number of large grid indices per bit map index.
        IYB=IYDV/16
C
C         Initialize the bit ma index.
        IXBI=0

        IF (IDEBF .AND. 4) WRITE(12,1002) IXL,IXDV,IYL,IYDV
1002    FORMAT(" ----",2("  LCM( 16,",I3,") = ",I5,3X))

C
C         Cycle through the large grid columns.
        DO 390 IXDVI=1,IXDV
        IF ((MOD(IXDVI,IXB) .NE. 1) .AND. (IXB .NE. 1)) GOTO 310
C
C         Initiate a new bit map column.
        IXBI=IXBI+1
        TYPE "<33>k<33>J"
        TYPE " New bit column, IXBI=",IXBI
C
C         Reset the counters.
        DO 305 IYBI=1,16
```

```
        ISUM(IYBI)=0
        ICNT(IYBI)=0
305     CONTINUE
310     CONTINUE
        IF ((MOD(IXDVI,IXD) .NE. 1) .AND. (IXD .NE. 1)) GOTO 320
C
C         Initiate new imge column.
        IXIP=IXIP+1
        CALL RVBLK(ICT,IDAT,IXIP,1,IYS,IYL)
320     CONTINUE
C
C         Reset row counters.
        IYBI=0
        IYII=0
C
C         Cycle through the large grid rows.
        DO 350 IYDVI=1,IYDV
        IF ((MOD(IYDVI,IYB) .NE. 1) .AND. (IYB .NE. 1)) GOTO 330
C
C         Initiate new bit map row.
        IYBI=IYBI+1
330     CONTINUE
        IF ((MOD(IYDVI,IYD) .NE. 1) .AND. (IYD .NE. 1)) GOTO 340
C
C         Initiate new image row.
        IYII=IYII+1
        IPIX=IDAT(IYII)
340     CONTINUE
C
C         Update the counts for this bit, using this image pixel.
        ISUM(IYBI)=ISUM(IYBI)+1
        IF ((IGMN .LE. IPIX) .AND. (IPIX .LE. IGMX))
     A            ICNT(IYBI)=ICNT(IYBI)+1
350     CONTINUE
        IF (MOD(IXDVI,IXB) .NE. 0) GOTO 370
C
C         This bit map column is done, get the results for each bit.
        IMASK=1
        IBMAP(IXBI)=0
        TYPE "<33>k"
        TYPE
        TYPE "<33>J  Bit values."
        DO 360 IYBI=1,16
        IPER=(ICNT(IYBI)*100)/ISUM(IYBI)
        TYPE "  SUM=",ISUM(IYBI),"  CNT=",ICNT(IYBI),"  Percent=",IPER
        IF (IPER .GT. ITHRB) IBMAP(IXBI)=IBMAP(IXBI) .OR. IMASK
        IMASK=ISHFT(IMASK,1)
360     CONTINUE
370     CONTINUE
390     CONTINUE

        IF (IDEBF .AND. 8) CALL PRBM(IBMAP)
```

```
400     CONTINUE

C
C         Write out the blob characterization.
C
C         Each word is a column.
        CALL OPEN(1,"JHDATA:NDAT.DT ",3,IERR)
        WRITE BINARY (1) IDN,SFX,SFY
        WRITE BINARY (1) (ICNTR(J),J=1,6)
        WRITE BINARY (1) (IBMAP(I),I=1,16)

        CALL CLOSE(1,IER)
        CALL RENAM("JHDATA:NDAT.DT ","JHDATA:BLOCK.DT ",IER)

        GOTO 100

500     CONTINUE
        TYPE "  Bit map region edge.  IYC=",IYC,"  IXC=",IXC
C
C         Bit map a 16x16 window.
C
C         Get the region limits for the bit map window.
        IXLFT=IXC-7
        IXRGT=IXLFT+15
        IYTOP=IYC-7
        IYBOT=IYTOP+15
C
C         Put a box cursor around it.
        CALL BOXCUR(ICT,18,18)
        CALL HCTAB(ICT,IXLFT-1,IYTOP-1)
C         Get region grey levels near the edge.
C               IDN=-1 for left side of edge
C                   +1 for right side of edge
        IXM=MIN0(IXC,IXC+IDN+IDN)
        IYM=IYC-1
        CALL RVBLK(ICT,IDAT,IXM,2,IYC,3)

C
C         Get the grey level range of values near the edge.
        IGMN=15
        IGMX=0
        DO 505 I=2,7
        IGMN=MIN0(IGMN,IDAT(I))
        IGMX=MAX0(IGMX,IDAT(I))
505     CONTINUE
C
C         Initialize the index, and cycle through the columns.
        IBPOS=1
        DO 540 IXLOC=IXLFT,IXRGT
C
C         Initially no bits set for this column.
        IWORD=0
```

```
      IF ((IXLOC .LT. IXMIN) .OR. (IXMAX .LT. IXLOC)) GOTO 530
C
C       Input the next column.
      CALL RVBLK(ICT,IDAT,IXLOC,1,IYTOP,16)
C
C       Initialize the indices, and scan the column entries.
      IDPOS=2
      IMASK=1
      DO 520 IYLOC=IYTOP,IYBOT
      IF((IYLOC .LT. YMIN) .OR. (IYMAX .LT. IYLOC)) GOTO 510
C
C       Check the grey value for in or out.
      IPIX=IDAT(IDPOS)
      IF ((IPIX .LT. IGMN) .OR. (IGMX .LT. IPIX)) GOTO 510
      IWORD=IWORD .OR. IMASK
510   CONTINUE
      IDPOS=IDPOS+1
      IMASK=ISHFT(IMASK,1)
520   CONTINUE
530   CONTINUE
      TYPE "<33>k"
      TYPE
      TYPE
      TYPE "  Map column =",IBPOS

      IBMAP(IBPOS)=IWORD
      IBPOS=IBPOS+1
540   CONTINUE
C
C       Write out the bit map.
      IF (IDEBF .AND. 16) CALL PRBM(IBMAP)
      IFLG=1
      CALL OPEN(1,"JHDATA:NDAT.DT",3,IER)
      WRITE BINARY (1) IFLG,(IBMAP(I),I=1,16)
      CALL CLOSE(1,IER)
      CALL RENAM("JHDATA:NDAT.DT","JHDATA:WBMAP.DT",IER)

      CALL HCTAB(ICT,IXLFT,255)

      GOTO 100


600   CONTINUE
C
C       Process line slice.
      IY=IYCTR-IYOFF
      TYPE "COMMAND = Get image slice.  Image #=",IMF,"  Line #=",IY
C
C       Set up IDIR parts common to both images.
      IDIR(1)=-1
      IDIR(2)=0
      IDIR(3)=IY
```

```
        IXL=IXMAX-IXMIN+1
        IXL1=IXL+1
        IDIR(4)=IXL-1
        IDIR(5)=IY
        IDIR(6)=IXL
C
C         Output one slice from selected image.
        IX=26
        IF (IMF .EQ. 2) IX=IX+140
        IDIR(7)=IMF
        CALL RVBLK(ICT,IDAT,IX,IXL,IY,1)
        IFIRST=IDAT(2)
        ILAST=IDAT(IXL1)
        CALL EXQVFL(IXL,IDAT(2),NFEAT,IFEAT,2)

        CALL OPEN(1,"JHDATA:NDAT.DT",3,IER)
        CALL WRBLK(1,1,IFBUF,1,IER)
        CALL CLOSE(1,IER)
        CALL RENAM("JHDATA:NDAT.DT","JHDATA:SLICE.DT",IER)

        GOTO 100


C
C         Set up the image.
800     CONTINUE
        TYPE "<33>k<33>J"
C
C         Check for the desired input mode.
        ACCEPT "Save the current image? (0=NO) ",IRP
        IF (IRP .NE. 0) GOTO 860

805     CONTINUE
        ACCEPT "Input new images? (0=NO) ",IRP
        IF (IRP .EQ. 0) GOTO 100

C
C         Clear the screen.
        CALL PXFILL(ICT,15,0,320,0,240)
C
C         Repeat for left and right halves of the pair.
        DO 856 IMGN=1,2
        IX=26
        IF (IMGN .NE. 1) IX=IX+140

C
C         Input from disk file.
850     CONTINUE
C
C         Prompt user for file name.
        IF (IMGN .EQ. 2) GOTO 8051
        TYPE "   Loading the LEFT image from disk. "
```

```
        TYPE "      Input the LEFT image file name."
        GOTO 8052
8051    CONTINUE
        TYPE "   Loading the RIGHT image from disk. "
        TYPE "      Input the RIGHT image file name."
8052    CONTINUE
        READ(11,1050) IMFIL(1)
1050    FORMAT(S40)
        TYPE
        CALL OPEN(1,IMFIL,1,IERR)
        IF (IERR .EQ. 1) GOTO 851
          TYPE " Open error on input file -- IERR=",IERR
          GOTO 850
851     CONTINUE
        IYP=0
        DO 855 I=0,14
        IB=I*4
        CALL RDBLK(1,IB,IDAT(2),4,IERR)
        IF (IERR .EQ. 1) GOTO 852
          TYPE "   Read error -- IERR=",IERR
          GOTO 850
852     CONTINUE
C
C         Output LEFT image to the OCTEK.
        CALL WVBLK(ICT,IDAT,IX,128,IYP,8)
        IYP=IYP+8
855     CONTINUE
        TYPE "  Image loaded."
        CALL CLOSE(1,IERR)
856     CONTINUE

        GOTO 100

860     CONTIUE
C
C         Repeat for left and right images.
        DO 870 IMGN=1,2
        IX=26
        IF (IMGN .NE. 1) IX=IX+140

C
C         Save the current image.
        TYPE "  Saving the current image to disk."
        IF (IMGN .EQ. 1) TYPE "   Input the LEFT image filename."
        IF (IMGN .EQ. 2) TYPE "   Input the RIGHT image filename."
        READ(11,1050) IMFIL(1)
        TYPE
        CALL OPEN(1,IMFIL,3,IERR)
        IF (IERR .EQ. 1) GOTO 861
          TYPE " Open error on output file -- IERR=",IERR
          GOTO 860
861     CONTINUE
```

```
        IYP=0
        DO 865 I=0,14
        IB=I*4
        CALL RVBLK(ICT,IDAT,IX,128,IYP,8)
        CALL WRBLK(1,IB,IDAT(2),4,IERR)
        IF (IERR .EQ. 1) GOTO 864
          TYPE "  Write error -- IERR=",IERR
          GOTO 860
864     CONTINUE
        IYP=IYP+8
865     CONTINUE
        TYPE " Saved."
870     CONTINUE

        CALL CLOSE(1,IERR)
        GOTO 805


900     CONTINUE
        TYPE "COMMAND= Terminate processing."
C
C         Terminate processing.
        CALL OPEN(1,"JHDATA:TRM.DT ",3,IERR)
        WRITE(1,1000)
1000    FORMAT("  NOVA processing terminated.")
        CALL CLOSE(1,IER)
        CALL RENAM("JHDATA:TRM.DT ","JHDATA:TERM.DT ",IER)

        CALL OREMOVE(ICT)

        STOP

        END
```

```
C
      SUBROUTINE EXQVFL(ILAST,IPIX,NFEAT,IFEAT,ITHR)
C
C       This is the Queen Victoria algorithm.  It is used on one
C     slice of pixel data to extract a list of edge feature regions.
C
C             Written by James R. Holten III, 9 Mar 85
C
C       Finds edges of significant slope.
C     ON ENTRY:
C             ILAST   The highest index for pixels in the buffer
C             IPIX    The buffer of pixels
C             ITHR    The desired noise limiting threshhold
C     ON EXIT:
C             NFEAT   The number of edge features found.
C             IFEAT   A table of features and their characteristics.
C             IFEAT(I,1)= The starting pixel index for the feature
C             IFEAT(I,2)= The ending pixel index for the feature
C             IFEAT(I,3)= The starting edge grey level
C             IFEAT(I,4)= The ending edge grey level
C

      DIMENSION IPIX(1),IFEAT(50,4)
      JLOC=1
      IPRE=1
      NFEAT=0
      NFMAX=50
      ISF=0
C
C       Start new slope search.
100   CONTINUE

C
C       Start where the last left off, and set up left end.
      ILOC=IPRE
      IVLE=IPIX(ILOC)
C
C       Get the first pixel to the right, if any left.
      IPRE=ILOC+1
      IF (IPRE .GT. ILAST) GOTO 300
      IVRE=IPIX(IPRE)
C
C       Is this the start of a new interval of slope?
      ICHG=IVRE-IVLE
      ISLP=ISGN(ICHG)
      IF (ISLP .EQ. 0) GOTO 100
      INEXT=IPRE
      IF (IABS(ICHG) .GT. ITHR) GOTO 120
C
C       Change of less than threshhold, alow up to one flat interval.
      IPASS=0
      IVPRE=IVRE
```

```
110     CONTINUE
        INEXT=INEXT+1
        IF (INEXT .GT. ILAST) GOTO 300
C
C         Get the next pixel.
        IVRE=IPIX(INEXT)
C
C         Does the slope continue?
        IDIFF=IVRE-IVPRE
        INSLP=ISGN(IDIFF)
        IF (INSLP .EQ. ISLP) GOTO 120
        IF (IDIFF .NE. 0) GOTO 100
        IF (IPASS .EQ. 1) GOTO 100
        IPASS=1
        GOTO 110

C
C         Valid slope.  How far does it go?
120     CONTINUE
C
C         Updte the total change over the feature interval.
        ICHG=IVRE-IVLE
C
C         Save the previous value and location before getting the next.
        IVPRE=IVRE
        IPRE=INEXT
        INEXT=INEXT+1
        IF (INEXT .GT. ILAST) GOTO 200
C
C         Check the next pixel for continued slope.
        IVRE=IPIX(INEXT)
        IDIFF=IVRE-IVPRE
        INSLP=ISGN(IDIFF)
        IF (INSLP .EQ. ISLP) GOTO 120
C
C         Not the same slope, is it a significant change?
        IF (IABS(IDIFF) .GT. ITHR) GOTO 200
C
C         No, see if the old slope continues beyond it.
        INEXT=INEXT+1
        IF (INEXT .GT. ILAST) GOTO 200
C
C         Get the next pixel, and check the new interval slope.
        IVRE=IPIX(INEXT)
        IDIFF=IVRE-IVPRE
        INSLP=ISGN(IDIFF)
        IF (INSLP .EQ. ISLP) GOTO 120
C
C         It does not continue, store the current feature and scan on.
        GOTO 200
C
C         The end of a slope feature was found, store it.
```

```
200     CONTINUE

        INUM=ILOC-JLOC+1
        IF (INUM .GT. 3) GOTO 220
C
C         Both averages are the same.
        ISUM=0
        DO 210 KLOC=JLOC,ILOC
        ISUM=ISUM+IPIX(KLOC)
210     CONTINUE
        SUM=ISUM
        IVAVE=(SUM/INUM)+.5
        IV1=IVAVE
        GOTO 240

220     CONTINUE
        IS1=0
        IS2=0
        DO 230 K=0,2
        IS1=IS1+IPIX(JLOC+K)
        IS2=IS2+IPIX(ILOC-K)
230     CONTINUE
        S1=IS1
        S2=IS2
        IVAVE=(S1/3.0)+.5
        IV1=(S2/3.0)+.5

240     CONTINUE
        IF (NFEAT .GE. NFMAX) GOTO 500

        NFEAT=NFEAT+1
C         Location of left end.
        IFEAT(NFEAT,1)=ILOC
C         Location of right end.
        IFEAT(NFEAT,2)=IPRE
C         Grey value of left end.
        IFEAT(NFEAT,3)=IV1
C         Grey value of right end.
        IFEAT(NFEAT,4)=IVPRE
C
C         Store the grey value in the previous feature also.
        IF (NFEAT .GT. 1) IFEAT(NFEAT-1,4)=IVAVE

C
C         Check for termination.  If not yet, then continue scan.
        IF (INEXT .GT. ILAST) GOTO 300
        INEXT=IPRE
        JLOC=IPRE
        GOTO 100
C
C         Finished with slice, return feature list.
300     CONTINUE
```

```
      IF (NFEAT .LE. 0) RETURN
      IF (INUM .GE. 1) GOTO 400
      IFEAT(NFEAT,4)=IPIX(ILAST)

      GOTO 410

400   CONTINUE

      ISUM=0
      INUM=0
      DO 405 K=0,2
      IF (JLOC+K .GT. ILOC) GOTO 405
      INUM=INUM+1
      ISUM=ISUM+IPIX(JLOC+K)
405   CONTINUE
      SUM=ISUM
      IFEAT(NFEAT,4)=(SUM/INUM)+.5

410   CONTINUE
C
C       Suppress overlapping edge features.
      IFLOC=1
      DO 490 ILOC=2,NFEAT
      IF (IFEAT(IFLOC,2) .LT. IFEAT(ILOC,1)) GOTO 430
      IWD1=IFEAT(ILOC,2)-IFEAT(ILOC,1)
      IWD2=IFEAT(IFLOC,2)-IFEAT(IFLOC,1)
      IF((IWD1 .GT. 1) .OR. (IWD2 .GT. 1)) GOTO 430
C
C       Combine the features.
      IFEAT(IFLOC,2)=IFEAT(ILOC,2)
      IFEAT(IFLOC,4)=IFEAT(ILOC,4)
      GOTO 490

430   CONTINUE
C
C       Copy the features as needed.
      IFLOC=IFLOC+1
      DO 450 INDX=1,4
      IFEAT(IFLOC,INDX)=IFEAT(ILOC,INDX)
450   CONTINUE
490   CONTINUE

      NFEAT=IFLOC
      RETURN

500   CONTINUE
      TYPE "-- EX1DEG -- TOO MANY FEATURES."
      TYPE
      RETURN

      END
```

```
C
      SUBROUTINE RADLN(ICT,IRAD,IDIR,NP,CAX,CAY)
C
C       This routine retrieves one radial line of data from the
C     current image on the OCTEK.  The line direction, start location,
C     and max allowable length are given.  It returns the array of
C     pixels, the actual number of pixels retrieved, and the line
C     ending location.
C
C     ON ENTRY:
C             ICT     The device control table for the OCTEK
C             IDIR    Array of directional line data
C               1     Direction index
C               2     X-start coordinate
C               3     Y-start coordinate
C               7     Image ID number
C             NP      Max number of points desired
C             CAX     X-index coefficient
C            CAY     Y-index coefficient
C
C     ON EXIT:
C             IRAD    The array of pixel values
C             IDIR    As above except for the following:
C               4     X-end coordinate
C               5     Y-end coordinate
C               6     Number of points actually stored
C
C     ROUTINES USED:
C             IRDPIX  Reads pixels from the OCTEK (from IACF4.LB)
C

      COMMON /IMAGE/ IXMIN,IXMAX,IYMIN,IYMAX,IMF
      DIMENSION ICT(1),IDIR(7),IRAD(1)

C
C       Only compute this constant once.
      SQRT2=SQRT(2.0)
C
C       Get the desired radial line start location.
      IXCTR=IDIR(2)+IXMIN
      IYCTR=IDIR(3)+IYMIN
      NDP=0
C
C       Get the initial data point location as default end point.
      IX=IXCTR
      IY=IYCTR
C
C       Limit the number of allowable points to desired range.
      IF (NP .LT. 1) NP=1
      IF (NP .GT. 60) NP=60

C
```

```
C         Loop through the points in the radial line.
        DO 100 I=1,NP
C
C         Convert from diagonal distance to horizontal.
        HYP=(I-1)*SQRT2
C
C         Get the projection onto the X and Y axes.
        IXLOC=(HYP*CAX)+IXCTR
        IYLOC=(HYP*CAY)+IYCTR
C
C         Is the next pixel location on the screen.
        IF ((IXLOC .LT. IXMIN) .OR. (IXLOC .GT. IXMAX)) GOTO 200
        IF ((IYLOC .LT. IYMIN) .OR. (IYLOC .GT. IYMAX)) GOTO 200
C
C         Get the current location and the index.
        IX=IXLOC
        IY=IYLOC
        NDP=NDP+1
C
C         Get the pixel value at this location.
        IRAD(NDP)=IRDPIX(ICT,IX,IY)
100     CONTINUE

200     CONTINUE
C
C         Save the end points and the length.
        IDIR(4)=IX-IXMIN
        IDIR(5)=IY-IYMIN
        IDIR(6)=NDP

        RETURN

        END
```

```
C
      SUBROUTINE RADSP(ICT,ISX,ISY,CAX,CAY,ICNTR)
C
C       Retrieves the ray data from the current quadrant in IDIR.
C     It stores the results out to IFILE.
C
C     ON ENTRY:
C             ICT     Device control table  for the OCTEK
C             IDIR    Array of image-relative location data.
C               1     Index of last direction processed.
C               2     X-center
C               3     Y-center
C               6     Number of points in the desired radial line.
C               7     Image ID flag, 1=left, 2=right
C             ISX     sign for the x coefficients.
C             ISY     sign for the y coefficients.
C             CAX     X coefficients for 4 directions
C             CAY     Y coefficients for 4 directions
C             ICNTR   Characterization of the center blob.
C
C     ON EXIT:
C             IDIR    Same as above, except as follows:
C               1     New last direction index
C               4     X-location of last line end
C               5     Y-location of last line end
C               6     Actual number of points in last line.
C
C     DEBUG FLAGS USED--(printouts enabled)
C             1       Radial arm information and number of features
C             2       Feature lists
C
C     ROUTINES USED:
C             RADLN           Retrieves a single radial line of
C                             pixels.
C             EXQVFL          Extracts features from a slice.
C             PRQVFL          Prints the feature list
C
      COMMON /FBUFF/ IFBUF,IDAT
      COMMON /DEBUG/ IDEBF
      DIMENSION IFEAT(50,4),IFBUF(256)
      DIMENSION ICT(1),IDIR(7),CAX(4),CAY(4)
      DIMENSION ICNTR(6)
      DIMENSION IDAT(1025)
      EQUIVALENCE (IFBUF(1),IDIR(1)),(IFBUF(8),NFEAT),
     A   (IFBUF(9),IFIRST),(IFBUF(10),ILAST),(IFBUF(11),IFEAT(1,1))

      TYPE "  RADSP called, quadrant=", IDIR(1),"  IMAGE ",IDIR(7)

      IQOFF=IDIR(1)
      NP=IDIR(6)

      IXMN=IDIR(2)
```

```
      IXMX=IXMN
      IYMN=IDIR(3)
      IYMX=IYMN
      IGMN=15
      IGMX=0

      DO 100 ID=1,4
      ID1=ID-1
      CX=ISX*CAX(ID)
      CY=ISY*CAY(ID)
      IDIR(1)=IQOFF+ID
      CALL RADLN(ICT,IDAT,IDIR,NP,CX,CY)

      IXL=IDIR(6)
      CALL EXQVFL(IXL,IDAT,NFEAT,IFEAT,1)

      IF (IDEBF .AND. 1) WRITE(12,1000) (IDIR(I),I=1,7),NFEAT
1000  FORMAT(" -- RADSP --  IDIR = ",7I5,"  NFEAT= ",I3)
      IF (IDEBF .AND. 2) CALL PRQVFL(NFEAT,IFEAT)

C
C       Get the line characteristics relative to the image.
      IFIRST=IDAT(1)
      ILAST=IDAT(IXL)
      IX1=IDIR(2)
      IX2=IDIR(4)
      IY1=IDIR(3)
      IY2=IDIR(5)
      IG1=IFIRST
      IG2=ILAST
      IF (NFEAT .LT. 1) GOTO 20
      INDX=IFEAT(1,1)-1
      RP=IDIR(6)
      XRAT=(IDIR(4)-IX1)/RP
      IX2=IX1+XRAT*INDX
      YRAT=(IDIR(5)-IY1)/RP
      IY2=IY1+YRAT*INDX
      IG2=IFEAT(1,3)
20    CONTINUE
C
C       Update the slice extremes.
      IXMN=MIN0(IXMN,IX1,IX2)
      IXMX=MAX0(IXMX,IX1,IX2)
      IYMN=MIN0(IYMN,IY1,IY2)
      IYMX=MAX0(IYMX,IY1,IY2)
      IGMN=MIN0(IGMN,IG1,IG2)
      IGMX=MAX0(IGMX,IG1,IG2)

100   CONTINUE

C
C       Update for other quadrants.
```

```
ICNTR(1)=MIN0(ICNTR(1),IXMN)
ICNTR(2)=MAX0(ICNTR(2),IXMX)
ICNTR(3)=MIN0(ICNTR(3),IYMN)
ICNTR(4)=MAX0(ICNTR(4),IYMX)
ICNTR(5)=MIN0(ICNTR(5),IGMN)
ICNTR(6)=MAX0(ICNTR(6),IGMX)

RETURN

END
```

```
C
      FUNCTION IGCD(IX,IY)
C
C       Computes the greatest common divisor of the two
C     given integers using the Cinese remainder theorem.
C
C       By James R. Holten III, 21 Oct 1985
C
      IX1=MAX0(IX,IY)
      IY1=MIN0(IX,IY)

100   CONTINUE
      IREM=MOD(IX1,IY1)
      IF (IREM .LE. 0) GOTO 200

      IX1=MAX0(IREM,IY1)
      IY1=MIN0(IREM,IY1)
      GOTO 100

200   CONTINUE

      IGCD=IY1
      RETURN
      END
```

```
C
      SUBROUTINE PRQVFL(NFEAT,IFEAT)
C
C       Print out the Queen Victoria algorithm feature list.
C
C
      DIMENSION IFEAT(50,4)

C
C       Printout the headers.
      WRITE(12,1000) NFEAT
1000  FORMAT("    1-Dimensional Slice Feature List"/
     A       "      There are ",I3," features in the slice.",/
     C       "   INDEX    START   STOP   GRYL   GRYR")

C
C       Loop through the list.
      DO 100 INDX=1,NFEAT

      WRITE(12,1001) INDX,(IFEAT(INDX,I),I=1,4)
1001  FORMAT(4X,I3,6X,2(I3,5X),2(I2,5X))

100   CONTINUE

      RETURN
      END
```

```
C
      SUBROUTINE PRBM(IBMAP)
C
C       Prints out a single bit map.
C
      DIMENSION IBMAP(16),IOUT(16)

      IMASK=1
      WRITE(12,1001)
1001  FORMAT(//," Bit map.")
      WRITE(12,1003)
1003  FORMAT(6X,".................")

      DO 200 IYBI=1,16
      DO 100 IXBI=1,16
      IOUT(IXBI)="  "
      IF (IBMAP(IXBI) .AND. IMASK) IOUT(IXBI)="**"
100   CONTINUE
      WRITE(12,1000) IYBI,(IOUT(I),I=1,16)
1000  FORMAT("  ",I3," .",16A1,".")
      IMASK=ISHFT(IMASK,1)
200   CONTINUE
      WRITE(12,1003)
      WRITE(12,1002)
1002  FORMAT(/," ")

      RETURN
      END
```

```
C
C       ESTAR.FR
C
C         This is the ECLIPSE main program to run the analysis on
C       epipolar lines, and get 3-D positions of the features in the
C       line.
C
C               Written 30 July 1985
C                       by James R. Holten III
C
C       VARIABLES USED:
C               IDEBF   Debug flags, each bit is a flag.
C                       1       print feature lists.
C                       2       print blob characterizations.
C                       3       print match lists.
C                       4       print position lists.
C                       5       print bit maps of blobs.
C
C       CALLED ROUTINES:
C               LCLN    Locates the regions along a single epipolar
C                       line.
C


      COMMON /CALIB/ DSEP,DVIRT
      COMMON /POSIT/ NP,POS
      COMMON /DEBUG/ IDEBF
      COMMON /FBUFF/ IFB1,IFB2
      COMMON /BLOBS/ NB1,IBLB1,NB2,IBLB2
      COMMON /MATCH/ NMATCH,IMATCH
      DIMENSION NMATCH(100),IMATCH(100,10,4)
      DIMENSION POS(640)
      DIMENSION IFB1(256),IFB2(256),IFEAT1(50,4),IFEAT2(50,4)
      DIMENSION IBLB1(17,100),IBLB2(17,100)
      EQUIVALENCE (IFB1(8),NFEAT1),(IFB1(11),IFEAT1(1,1))
      EQUIVALENCE (IFB2(8),NFEAT2),(IFB2(11),IFEAT2(1,1))

      TYPE
      TYPE "ROBOT VISION SYSTEM"
      TYPE
      TYPE "          By James R. Holten III"
      TYPE
      ACCEPT "  Camera seperation = ",DSEP
      ACCEPT "  Virtual image distance = ",DVIRT

      IDEBF=0
      TYPE
      TYPE "The debugger options are:"
      TYPE "          1--Print feature lists."
      TYPE "          2--Print blob characterizations."
      TYPE "          4--Print match lists."
      TYPE "          8--Print position lists."
      TYPE "         16--Print bit maps."
```

```
        TYPE "          32—Print compared bit maps."
        TYPE
        TYPE "  Add together the options desired."
        TYPE
        ACCEPT " Desired debug flags (0 for none) = ",IDEBF

100     CONTINUE
        TYPE
        ACCEPT "  Which horizontal line? (0 to 119)  IY=",IYLOC
        IF ((IYLOC .LT. 0) .OR. (119 .LT. IYLOC)) GOTO 200
        CALL LCLN(IYLOC)
        GOTO 100
200     CONTINUE

        STOP

        END
```

```
C
      SUBROUTINE LCLN(IYLOC,IFLG)
C
C       Processes single epipolar line from the images, and
C     generates a camera-relative model.
C
C       by James R. Holten III, 19 Oct 1985
C
C     ON ENTRY:
C             IYLOC   The Y-index location for the slices across
C                       the images.
C
C     ON EXIT:
C             All results are passed in the common areas.
C
C     ROUTINES USED:
C             CFEAT   Gets the feature lists for the epipolar line.
C             NBLOB   Characterizes a single blob
C             MASL    Matches the features
C             LCSL    Locates the positions of the feature edges.
C             PNPL    Passes the camera-relative positions on for
C                     display or use.
C

      COMMON /DEBUG/ IDEBF
      COMMON /POSIT/ NP,POS
      COMMON /FBUFF/ IFB1,IFB2
      COMMON /BLOBS/ NB1,IBLB1,NB2,IBLB2
      COMMON /MATCH/ NMATCH,IMATCH
      DIMENSION IFB1(256),IFB2(256),IFEAT1(50,4),IFEAT2(50,4)
      DIMENSION IBLB1(17,100),IBLB2(17,100)
      DIMENSION NMATCH(100),IMATCH(100,10,4),POS(640)
      EQUIVALENCE (IFB1(8),NFEAT1),(IFB1(11),IFEAT1(1,1))
      EQUIVALENCE (IFB2(8),NFEAT2),(IFB2(11),IFEAT2(1,1))

C
C       Clearing the files.
      CALL DELETE("JHDATA:SLICE.FR")
      CALL DELETE("JHDATA:WBMAP.DT")
      CALL DELETE("JHDATA:BLOCK.DT")
C
C       Get the features for the epipolar slices.
      TYPE " Getting slices from IY = ",IYLOC
      CALL CFEAT(IYLOC)
      IF ((IDEBF .AND. 1) .EQ. 0) GOTO 110
      CALL PRQVFL(NFEAT1,IFEAT1)
      CALL PRQVFL(NFEAT2,IFEAT2)
110   CONTINUE
C
C       Set all the blob flags to "no bit map saved".
      DO 120 INDX=1,100
C
```

```
C         First location is the flag, 0 for no map, 1 for yes.
          IBLB1(1,INDX)=0
          IBLB2(1,INDX)=0
120       CONTINUE

C
C         Attempt to match the features and their blobs.
          TYPE " Matching feature lists.  N1 = ",NFEAT1,"   N2 =",NFEAT2
          CALL MASL(IYLOC,NFEAT1,IFEAT1,NFEAT2,IFEAT2,IBLB1,IBLB2,
      A     NMATCH,IMATCH)
          IF ((IDEBF .AND. 2) .EQ. 0) GOTO 230
          DO 210 ILOC=1,NFEAT1
          WRITE(12,1000) ILOC
1000      FORMAT("  MAP #",I3)
          IF (IBLB1(1,ILOC) .EQ. 1) CALL PRBM(IBLB1(2,ILOC))
          IF (IBLB1(1,ILOC) .NE. 1) WRITE(12,1001)
1001      FORMAT("   There is no map stored.")
          WRITE(12,1002)
          IF (IBLB1(1,ILOC+50) .EQ. 1) CALL PRBM(IBLB1(2,ILOC+50))
          IF (IBLB1(1,ILOC+50) .NE. 1) WRITE(12,1001)
210       CONTINUE
          DO 220 ILOC=1,NFEAT2
          WRITE(12,1000) ILOC
          IF (IBLB2(1,ILOC) .EQ. 1) CALL PRBM(IBLB2(2,ILOC))
          IF (IBLB2(1,ILOC) .NE. 1) WRITE(12,1001)
          WRITE(12,1002)
1002      FORMAT("  ")
          IF (IBLB2(1,ILOC+50) .EQ. 1) CALL PRBM(IBLB2(2,ILOC+50))
          IF (IBLB2(1,ILOC+50) .NE. 1) WRITE(12,1001)
220       CONTINUE
230       CONTINUE

          IF (IDEBF .AND. 4) CALL PMAL(NFEAT1,NMATCH,IMATCH)

C
C         Locate them in 3-D relative to the camera coordinates.
          TYPE " Computing locations."
          CALL LCSL(IYLOC,NFEAT1,NMATCH,IMATCH,POS)
          NP=NFEAT1
          IF (IDEBF .AND. 8) CALL PRFPOS(NP,POS)

C
C         Pass the descriptions out to a file for further processing.
          TYPE " Storing positions."
          CALL PNPL(IER)

          RETURN
          END
```

```
C
      SUBROUTINE CFEAT(IYLOC)
C
C       Requests and receives fature lists for epipolar slices
C     from the NOVA.
C
C       by James R. Holten III, 19 Oct 1985
C
C     ON ENTRY:
C             IYLOC   The y-location for the two slices.
C
C     ON EXIT:
C             All values are passed in the common area.
C
C     ROUTINES USED:
C             CNOVA   Sends the command file to the NOVA.
C             NWAIT   Waits until the file is present.
C             RENAM, OPEN, RDBLK, CLOSE, DELETE   System I/O.
C
      COMMON /FBUFF/ IFB1,IFB2
      DIMENSION IFB1(256),IFB2(256)
      DIMENSION INOVAC(6)

C
C       Put out the image 1 slice command.
      INOVAC(1)=2
      INOVAC(2)=1
      INOVAC(3)=0
      INOVAC(4)=IYLOC
      INOVAC(5)=128
      INOVAC(6)=0
      CALL DELETE("JHDATA:SLICE.DT")
      CALL CNOVA(INOVAC)

C
C       Put out the image 2 slice command.
      INOVAC(2)=2
      CALL CNOVA(INOVAC)

C
C       Wait for the first results, then input the features.
      CALL NWAIT("JHDATA:SLICE.DT ",1)
      CALL RENAM("JHDATA:SLICE.DT ","JHDATA:SL.DT ",IER)
      CALL OPEN(1,"JHDATA:SL.DT ",1,IER)
      CALL RDBLK(1,1,IFB1,1,IER)
      CALL CLOSE(1,IER)
      CALL DELETE("JHDATA:SL.DT ")

C
C       Wait for the second, then input it.
      CALL NWAIT("JHDATA:SLICE.DT ",1)
      CALL OPEN(1,"JHDATA:SLICE.DT ",1,IER)
```

```
CALL RDBLK(1,1,IFB2,1,IER)
CALL CLOSE(1,IER)
CALL DELETE("JHDATA:SLICE.DT ")

RETURN
END
```

```
C
      SUBROUTINE PRQVFL(NFEAT,IFEAT)
C
C       Print out the Queen Victoria algorithm feature list.
C
C
      DIMENSION IFEAT(50,4)

C
C       Printout the headers.
      WRITE(12,1000) NFEAT
1000  FORMAT("    1-Dimensional Slice Feature List"/
     A         "     There are ",I3," features in the slice.",/
     C         "   INDEX    START   STOP   GRYL   GRYR")

C
C       Loop through the list.
      DO 100 INDX=1,NFEAT

      WRITE(12,1001) INDX,(IFEAT(INDX,I),I=1,4)
1001  FORMAT(4X,I3,6X,2(I3,5X),2(I2,5X))

100   CONTINUE

      RETURN
      END
```

```
C
      SUBROUTINE MASL(IYLOC,NFEAT1,IFEAT1,NFEAT2,IFEAT2,IBLB1,IBLB2,
     A     NMATCH,IMATCH)
C
C       Matches slice features using lists of feature match guesses.
C
C             Written by James R. Holten III, 2 May 1985
C
C       ON ENTRY:
C             IYLOC           vertical position within the two images.
C             NFEAT1,NFEAT2   max feature indices for both slices.
C             IFEAT1,IFEAT2   feature description lists.
C             IBLB1,IBLB2     lists of bit maps for the features.
C
C       ON EXIT:
C             NMATCH(INDX)    number of possible matches in the match
C                             list for feature #INDX
C             IMATCH(INDX,I,1)        slice 2 feature which matches
C                             feature #INDX of slice 1.
C             IMATCH(INDX,I,2)        measure of match mismatch.
C             IMATCH(INDX,I,3)        adjusted "best" match location
C                             in slice 1.
C             IMATCH(INDX,I,4)        adjusted "best" match location
C                             in slice 2.
C
C
      COMMON /DEBUG/ IDEBF
      DIMENSION IBLB1(17,100),IBLB2(17,100)
      DIMENSION IFEAT1(50,4),IFEAT2(50,4)
      DIMENSION NMATCH(100),IMATCH(100,10,4)
C
C       Left and right edges of each feature are matched seperately.
C     The left edges are in IMATCH indices (0<INDX<50), and the
C     right edges are at indices (50<INDX<100)
C
C       This is a local match table for passing the entries.
      DIMENSION LMATCH(8)

      IPOSL=1
C
C       Scan features from slice #1.
      DO 999  INDX=1,NFEAT1
C
C       Set the index for the right edge matches.
      KNDX=INDX+50
C
C       Initialize the number of left and right matches for this
C     feature.
      INUML=0
      INUMR=0
C
C       Get the left and right edge positions for this feature.
```

```
      ILOCL=IFEAT1(INDX,1)
      ILOCR=IFEAT1(INDX,2)
C
C       Get the maximum allowed deviation for match searches.
      ILOCD=(ILOCR-ILOCL)/2

      IRGDL=ILOCL
      IF (INDX .GT. 1) IRGDL=(ILOCL-IFEAT1(INDX-1,2))/2
      IWL=IRGDL
      IF (IRGDL .GT. ILOCD) IRGDL=ILOCD
      IF (IRGDL .LE. 0) IRGDL=0

      IRGDR=NPIX1-ILOCR+1
      IF (INDX .LT. NFEAT1) IRGDR=(IFEAT1(INDX+1,1)-ILOCR)/2
      IWR=IRGDR
      IF (IRGDR .GT. ILOCD) IRGDR=ILOCD
      IF (IRGDR .LE. 0) IRGDR=0


C
C       Get the left and right grey levels.
      IGRYL=IFEAT1(INDX,3)
      IGRYR=IFEAT1(INDX,4)
C
C       Scan the feature list for slice #2.
      DO 900 JNDX=1,NFEAT2
      LNDX=JNDX+50
C
C       Get the left and right edge locations.
      JLOCL=IFEAT2(JNDX,1)
      JLOCR=IFEAT2(JNDX,2)
C
C       Get the maximum allowed deviation for the matches.
      JLOCD=(JLOCR-JLOCL)/2

      JRGDL=JLOCL
      IF (JNDX .GT. 1) JRGDL=(JLOCL-IFEAT2(JNDX-1,2))/2
      JWL=JRGDL
      IF (JRGDL .GT. JLOCD) JRGDL=JLOCD
      IF (JRGDL .LE. 0) JRGDL=0

      JRGDR=NPIX2-JLOCR+1
      IF (JNDX .LT. NFEAT2) JRGDR=(IFEAT2(JNDX+1,1)-JLOCR)/2
      JWR=JRGDR
      IF (JRGDR .GT. JLOCD) JRGDR=JLOCD
      IF (JRGDR .LE. 0) JRGDR=0

C
C       Get the left and right grey levels.
      JGRYL=IFEAT2(JNDX,3)
      JGRYR=IFEAT2(JNDX,4)
C
```

```
C         Compare the left levels for a near match.
          IF (IABS(IGRYL-JGRYL) .GT. 1) GOTO 500
C
C         Near match case for left edge.
C
C         Get the max window size for the left edge.
          IWSL=IWL
          IF (IWSL .GT. JWL) IWSL=JWL
C
C         Left edge match refinement.
          LMATCH(1)=JNDX
          LMATCH(2)=9999
          LMATCH(3)=ILOCL
          LMATCH(4)=JLOCL
          LMATCH(5)=IRGDL
          LMATCH(6)=JRGDL
          LMATCH(7)=-1
          LMATCH(8)=IYLOC
C
C         Get refined position and measure of quality.
          TYPE "   Comparing ",INDX,"   and  ",JNDX
          IF (IDEBF .AND. 32) WRITE(12,1000) INDX,JNDX
1000      FORMAT(/,"  Comparing left sides of ",I3,"  and ",I3)

          CALL MARF(LMATCH,IBLB1(1,INDX),IBLB2(1,JNDX))

          IF (IDEBF .AND. 32) WRITE(12,1001)
1001      FORMAT(//," ")

          IF (LMATCH(3) .LE. LMATCH(4)) GOTO 500
C
C         If the list is empty add it at the top.
          IF (INUML .GE. 1) GOTO 100
C
C         Empty until now.
          INUML=1
          ISPT=1
          GOTO 200

100       CONTINUE
C
C         Compare to worst (bottom) element.
          IQM=IMATCH(2)
          IF (IQM .LT. IMATCH(INDX,INUML,2)) GOTO 120
C
C         Worse than or equal to bottom.  If there is room, put
C         it at the bottom.
          IF (INUML .GE. 10) GOTO 500
          INUML=INUML+1
          ISPT=INUML
          GOTO 200
```

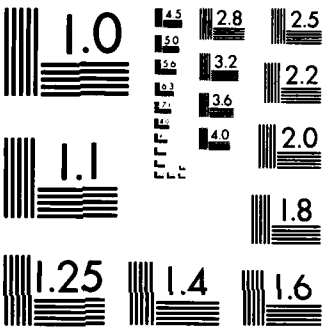
```
120     CONTINUE
C
C         Better than the bottom element.
        INUM=INUML
        IF (INUML .LT. 10) INUML=INUML+1
C
C         Scan for the insert position.
        DO 150 I=1,INUM
        ISPT=INUM-I+1
        IF (ISPT .EQ. 10) GOTO 140
C
C         Move current entry down.
        DO 130 LINDX=1,4
        IMATCH(INDX,ISPT+1,LINDX)=IMATCH(INDX,ISPT,LINDX)
130     CONTINUE

140     CONTINUE
C
C         Check the next entry position.
        IF (ISPT .LT. 2) GOTO 150
        IF (IQM .GE. IMATCH(INDX,ISPT-1,2)) GOTO 200
150     CONTINUE
        ISPT=1

200     CONTINUE
C
C         Insert at ISPT.
        DO 210 LINDX=1,4
        IMATCH(INDX,ISPT,LINDX)=LMATCH(LINDX)
210     CONTINUE

C  ==========   end of left edge processing.

500     CONTINUE
        IF (IABS(IGRYR-JGRYR) .GT. 1) GOTO 900
C
C         Near match case for right edge.


C
C         Get the max window size for the right edge.
        IWSR=IWR
        IF (IWSR .GT. JWR) IWSR=JWR

C
C         Right edge match refinement.
        LMATCH(1)=LNDX
        LMATCH(2)=9999
        LMATCH(3)=ILOCR
        LMATCH(4)=JLOCR
        LMATCH(5)=IRGDR
        LMATCH(6)=JRGDR
```

```
        LMATCH(7)=+1
        LMATCH(8)=IYLOC
C
C         Get the refined position and measure of quality.
        TYPE "   Comparing ",KNDX," and ",LNDX
        IF (IDEBF .AND. 32) WRITE(12,1002) KNDX,LNDX
1002    FORMAT(/," Comparing right sides of ",I3," and ",I3)

        CALL MARF(LMATCH,IBLB1(1,KNDX),IBLB2(1,LNDX))

        IF (IDEBF .AND. 32) WRITE(12,1001)

        IF (LMATCH(3) .LE. LMATCH(4)) GOTO 900
C
C         If the list is empty add it at the top.
        IF (INUMR .GE. 1) GOTO 600
C
C         Empty until now.
        INUMR=1
        ISPT=1
        GOTO 700

600     CONTINUE
C
C         Compare the worst (botom) element.
        IQM=LMATCH(2)
        IF (IQM .LT. IMATCH(KNDX,INUMR,2)) GOTO 620
C
C         Worse than or equal to bottom element.
C
C         Put it at the bottom if there is room.
        IF (INUMR .GE. 10) GOTO 900
        INUMR=INUMR+1
        ISPT=INUMR
        GOTO 700

620     CONTINUE
C
C         Better than the bottom.
        INUM=INUMR
        IF (INUMR .LT. 10) INUMR=INUMR+1
C
C         Scan for the insert position.
        DO 650 I=1,INUM
        ISPT=INUM-I+1
        IF (ISPT .EQ. 10) GOTO 640
C
C         Move current entry down.
        DO 630 LINDX=1,4
        IMATCH(KNDX,ISPT+1,LINDX)=IMATCH(KNDX,ISPT,LINDX)
630     CONTINUE
```

```
640     CONTINUE
C
C         Check for the next entry position.
        IF (ISPT .LT. 2) GOTO 650
        IF (IQM .GE. IMATCH(KNDX,ISPT-1,2)) GOTO 700
650     CONTINUE
        ISPT=1

700     CONTINUE
C
C         Insert at ISPT.
        DO 710 LINDX=1,4
        IMATCH(KNDX,ISPT,LINDX)=LMATCH(LINDX)
710     CONTINUE

C ============   end of right edge processing.
900     CONTINUE
        IF (IDEBF .AND. 32) WRITE(12,2000) INDX,INUML,KNDX,INUMR
2000    FORMAT(" Match lists generated.",/,"   INDX=",I4,
     A     "   INUML=",I4,/,"   KNDX=",I4,"   INUMR=",I4,/)

        NMATCH(INDX)=INUML
        NMATCH(KNDX)=INUMR
999     CONTINUE

        RETURN
        END
```

AD-A164 282 A ROBOT VISION SYSTEM(U) AIR FORCE INST OF TECH 3/3
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
J R HOLTEN DEC 85 AFIT/DS/ENG/85D-1

UNCLASSIFIED F/G 17/7 NL



MICROCOPY RESOLUTION TEST CHART

```
C
      SUBROUTINE MARF(LMATCH,IBM1,IBM2)
C
C       Refines the current match guess in LMATCH  and stores
C     improved locations and a measure of mismatch (quality of
C     match where small values are better).
C
C
C       ON ENTRY:
C            LMATCH(1)       The match in the opposite image.
C            LMATCH(3)       initial slice 1 x-location guess.
C            LMATCH(4)       initial slice 2 x-location guess.
C            LMATCH(5)       max map shift for slice 1.
C            LMATCH(6)       max map shift for slice 2.
C            LMATCH(7)       edge side flag, -1=left,+1=right
C            LMATCH(8)       The vertical location index.
C
C       ON EXIT:
C            LMATCH(2)       best (smallest value) quality measure
C                            found in the search space around the
C                            guess.
C            LMATCH(3)       best position in slice 1.
C            LMATCH(4)       best position in slice 2.
C
      COMMON /DEBUG/ IDEBF
      DIMENSION LMATCH(8)
      DIMENSION IBM1(17),IBM2(17),IBLK(32)
C
C       Get the location and retrieve maps from the NOVA if necessary.
      IYLOC=LMATCH(8)
      IES=LMATCH(7)

      IF (IBM1(1) .NE. 1) CALL NBLOB(1,LMATCH(3),IYLOC,IES,IBM1(1))
      IF (IBM2(1) .NE. 1) CALL NBLOB(2,LMATCH(4),IYLOC,IES,IBM2(1))
      IF ((IDEBF .AND. 32) .EQ. 0) GOTO 70
C
C       Write out compared bit maps on debug flag 32.
      WRITE(12,1000)
1000  FORMAT("  Comparing bit maps.")
      WRITE(12,1003)
1003  FORMAT(1X,2(4X,"................."))

      IMASK=1
      DO 40 J=1,16
      DO 20 I=1,16
      IBLK(I)="  "
      IBLK(I+16)="  "
      IF (IBM1(I+1) .AND. IMASK) IBLK(I)="**"
      IF (IBM2(I+1) .AND. IMASK) IBLK(I+16)="**"
20    CONTINUE
      IMASK=ISHFT(IMASK,1)
      WRITE(12,1001) (IBLK(I),I=1,32)
```

```
1001    FORMAT(" ",2(4X,".",16A1,"."))
40      CONTINUE
        WRITE(12,1003)

70      CONTINUE

C
C         Get the maximum map shift for this pair.
        IWST=MIN0(LMATCH(5),LMATCH(6))
C
C         Set up to get the position with the best match of
C       edge shape.
        MNVAL=256
        MLOC=0
C
C         Cycle through the allowable positions.
        NIWST=-IWST
        DO 200 IPOS=NIWST,IWST
        IOFF=IABS(IPOS)
        INUM=16-IOFF
        IF (IPOS .LT. 0) CALL BMCP(INUM,IBM1(2),IBM2(IOFF+2),N1,N2,ND)
        IF (IPOS .GE. 0) CALL BMCP(INUM,IBM1(IOFF+2),IBM2(2),N1,N2,ND)
        NTOT=N1+N2
        IVAL=256
        IF ((NTOT .NE. 0) .AND. (ND .LT. 128) IVAL=(ND*256)/NTOT
C
C         If this is better than previous ones, then update the markers.
        IF (MNVAL .LT. IVAL) GOTO 200
        IF (MNVAL .NE. IVAL) GOTO 100
C
C         If they are equal, then take the one closest to the center.
        IF (IABS(MLO) .LE. IABS(IPOS)) GOTO 200
100     CONTINUE
C
C         Update the position and best value.
        MNVAL=IVAL
        MLOC=IPOS
200     CONTINUE
        IF (IDEBF .AND. 32) WRITE(12,1002) MLOC,MNVAL
1002    FORMAT(/,"  Best match is at shift = ",I2,"   Quality =",
     A     I4,//)
C
C         Update the "best location" on image #2 only.
        LMATCH(4)=LMATCH(4)+MLOC
        LMATCH(2)=MNVAL

        RETURN
        END
```

```
C
      SUBROUTINE BMCP(INUM,IBM1,IBM2,N1,N2,ND)
C
C       Compares bit maps and returns the number of bits in each
C     and the number of bits different between them.
C
C     ON ENTRY:
C             INUM            Number of words to compare.
C             IBM1,IBM2       Bit maps of the blobs.
C
C     ON EXIT:
C             N1,N2   The number of bits in each.
C             ND      The number of bits in the differences map.
C
C     ROUTINES USED:
C             ICBIT (A function) Counts the number of bits set in
C                     a single word.
C
      DIMENSION IBM1(16),IBM2(16)

      N1=0
      N2=0
      ND=0
      DO 100 ILOC=1,INUM
C
C       This implements an XOR.
      IWD1=IBM1(ILOC)
      IWD2=IBM2(ILOC)
      IAND=IWD1 .AND. IWD2
      IOR=IWD1 .OR. IWD2
      IXOR=(.NOT. IAND) .AND. IOR
C
C       Now count the number of bits set in each, updating totals.
      N1=N1+ICBIT(IWD1)
      N2=N2+ICBIT(IWD2)
      ND=ND+ICBIT(IXOR)
100   CONTINUE

      RETURN
      END
```

```
C
      FUNCTION ICBIT(IWORD)
C
C       Counts the bits set to 1 in IWORD.
C
C     ON ENTRY:
C             IWORD   The word of bits to count.
C
C     ON EXIT:
C             The arguement is passed as the function value.
C
      IMASK=1
      ICNT=0
      DO 100 I=1,16
      IF (IMASK .AND. IWORD) ICNT=ICNT+1
      IMASK=ISHFT(IMASK,1)
100   CONTINUE

      ICBIT=ICNT
      RETURN
      END
```

```
C
      SUBROUTINE PMAL(NFEAT,NMATCH,IMATCH)
C
C       Prints out the match lists for right and left edges of
C       each feature.
      DIMENSION NMATCH(100),IMATCH(100,10,4)

      WRITE(12,1000) NFEAT
1000  FORMAT("1    Feature matches, left and right edges.  ",
     A    "Number of features=",I3)
      WRITE(12,1001)
1001  FORMAT(" IM#1",/," INDX",/," # OF",/," FEATS  ID ",
     A    "  1    2    3    4    5    6    7    8    9   10",/)

      DO 100 INDX=1,NFEAT
      KNDX=INDX+50

      WRITE(12,1020)
      NL=NMATCH(INDX)

      IF (NL .LT. 1) GOTO 20

      WRITE(12,1010) INDX,(IMATCH(INDX,I,1),I=1,NL)
      WRITE(12,1011)      (IMATCH(INDX,I,2),I=1,NL)
      WRITE(12,1012) NL,  (IMATCH(INDX,I,3),I=1,NL)
      WRITE(12,1013)      (IMATCH(INDX,I,4),I=1,NL)
      GOTO 40

20    CONTINUE
      WRITE(12,2010) INDX
      WRITE(12,2011)
      WRITE(12,2012) NL
      WRITE(12,2013)

40    CONTINUE

      WRITE(12,1021)
      NR=NMATCH(KNDX)

      IF (NR .LT. 1) GOTO 70

      WRITE(12,1010) KNDX,(IMATCH(KNDX,I,1),I=1,NR)
      WRITE(12,1011)      (IMATCH(KNDX,I,2),I=1,NR)
      WRITE(12,1012) NR,  (IMATCH(KNDX,I,3),I=1,NR)
      WRITE(12,1013)      (IMATCH(KNDX,I,4),I=1,NR)
      GOTO 90

70    CONTINUE
      WRITE(12,2010) KNDX
      WRITE(12,2011)
      WRITE(12,2012) NR
      WRITE(12,2013)
```

```
90      CONTINUE
1010    FORMAT("  ",I3," INDX ",10(1X,I3,1X))
1011    FORMAT("      QUAL ",   10(1X,I3,1X))
1012    FORMAT("  ",I3," LOC1 ",10(1X,I3,1X))
1013    FORMAT("      LOC2 ",   10(1X,I3,1X))

1020    FORMAT(" ------------------------------------------------")
1021    FORMAT(" ")

2010    FORMAT("  ",I3," INDX  ---   *** NO MATCHES ****")
2011    FORMAT("      QUAL  ---")
2012    FORMAT("  ",I3," LOC1  ---")
2013    FORMAT("      LOC2  ---")

100     CONTINUE
        WRITE(12,1020)
        RETURN
        END
```

```
C
      SUBROUTINE PRBM(IBMAP)
C
C       Prints out a single bit map.
C
      DIMENSION IBMAP(16),IOUT(16)

      IMASK=1
      WRITE(12,1001)
1001  FORMAT(//," Bit map.")
      WRITE(12,1003)
1003  FORMAT(6X,".................")

      DO 200 IYBI=1,16
      DO 100 IXBI=1,16
      IOUT(IXBI)="  "
      IF (IBMAP(IXBI) .AND. IMASK) IOUT(IXBI)="**"
100   CONTINUE
      WRITE(12,1000) IYBI,(IOUT(I),I=1,16)
1000  FORMAT(" ",I3," .",16A1,".")
      IMASK=ISHFT(IMASK,1)
200   CONTINUE
      WRITE(12,1003)
      WRITE(12,1002)
1002  FORMAT(/," ")

      RETURN
      END
```

```
C
      SUBROUTINE LCSL(IYLOC,NFEAT1,NMATCH,IMATCH,POS)
C
C       Generates a list of camera relative coordinates for the
C     matched blobs in the match list.  Uses only the best match in
C     each list in IMATCH.
C
C             by James R. Holten III, 21 Oct 1985
C
C     ON ENTRY:
C             IYLOC   Vertical position index for the slice.
C             NFEAT1  Number of features in the reference slice.
C             NMATCH  Array inluding the length of each list.
C             IMATCH  Array of lists of potential matches.
C                     for each match
C                             1--index of potential match
C                             2--measure of "goodness" of match
C                             3--position of match in refimage.
C                             4--position of match in other image.
C
C     ON EXIT:
C             POS     Array of positions, for each
C                     1--range
C                     2--horixontal offset to right
C                     3--vertical offset down
C                     4--range error estimate
C                     5--horizontal error estimate
C                     6--vertical error estimate
C                             (see dissertation App A for errors)
C

      COMMON /CALIB/ DSEP,DVIRT
      COMMON /BLOBS/ NB1,IBLB1,NB2,IBLB2
      DIMENSION IBLB1(17,100),IBLB2(17,100)
      DIMENSION NMATCH(100),IMATCH(100,10,4),POS(6,100)

C
C       Compute a constant.
      DCONS=DSEP*DVIRT
      DVI=1.0/DVIRT
      DD=IYLOC-60
      ELMAX=0.5
      ERMAX=0.5
      EDISP=ELMAX+ERMAX
      EDMAX=0.5


C
C       Cycle through the features in the reference blob list.
      DO 100 INDX=1,NFEAT1
      JNDX=INDX+50
C
```

```
C         First match the left edges.
          POS(1,INDX)=-1
          IF (NMATCH(INDX) .LE. 0) GOTO 15
C
C         Get the best match from the match list.
          NM1=NMATCH(INDX)
          DO 10 J=1,NM1
          DL1=IMATCH(INDX,J,3)-60
          DR1=IMATCH(INDX,J,4)-60
C
C         Get the range.
          DISP1=D1-DR1
          IF (DISP1 .LT. 1) GOTO 5
          IF (EDISP .GE. DISP1) GOTO 5
          DP1=DCONS/DISP1
          POS(1,INDX)=DP1
C         x-offset
          DX1=DL1*DP1*DVI
          POS(2,INDX)=DX1
C         y-offset
          DY1=DD*DP1*DVI
          POS(3,INDX)=DY1
C
C         Get the range error.
          DENOM=DISP1*DISP1-DISP1*EDISP
          EPMAX=DCONS*EDISP/DENOM
          POS(4,INDX)=EPMAX
C         x-offset error.
          EXMAX=(ABS(DL1)*EPMAX+DP1*ELMAX+EPMAX*ELMAX)*DVI
          POS(5,INDX)=EXMAX
C         y-offset error.
          EYMAX=(ABS(DD)*EPMAX+DP1*EDMAX+EPMAX*EDMAX)*DVI
          POS(6,INDX)=EYMAX
5         CONTINUE
          IF (POS(1,INDX) .GT. 1.0) GOTO 15
10        CONTINUE
15        CONTINUE
C
C         Now get the right edges of the regions.
          POS(1,JNDX)=-1
          IF (NMATCH(JNDX) .LE. 0) GOTO 35
C
C         Compute the locations for the best match.
          NM2=NMATCH(JNDX)
          DO 30 J=1,NM2
          DL2=IMATCH(JNDX,J,3)-60
          DR2=IMATCH(JNDX,J,4)-60
C
C         The range.
          DISP2=DL2-DR2
          IF (DISP2 .LT. 1) GOTO 25
          IF (EDISP .GE. DISP2) GOTO 25
```

```
      DP2=DCONS/DISP2
      POS(1,JNDX)=DP2
C       x-offset
      DX2=DL2*DP2*DVI
      POS(2,JNDX)=DX2
C       y-offset
      DY2=DD*DP2*DVI
      POS(3,JNDX)=DY2
C
C       Get the range error.
      DENOM=DISP2*DISP2-DISP2*EDISP
      EPMAX=DCONS*EDISP/DENOM
      POS(4,JNDX)=EPMAX
C       x-offset error.
      EXMAX=(ABS(DL2)*EPMAX+DP2*ELMAX+EPMAX*ELMAX)*DVI
      POS(5,JNDX)=EXMAX
C       y-offset error.
      EYMAX=(ABS(DD)*EPMAX+DP2*EDMAX+EPMAX*EDMAX)*DVI
      POS(6,JNDX)=EYMAX
25    CONTINUE
      IF (POS(1,JNDX) .GT. 1.0) GOTO 35
30    CONTINUE
35    CONTINUE


100   CONTINUE


      RETURN
      END
```

```
C
      SUBROUTINE PRFPOS(NP,POS)
C
C       Prints out the list of feature positions.
C
      COMMON /CALIB/ DSEP,DVIRT
      DIMENSION POS(6,100)

      WRITE(12,1000) NP,DVIRT,DSEP
1000  FORMAT("1    Feature 3-D positions.  NP=",I3,"  dv=",F6.2,
     A    "  ds=",F6.3,/,
     B    " INDX      dp        dx        dy        ep",
     C    "        ex        ey")

      WRITE(12,1004)
      DO 200 INDX=1,NP
C
C       Repeat for right and left edges.
      DO 150 J=1,2
      JNDX=(J-1)*50+INDX
      IF (POS(1,JNDX) .GT. 0) GOTO 100
      WRITE(12,1001) JNDX
1001  FORMAT(" ",I3,10X,"No data point.")
      GOTO 150

100   CONTINUE
      WRITE(12,1002) JNDX,(POS(I,JNDX),I=1,6)
1002  FORMAT(" ",I3,5X,6(F7.3,3X))
150   CONTINUE
      WRITE(12,1004)
1004  FORMAT(" ------------------------------------------")
200   CONTINUE

      WRITE(12,1003)
1003  FORMAT(//,"1 ")

      RETURN
      END
```

```
C
      SUBROUTINE PNPL(IER)
C
C       Saves the position list out on disk for later use or
C     display.
C
      COMMON /POSIT/ NP,POS
      DIMENSION POS(640)
C
C       Write it out to a file.
      CALL DELETE("JHDATA:POSIT.DT")
      CALL OPEN(1,"JHDATA:EDAT.DT",3,IER)
      CALL WRBLK(1,0,NP,5,IER)
      CALL CLOSE(1,IER)
      CALL RENAM("JHDATA:EDAT.DT","JHDATA:POSIT.DT",IER)

      RETURN
      END
```

Vita

James R. Holten III was born on 18 April, 1949, in Paso Robles, California, to Mr. and Mrs. James R. Holten jr. He graduated from Illinois Valley High School, Cave Junction, Oregon, in 1967. In 1973 he graduated from Oregon State University with a Bachelor of Science in Mathematics and a Bachelor of Science in Computer Science. After graduation he enlisted in the Air Force, and in 1975 was admitted into Officer Training School. After commissioning on 16 July, 1975, he spent six years as a Missle Warning Programming Officer on phased array warning sites at Eglin Air Force Base, Florida; Otis Air Force Base, Massachusetts; and Beale Air Force Base, California. During this time he maintained computer programs for communications, radar function control, real time operating systems, and automated fault detection and isolation. In 1980 he entered the Air Force Institute of Technology, graduating in December 1982 with a Master of Science in Computer Systems. He is married to the former Raymona A. Clinkingbeard of Ft. Walton Beach, Florida, and they have six children, Erin, Donald, James, Aghavni, Arlene, and Roger.

Permanent Address: 5839 Westside Rd.
Cave Junction, Oregon 97523

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A164 202

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/DS/ENG/85D-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, OH, 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT NO.
11. TITLE (Include Security Classification): A ROBOT VISION SYSTEM (Unclassified)	
12. PERSONAL AUTHOR(S): James R. Holten III	
13a. TYPE OF REPORT: PhD Dissertation; 13b. TIME COVERED: FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): 85 Dec. — 15. PAGE COUNT: 206
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	
17	08	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Computer Vision, Stereo Robot Vision,

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Short Abstract--- Chairman: Dr. Matthew Kabrisky

Robot vision algorithms are discussed, including camera characterization, image feature extraction, image feature registration, and feature 3-space location estimation. Some aspects of stereo camera vision allow simplifications, but accuracy can be a problem at a distance. An alternative technique is suggested for further study.

Approved for public release: IAW AFR 190-1.
7 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Matthew Kabrisky, Professor	22b. TELEPHONE NUMBER (Include Area Code): (513) 255-3276 — 22c. OFFICE SYMBOL: AFIT/ENG

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

END

FILMED

3 -86

DTIC