A132720

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 18249.2-EL	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Computer Analysis of Time-varying Imagery	5. TYPE OF REPORT & PERIOD COVERED: Final Report 2/82 - 1/83 (1 Feb 82 - 31 Jan 83)
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Wesley E. Snyder, Sarah A. Rajala	8. CONTRACT OR GRANT NUMBER(s): DAAG-29-82-K-0070
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Electrical Engineering Department, North Carolina State University, Raleigh, NC 27650	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709	12. REPORT DATE: Jan. 1983
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

Use title on 1473 per ARO

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Moving target detection, window tracking and predicting, descriptor matching, relaxation labeling, motion interpretation and understanding, moving target representation, production system, time-varying imagery, background motion analysis.

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

This report documents the study of computer analysis of time-varying imagery. The research is on the development of techniques which locate, track, identify and (cont. on next page)

over

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

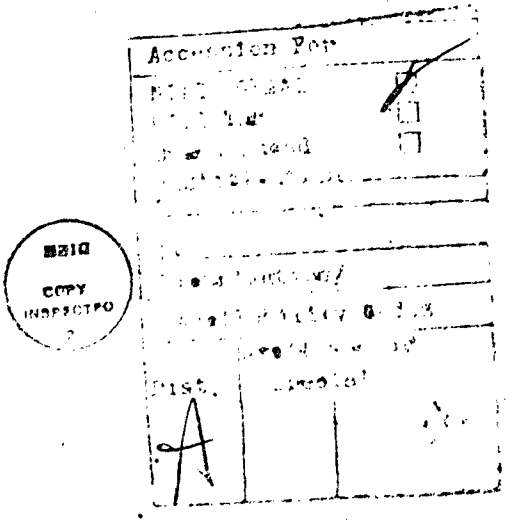
UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

characterize a single rigid target moving in an image sequence in which the camera platform is also moving.

The accomplished work is more than what was originally proposed. A summary of the primary products of this research follo

1. A moving target detection algorithm;
2. A window tracking and predicting algorithm;
3. A corner matching algorithm;
4. A near natural language mechanism for describing motion;
5. A procedure for extracting target images;
6. A background motion estimation algorithm; and
7. A database of time-varying stereo imagery.

Further research based on the accomplished work is briefly discussed.



Studies in Time-Varying Imagery

FINAL REPORT

Submitted to

Army Research Office

Covering the period February 1, 1982 - January 31, 1983

by

Wesley E. Snyder
Associate Professor

Sarah A. Rajala
Assistant Professor

Department of Electrical and Computer Engineering

N C S U

May 15, 1983

TABLE OF CONTENTS

1. INTRODUCTION

For the past year, we have performed a study of the computer analysis of time-varying imagery. The original goal of this study was to develop techniques which locate, track, identify and characterize a single rigid target moving in an image sequence in which the camera platform (the observer) is also moving. However, the work which has been accomplished is more than that proposed. The requirement of a single target is dropped. In addition, the work on motion description has been completed.

The first progress report describing a motion detection algorithm was made and submitted to ARO on June 30, 1982. This final report will cover the progress during the period between July 1, 1982 and January 31, 1982.

A paper [1] describing a developed motion detection algorithm was published on the Proceedings of the 6th International Conference on Pattern Recognition in October 1982 (see Appendix A).

The participating scientific personnel in this study are Wesley E. Snyder, Sarah A. Rajala and I-Sheng Tang. Mr. Tang completed his doctoral research work in January 1983. A part of his dissertation [2], concerned with computer analysis of motion in time-varying imagery containing multiple rigid moving objects, is included in Appendix C.

2. RESEARCH PROBLEM

The original goal of this study was to develop techniques which locate, track, identify, and characterize a single, rigid target moving in an image sequence in which the camera platform is also moving. The specific research tasks are as follows:

1. Obtain a realistic data base;
2. determine the optical flow;
3. process the optical flow, and
4. develop a model of the target.

Items 1-3 have been completed successfully within the stated assumption. Item 4 has been addressed, and progress made toward a solution. In addition, it has been possible to relax some of the assumptions stated in the proposal.

The requirement of dealing with a single moving target has been dropped; multiple moving targets (up to 3 targets) is manageable by the developed techniques. In addition, a near-natural language description of the motion of each moving target has been developed.

3. DATA BASE

Primarily, there are two classes of time-varying imagery for this study. The first class, those with high detail, is exemplified by a real-world image sequence containing a street scene. The second class, rapid motion but simple background, is represented by several laboratory generated sequences of radio controlled toy cars having different types of movement. These image sequences were used for the study of the problems under the condition of multiple targets moving in an image sequence with a stationary camera. The second class of time-varying imagery is comprised of two synthetic image sequences and several real-world FLIR image sequences (from Martin Marietta Aerospace, Orlando, FL). These image sequences were used for study of background motion under the condition that a single target is moving in an image sequence with a moving camera (sensor).

These test image sequences are shown in Appendix B.

4. SUMMARY OF RESULTS

In this section, a summary of the results of this study to data are listed in the following:

1. Moving target detection algorithm

A simple frame differencing algorithm is followed by region growing on the thresholded difference picture. The motion areas are then linked into a graph structure. Rectangular windows are placed around the linked areas. Each window may contain a single target, a part of a target (e.g. when occluded by a foreground), or more than one target (e.g. when occluded by an another target).

2. Window tracking and predicting algorithm.

The moving target detection algorithm provides a set of windows which are placed around the targets. In this algorithm, a basic set of mapping rules are developed to handle the tracking from frame-to-frame. In addition, a set of rules are used to correct for imprecise windows due to jitter and time-varying noise. Another set of rules are also developed to predict target locations when occluded or when a window containing two targets must be split. This algorithm is capable of handling a variety of situations, particularly in the occlusion problems.

3. Corner matching algorithm.

A corner matching algorithm, using relaxation labeling, to derive the direction of motion, in the 2-D sense, was developed. This algorithm is capable of searching for corresponding corners of a target in a pair of consecutive frames even when partial occlusion occurs.

4. Motion description output.

A capability of near-natural language description of the motion is implemented. Qualitative description of the target's motion and relationships between targets in an image sequence is provided through processing of the results of the low-level image analysis.

5. Extraction of moving targets.

A procedure was developed for extracting images of moving targets. Approximation of a target's image can be acquired. This result will provide a further study of target description.

6. Background motion estimation.

In the case of a moving camera (sensor) producing an image sequence, the apparent background motion reflects the sensor motion. If apparent background motion can be analyzed, then the analysis can be coupled with a camera model to provide ground topology or sensor platform motion. In addition, knowledge of the background motion can make frame-to-frame registration possible. The algorithms developed for a stationary camera then applicable for a processed image sequence generated by a moving camera.

A transform-based approach (see Appendix D) is developed to estimate background motion in a FLIR image sequence. The calculation of sensor motion parameters is currently being investigated. The current algorithm is applicable for translational sensor motion only.

5. CONCLUDING REMARKS

We have accomplished the proposed research on a number of aspects of the motion analysis problem in time-varying imagery. The developed techniques are capable of locating, tracking, identifying, and characterizing a single, rigid target moving in an image sequence. With moving camera, a transform-based algorithm is capable of estimating the background motion. However, incorporating the information of background motion with the developed techniques under the coudition of a stationary camera remains for further investigation.

In addition to the accomplished work stated in above, the capability of handling more than one target in the developed techniques and providing near-natural language output in the developed system have been accomplished.

The accomplished work has concentrated on motion analysis. A further extension of this work might focus on target description, incorporating motion information. Another possible extension would be the investigation of the feasible of hardware implementation, described in the next paragraph, and the appropriate system architecture.

The implemented moving target detection algorithm can be modified to perform the differencing and region growing steps in a parallel computation. These two major bottlenecks might be removed by a hardware implementation of the modified algorithm. Another possibility in hardware implementation is the window tracking algorithm. Since a window is simply represented by a quadruple (left, right, top, bottom), this algorithm has potential to be implemented in hardware.

We feel the accomplished work is adequate for providing the basis of the aforementioned future research. A proposal details the future research has been submitted to the ARO.

6. APPENDICES

APPENDIX A

"Extraction of Moving Objects in Dynamic Scenes"

by

I-Sheng Tang, Wesley E. Snyder and Sarah A. Rajala

EXTRACTION OF MOVING OBJECTS IN DYNAMIC SCENES*

I-Sheng Tang, Wesley E. Snyder and Sarah A. Rajala

Department of Electrical Engineering
North Carolina State University
Raleigh, North Carolina

Abstract

This paper presents a method of identifying the images of moving objects in real world dynamic scenes where both the moving objects and the background are nonhomogeneous. We show that region growing on a thresholded difference picture followed by linking neighboring regions determines a window around the image of each moving object. Once the windows have been determined, we can separate images of moving objects from stationary scene components using a simple pixel-based process. Some refining processes are discussed, and some experiments are demonstrated.

1. Introduction

The existing schemes for extracting the images of moving objects from dynamic scenes encounter difficulties when the scenes contain nonhomogeneous moving objects and a nonhomogeneous background and the contrast between objects and their surrounding background is low.

Earlier efforts [3,8] are primarily to analyzing objects having translational movement and a sufficiently contrasting background. Jain et al. [4] utilize the properties of first and second-order difference pictures to extract the image of a moving object which may have simultaneous translational and rotating movements. However, the restriction of a sufficient contrast against the stationary scene component has not been alleviated.

If the contrast is well defined, classifying regions in a difference picture and using region growing and region decaying processes to extract the images of moving objects [5] can be used. However, a recent report [6] shows that the classification algorithm is not successful in handling an image sequence with a low contrast, nonhomogeneous background and nonhomogeneous objects.

In this paper, we describe an approach for extracting the images of moving objects under the conditions of nonhomogeneous moving objects, nonhomogeneous background and poor contrast between objects and background.

* This work was supported by the Army Research Office under Grant DAAG-29-82-K-0070.

2. Assumptions About Dynamic Scences

We first describe some assumptions about the analyzed dynamic scenes. This will also indicate some of the limitations of the proposed algorithm.

We first assume that the direction of illumination is fixed so that variations of grayvalue at a fixed pixel position are mostly due to motion. We assume that the television camera is stationary so that spatial coordinates of images are fairly well aligned. No radical change in the shape and position for the same moving object in any pair of consecutive frames is allowed. We assume the images of moving objects have motion over at least several pixels per frame in at least one direction. This will restrict our attention to objects which are not slowly moving. The images of moving objects should also have a reasonably large size. Although we assume the condition of poor contrast between the images of moving objects and the background, we should mention that "poor contrast" means difficulty for automated edge detection.

3. Description of the Algorithm

We approach this problem in two major steps. First we try to place a window around an image of the moving object. Then we extract the image of the moving object within the window.

The algorithm for placing a window around an image of a moving object is the following:

1. generate a difference picture (DP),
2. generate a thresholded difference picture (TDP) with a threshold value (THD),
3. segment TDP into regions (STDP), and,
4. link neighboring regions, in which each region contains pixels above a threshold, then place a window around each moving object and delete small windows.

The first step is simply the subtraction of the gray level at each pixel position in one frame with respect to the corresponding pixel in the other frame followed by taking the absolute value of the result of subtraction.

Next, the DP is thresholded. All the values less than THD are set to zero.

CH1801-0/82/0000/1143 $00.75 © 1982 IEEE

The above two steps are essentially a high-pass (temporal) filtering process. Because we suppress low frequencies represented by slow changes in gray level at the same pixel position between two consecutive frames, the analyzed dynamic image sequence can not allow images of very slow moving objects.

We then apply a region growing technique [10] on TDP in step 2. Generally, the largest region happens to be the stationary component plus homogeneous parts of the images of the moving objects. This region will be disgarded in step 4. Because of noise from image recording processes and insufficient evidence of the existence of a moving object, regions with a small number of pixels will also be disgarded.

In the last step, we use a single-linkage algorithm [2] to link neighboring regions together. We represent each region as a point in a two-dimensional image plane at the center of gravity of its extremes. The minimum distance between clusters K1 and K2 is measured as

$$Dmin(K1,K2)=\min||X1-X2||, \text{ where } X1\in K1 \text{ and } X2\in K2.$$

The algorithm will be terminated when Dmin between nearest clusters exceeds a threshold. The results can be thought of as partitioning a minimal spanning tree by breaking any edge with length over a threshold.

The segmented, thresholded difference pictures (STDPs) have a large number of regions located in the overlapped interior of the images of the moving objects, while homogeneous images of moving objects do not have this same property. Thus, a single-linkage algorithm is sufficient. If the images of moving objects are not close to each other, then it is not likely that a given window will contain two or more moving objects. Windows should be conservative, so the extremes of the linked regions in the x and y direction are used to form windows.

The algorithm described above can be understood more easily through the use of some simple examples. For simplicity, assume the background environments are the same for images of a moving object in frame i and i+1. Then for a homogeneous region in each frame representing the image of a moving object having a translational movement parallel to their edges generates two regions in the difference picture (refer to figure 1). However, more regions can be generated in the difference picture of nonhomogeneous images of a moving object (e.g. five regions constitute the image of a moving object in figure 2).

Figure 3 shows that the proposed algorithm works well if some regions constitute an image of a moving object having poor contrast against the background. In figure 3, two regions, one in the upper right and the other in the lower left, do not have sufficient contrast against their backgrounds. Using the algorithm described above, the determined window still contains the image of the moving object.

The above algorithm can be applied for each pair of consecutive frames through the whole image sequence, resulting in the detection of motion. The next step is the extraction of the image of the moving object within each window.

In the poor contrast situation, the extraction is not reliable using only a single frame difference pair. Information must be accumulated over time. Knowing that the image of a moving object only temporarily occupies a specific region in the picture, we can extract the moving object utilizing information derived from a partial image sequence in which the moving object is not located within the corresponding windows. That is, having detected motion within a window, we search for a set of frames in which the moving object has left that window. From those frames, we can accurately model the background and thus extract the moving object precisely.

Two pixel-based processes for extracting moving objects are used. Suppose we have n frames, say from j to j+n-1. Further, suppose the moving object is not located within the corresponding window in frame i, the desired frame for extracting the image of the moving object. The n absolute difference pictures between frame i and frame j through j+n-1 are formed. At each pixel position the number of these n difference pictures whose value exceeds a threshold is counted. If this count exceeds a threshold p where $0\le p\le n$, then a "1" is marked to denote a moving object. Otherwise, a "0" is marked to indicate background. Alternately, we can compute mean and standard deviation of gray level at each pixel position for the n frames. Then we accept a pixel as belonging to the image of a moving object if its gray level in frame i is outside some multiple of standard deviation around the mean.

4. Results

A real world traffic scene is used to test the proposed algorithm. There are four moving objects in the scene: a black car (CAR) going east, a taxicab (CAB) turning right and away, a black wagon (WAGON) going west, and a person in the left upper corner walking down the street. The black cars, CAR and WAGON, have poor contrast against the background. The images of the moving objects are nonhomogeneous, especially the CAR and WAGON. The image sequence also has foreground, a tree, in the right lower corner of the image.

All the images of the PERSON are partially detected, except in frame 3. The detection failure is due to the size of the image of the PERSON and the fact that the bottom part of the PERSON has a very low contrast against its background. The CAR in each frame is detected; likewise the CAB is detected in each frame. However, a larger segmented region close to those of CAB is mislinked in frame 2 and frame 4 due to noise. In detecting the WAGON, due to the interference of the foreground and background the results in frame 7 and frame 8 are not satisfactory. Due to limited space, we only show the results of the window placing algorithm applied on frame 1 and frame 8 in figure 4.

Table 1 shows the coordinates of the windows containing the images of the moving objects. This data indicate that the CAR and WAGON are proceeding in approximately translational movement and moving into the visual field. The CAB is at about the same area except the radical change in the size of the window in frame 2 and frame 4 is due to noise. The PERSON is walking down in a southwest direction.

Figure 5(a) and (b) are the segmented images of CAR.

5. Discussion

Our experiments show that the algorithm is capable of extracting multiple images of moving objects in a nonhomogeneous dynamic image sequence. However, the algorithm is dedicated to handling the situation of insufficient edge information. Otherwise, the approach in [5] is preferred. This peripheral process [7] in motion analysis will direct the attention of higher level processes and reduce the computational burden.

We can refine our results (e.g. the windows containing the CAB in frame 2 and frame 4) using the assumption that no radical change in the shapes and positions of the images of moving objects can occur. A more elaborate method such as the matching of regions [1,9] in the corresponding windows may improve the result.

In the process of moving object extraction, isolated pixels may be deleted or added as members of an image of a moving object depending on their connectedness with respect to their neighboring pixels. However, further study will be required to implement such a technique.

6. Acknowledgement

We would like to thank professor H.-H. Nagel of the University of Hamburg for providing the traffic scene.

References

1. L. Dreschler and H.-H. Nagel, "Using 'affinity' for extracting images of moving objects from TV-frame sequence," IfI-H-8-44178, University of Hamburg, Feb. 1978.

2. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, New York, 1973.

3. R. Jain, D. Militzer, and H.-H. Nagel, "Separating nonstationary from stationary scene components in a sequence of real world TV-images," IJCAI 77, Vol. 2, pp. 612-618, 1977.

4. R. Jain and H.-H. Nagel, "On the analysis of accumulative difference pictures from image sequences of real world scenes," PAMI-1, No. 2, pp. 206-214 April 1979.

5. R. Jain, W. N. Martin, and J.K. Aggarwal, "Segmentation through the detection of changes due to motion," Computer Graphics and Image Processing, Vol. 11, pp. 13-34, Sept. 1979.

6. R. Jain, "Extraction of motion information from pheripheral processes," PAMI-3, No. 5, pp. 489-503, Sept. 1981.

7. W. N. Martin and J.K. Aggarwal, "Survey: Dynamic scene analysis," Computer Graphics and Image Processing, Vol. 7, pp. 356-374, June 1978.

8. H.-H. Nagel, "Formation of an object concept by analysis of systematic time variations in the optically perceptible environment," Computer Graphics and Image Processing, vol. 7, pp. 149-194, April 1978.

9. K. Price and R. Reddy, "Change detection and analysis in multispectral images," IJCAI 77, Vol. 2, pp. 619-625, 1977.

10. W. E. Snyder and A.E. Cowart, "An iterative approach to region growing," ICPR 80, Vol. 1, pp. 348-351, Dec. 1980.

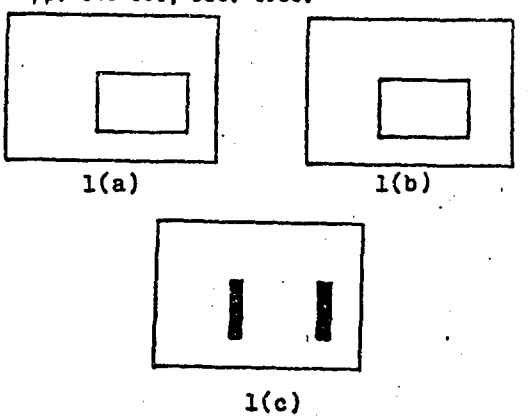


Fig. 1 (a) Frame i, (b) Frame i+1, (c) DP generated from frame i and i+1.

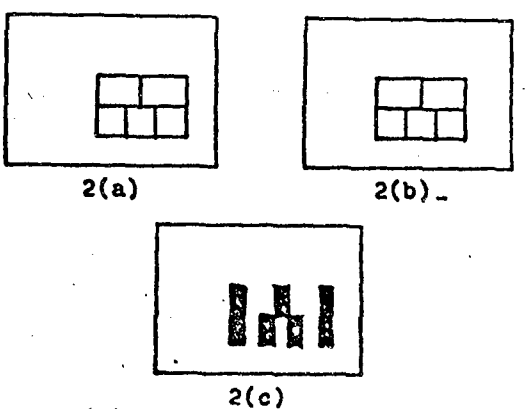


Fig. 2 (a) Frame i, (b) Frame i+1, (c) DP generated from frame i and i+1.

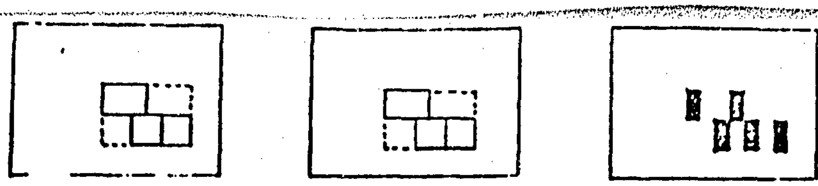
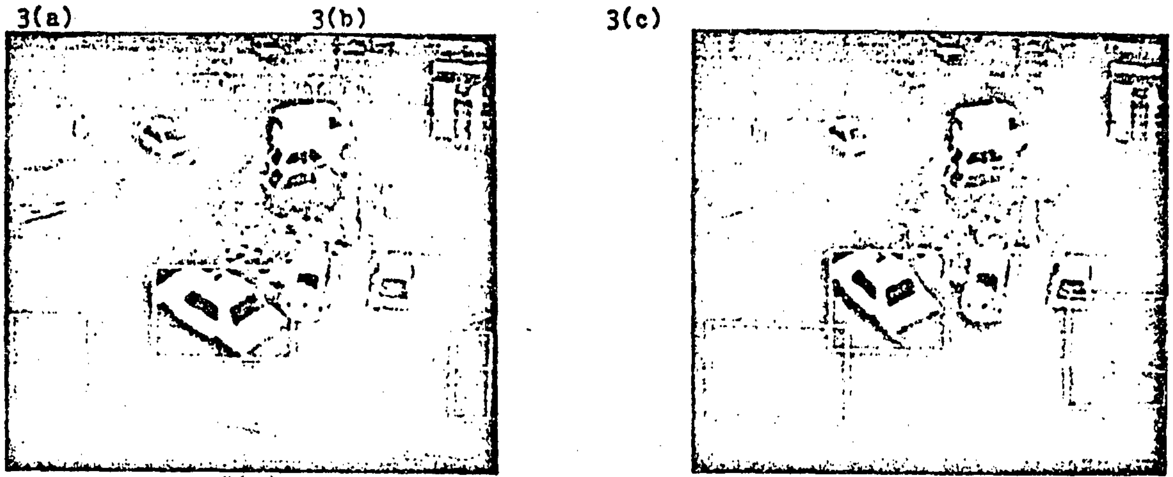


Fig. 3 (a) Frame i, (b) Frame i+1, (c) DP generated from i and i+1.

3(a) 3(b) 3(c)



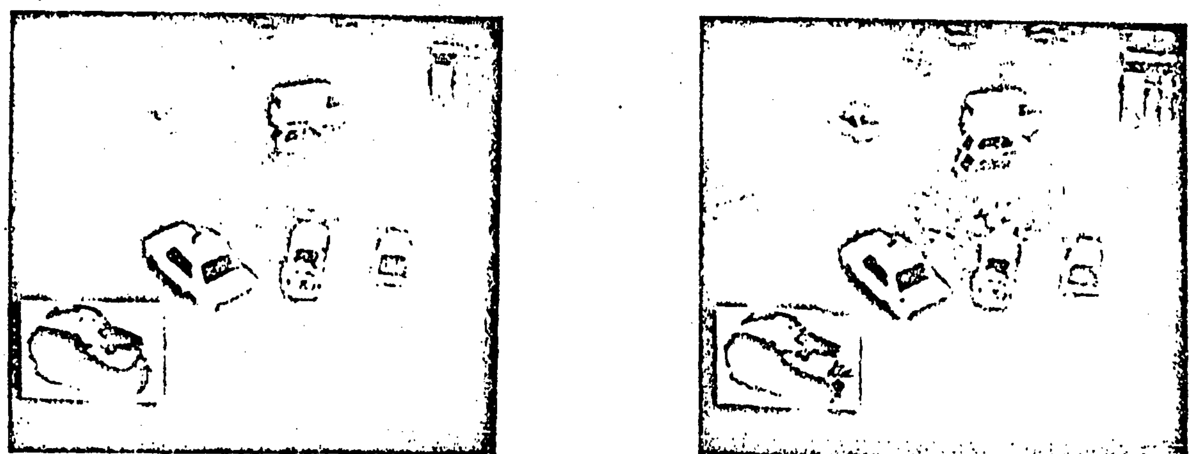
4(a) 4(b)

Fig. 4 (a) Frame 1, (b) Frame 8.

Table 1

[illegible]	MOVING OBJECT			
	[illegible]	CAR	CAR	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	NIL	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

*(A,B,C,D) denotes four coordinates of the extremes, (B,A), (B,C), (D,A) and (D,C), of a window. The first element, e.g. B of (B,A), represents y coordinate. The second element, e.g. A of (B,A), represent x coordinate.

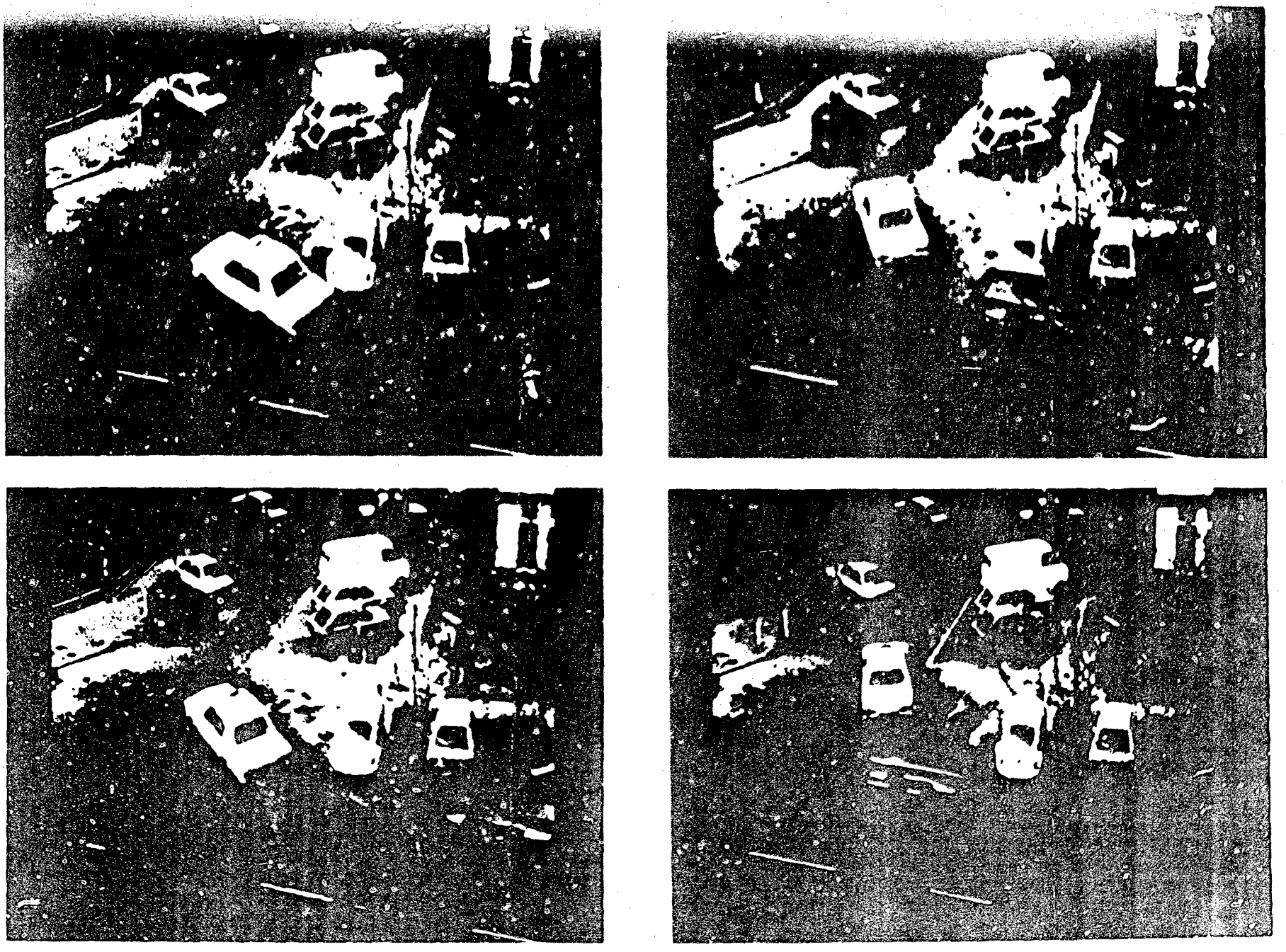


5(a) 5(b)

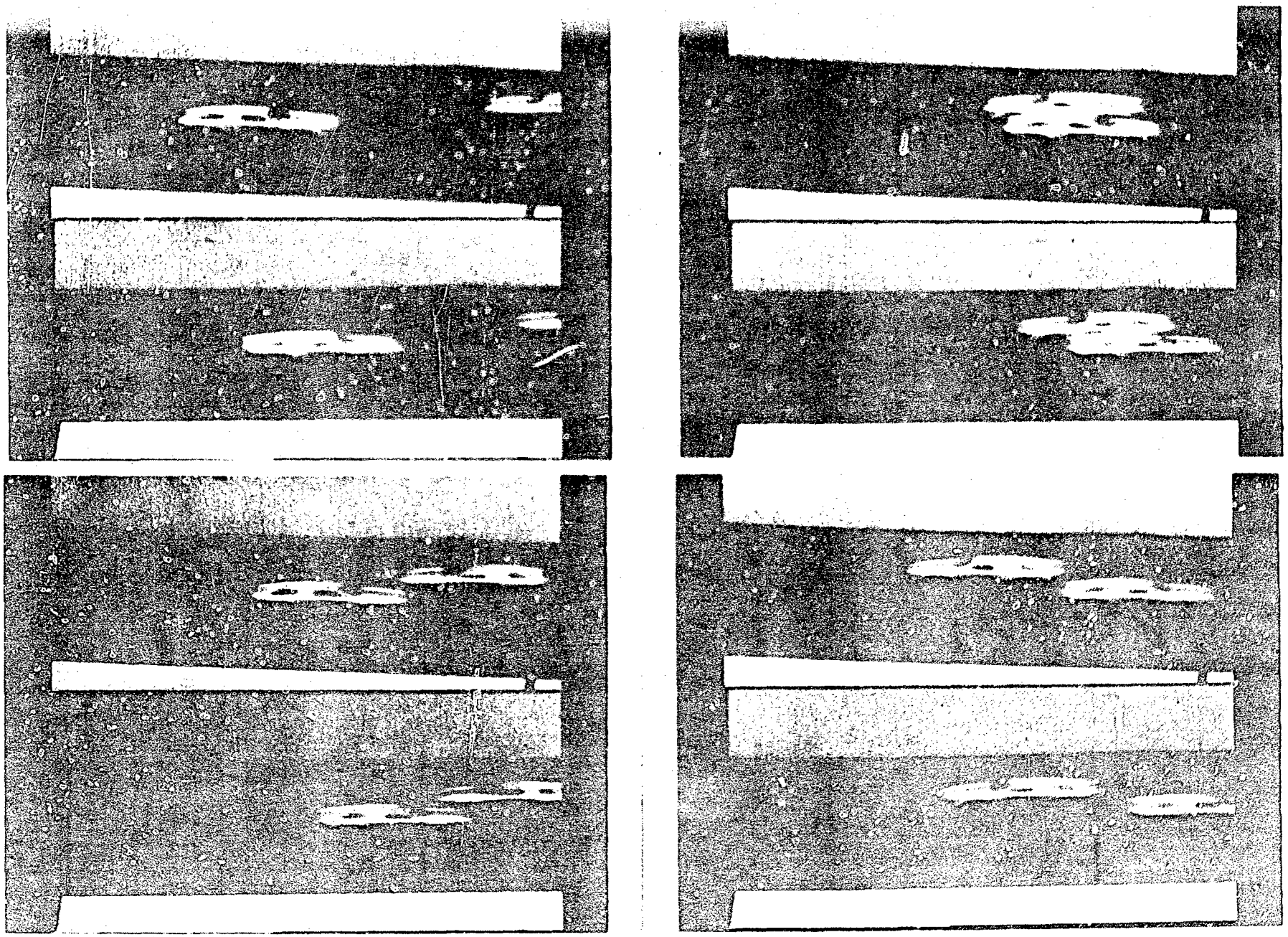
Fig. 5 Extracted image of CAR in frame 8 using (a) voting, (b) mean and standard deviation.

APPENDIX B

Image Data Bases



Sequence 1. A series of 32 frames with rigid body motion, including translation and rotation. From Hamburg, Germany (courtesy A. Nagle).



Sequence 2. A series of stereo/TVI frames, one field from each of two cameras. Several sequences are available, including high and low velocities, occlusion, one and two objects (courtesy MA/com Labs).

APPENDIX C

Excerpts from Tang's Thesis

3 REPRESENTATION OF MOVING OBJECTS

In this dissertation, the major concern is analysis of object movements which are derived from low-level processing of digitized image sequences. Some representation of the moving objects is required in order to facilitate storage and to manipulate the information associated with each individual object. The representation of such knowledge is one of the major research areas in artificial intelligence [9], [74].

Types of knowledge can be classified as objects, events, performance, and meta-knowledge [9]. Generally, the representation schemes are: logic, procedural representation, production systems, frames, etc. Depending on the tasks to be accomplished, some representation schemes are more suitable than others. This chapter describes the representation of rigid moving objects in this study.

Described with a PASCAL-like language, a rigid moving object is represented in the following manner.

```
object = record
         object-type : (vehicle, others);
         frame-time: integer;
         label : (obj1, obj2, obj3, ooj4);
         extremes : array [1..4] of 1..512;
         center : array [1..2] of 1..512;
         location : (c-c, c-l, c-r, t-c,
            t-l, t-r, b-c, b-l, b-r);
         horizontal-length : integer;
         vertical-length : integer;
         area : integer;
         no-mv-obj-next : integer;
         no-mv-obj-current : integer;
         no-mv-obj-previous : integer;
         cfptr : array [1..4] of integer;
```



```
         spfptr : integer;
         snfptr : integer;
         moving-direction : (s, n, w, e,
            ne, nw, se, sw, unknown);
         case confidence : boolean of
            true : (heading : integer);
            false : (condition :
                     (no-confidence,
                     not-available))
end;
```

Obviously, the variable object-type is restricted to the vehicle in the image sequences. Other types of moving objects may be included in the system. It should be noted, however, that if two or more types of objects appear in an image sequence, it is likely that an object type recognition process is required, especially if mutual occlusion occurs.

The variable frame-time indicates the time (frame number) at which this object descriptor is valid.

The variable label contains an object's label. For an entire image sequence, if an object stops its movement, the label associated with it will not be used for a newly appearing moving object. If the object resumes its movement, a new label will be assigned. The fact that the two labels represent the same object is detected in later processing.

The variable extremes represents the four extremes of a rectangular window in a compact form. For example, (A,B,C,D) denotes four extreme positions, (B,A), (B,C), (D,A), and (D,C) of a window. The first element, e.g., B of (B,A), represents the y-coordinate. The second element, e.g., A of (B,A) represents the x-coordinate. The variable center represents the position of the center point of a win-

dow represented by the extremes. Furthermore, the center is encoded symbolically to represent the location of a window containing a moving object in an image plane. An image plane is partitioned into nine regions shown in figure 3.1. Each region is labeled to represent its location. When a center is located at a particular region, then the label of that region is assigned to the variable location.

The variables horizontal-length and vertical-length represent the lengths of the horizontal and vertical sides, respectively, of a window. The multiplication of these two values gives a value contained in the variable area.

The variable no-mv-obj-next indicates the number of moving objects in the next frame. Similarly, the variable no-mv-obj-current and no-mv-obj-previous represent the number of moving objects in the current and previous frames.

The variable cfptr is a pointer to descriptors of objects at the same frame time so that the information about other objects can be easily accessed.

The variables spfptr and snfptr are pointers to descriptors for the same moving object at the previous and next frame times, respectively.

The variables cfptr, spfptr, and snfptr can be considered arcs representing relations between moving objects in a semantic network (see figure 3.2). In figure 3.2, each node represents an object and each arc represents a relation. An object is labeled as OBJx(n) to denote the object

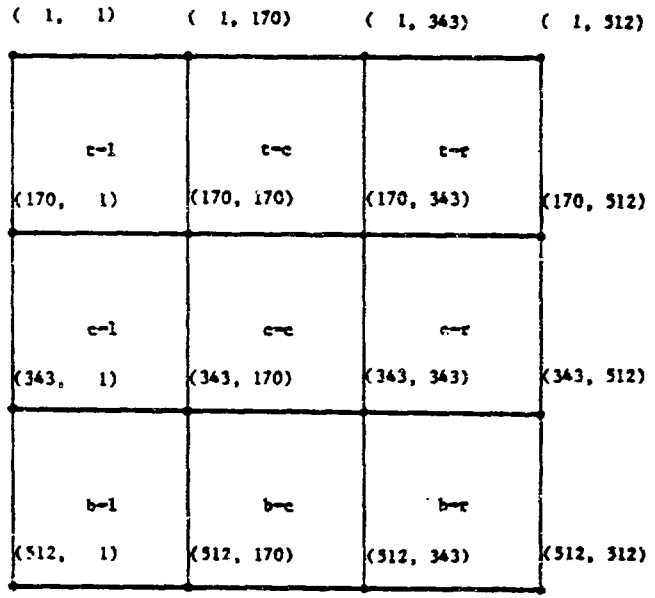


Figure 3.1 A partition of an image plane in location encoding

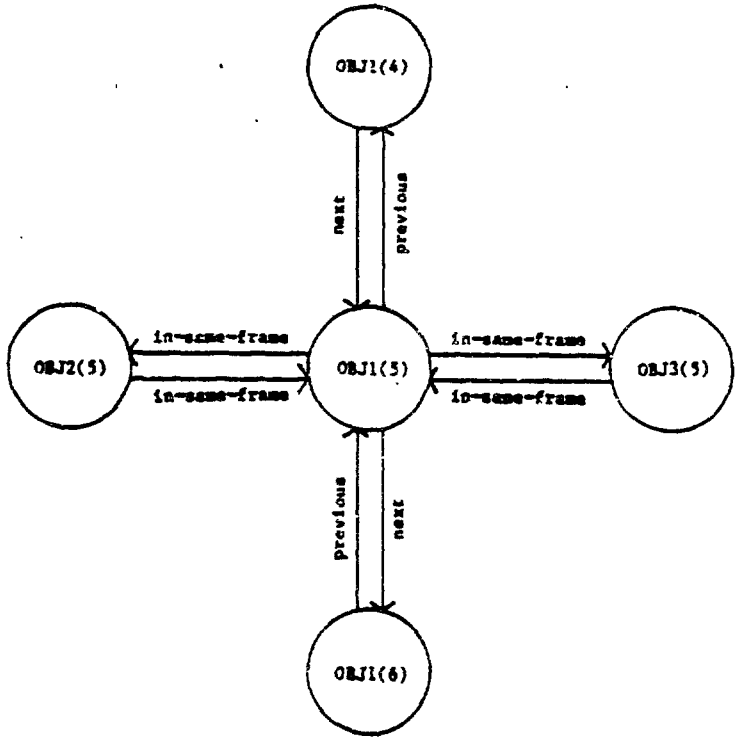


Figure 3.2 Connections between object OBJ1(5) and its neighbors in the semantic network

x at frame time n. An arc can be labeled as "next," "previous," or "in-same-frame," depending on the relation between the two connected nodes.

The variable moving-direction represents the direction of object movement. The symbols representing directions are generated from encoding numeric values. For example, a displacement of 8 and 2 pixels in x- and y-directions (approximately 14 degrees) is encoded as "e," which denotes that the movement is approximately eastbound. Figure 3.3 illustrates the direction quantization used in this system. The special symbol "unknown" is for the objects in the last frame because their moving directions are not available.

If occlusion occurs, the acquired moving direction may not generate a consistent result. This will force us to predict the moving direction based on the previous direction. The variable confidence indicates the confidence of the acquired information about the moving direction. If it is based on the prediction, the numeric value for the moving direction will not be saved. Otherwise, the numeric value for the moving direction will be kept for further use. The variable condition indicates that the acquired moving direction is either not available or not reliable, when the variable confidence has a value of false.

It is worth noting that the symbolic encoding of numeric values for moving directions provides not only for an abstract level manipulation in future processing, but also

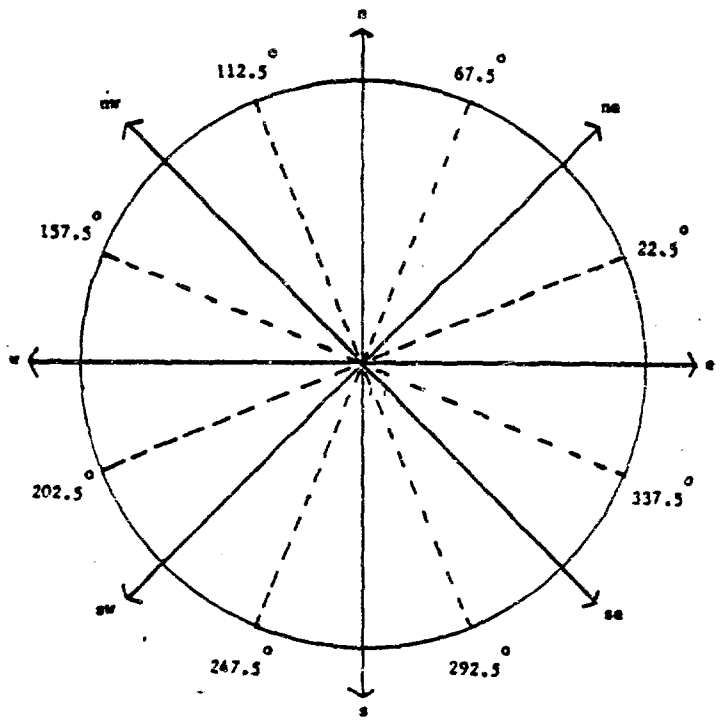


Figure 3.3 A direction quantization

compensates for the imprecise data from the corner matching process, in which the numeric values for moving directions are acquired. A small variation in the numeric values should not be reflected in their encoding. For example, an object's moving directions are 180 and 195 degrees at times n and (n+1), respectively. They are encoded with "w" which means westbound. The same idea was pointed out in [10] in a symbolic encoding ECG signal (p. 372) and in [9] in a general discussion about the scope and grain size of a representation scheme (p. 147-148).

In summary, a uniform representation of moving objects is presented. Each moving object is an integral unit comprising its physical properties and other important information. Objects having temporal adjacency are linked for easy access. Having now described the representational structure for moving objects, in the next chapter the utilization of this structure to analyze the motion of objects is shown.

4 THE IMPLEMENTED SYSTEM

4.1 Overview

The underlying purpose of this system is to describe the motion of objects and the relationships between moving objects in image sequences. The block diagram shown in figure 4.1 illustrates the implemented system.

The implemented system accepts real-world image sequences as its input. It begins with locating the moving objects, represented by windows in which the moving objects are contained, in each frame. Next, the same object in each pair of consecutive frames is identified by a tracking algorithm. When two or more objects are too close together in the image plane, resulting in a single larger window containing both of them, or when an object is partially occluded by a stationary foreground, the locations of the moving objects are predicted, based on their previous velocities. The direction of movement for each object in each frame is derived by a feature matching algorithm. Corners are used as the features because the analyzed moving objects in the test data base have corners as prominent descriptors. The information from the above processes is collected in a compact form for each moving object. Finally, the system interprets the information so that object movements and their relationships are described.

One basic problem encountered in the motion analysis of TVI is locating the moving objects (sometimes referred to as

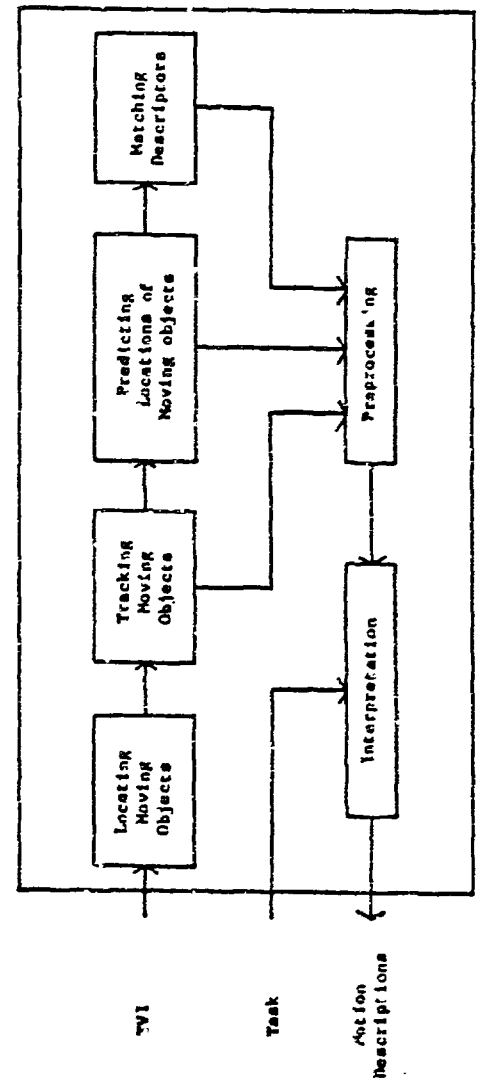


Figure 4.1 System block diagram

"moving target indication," or MTI). This is similar to the peripheral process [38] of the human visual system. At this stage, the locations of moving objects in each frame should be extracted and passed forward to be used for tracking those objects.

Tracking improves the results of the MTI step and provides motion information. The results of this process are then passed to processes which perform prediction.

The results obtained from tracking are used to predict the locations of the moving objects in each frame. The MTI step only detects the presence of motion in a small area. If two moving objects should pass close to each other, the MTI process will lump them together into one window. If a window contains more than one moving object, the window must be split, and the splitting algorithm should be based on the velocity estimated from previous frames. Conversely, if more than one window contains the same moving object, the same algorithm merges the windows. The result of this "split or merge" step is that each window encloses only one moving object.

After splitting or merging, the locations of the moving objects are known. The next process matches descriptors contained in two windows having the same moving object, where each window is in a different frame. The matching step constructs the descriptors and matches them from frame to frame. Hence, the direction of an object's movement is

acquired.

The process step prior to interpretation of the acquired motion information is referred to as preprocessing. Windows are classified into one of three types. Inconsistent motion information is examined and corrected. The information about the moving objects and the windows is packed concisely. Hopefully, the information passed to the next block is adequate and accurate.

The last stage interprets the information from the previous processes. When a task is assigned by the system user, this process acquires the desired information and gives a description of an object's movement.

In what follows, each block is described and discussed in detail.

4.2 Locating Moving Objects

4.2.1 Purpose

Locating moving objects (also referred to as "moving target indication") is the first process in the implemented system. The function of this process is similar to that of the periphery process described in [38] for detecting areas of interest in the periphery of the visual field. This process in motion analysis will direct the attention of the system and hence reduce the computational burden. The result of locating moving objects must be reasonably good, if not perfect; otherwise, any result derived from it is suspect and unreliable. Figure 4.2 shows the input and out-

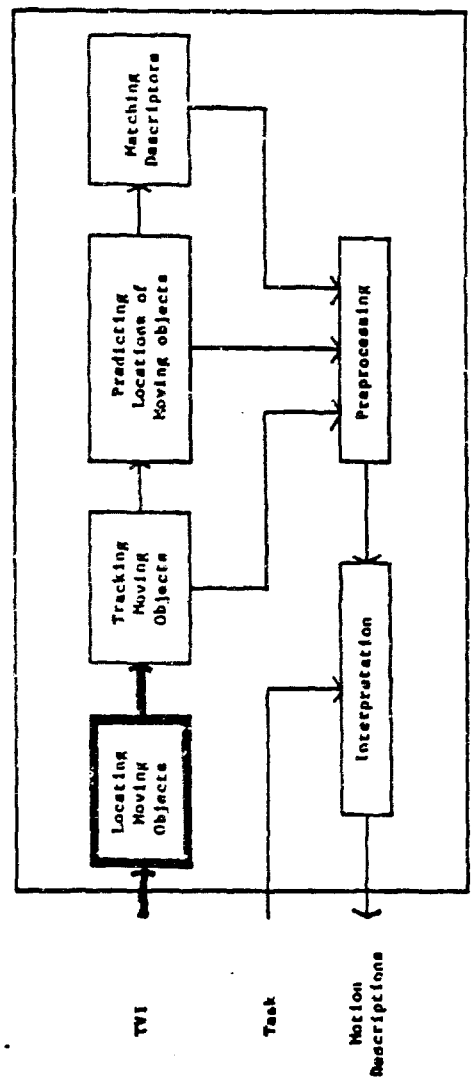


Figure 4.2 Input and output of the process of locating moving objects

put of this process.

4.2.2 Representation of Moving Objects at the Current Stage

Representation of a moving object in an image frame might be a precise registration of yes/no or 0/1 at each pixel position [28], a compact but less precise rectangular window containing an image of a moving object [29], or a set of extracted descriptors [15]. Since the objective of this study is tracking in three dimensions and not object description, we do not utilize region-to-region matching, but rather use a rectangle, i.e. window, to surround an image of a moving object, or images of multiple moving objects when they are close together in an image frame, or part of an image of a moving object when occlusion occurs.

4.2.3 Algorithm for Locating Moving Objects

The algorithm for placing a window around an image of a moving object is as follows:

1. generate a difference picture (DP),
2. generate a thresholded difference picture (TDP) with a threshold value (THD),
3. segment TDP into regions (STDP), and
4. link neighboring regions, in which each region contains pixels above a threshold; then place a window around each moving object and delete small windows.

The first step is simply the subtraction of the gray-value at each pixel position in one frame from the

corresponding pixel position in the other frame, followed by taking the absolute value of the result of the subtraction. This step is essentially a high-pass (temporal) filtering process. A typical result is shown in figure 4.3. For the purpose of illustration, the binary DP is given.

Next, the DP is thresholded. All the values less than THD are set to zero. A typical result is shown in figure 4.4. For the purpose of illustration, the binary TDP is given.

The next step is applying a region growing technique [59] on the TDP in step 2. Generally, the largest region happens to be the stationary component plus homogeneous parts of the image of the moving objects. This region will be discarded in step 4. Noise from the image recording process will result in small regions which will also be discarded. A typical result is shown in figure 4.5. Again, for the purpose of illustration, the binary picture is given.

In the last step, a single-linkage algorithm [18] is applied to link neighboring regions. Each region is represented as a point in a 2-D image plane at the center of its extremes. The minimum distance between clusters K1 and K2 is measured as:

$$Dmin(K1, K2) = \min|X1 - X2|,$$

where $X1 \in K1$ and $X2 \in K2$.

The algorithm will be terminated when Dmin between the

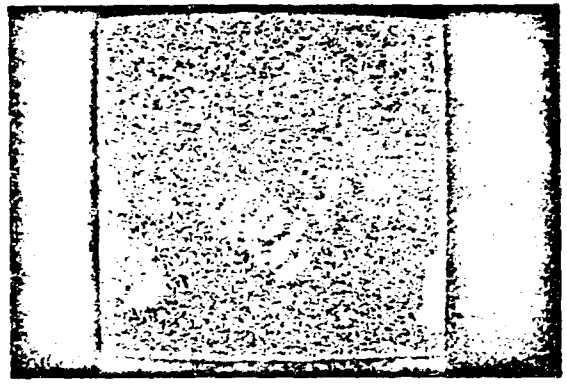


Figure 4.3 A binary difference picture

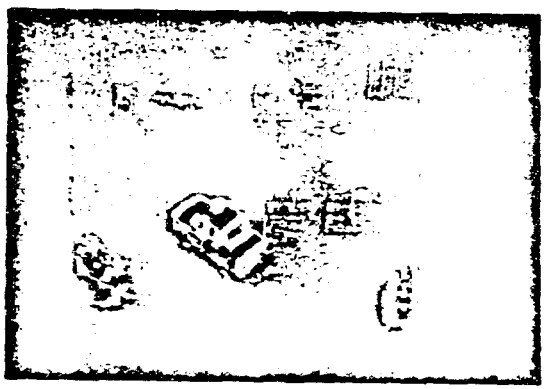


Figure 4.4 A binary thresholded difference picture

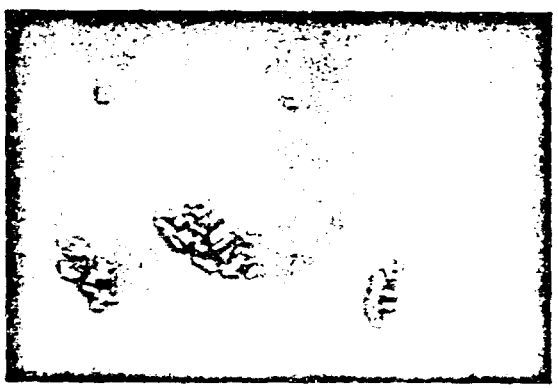


Figure 4.5 A binary thresholded difference picture after removing small regions

nearest clusters exceeds a threshold. The results can be thought of as partitioning a minimal spanning tree by breaking any linkage having a distance over a threshold. A result of applying the single-linkage algorithm is shown in figure 4.6. The extremes of the linked regions in x- and y- direction are used to form windows, and small windows are deleted (see figure 4.7).

The above algorithm can be applied for each pair of consecutive frames through the whole image sequence.

4.2.4 Results

Figures 4.7 to 4.14 are the results of locating the moving objects for frames 1, 5, 11, 16, 19, 25, 29 and 32 in image sequence one. The PERSON is rejected for further analysis due to its rather small images. Table 1 in the Appendix shows the results of locating the moving objects for image sequence one.

The results shown in figures 4.7 to 4.14 are encouraging. Of course some results are not perfect. For example, a small region is picked up in frame 29. Also in frame 19, a small window at the bottom of the image frame should be linked to the larger one above it. In frames 29 and 32, two windows contain only part of the CAR.

All the unsatisfactory results mentioned above are corrected fixed in the processes of tracking and predicting, which will be described in Section 4.3 and Section 4.4, respectively.

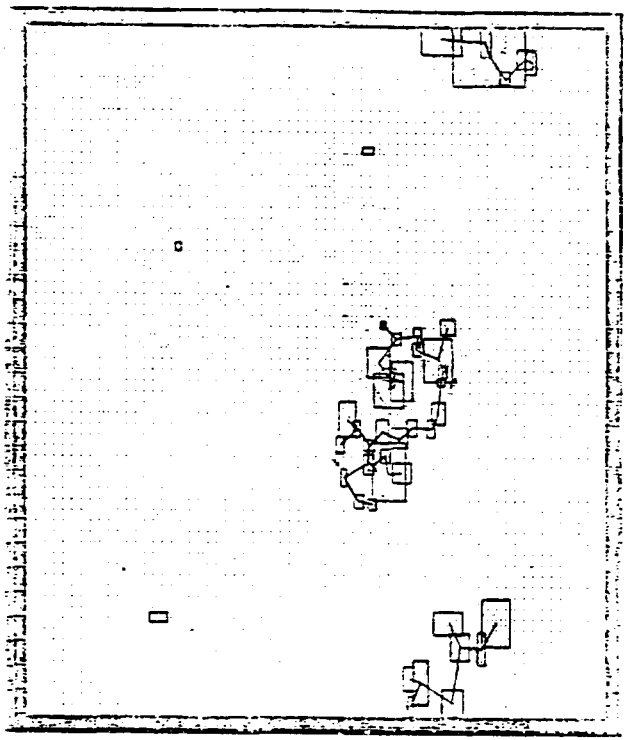


Figure 4.6 A result of applying the single-linkage algorithm

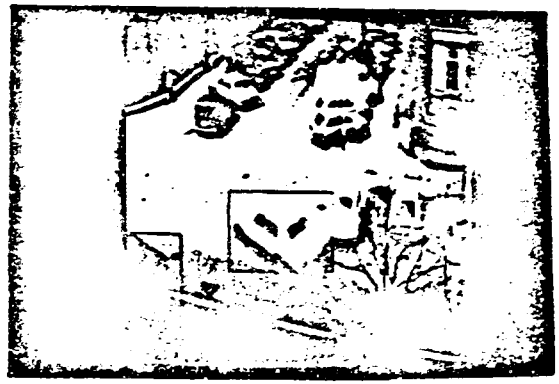


Figure 4.7 Detected moving objects in frame 1 of image sequence one

Figure 4.8 Detected moving objects in frame 5 of image sequence one

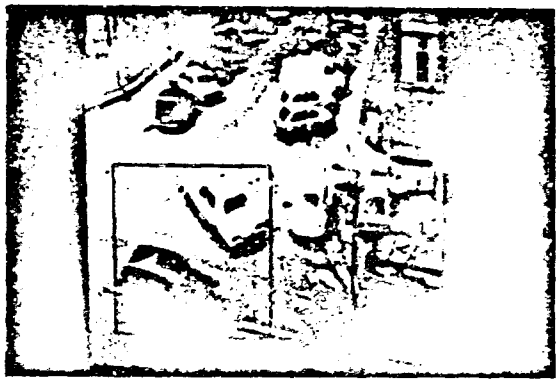


Figure 4.9 Detected moving objects in frame 11 of image sequence one

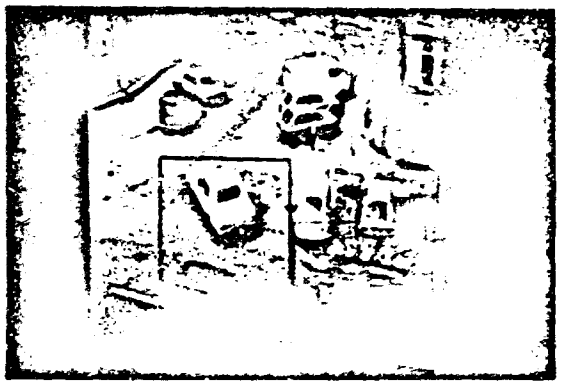


Figure 4.10 Detected moving objects in frame 16 of image sequence one

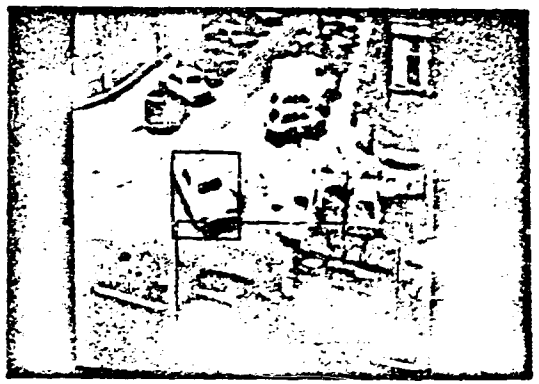


Figure 4.11 Detected moving objects in frame 19 of image sequence one

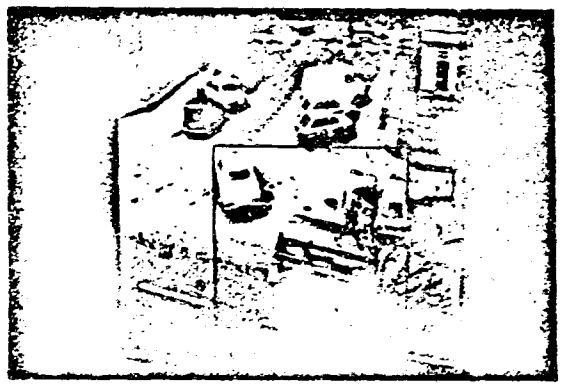


Figure 4.12 Detected moving objects in frame 25 of image sequence one

Figure 4.13 Detected moving objects in frame 29 of image sequence one

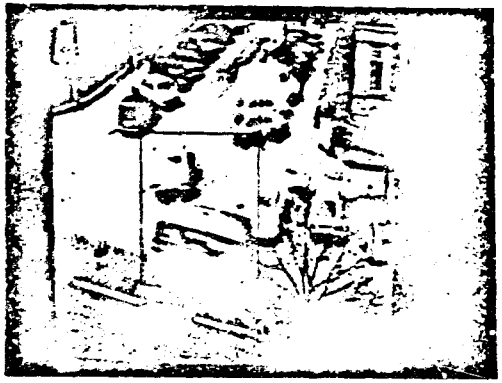


Figure 4.14 Detected moving objects in frame 32 of image sequence one

Figures 4.15 to 4.18 are the results of locating the moving objects for frames 1, 3, 5, and 9 in image sequence two. Figures 4.19 to 4.24 are the results for frames 1, 3, 5, 9, 11, and 13 in image sequence three. The results demonstrate that the algorithm is capable of locating a single moving object in a frame with satisfactory performance. Tables 2 and 3 in the Appendix show the results of locating moving objects for image sequences two and three.

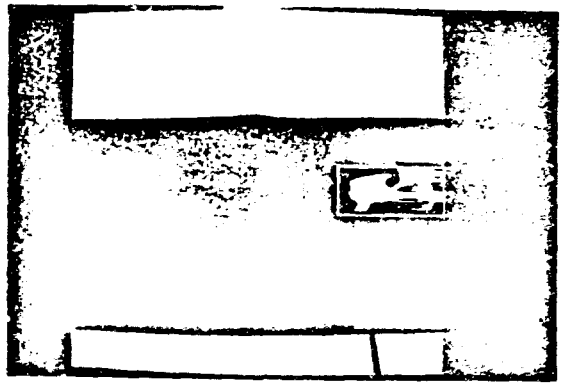


Figure 4.15 Detected moving object in frame 1 of image sequence two

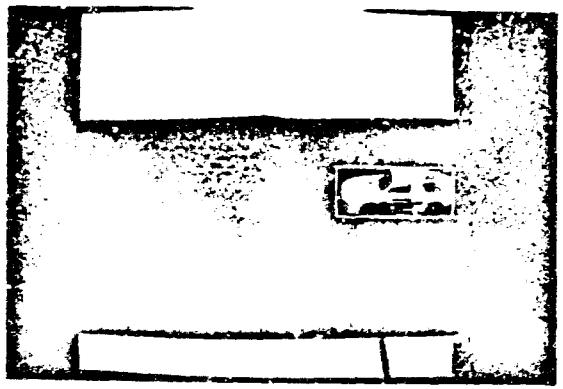


Figure 4.16 Detected moving object in frame 3 of image sequence two

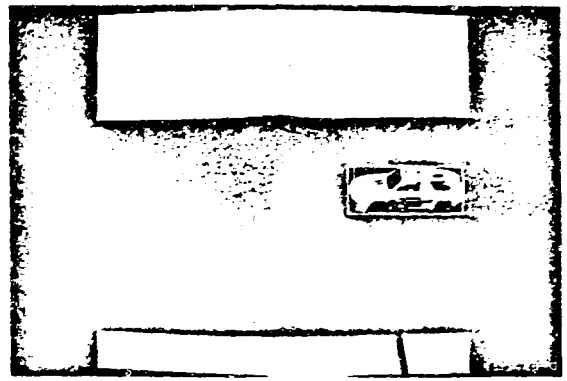


Figure 4.17 Detected moving object in frame 5 of image sequence two

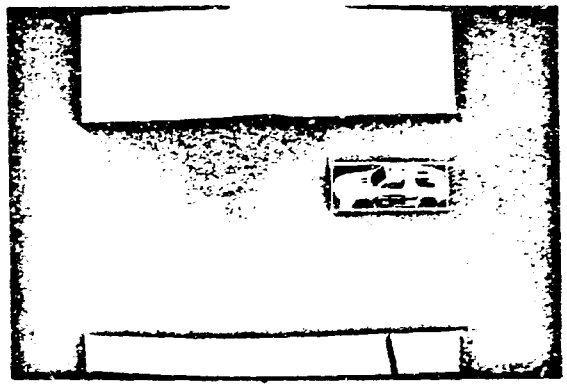


Figure 4.18 Detected moving object in frame 9 of image sequence two

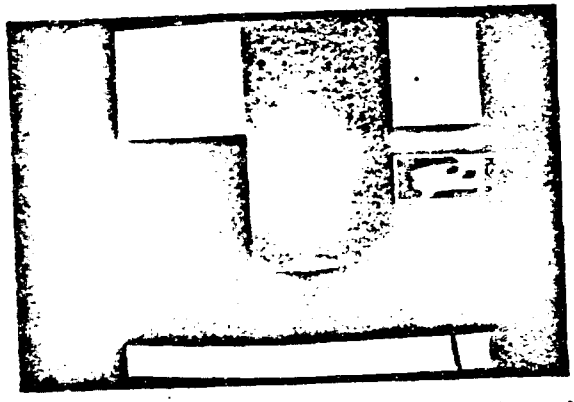


Figure 4.19 Detected moving object in frame 1 of image sequence three

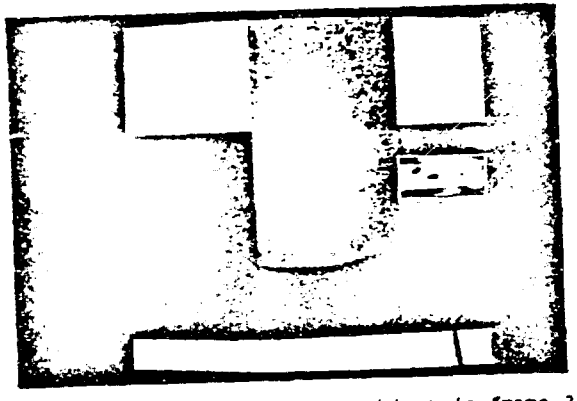


Figure 4.20 Detected moving object in frame 3 of image sequence three

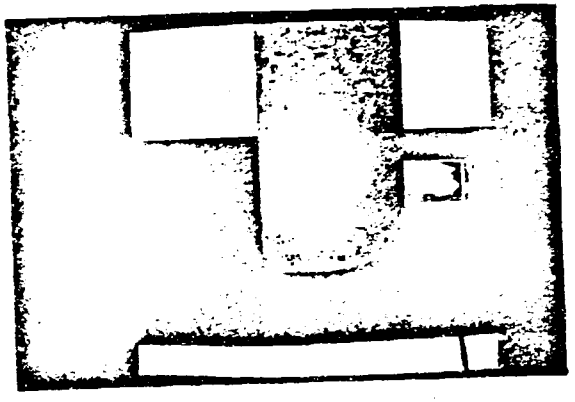


Figure 4.21 Detected moving object in frame 5 of image sequence three

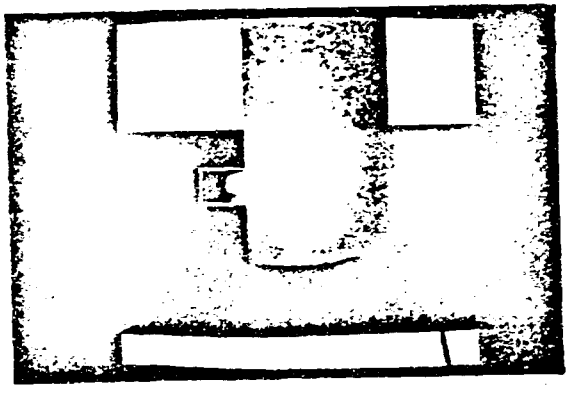


Figure 4.22 Detected moving object in frame 9 of image sequence three

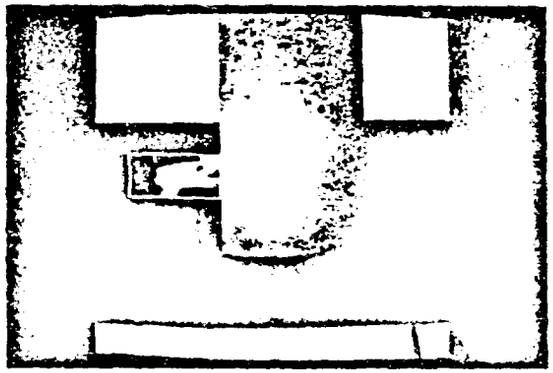


Figure 4.23 Detected moving object in frame 11 of image sequence three

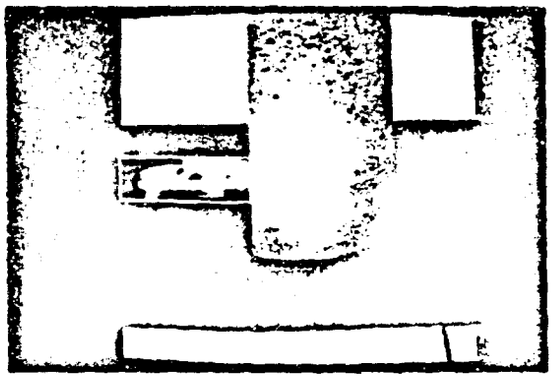


Figure 4.24 Detected moving object in frame 13 of image sequence three

4.3 Tracking Moving Objects

4.3.1 Introduction

In research on the computer analysis of motion, the tracking of moving objects from frame to frame is probably one of the most important issues, because it serves as a vehicle toward motion estimation [8], motion description [53], and moving object description [17], [37].

Approaches to object tracking are in a variety of forms, depending on the types of descriptors which characterize the images of the moving objects in each frame. Descriptors can vary from a simple point to a complex relational structure.

This section does not intend to provide a survey of the literature, but to describe the tracking algorithm implemented in the system. For the approach to tracking by established correspondence between points, the survey paper by Aggarwal, et al. [2] is a valuable reference. Other papers, such as [8], [21], [22], [39], [40], [52], represent different approaches.

4.3.2 Purpose

To avoid confusion, it should be pointed out that tracking moving objects is, in a sense, tracking windows containing images of moving objects.

In the implemented system, the process of tracking moving objects serves two purposes: to improve results of the MTI step described in Section 4.2 and to track moving ob-

jects, thus providing motion information for further processing. Failures of the MTI step are due to line jittering and other sources of noise in image recording, occasionally inducing fairly large noise regions in difference pictures (see figure 4.13). In a more simplistic system, these noise regions might be interpreted as moving objects. The latter purpose serves to extract intermediate motion information. For example, two windows merging into one may indicate that images of the two moving objects contained in the windows are close together in the image frame. Therefore, this piece of information can direct the attention of the system for further processing.

Figure 4.25 shows the input and outputs of this process.

4.3.3 The Tracking Algorithm

An algorithm is implemented under the constraint of no radical change in shape and position for the same moving object in a pair of consecutive frames. The approach to window tracking is to match windows between a pair of consecutive frames. Also an important property -- consistency -- is applied to guide the matching so that the result of MTI can be improved. The property of consistency means that the change in contents and directions of movement of windows should cohere over several frames.

Let A(i) = (A(i,1), A(i,2), ..., A(i,k)) be the set of all matched windows in the ith frame. Similarly, let B(i) =

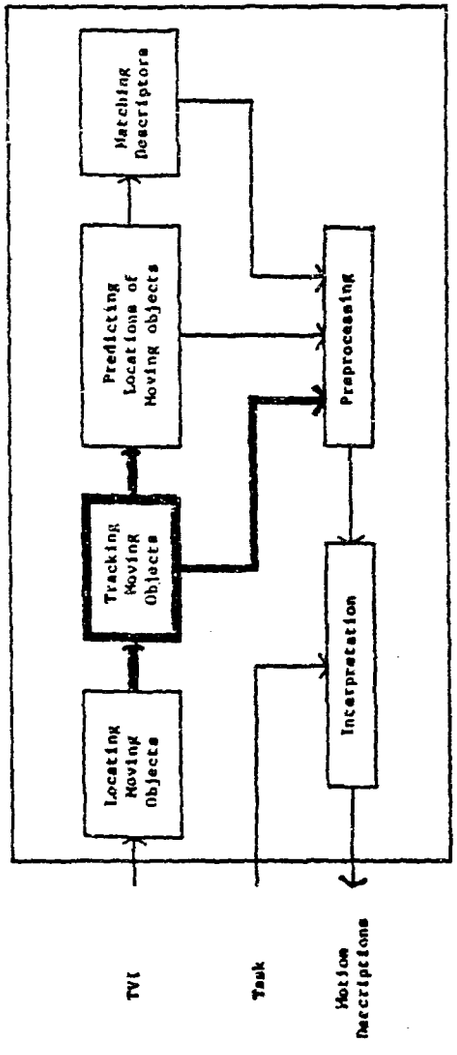


Figure 4.25 Input and outputs of the process of tracking moving objects

{B(i,1), B(i,2), ..., B(i,l)} be the set of all matched windows in the (i+1)th frame. Also let a(i) = {A(i,k+1), A(i,k+2), ..., A(i,m)} and b(i) = {B(i,l+1), B(i,l+2), ..., B(i,n)} be a set of unmatched windows in the ith and (i+1)th frames, respectively. The index, i, represents the matching frame i to (i+1). A "matched window" means the window is trackable. Conversely, an "unmatched window" means the window is not trackable.

The tracking algorithm must find matches between sets A(i) and B(i). Also, it must handle the appearance of a(i) and b(i).

We can consider the matching process as a mapping between A(i) and B(i). It must take into account the following possibilities in mapping windows:

(1) A(i,i1) --> B(i,j1),

(2) A(i,i1) --> B(i,j1) + B(i,j2),

(3) A(i,i1) --> B(i,j1) + B(i,j2) + B(i,j3),

(4) A(i,i1) --> B(i,j1) + B(i,j2) + B(i,j3) + B(i,j4),

(5) A(i,i1) + A(i,i2) --> B(i,j1),

(6) A(i,i1) + A(i,i2) + A(i,i3) --> B(i,j1),

(7) A(i,i1) + A(i,i2) + A(i,i3) + A(i,i4) --> B(i,j1),

where A(i,i1), A(i,i2), A(i,i3), and A(i,i4) ϵ A(i), B(i,j1), B(i,j2), B(i,j3), and B(i,j4) ϵ B(i), and the symbol "+" denotes an "associated" operation of windows. The mapping of A(i,i1) + A(i,i2) --> B(i,j1) means that windows A(i,i1) and A(i,i2) merge to one window, B(i,j1), in the next frame. On the other hand, A(i,i1) --> B(i,j1) + B(i,j2) means that window A(i,i1) splits into two windows, B(i,j1) and B(i,j2), in the next frame.

Another possible mapping is

(8) A(i,i1) --> O(i,B),

where A(i,i1) ϵ a(i) and the symbol O(i,B) denotes no matchable windows in B(i) U b(i) This kind of mapping can happen when there is a "noise window" or a moving object which stops its movement.

The criterion for a valid matching is a measure of

$$M = (LU1(x) - LU2(x))^2 + (LU1(y) - LU2(y))^2 + (LR1(x) - LR2(x))^2 + (LR1(y) - LR2(y))^2,$$

where LU1(x) and LU1(y) are the coordinates, in x- and y-directions, respectively, of the upper left extreme of a window in one frame. Similarly, LR1(x) and LR1(y) are the coordinates of the lower right extreme. The LU2(x) and LU2(y) are coordinates of the upper left extreme of a window in the other frame. Similarly, LR2(x) and LR2(y) are the coordinates of the lower right extreme. If M is less than a threshold, then a valid mapping is constructed. The threshold is based on the assumption of the maximum velocity for a window in a pair of consecutive frames at any direction.

The following rules based on the property of consistency for guiding the tracking are listed below:

(9) If (A(i,i1) --> B(i,j1) + B(i,j2) and

(A(i,i2) —> B(i,j1)), then (A(i,i1) @ A(i,i2)) and (B(i,j1) @ (B(i,j2));

(10) If (A(i,i1) —> B(i,j1)), (B(i,j2) ε b(i)) and (B(i+1,j2) —> C(i+1,k1)), then check the possibility of (B(i,j1) @ B(i,j2);

(11) If A(i,j1) is surrounded by A(i,j2), then delete A(i,j1);

(12) If (A(i,i1) + A(i,i2) —> B(i,j1)) and (B(i+1,j1) —> C(i+1,k1) + C(i+1,k2)), then split B(i,j1) into two windows based on averaging the extremes of the windows between A(i,i1) and C(i+1,k1), and (A(i,i2) and C(i+1,k2), respectively;

(13) If (A(i,i1) —> B(i,j1) + B(i,j2)) and (B(i+1,j1) + B(i+1,j2) --> C(i+1,k1)), then (B(i,j1) @ B(i,j2));

(14) If (A(i,i1) --> O(i,B)), then delete A(i,i1);

(15) If (A(i,i1) + A(i,i2) —> B(i,j1)) and the area of A(i,i1) is less than 2 percent of A(i,i2), then delete A(i,i1);

(16) If (A(i,i1) —> B(i,j1)) and (B(i+1,j1) --> C(i+1,k1)), then check the area of B(i,j1);

(17) If (A(i,i1) —> B(i,j1)) and (B(i+1,j1) --> C(i+1,k1)), then check the area of C(i+1,k1);

(18) If (A(i,i1) + A(i,i2) —> B(i,j1)) and (B(i+1,j1) —> C(i+1,k1)), then check the area of B(i,j1).

The symbol "@" denotes a "merging" operation.

The sequence of applying the above 18 rules is described in figure 4.26. It is worth noting that whenever rule 9, 10, or 11 has been applied, the algorithm returns to the basic rules 1 to 8 because change due to applying any one of these rules will greatly influence previously established mapping. The priority arrangement in applying rules 1 to 7 is also worthy of note. The priorities, from the highest to the lowest, are rules 1, 2, 5, 3, 4, 6, and 7. Rule 8 is applied automatically when rules 1 to 7 have been applied.

Rules 16, 17, 18 are used for checking and correcting abrupt changes in areas. Any inconsistency in the changes of areas will be detected, and the areas in the previous and/or the next frame are used in correction.

4.3.4 Results

The results of tracking moving objects in image sequence one are shown in table 4 (see the Appendix). Also the improved results of locating moving objects, due to the guidance of consistency, are shown in figures 4.27 to 4.40.

The results of this process will be used to predict ob-

jects' locations, as described in the next section.

In image sequences two and three, there is only one moving object and the results of MTI are adequate for tracking.

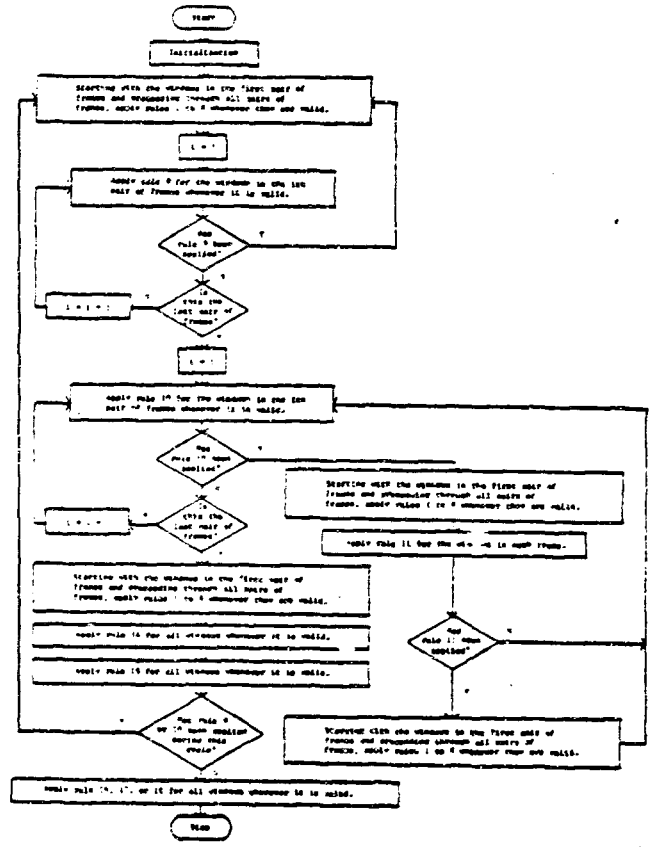


Figure 4.26 Flowchart of the tracking algorithm

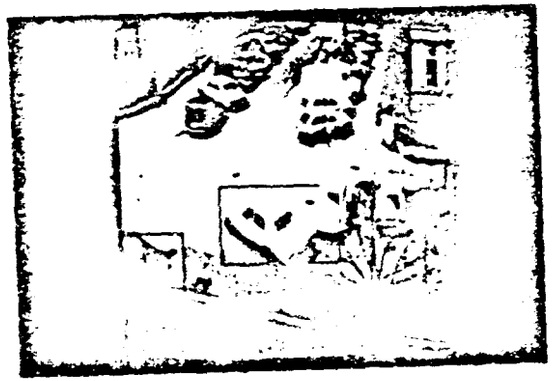


Figure 4.27 Detected moving objects in frame 2 of image sequence one

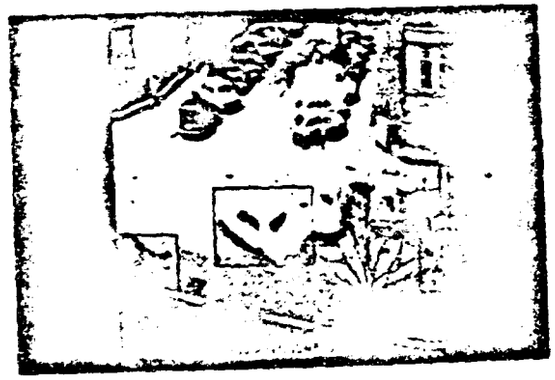


Figure 4.28 Improved result of detecting moving objects in frame 2

Figure 4.29 Detected moving objects in frame 9 of image sequence one

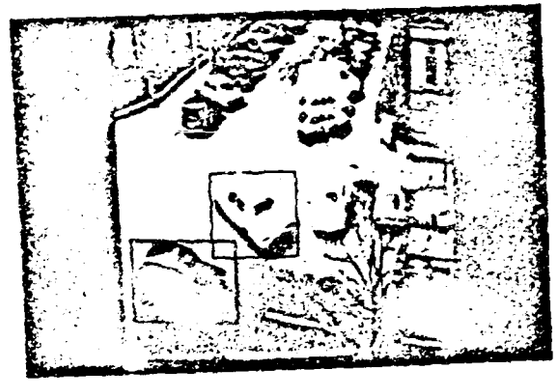


Figure 4.30 Improved result of detecting moving objects in frame 9

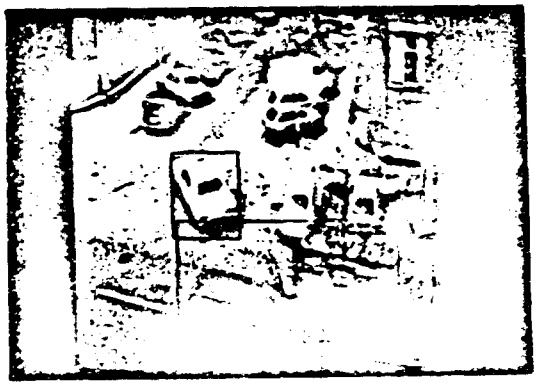


Figure 4.31 Detected moving objects in frame 19 of image sequence one

Figure 4.32 Improved result of detecting moving objects in frame 19

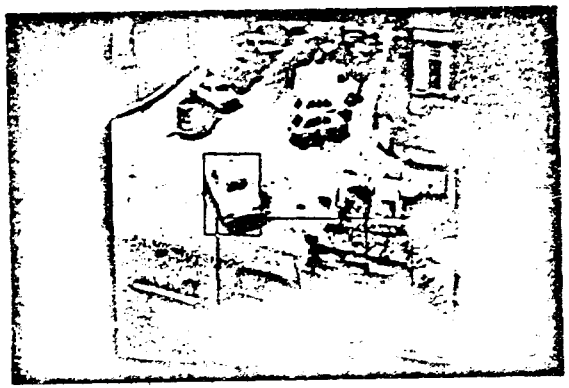


Figure 4.33 Detected moving objects in frame 20 of image sequence one

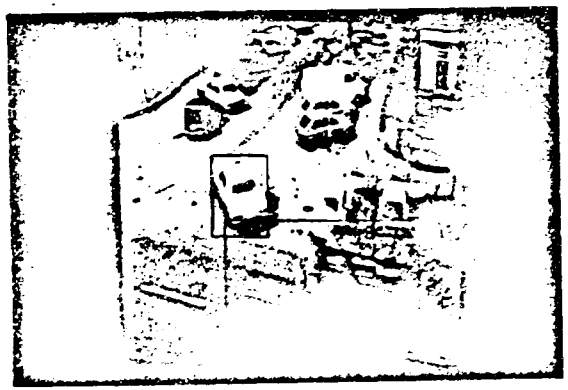


Figure 4.34 Improved result of detecting moving objects in frame 20

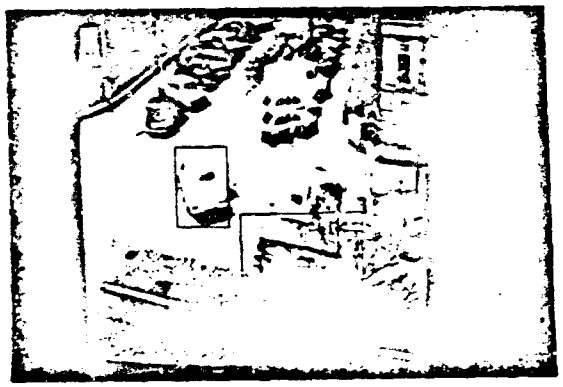


Figure 4.35 Detected moving objects in frame 23 of image sequence one

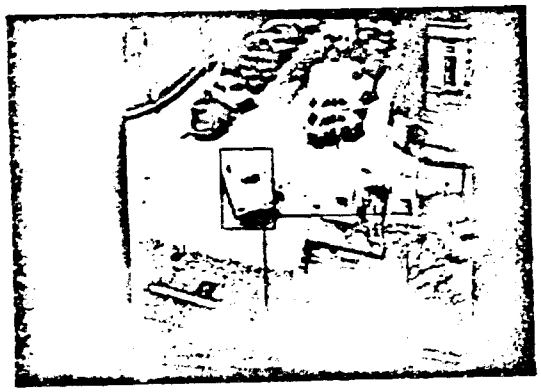


Figure 4.36 Improved result of detecting moving objects in frame 23

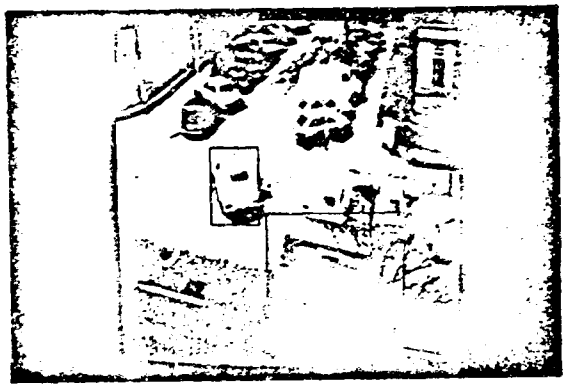


Figure 4.37 Detected moving objects in frame 24 of image sequence one

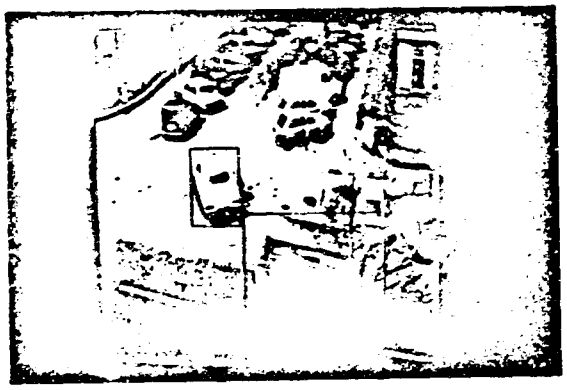


Figure 4.38 Improved result of detecting moving objects in frame 24

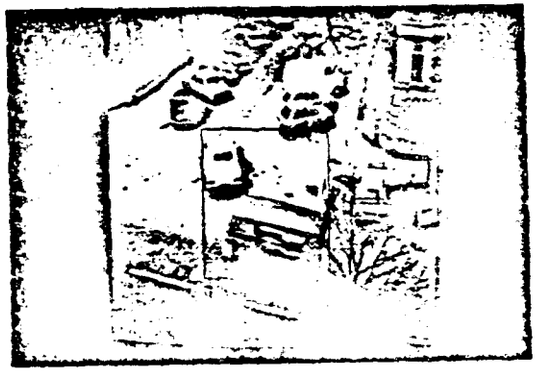


Figure 4.39 Detected moving objects in frame 29 of image sequence one

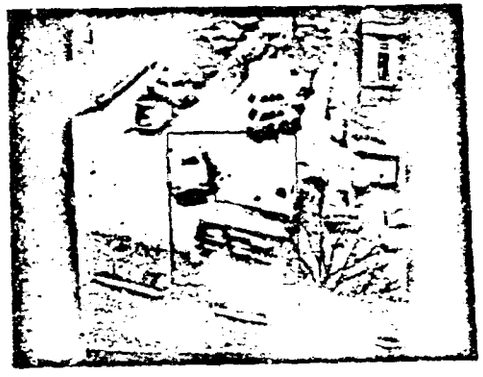


Figure 4.40 Improved result of detecting moving objects in frame 29

4.4 Predicting Locations of Moving Objects

4.4.1 Purpose

Thus far, some information about objects' motions by tracking windows containing moving objects has been acquired. In the next step, an attempt is made to predict the locations of the windows, each containing only one moving object, when images of two or more moving objects are close together, resulting in a single larger window. Or, when occlusion occurs, two or more windows will contain the same moving object.

The occlusion is restricted to the types shown in figures 4.13 and 4.14, where the images of objects are partially occluded.

The input and outputs of this process are illustrated in figure 4.41.

4.4.2 Assumptions

Two assumptions are embedded in the algorithm:

1. Each window in the first frame should contain only one object's image.
2. The images of two or more objects contained in the same window should not change their speeds and directions significantly.

The above assumptions are necessary because the implemented system does not include a recognition process. So instead of recognizing objects and then locating windows, windows are tracked and predicted.

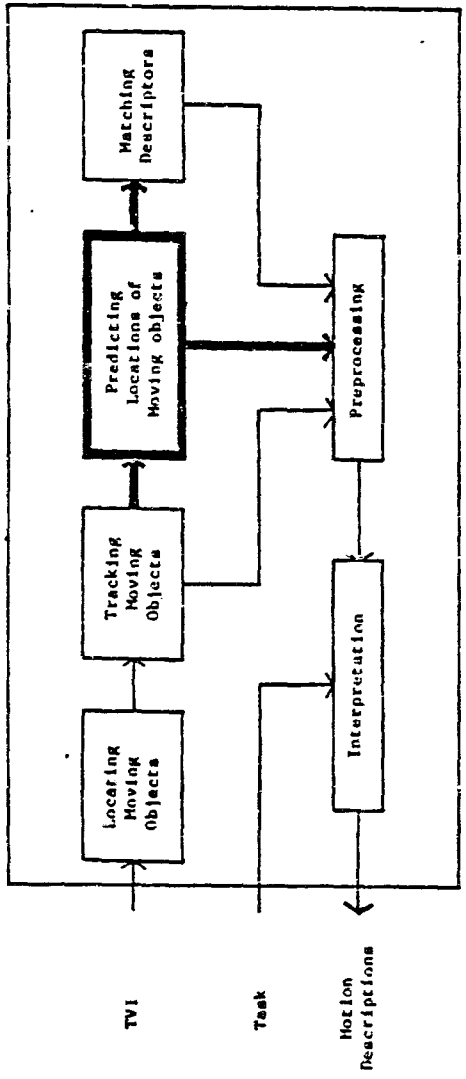


Figure 4.41 Input and outputs of the process of predicting locations of moving objects

4.4.3 The Predicting Algorithm

The notations introduced in Section 4.3 are used here, and rules 1 to 8 and rule 16 will be applied for the purpose of tracking and improving results, respectively. The criterion for a valid match is also used here.

Rules 1 to 8 and rule 16 are restated below:

(1) A(i,i1) --> B(i,j1),

(2) A(i,i1) --> B(i,j1) + B(i,j2),

(3) A(i,i1) --> B(i,j1) + B(i,j2) + B(i,j3),

(4) A(i,i1) --> B(i,j1) + B(i,j2) + B(i,j3) + B(i,j4),

(5) A(i,i1) + A(i,i2) --> B(i,j1),

(6) A(i,i1) + A(i,i2) + A(i,i3) --> B(i,j1),

(7) A(i,i1) + A(i,i2) + A(i,i3) + A(i,i4) --> B(i,j1),

(8) A(i,i1) --> O(i,B),

(16) If (A(i,i1) --> B(i,j1)) and (B(i+1,j1) --> C(i+1,k1)), then check the area of B(i,j1);

The additional rules for the purpose of prediction are introduced in the following:

(a) If (A(i,i1) --> B(i,j1) + B(i,j2)) and (A(i,i2) --> B(i,j1)), then (A(i,i1) --> B(i,j2));

(b) If (A(i,i1) --> O(i,B)), then (B(i,j1) @ B(i,j2) @ ... @ B(i,jn)), excluding new windows due to the appearance of new moving objects or windows which stop their movements;

(c) If (A(i,i1) + A(i,i2) --> B(i,j1)), then do the

prediction;

(d) If (A(i,i1) + A(i,i2) +A(i,i3) --> B(i,j1)), then

do the prediction.

Rule (a) is introduced here because the condition stated in rule (a) may occur under the matching criterion. Rule (b) merges windows into a larger one when mapping cannot be established.

In applying rules (c) and (d), a speed of window boundary is estimated by previous observations. Two important variables, THD and COMP, are used to control the estimation: the former filters out slow motion and possible noise, and the latter compensates for the estimation. When a speed is less than THD, it is set to zero; otherwise, it has the previous speed minus or plus THD, depending on the boundary.

To estimate the location of a window's right vertical boundary n frames after the current frame, the following relation is used:

EBL = CBL + (PAS + THD) * n - (n - 1) * THD / COMP,

where EBL = estimated boundary's location, CBL = current boundary's location, and PAS = previous averaged speed.

The event of encountering-and-leaving shown in figure 4.42 also requires careful handling. It is worth noting that the window containing two moving objects shrinks and then expands during this kind of phenomenon. The algorithm should be able to detect this phenomenon and "lock" the tracked boundary. This kind of function is included in

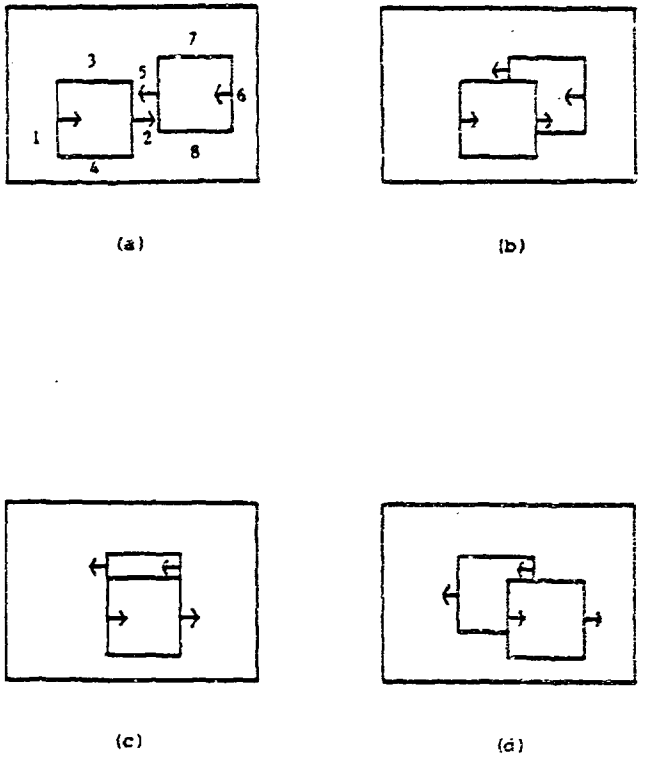


Figure 4.42 Illustration of the event of encountering-and-leaving. (a) Two objects are about to encounter, (b) Two objects contained in a window, (c) Each object keeps moving in the same direction and at the same speed, (d) Two objects are about to leave each other. The arrows show directions of motion.

predicting window locations. The algorithm will predict the location of boundaries 2, 3, 5, and 8 in figure 4.42(b) and (c). But it will predict the location of boundaries 1, 3, 6, and 8 in figure 4.42(d).

The flowchart in figure 4.43 provides details about this process.

4.4.4 Results

Figures 4.44 to 4.49 shows some results of predicting in image sequence one. Comparing these with the results shown in Sections 4.2 and 4.3, it can be seen how the system gradually refines and acquires motion information. The results of prediction are listed in table 5 (see the Appendix).

The results of this process can be used to match an object's descriptors contained in two windows, one in each frame, containing the same moving object. This process will be described in Section 4.5. Also the results can be used in classifying windows which will be described in Section 4.6.

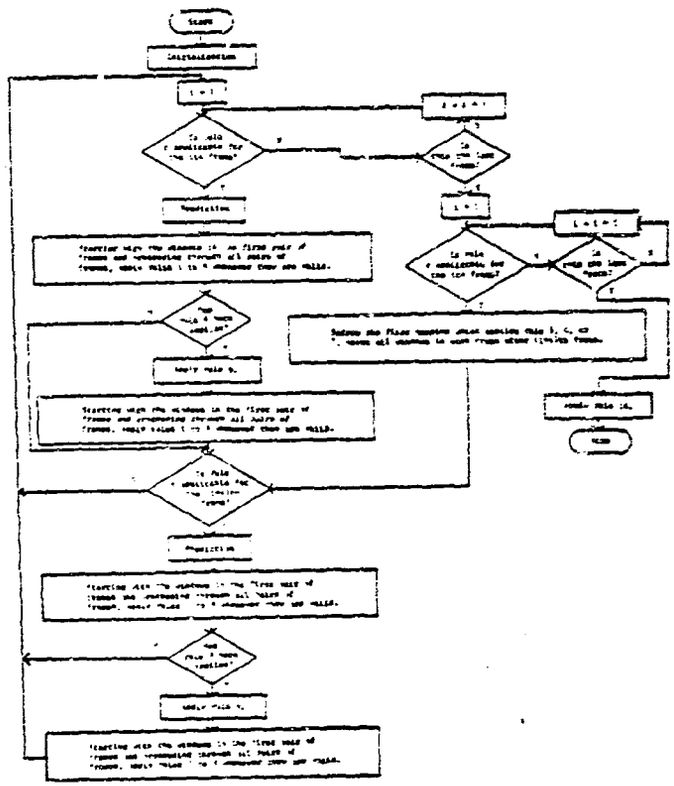


Figure 4.43 Flowchart of predicting locations of moving objects

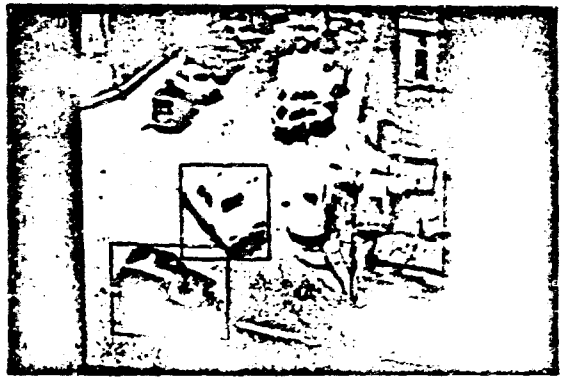


Figure 4.44 Predicted locations of moving objects in frame 11 of image sequence one

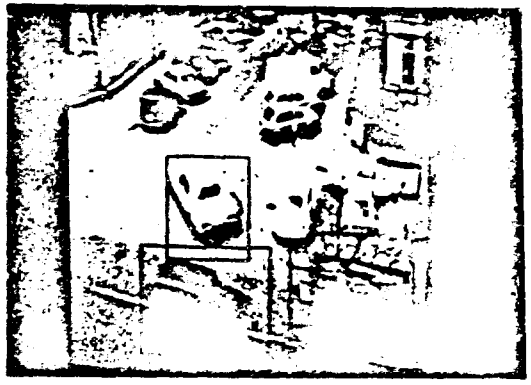


Figure 4.45 Predicted locations of moving objects in frame 16 of image sequence one

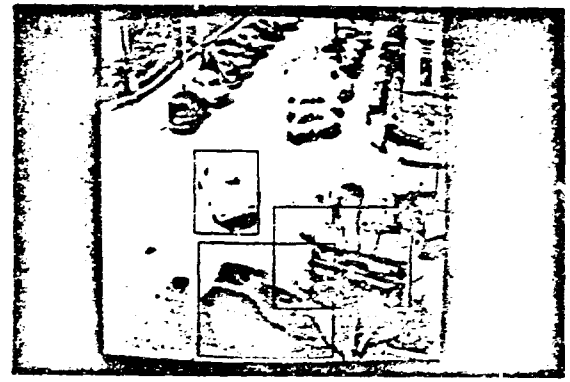


Figure 4.46 Predicted locations of moving objects in frame 19 of image sequence one

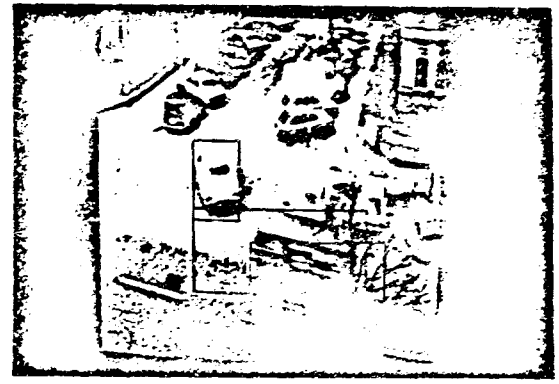


Figure 4.47 Predicted locations of moving objects in frame 25 of image sequence one

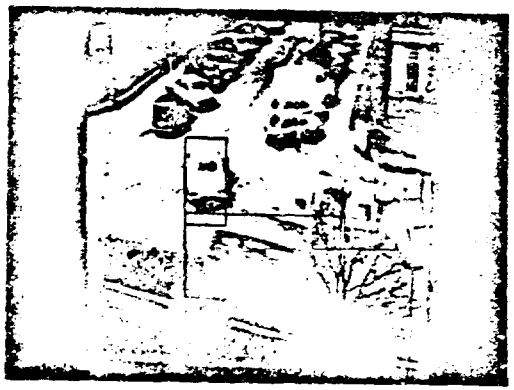


Figure 4.48 Predicted locations of moving objects in frame 29 of image sequence one

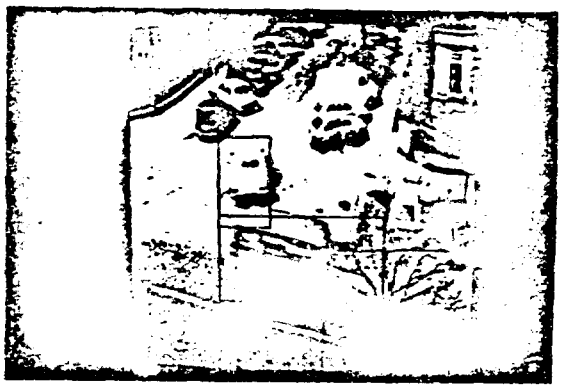


Figure 4.49 Predicted locations of moving objects in frame 32 of image sequence one

4.5 Matching Descriptors

4.5.1 Purpose

From the previous processes, information about the locations of moving objects and the association of windows containing the same object in different frames is acquired. An attempt is now made to derive each object's direction of movement by matching descriptors within a pair of windows, one in each frame, containing the same moving object. Each object's direction history can be determined if there is success in matching descriptors from frame to frame.

The input and output of this process are shown in figure 4.50.

4.5.2 Corner Matching

Corners as descriptors. In selecting descriptors for a matching process, three questions are the major concern:

1. What are the descriptors generally occurring in a wide variety of scenes? And what are the descriptors suitable for analyzing motion in the processed image sequences?
2. Among these descriptors, what are the major differences, i.e. the information represented by the descriptors?
3. What detectors are suitable for the desired application, i.e. to determine their performances and computational complexities?

Research in physiology and psychology relating to this

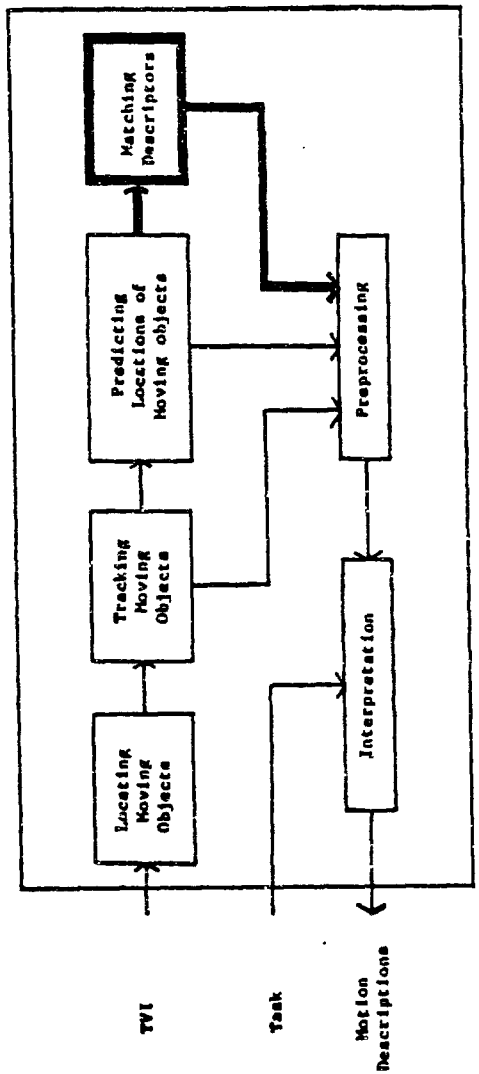


Figure 4.50 Input and output of the process of matching descriptors

study has been examined: Attneave [3] points out that the angles provide a basis for giving a simplified description of shapes, since the shapes can be approximated by polygons whose vertices are located at the angles. Barlow, et al. [6], discuss research in processing pictorial information in distinct research areas: neurophysiology, psychology, and computer science. In all these areas, detecting edges, lines, and the terminations or junctions of contours in all types of pictorial analysis is vital. Ullman [67] shows that the matching process of an object's motion is low-level. He uses so-called "correspondence tokens" — such as edges, bars, and small blobs — to establish matches.

From the reports cited above, one can infer that points, edges, lines, and corners probably are good candidates for descriptor matching.

Corners are selected as the descriptors in the matching process. A corner probably contains as much information as a line segment, e.g., its orientation, sharpness, and location. Also it requires less computation (compared with the additional process of thinning and linking broken lines after edge detection). A report by Nagel [45] points out the difficulty of region or contour matching in analyzing TVI. The reason for this difficulty is that the segmentation algorithm cannot reliably handle the systematic changes of the grayvalues in the image of moving objects. In using the corner detector, the problem of reliability remains.

However, experiments show that a reasonable result can be achieved in removing noise corners and acquired information about object motion. The algorithm is described below.

The corner matching algorithm. The first step is to find corners in the windows containing the same moving object, one in each frame. Since the evaluation is made on the rate of change of gradient direction along an edge and multiplies this rate by the gradient magnitude, the sign of the value depends on the first term. The positive values are selected so that after thresholding and local nonmaximum suppression, each surviving corner will be the local maximum and will have the same characteristics -- a dark corner intruding into a white background.

The reason for selecting positive values is that the detected corners are mostly on the moving object, although some corners may be picked up in the background and foreground.

The number of corners should be in a reasonable range. Too many corners may increase the possibility of spurious matches and computation cost. On the other hand, it is necessary to find enough corners to support evidence of compatibility.

For each corner in the reference window, an initial probability is assigned to each possible matched corner in the target window. Then the probabilities are updated. Notationally, let P(i, k, Δx, Δy) be the probability of matching the ith corner in the reference window to the corner in the target window with displacement of dx and dy in x- and y-directions, respectively, at the kth iteration of the updating process. The corners in the target window should be between a maximum and a minimum possible displacement. Thus, P(i,k = 0, Δx, Δy) is an initial probability, and the sum of the probabilities over possible matches for the ith corner is unity. The initial probabilities are determined from

$$W(i, k = 0, \Delta x, \Delta y) = 1 / (1 + c * D(\Delta x, \Delta y)),$$

where c is a constant factor, $D(\Delta x, \Delta y)$ denotes $(\Delta x^2 + \Delta y^2)$, and i and k are defined as in P(i, k, Δx, Δy).

The reason for using the squared distance as a measurement of initial probability instead of using the squared grayvalue difference between the small windows, as [8] did, is the possible abrupt change, due to motion, of background characteristics near the exterior corners. Another reason will be explained later in discussing the sensitivity of direction measurement.

Here a special notation, #, is introduced for "undefined displacement." This is because a corner might disappear due to occlusion, movement out of the view of the observer, or a local nonmaximum. The "undefined displacement" is denoted as W(i, k, Δ#, Δ#). The W(i, k = 0, Δ#, Δ#) is derived from subtracting the maximum of W(i, k = 0, Δx, Δy) from one, i.e.

$$W(i, k = 0, \Delta\theta, \Delta\theta) = 1 - \max(W(i, k = 0, \Delta x, \Delta y)).$$

Normalizing the sum over $W(i, k = 0, \Delta x, \Delta y)$, including $W(i, k = 0, \Delta\theta, \Delta\theta)$, to the value of one, the initial probabilities for the ith corner in a reference window are established.

The third step is to update probabilities for each corner. It is assumed that the moving directions (ang(Δx, Δy)) for a set of neighboring corners vary smoothly, so a criterion for increasing a probability for a macch is the existence of neighbors having similar directions. This assumption is a relaxed form of the assumption: "The velocity field of motion within the image of a rigid object varies continuously almost everywhere" [36], [58], [75]. This requirement is relaxed because it is considered that sparsely distributed corners might violate the assumption.

It is worth noting that the sensitivity of direction measurement decreases when the distance increases, e.g., the difference between (ang(4, 2) - ang(4, 3)) and (ang(12, 6) - ang(12, 7)) is about 7 degrees, although the deviations in Δy are the same, both having only one pixel. This explains the reason for selecting the squared distance as a measurement of similarity in initial probability estimation.

The updating equations are

$$W(i, k + 1, \Delta x, \Delta y) = P(i, k, \Delta x, \Delta y) * a + b * Q(i, k, \Delta x, \Delta y),$$

and

$$W(i, k + 1, \Delta\theta, \Delta\theta) = P(i, k, \Delta\theta, \Delta\theta),$$

where a and b are constants.

The $Q(i, k, \Delta x, \Delta y)$ is defined as

$$Q(i, k, \Delta x, \Delta y) = \sum_{i \neq j,\ j \text{ is a neighbor of } i} \left[\sum_{d(\Delta x, \Delta y, \Delta x', \Delta y') < \text{threshold}} P(i, k, \Delta x', \Delta y') / (1 + d(\Delta x, \Delta y, \Delta x', \Delta y') * d * \sqrt{(D(\Delta x, \Delta y))}) \right],$$

where "<>" denotes "is not equal to," d is a constant, and $d(\Delta x, \Delta y, \Delta x', \Delta y') = |\text{ang}(\Delta x, \Delta y) - \text{ang}(\Delta x', \Delta y')|$.

As pointed out in [8], the purpose of constant a is to delay the total suppression of unlikely matches; the purpose of constant b is to determine the rate of convergence.

Usually, after ten iterations of updating, each corner is selected which has a match with a probability over a threshold in matching a corner in the target window. These surviving corners are locally similar in the directions of movement.

Because the constraint of local similarity in the direction propagates over the surviving corners, a further assumption is made: There is a prominent cluster in the direction space. Therefore, corners which do not have directions belonging to the cluster are discarded. This step will filter out the most spurious matches due to detected corners in the background enclosed by the windows.

Averaging directions over the member in the cluster, a direction representing the prominent moving direction of an object is acquired.

4.5.3 Results

Figures 4.51 to 4.53 illustrate detected corners on the CAB in three consecutive frames in image sequence one. These figures show that even the detected corners in the background vary from frame to frame. The difficulty of tracking the same corner in the view from frame to frame is demonstrated.

Figures 4.54 to 4.57 show the matched corners on the CAB in the frames illustrated in figures 4.51 to 4.53. Table 6 shows the results of corner matching over the entire image sequence one (see the Appendix). A numeric value followed by the symbol "$" means an inconsistent result due to the influence of the foreground, a tree in the lower right corner in the image sequence. Figures 4.58 to 4.59 show the detected corners within a window containing the WAGON in frame 16 and the detected corners within a window containing the CAR in frame 19. These figures show that a lot of corners are generated due to the existence of the tree.

In testing image sequences two and three, figures 4.60 to 4.67 show some results of the detected and matched corners. Tables 7 and 8 in the Appendix show the results of corner matching over image sequences two and three, respectively.

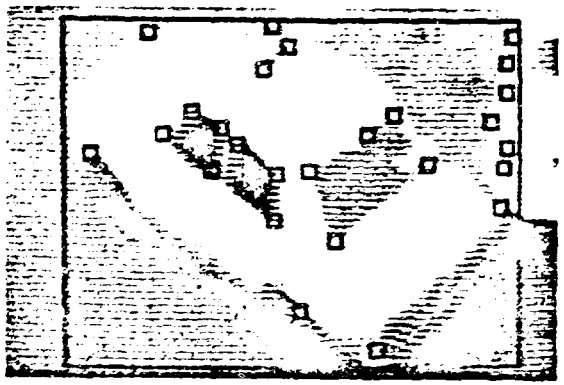


Figure 4.51 Detected corners on the CAB in frame 1 of image sequence one

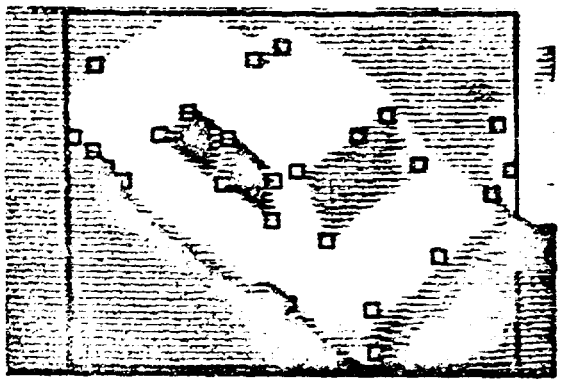


Figure 4.52 Detected corners on the CAB in frame 2 of image sequence one

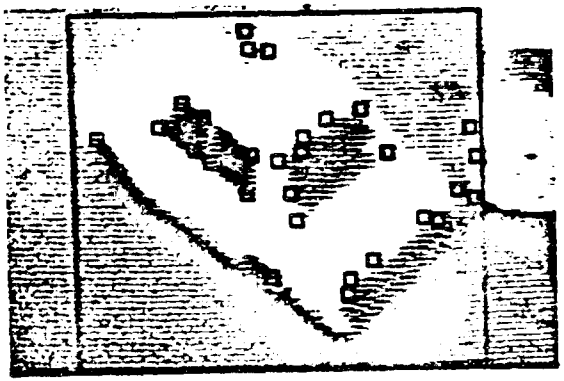


Figure 4.53 Detected corners on the CAB in frame 3 of image sequence one

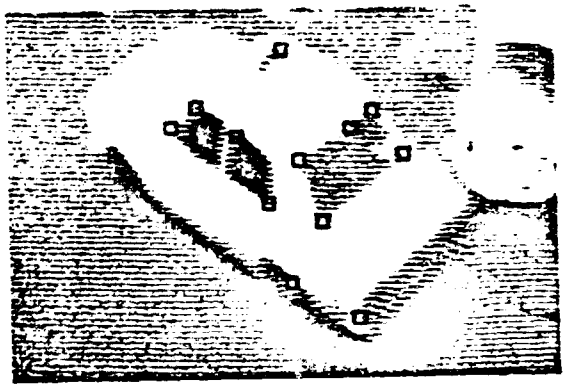


Figure 4.54 Matched corners on the CAB in frame 1 (in matching corners between frames 1 and 2)

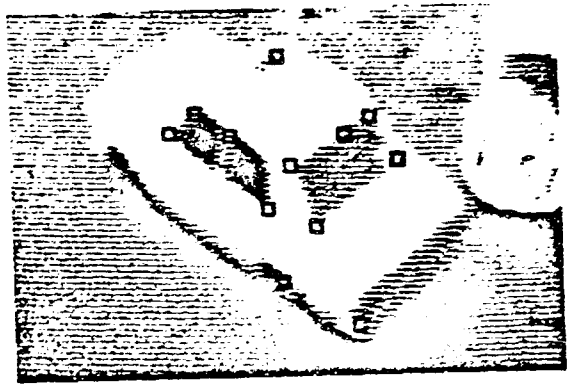


Figure 4.55 Matched corners on the CAB in frame 2 (in matching corners between frames 1 and 2)

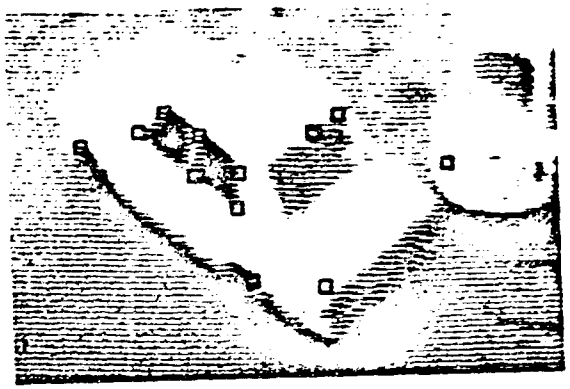


Figure 4.56 Matched corners on the CAB in frame 2 (in matching corners between frames 2 and 3)

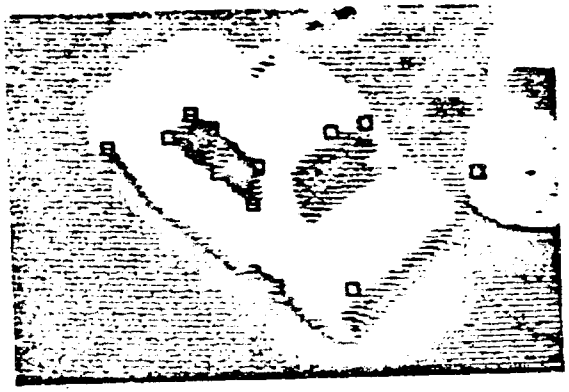


Figure 4.57 Matched corners on the CAB in frame 3 (in matching corners between frames 2 and 3)

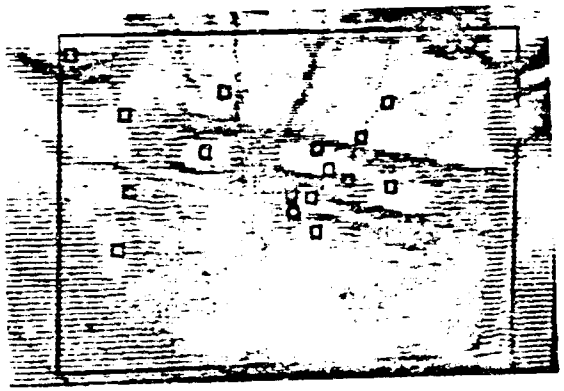


Figure 4.58 Detected corners on the WAGON in frame 16 of image sequence one

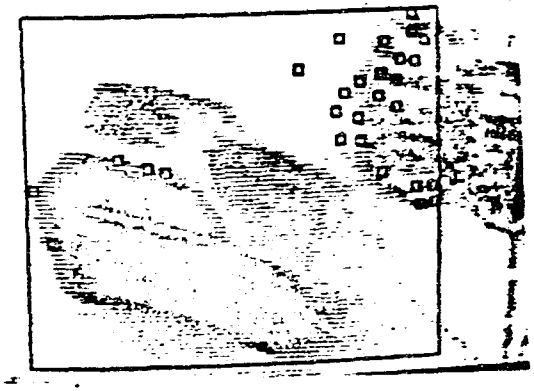


Figure 4.59 Detected corners on the CAR in frame 19 of image sequence one

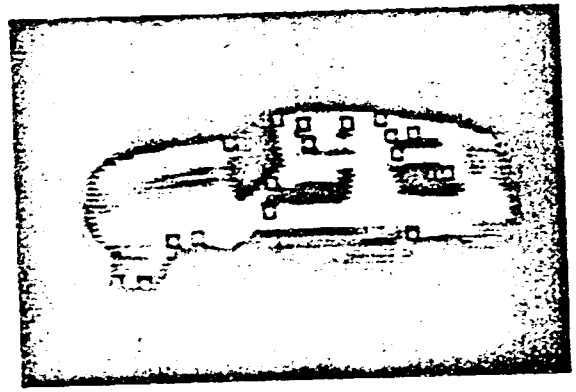


Figure 4.60 Detected corners on the car in frame 3 of image sequence two

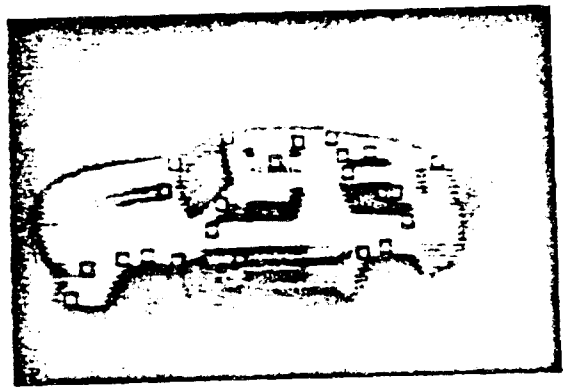


Figure 4.61 Detected corners on the car in frame 4 of image sequence two

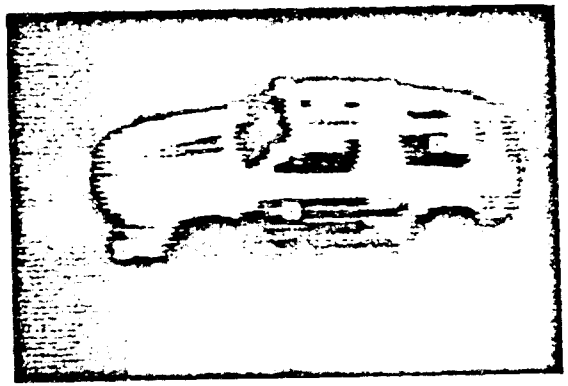


Figure 4.62 Matched corners on the car in frame 3 (in matching corners between frames 3 and 4 in image sequence two)

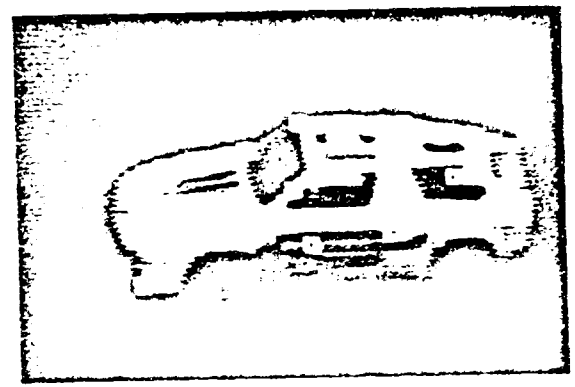


Figure 4.63 Matched corners on the car in frame 4 (in matching corners between frames 3 and 4 in image sequence two)

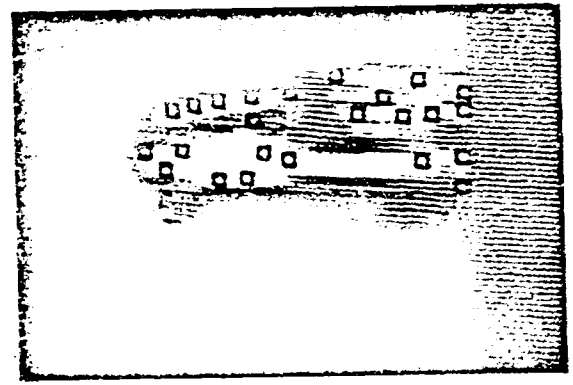


Figure 4.64 Detected corners on the car in frame 12 of image sequence three

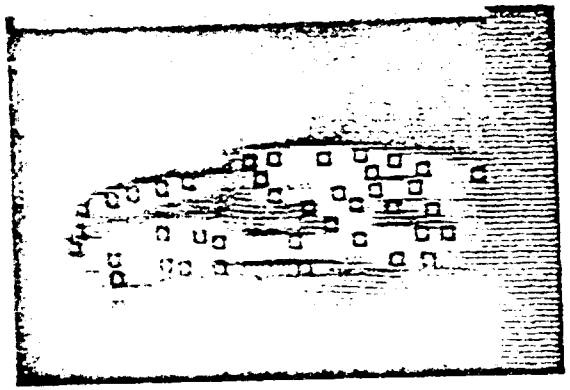


Figure 4.65 Detected corners on the car in frame 13 of image sequence three

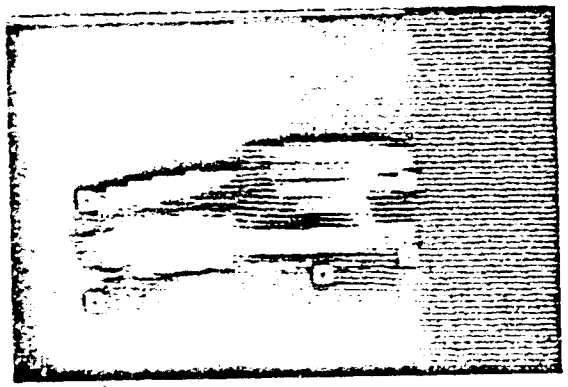


Figure 4.66 Matched corners on the car in frame 12 (in matching corners between frames 12 and 13 in image sequence three)

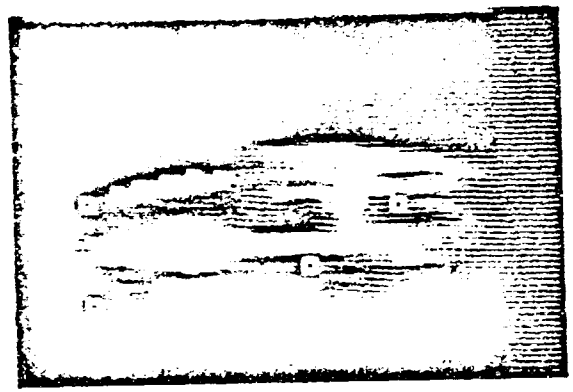


Figure 4.67 Matched corners on the car in frame 13 (in matching corners between frames 12 and 13 in image sequence three)

4.6 Preprocessing

4.6.1 Purpose

As pointed out in Section 4.1, the purposes of preprocessing are to classify windows, to examine and then correct inconsistent motion information, and to collect the information about moving objects and windows in compact forms.

The representation of moving objects is described in Chapter 3. In this section, the procedures which classify window types and examination and correction of inconsistent motion information are described. The inputs and output of this process are shown in figure 4.6B.

4.6.2 Window Type Classification

Recalling the tracking algorithm introduced in Section 4.3, the system is able to acquire information about a set of windows in one frame matching a set of windows in the next frame. An interesting result is that several windows may merge into one window because two or more moving objects are close together in a frame. Conversely, one window may split into several windows because moving objects are separating from each other in a frame or because occlusion occurs (refer to table 4).

Also recalling the predicting algorithm introduced in Section 4.4, the system is able to acquire information about the location of each moving object in each frame over the entire image sequence one.

An attempt is made to classify windows based on the

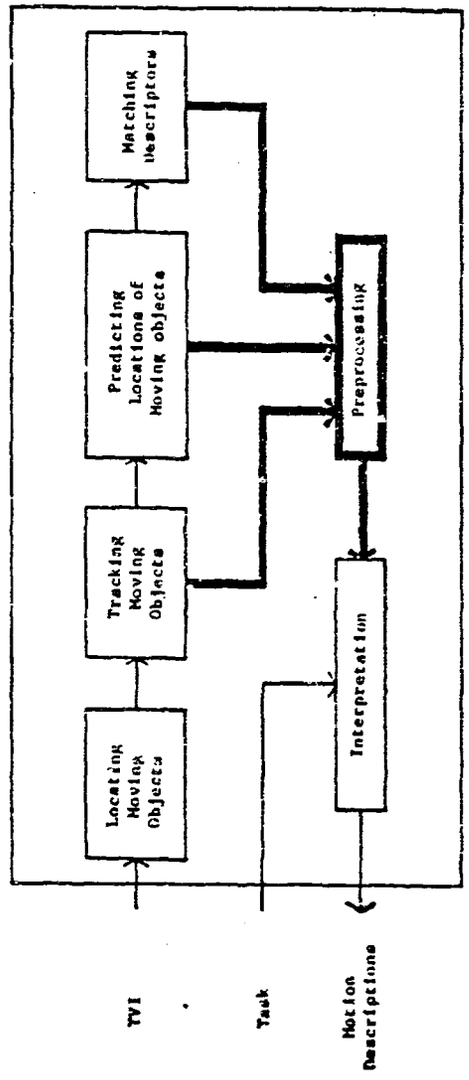


Figure 4.6B Inputs and output of the process of preprocessing

above information. The term "window type classification" means the labeling of a window as "regular," "merged," or "split." A window containing only one whole moving object is labeled "regular." A window containing part of a moving object is labeled "split." And a window containing more than one moving object is labeled "merged."

A classification algorithm which is capable of handling three or fewer windows in each frame is developed. The classification procedure is best described by using the flowchart in figure 4.69. If there is only one window in a frame, then certainly all the images of moving objects are in the same window, which is of type "regular" or "merged." If there are two windows in a frame, then there are three possibilities: each of the windows contains part of a moving object, each of the windows contains a distinct moving object, or one window contains one moving object and another window contains two moving objects. If there are three windows in a frame, then there are three possibilities: one window contains one moving object and the other two windows contain only part of a moving object, each window contains one moving object, or one window contains two moving objects and the other two windows contain only part of a moving object. The above eight possibilities are examined based on the sum of squared differences between predicted object locations and window locations, each represented by a quadruple

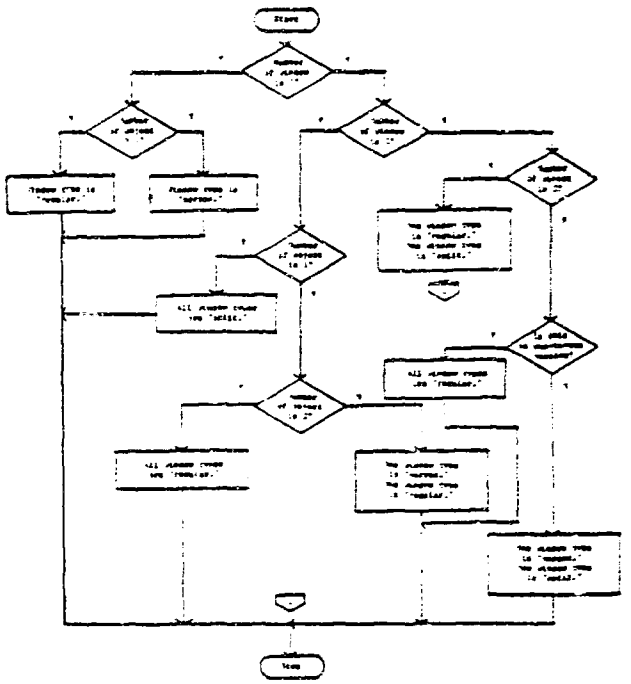


Figure 4.69 Flowchart of window classification

(LV, UH, RV, LH),

where LV and RV represent the x-coordinates of the left and right sides of the vertical boundaries, respectively, and UH and LH represent the y-coordinates of the upper and lower horizontal boundaries, respectively.

After classifying the windows, the information about the windows is gathered in a compact from. Describing with PASCAL-like language, the representation is in the following.

```
window = record
         window-type : (merged, split, regular);
         frame-time : integer;
         window-extremes : array [1..4] of 1..512;
         no-mv-obj : integer;
         obj-within-window : array [1..4] of label;
         no-window-next : integer;
         nwptr : array [1..4] of integer;
         no-window-current : integer;
         cwptr : array [1..4] of integer;
         no-window-previous : integer;
         pwptr : array [1..4] of integer
         end;
```

The variable window-type denotes the type of a window which is derived from the algorithm described above.

The variable frame-time indicates the frame number at which this description is valid.

The variable window-extremes is a quadruple which is used to represent a window through this work (refer to Chapter 3).

The variable no-mv-obj indicates the number of moving object(s) within in a window. If a window contains only part of a moving object, the variable still has a value of

1. The label(s) of the object(s) are stored in the variable obj-within-window.

. The variable no-window-next indicates the number of windows in the next frame. The addresses of these window descriptors are stored in the variable nwptr. Similarly, the variable no-window-current and no-window-previous indicate the number of windows in the current and previous frames, respectively. Also the variable cwptr and pwptr contain the addresses of the windows in the current and previous frames, respectively.

4.6.3 Examining and Correcting Inconsistent Motion Information

Before gathering information about each moving object, each object's moving direction should be examined and corrected in order to have consistent motion information. Also some additional information can be derived at this stage.

First, changes of moving direction in the numeric level are examined. Changes to be examined are restricted to changes of the original moving direction to its adjacent one. For example, if an object's moving direction is south, encoded as "s" in the current frame, then its adjacent moving direction is southwest or southeast. This examination is required due to the possible imprecision in the corner matching process. It is possible that an object moving in a 20 degree direction, encoded as "s," in the current frame, while it moves in a 27 degree direction in the next frame.

results in encoding as "ne." Such a small amount of deviation should be neglected, and the new direction is still encoded as "e."

Second, if there is an abrupt change in the moving direction, then the consistency of moving direction in the three consecutive frames is examined. For example, an object changes its moving direction from "s" to "n," but it changes back to "s" immediately. This phenomenon might come from the inaccuracy of corner matching. Therefore, a numeric value is assigned from averaging its previous and next frames. Also the moving direction is changed from "n" to "s."

Third, the consistency in the direction of motion of each moving object is checked after performing the above two steps. When an object's direction is not the same as in the previous frame, the consistency constraint only allows transit to an adjacent direction and within a 45 degree difference. For example, a moving direction encoded as "s" can only transit to "sw" or "se" and within a 45 degree difference. If such consistency cannot be maintained, the moving direction is replaced so that its value is the same as the one in the previous frame. Also this piece of information is marked as "no-confidence." This phenomenon might result from occlusion. Since occlusion may exist for several frames, it is assumed that the object's moving direction does not change during the course of occlusion. This assumption enables the system to resume the consistency examination after occlusion.

4.6.4 Results

Table 9 in the Appendix shows the results of window classification on the data listed in table 4. Also the moving objects contained in each window are listed. All windows in image sequence one are correctly classified.

Tables 10 to 12 in the Appendix show the results of examining and correcting moving directions bassed on the data listed in taole 6. The symbol "$" following the datum indicates that piece of datum has been corrected. All inconsistent data have been satisfactorily corrected.

4.7 Interpretation

4.7.1 Purpose

The ultimate objective of this system is to derive descriptions about object movements. Information about the moving objects and windows is gathered in compact forms, described in Section 4.6, to facilitate its manipulation. Based on the information and the task requested by a system user, the system is capable of providing descriptions of the movements of a specified object. The block diagram in figure 4.70 shows the inputs and output of the information flow of this process.

4.7.2 Implementation

Overview. A set of primitive movement descriptors and events has been determined and developed from which higher

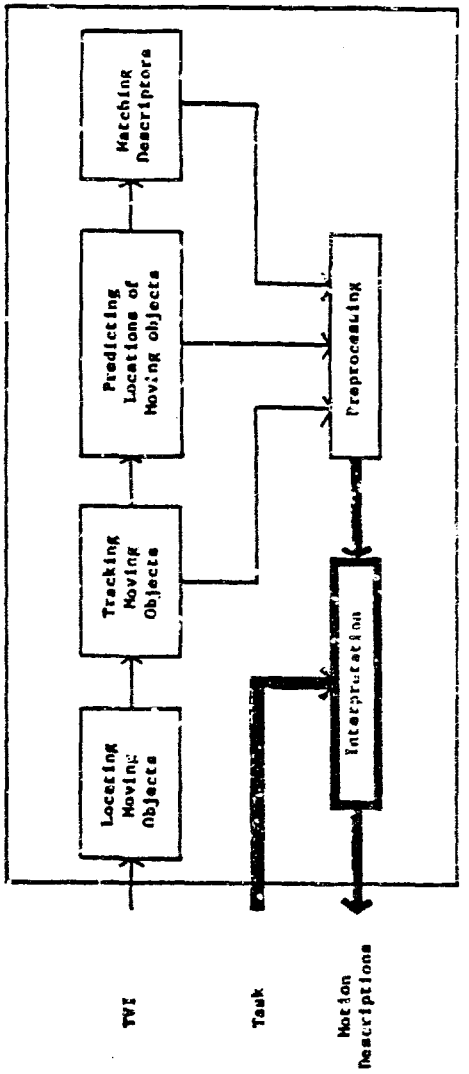


Figure 4.70 Inputs and output of the process of interpretation

level events are derived. When the system accepts a task from its user, the set of primitive descriptors and events is examined. The existence of a single descriptor/event or a set of these descriptors and/or events triggers the selected first-level event verifications. Then the existence of the first-level events triggers the selected second-level event verifications.

The verificatoin processes for the first- and second-level events are implemented via a two-level production system. A production system [9], [71] is comprised of a set of production rules. The formalism of "IP-THEN" (condition-action) is used as the mechanism for representing knowledge about motion interpretation. The results of verification are arranged to generate motion descriptions for the object specified . Other information, such as the appearance of a new object in the visual field, is also provided for the user's attention.

Figure 4.71 shows a typical interpretation process of the implemented system. The primitive movement descriptors and events are derived from low-level image sequence analysis, e.g., the "location change" is a primitive movement descriptor, while the "the disappearance of an object" is a primitive movement event. The existence of these descriptors and/or events triggers the first-level event verifications. In this figure, the primitive movement descriptors and/or events P1 and P2 trigger the verification of the first-level

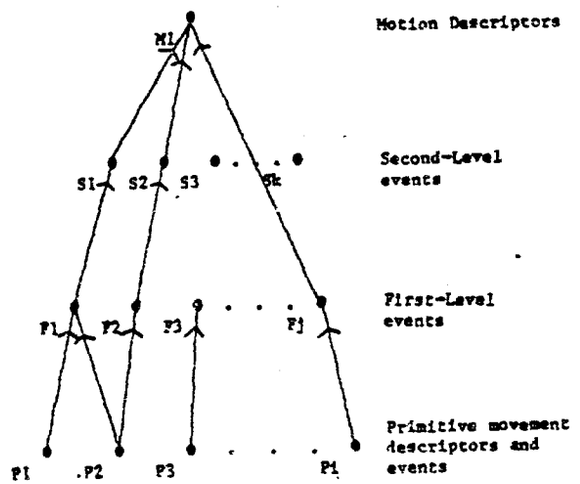


Figure 4.71 Illustration of interpretation process

event F1. Similarly, F1 can trigger the verification of S1, a second-level event. The arrows represent the flow of information from one level to another. The function of the top level is to receive and edit information from the first- and second-level events in order to generate motion descriptions as the system's output.

Production system. A production system is comprised of a set of rules. Each rule is a statement "If this condition holds, then do this action." For example, if there is a change in object's direction of motion, then the system verifies the event of turning left/right. In the system, each left-hand side (condition side) is a set of flags reflecting the situation while each right-hand side (action side) is a program verifying an event. The verified events then set up another set of flags for their next highest level event verifications.

The major motivations for production system implementation are its flexibility and naturalness. The flexibility, due to high modularity, provides easy expansion of the implemented system in the future.

The implemented production system has two levels. The first-level has primitive movement descriptors and events as inputs which activate a set of verifications. The verifications resulting from the second-level inputs activate another set of verifications.

Primitive movement descriptors and events. The set of

primitive movement descriptors and events is described below. They trigger the first-level event verifications.

P1. The possible appearance of a new object(s):
This information is derived from the variable spfptr in the data structure representing an object. When the variable has a value of zero, it means an object can not link back to the previous frame. Thus it indicates the possibility of a new object appearing in the visual field.
The information about the possible appearance of a new object(s) is represented by: the number of new objects, the objects' labels, and the frame times.

P2. The disappearance of an object(s): This information is derived from the variable snfptr in the data structure representing an object. When the variable has a value of zero, it means an object can not link forward to the next frame. To avoid confusion, it should be pointed out that the meaning of "disappearance" is in the sense of an object which can not be tracked. This may indicate an object which stops its movement.
The information about the disappearance of an object(s) is represented by: the number of objects, the objects' labels, and the frame times.

P3. The appearance of an object in the image boundary(ies): This information is derived from the variable extremes in the data structure representing an object. This variable indicates a rectangular window, containing an object, represented by a quadruple

(LV, UH, RV, LH),

where LV and RV represent the x-coordinates of the left and right sides of the vertical boundaries, respectively, and UH and LH represent the y-coordinates of the upper and lower horizontal boundaries, respectively. For an image of 512x512 in size, if LV UH is less than 11 or if RV or LH is greater than 502, the system will register this event. The verification of the event of entering/leaving the visual field in the next highest level will be based on this piece of information.
The information about the appearance of an object in the image boundary(ies) is represented by: the number of appearances, the object's label, and the frame times.

P4. The appearance of "no confidence" in deriving information of an object's moving direction:

This information is derived from the variable confidence in the data structure representing an object. A flag is set up when this event is true.

P5. The appearance of location changes: This information is derived from the variable location in the data structure representing an object. When this variable in the current frame has a different location compared to the same variable of the same object in the next frame, the change will be registered.

The information about location changes for an object is represented by: the number of location changes, the object's label, the frame times, and the current and next locations.

P6. The appearance of changes in direction of motion: This information is derived from the variable moving-direction in the data structure representing an object. When this variable in the current frame has a different moving direction compared to the same variable of the same object in the next frame, the change will be registered.

The information about moving direction changes for an object is represented by: the number of moving direction changes, the object's label, the frame times, and the current and next moving directions.

P7. The appearance of vertical boundary position changes of windows containing the same object: This information is derived from the variable extremes in the data structure representing an object. The left vertical boundary position, i.e. LV, in the current frame is compared with the corresponding one of the same object in the next frame. If there is a difference over a threshold, the significance of this change is registered. Similarly, the change of the right vertical boundary position is examined. Only the maximum difference of these two position changes is kept.

The information of vertical boundary position changes of windows containing the same object is represented by: the number of vertical boundary position changes, the object's label, and the frame times.

P8. The appearance of horizontal boundary position changes of windows containing the same object: This information is derived in a manner similar to the procedure described above.

P9. The appearance of changes in the type of window containing the same object: This informa-

tion is derived from the variable window-type in the window descriptor data structure. A window may be classified as "regular," "split," or "merged." The type "regular" denotes a window containing one object. The type "split" denotes a window containing only a part of an object. And the type " merged" denotes a window containing more than one object. When this variable in the current frame has a different value compared with its value for the window containing the same object in the next frame, the change will be registered. The information of window type changes containing the same object is represented by: the number of window type changes, the object's label, the frame times, and the current and next window types.

First-level events. The set of first-level events is described in the following. Their verifications are triggered by primitive movement descriptors and events.

P1. Newly appearing object(s): The existence of P1 triggers the verification of the appearance of a new object(s). A new object might originally be stationary but then start its movement. Or it is entering the visual field. However, there is an another possibility -- an

object which stopped its movement at an earlier time in the period of observation might start its movement again. The last possibility should be detected and excluded to avoid providing false information containing the appearance of a new object(s). The existence of P1 is confirmed by determining that no object which stopped its movement in approximately the same region started up again. This done simply by computing the differences of the extremes for a window containing a possible new moving object and a window containing an object which stopped its movement. When any difference is greater than a threshold, then the appearance of a new object is confirmed. The information about the appearance of a new object(s) is represented by: the number of new objects, the objects' labels, and the frame times.

P2. Object stopped its movement: The existence of P2 triggers the verification of an analyzed object which stops its movement. When the existence of P2 is not due to a failure of the tracking algorithm described in Section 4.3, the object has not necessarily stopped its movement. An object leaving the visual field

may also result in the existence of P2. This possibility is detected and excluded to avoid providing false information concerning an object which stops its movement. The existence of P2 is confirmed by determining an analyzed object which does not appear in the image boundary(ies).

The information about an object which stops its movement is represented by: the object's label and frame time.

P3. Object entering/leaving the visual field: The existence of P3 triggers the verification of an analyzed object entering/leaving the visual field. Several reasons may result in a window containing an analyzed object in the image boundary(ies). First, of course, is an object which is entering or leaving the visual field. Second, the noise in the image recording process may cause artifacts in the region growing described in Section 4.2. This results in a larger window in the image boundary(ies) when an object's image is close to, but not yet in the image boundary(ies). Third, the prediction algorithm, described in Section 4.4, attempts to predict the location of an analyzed object in the form of a larger window containing the object. This may result in having an object's image in the image boundary(ies).

The last two cases can be detected and excluded in the information processing by disregarding any information about an object which is in the image boundary(ies) when it does not last at least three time units. The strategy for verifying an object entering/leaving the visual field is as follows:

1. Disregard any information about an object which is in the image boundary(ies) when it it does not last at least three time units.
2. Analyze the location and direction of motion of an object at the time when it completely enters/leaves the visual field. If an object does not completely enter/leave the visual field, the latest time of an object in the period of observation is chosen.
3. Compare the values of the variable area (in representing an object) in the first and last times.

As pointed out earlier, step 1 is used to exclude the unreliable information. In step 2, the event of entering/leaving the visual field

can be determined by an object's location and direction of motion. When an object is close to but not yet in the image boundary(ies), and if its movement is parallel to an image boundary, there are two possibilities: (1) an object is moving parallel to an image boundary or (2) an object is entering/leaving the visual field. This ambiguity is due to the quantization of moving direction. This is handled in step 3. In that step, a ratio of areas in the first and last times is computed. If an object is entering the visual field, then the ratio should be far less than one. Conversely, if an object is leaving the visual field, then the ratio is much greater than one.

The information about an object entering/leaving the visual field is represented by: the object's label, the type(s) (entering/leaving), and the frame time(s).

F4. Object partially occluded by a stationary object: The existence of P4 triggers the verification of an analyzed object which is occluded by a stationary object. Recalling the consistency checking described in Section 4.6, an object's moving direction may be marked as

"no-confidence" when consistency can not be held. This will give the interpretation process a "good guess" of the occurrence of occlusion.

The information about an object occluded by a stationary object is represented by: the number of occlusions, the object's label, and the start frame time(s) and end frame time(s).

F5. Object partially occluded by a stationary object: Alternately, the existence of P9 can also trigger the verification of an analyzed object occluded by a stationary object. In this case, a further examination of the occurrence of a change in window types from "regular" to "split" is performed. If windows contain the analyzed object, then the duration of the "split" is computed as the period of occlusion. Again, this is a "good guess." Compared with F4, F5 provides better information in testing image sequence one. The information representation is the same as in F4.

F6. Object moving across the visual field: The existence of P5 triggers the verification of an analyzed object which moves across the visual field. The existence of F6 can be determined by comparing the first and last lo-

cations of an analyzed object in the period of observation. If the variable location has a value of "t-l," "c-l," or "b-l" at the time the object first appears and a value of "t-r," "c-r," or "b-r" at the time the object last appears, or vice versa, then the existence of an object moving across the visual field is confirmed.

The information about an object moving across the visual field is represented by: the object's label, the beginning location, and the last location.

F7. Object turning left/right: The existence of P6 triggers the verification of an analyzed object turning left/right. This is simply done by transforming the original direction of motion and the new direction of motion into "left/right." For example, if an object moving northwest in the beginning is now moving north, then these two pieces of information result in a transformation to "right."

The information about an object turning left/right is represented by: the number of turns, the object's label, the turning direction(s) (left/right), the moving direction(s) before turn(s), the moving direction(s) after turn(s), and the frame time(s).

F8. Object moving east, west, northeast, northwest, southeast, or southwest: The existence of P7 triggers the verification of an object's moving direction. An object's moving direction is determined by the moving direction at the first frame time in the list of significant changes in vertical boundary positions.

The directional information is represented by: the object's label and the frame time.

F9. Object moving south, north, southeast, southwest, northeast, or northwest: In a manner similar to F8, the existence of P6 triggers the verification of an object's direction. The information representation is the same as in F8.

F10. Object moving south, north, east, west, southeast, southwest, northeast, or northwest: The existence of P7 and P8 triggers the verification of an object's direction. The verification and information representation are the same as in F8 and F9.

F11. Object completely occluded by a stationary object: The existence of P2 also triggers the

verification of an analyzed object which is completely occluded by a stationary object. If a moving object is not in the image boundary(ies) in frame i, and it has more than a 20% deviation in the ratio of areas in frames i and (i-1), then the event of the object is completely occluded by a stationary object is confirmed. Notice that frame (i+1) is the time when the object cannot be tracked. The information about an object completely occluded by a stationary object is represented by: the object's label, and the frame time.

Second-Level events. The set of second-level events is described in this section. Their verifications are triggered by the first-level events.

S1. Object moving again: The existence of F2 triggers the verification of an analyzed object which stopped its movement at an earlier time and started up its movement again. If an object stopped its movement at an earlier time, then the existence of S1 is confirmed by determining that the analyzed object started its movement approximately in the same region. This is accomplished by computing the differences of the extremes for a window containing an object (which stopped its movement) and a

window containing a possible new object. When each difference is less than a threshold, the existence of S1 is confirmed.

The information about an object which stopped its movement at an earlier time and started up its movement again is represented by: the number of stop-start's, the object's label(s), the object's previous label(s), the frame time(s) for starting movement(s), and the frame time(s) for previously stopped movement(s).

It is worth noting that even if it is the same object, a new object label is assigned each time the object restarts.

S2. Events of encountering, leaving, passing, and encountering-and-leaving: The existence of F8, F9, or F10 triggers the verification of the events of encountering, leaving, encountering-and-leaving, and passing. The illustration of these events is shown in figure 4.72. The definition of encountering is two objects moving in opposite directions and enclosed in the same window. The instant that the two objects are first enclosed in the same window is defined as the time of encountering. The event defined to be "leaving" occurs when

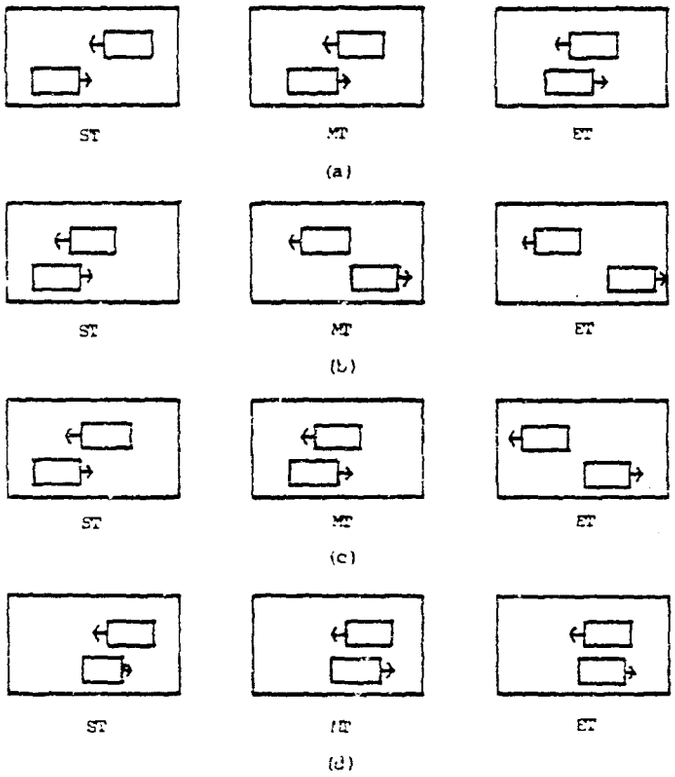


Figure 4.72 Illustration of the events of (a) encountering, (a) leaving, (c) encountering-and-leaving (d) passing. ST, MT, and ET denote the start time, middle time, and end time, respectively, in the period of observation.

two objects are moving in opposite directions and are enclosed in the same window, and one frame later are in different windows. The instant that the two objects are last enclosed in the same window is defined to be the time of leaving. Encountering-and-leaving is defined as the combination of the above two definitions. The event of passing is defined as the time between encountering and leaving. The verification of S2 is performed by the procedure described below:

1. Find at least one window, in the period of observation, which has window type "merged" and contains an analyzed object; otherwise, stop this procedure.
2. Find a set of windows, continuous in time, each enclosing the analyzed object and having window type "merged." This means the analyzed object and other object(s) are enclosed within the same window during the period.
3. Establish a list of objects which are enclosed by the same window as the analyzed object during the period.
4. Delete each object, listed in Step 3, which does not move in a direction op-

posite to the analyzed object's moving direction.

5. Determine the start and end times of the event between each surviving object in step 4 and the analyzed object.
6. Classify events (described below).
7. Go to step 3 if there is an another set of windows which meets the conditions described in step 2; otherwise, stop this procedure.

In step 6, the classification of events is done by examining the window type, one time unit before the start time, and the window type, one time unit after the end time. If, one time unit before the start time, the window does not contain both the analyzed object and the object in the list, then it is classified as the event of encountering. If the window, one time unit after the end time, does not contain both the analyzed object and the object in the list, then it is classified as the event of leaving. If the above two conditions hold, then it is classified as the event of encountering-and-leaving. Conversely, if the above two conditions do not hold, then it is classified as the event of passing.

The information about the event(s) is represented by: the number of event(s), the type(s) of event(s), the objects' labels in the object list, and the start time(s) and end time(s).

Control of verification. The interpretation process is divided into two levels, first- and second-level events, which inherently provide the order of verification. The first level has higher priority than the second level. However, within each level, the order of verification is not important because the motion events are independent of each other in time. After executing a rule, the rule will not be executed again.

It is worth noting that the verifications of F4 and F5, F8 and F10, and F9 and F10 in the first level required special control. Since F4 is able to provide better information than F5, the verification of F4 is selected and verification of F5 is voided, if both requests for verification occur. For the same reason, if verification of F8 and F10, or F9 and F10 can be triggered, only F10 will be verified.

Input. At this time, the input of the interpretation process is a triple

(ST, ET, OBJECT),

where ST and ET denote the start and end times for the period of observation, and OBJECT denotes the object's label which specifies the object of interest.

It is presumed that all the motion descriptions and related events concerning the analyzed object are requested by the system user. However, this process can be expanded in response to a specific request in the future. For example, if a system user only wants to see whether there is an event of encountering for the analyzed object, the interpretation process only examines and provides the required information. Thus, the input of the interpretation process will be a quadruple

(ST, ET, OBJECT, TASK),

where TASK denotes the required task to be accomplished by the interpretation process.

Output. The output of the interpretation process is English sentences which provide motion descriptions and related events. Several examples of the output are shown in the next subsection.

4.7.3 Examples of Output

Example 1: (1,30,obj1). The input is a (1,30,obj1), where obj1 is the label of the CAR in the analyzed image sequence one. The output is in the following:

Object obj1 in the beginning is going northwest
then turns to its rights and now is going north.

Example 2: (1,30,obj2). The CAR is labeled as obj2 in the analyzed image sequence one.

Object obj2 is entering the visual field at frame 1.
Object obj2 is going east.
Object obj2 encounters obj3 at frame 17 and leaves
at frame 27.

Object obj2 is moving across the visual field from
left to right.
Object obj2 is partially occluded by a stationary
object beginning from frame 19 to frame 30.

Example 3: (1,30,obj3). The WAGON is labeled as obj3 in the analyzed image sequence one.

Object obj3 is entering the visual field at frame 1.
Object obj3 is going west.
Object obj3 encounters obj2 at frame 17 and leaves
at frame 27.
Object obj3 is partially occluded by a stationary
object beginning from frame 15 to frame 20.

Example 4: (1,20,obj2). This example demonstrates the verification of the event encountering in the analyzed image sequence one.

Object obj2 is entering the visual field at frame 1.
Object obj2 is going east.
Object obj2 encounters obj3 at frame 17.
Object obj2 is partially occluded by a stationary
object beginning from frame 19 to frame 20.

Example 5: (20,30,obj3). This example demonstrates the verification of the event leaving in the analyzed image sequence one.

Object obj3 is going west.
Object obj3 leaves obj2 at frame 27.

Example 6: (18,25,obj2). This example demonstrates the verification of the event of passing in the analyzed image sequence one.

Object obj2 is going east.
Object obj2 and object obj3 pass each other from
frame 18 to frame 25.
Object obj2 is partially occluded by a stationary
object beginning from frame 19 to frame 25.

Example 7: (1,10,obj1). This example demonstrates the verification of the event S1 in the analyzed image sequence

two.

Object obj1 is entering the visual field at frame 1.
Object obj1 is going west.
Object obj1 stops its movement at frame 7.
Its new label is obj4. Object obj4 begins its movement again at frame 9.
Object obj4 is going west.

Example 8: (1,14,obj1). This example demonstrates the verification of the event F11 in the analyzed image sequence three.

Object obj1 is entering the visual field at frame 1.
Object obj1 is going west.
Object obj1 is completely occluded by a stationary object at frame 7.
New object appears at frame 9. It is labeled as obj4.
Object obj4 is going west.

5 EXTRACTION OF IMAGES OF MOVING OBJECTS

5.1 Purpose

Since the major concern of this study is the description of object movement, the work described in Chapter 4 does not involve the extraction of object images. It is known that each object is surrounded by a rectangular window; however, whether a given pixel within that window belongs to the image of a moving object is unknown. The extracted object image resulting from the procedure described in the next section provides a 2-D model of object grayvalue distribution.

5.2 Procedure

The procedure for extracting the image of a moving object is as follows:

1. generate a binary thresholded difference picture (BTDP) in the window containing an object image,
2. segment the BTDP in step 1 into regions,
3. delete the noise regions,
4. find the boundary image points of the binary resulting from step 3,
5. identify the convex hull points from the set of boundary points,
6. construct the convex hull, and
7. substitute a grayvalue for each pixel within the convex hull.

Step 1 is accomplished by subtracting the grayvalue at each pixel position in one frame from the corresponding pixel in the adjacent frame. Then the absolute value of the result of the subtraction is taken. A pixel position with a significant change in its grayvalue is marked as a black point, or, conversely, as a white point. Thus, a binary thresholded difference picture is generated. This step results in extracting the object boundary. Figure 5.1 illustrates a typical result after performing this step.

The second step is the segmentation of the binary thresholded ifference picture into regions. The region growing algorithm described in [59] is applied.

In step 3, noise black regions are deleted. Generally, the largest black region is kept if most object boundaries are well-connected (see figure 5.1) to form the largest region. However, regions whose areas are over a threshold are kept if most object boundaries are not well-connected (see figure 5.2) to form the largest region. Applying one of the noise region deleting schemes, the remaining black regions are located on an object's boundaries and its internal areas. Figures 5.3 and 5.4, respectively, illustrate the result after performing this step on the previous outcome.

Next, the search of boundary points is performed. For each scan line, the extreme points are registered if they are black points. Figure 5.5 illustrates the result after performing this step in figure 5.3.

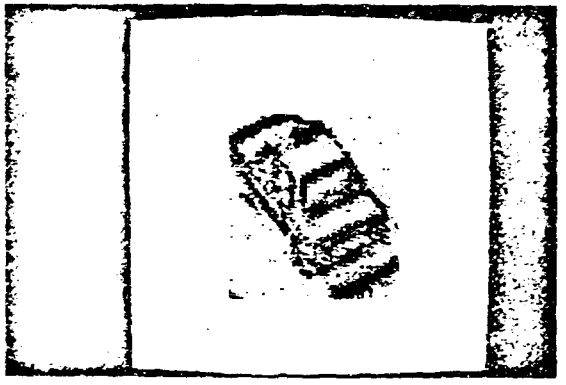


Figure 5.1 Illustration of a BTDP (with well-connected boundaries)

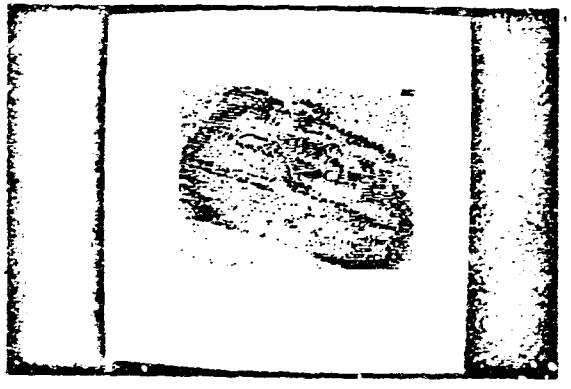


Figure 5.2 Illustration of a BTDP (without well-connected boundaries)

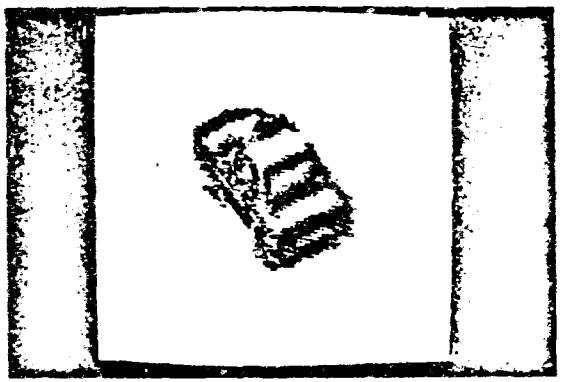


Figure 5.3 Illustration of a BTDP after removing noise regions in figure 5.1

Figure 5.4 Illustration of a BTDP after removing noise regions in figure 5.2

Generally, since the black points resulting from performing step 4 can not either form a closed contour or have a good approximation of an object's boundary, a convex hull [18] has to be determined, resulting in an approximation of an object's boundary. A subset of points which are found in step 4 should be identified as convex hull points.

In step 5, the algorithm of identifying convex hull points [31] is performed. If there are n boundary points, the algorithm is described as follows:

(1) Set an origin point outside the n boundary points, say at the lower left corner.

(2) Find a boundary point whose measured angle is minimum. An angle is determined by a radial arm from the origin, in a counterclockwise direction from the horizontal line passing the origin. For equal minimum angles, pick the point closest to the origin. Thus, a convex hull point is identified.

(3) Change the origin as the last convex hull point found in (2), and go back to (2) until the first convex hull point is found again.

Figure 5.6 shows the identified convex hull points in figure 5.5.

Next, the convex hull is constructed by connecting each adjacent pair of convex hull points. Figure 5.7 illustrates the result after performing this step in figure 5.6.

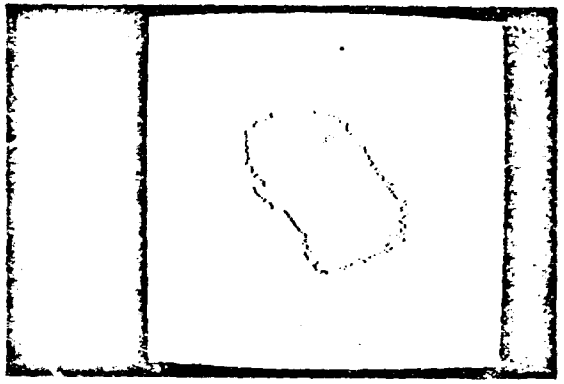


Figure 5.5 Found boundary points in figure 5.3

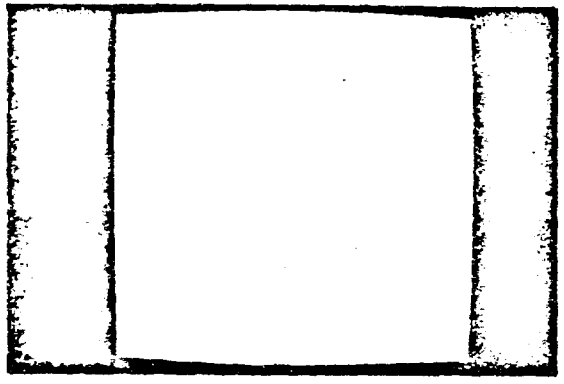


Figure 5.6 Identified convex hull points in figure 5.5

The last step is simply the substitution of a grayvalue for each pixel within the convex hull. Thus, an object grayvalue distribution is acquired. Figure 5.8 illustrates the result after performing this step on the previous outcome.

5.3 Results

Frames 1, 5, 11, 16, 19, 25, 29, 32 of the first image sequence were selected to test the proposed procedure. The results are illustrated in figures 5.9 to 5.14. It is worthy to note that even the images of the CAR and WAGON, which are partially occluded by the tree, are extracted.

In addition, since the images of the CAB are well-contrasted against their background, a simple thresholding technique [56] can delete a part of the background included within a convex hull. This step is done by changing each pixel position having a grayvalue between the range of the background grayvalues (compare figure 5.15 with figure 5.16). A small number of pixels in the internal areas of an object image will also be changed to have the background grayvalue. However, the outcome does not attract the observer's attention.

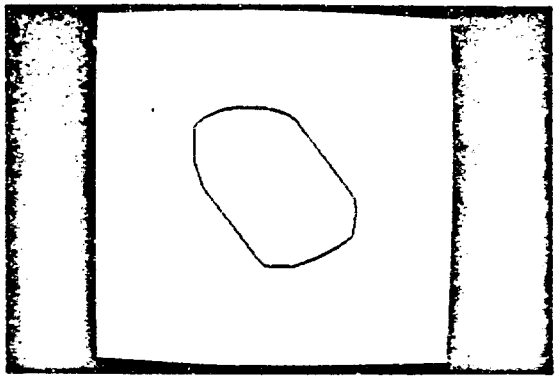


Figure 5.7 Illustration of a convex hull

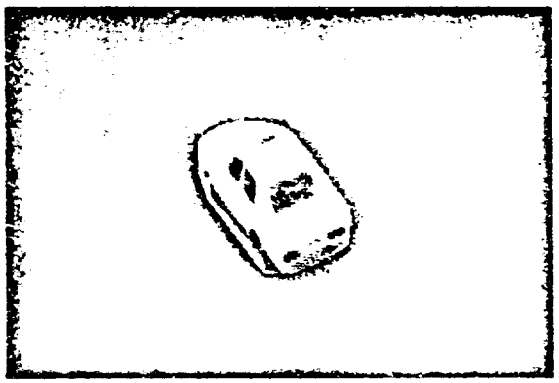


Figure 5.8 Illustration of an extracted object image.

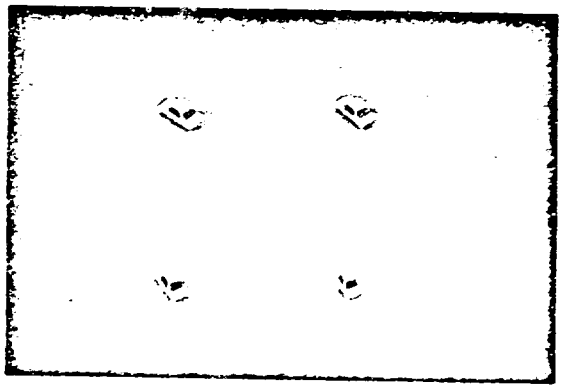


Figure 5.9 Extracted images of the CAB (in frames 1, 5, 11, and 16 of image sequence one)

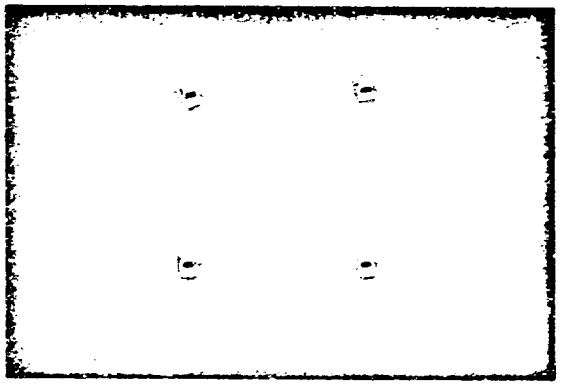


Figure 5.10 Extracted images of the CAB (in frames 19, 25, 29, and 32 of image sequence one)

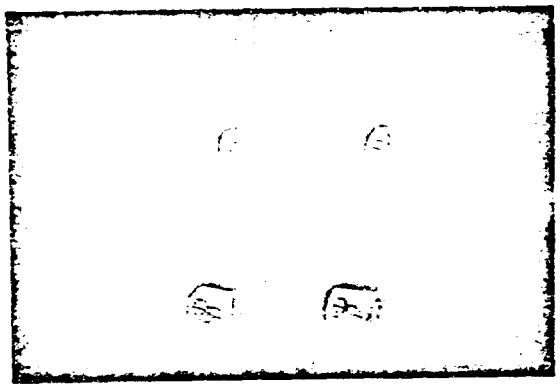


Figure 5.11 Extracted images of the WAGON (in frames 1, 5, 11, and 16 of image sequence one)

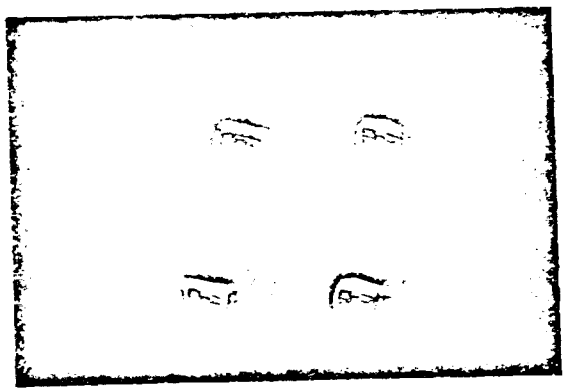


Figure 5.12 Extracted images of the WAGON (in frames 19, 25, 29, and 32 of image sequence one)

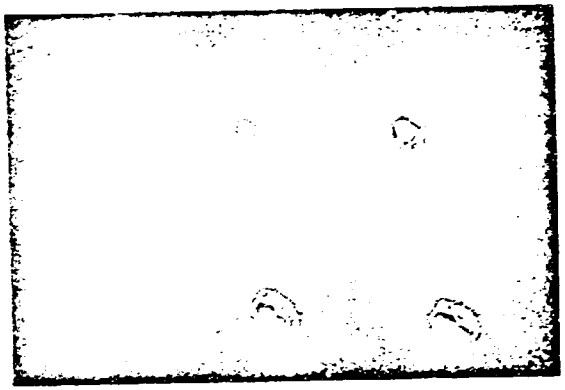


Figure 5.13 Extracted images of the CAR (in frames 1, 5, 11, and 16 of image sequence one)

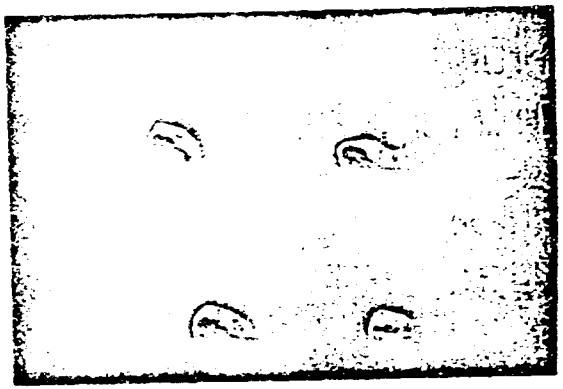


Figure 5.14 Extracted images of the CAR (in frames 19, 25, 29, and 32 of image sequence one)

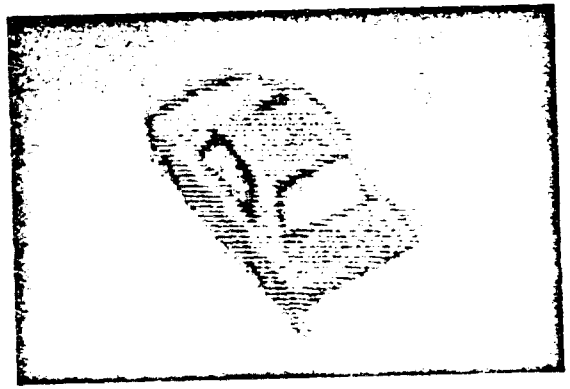


Figure 5.15 Illustration of an extracted image of the CAB

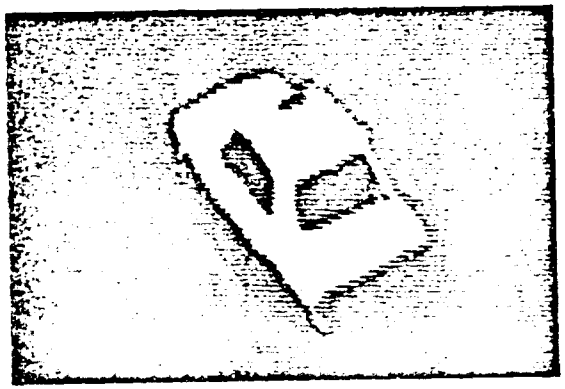


Figure 5.16 Illustration of thresholding the image of the CAB in figure 5.15

APPENDIX D

"Displacement Field Calculation by the Motion Detection Transform with Applications in FLIR Imagery"

by

Margie Groves

ABSTRACT

Our research examines a new transform-based technique for calculating the displacement field of a dynamic sequence generated by a single moving optical sensor. The transform, based on the one dimensional Fourier transform, computes displacement vectors from raw positional intensity data. Variations on two existing methods of analysis of displacement fields are employed to extract additional information. We use a variation on Holben's MTI (Moving Target Identification) algorithm to effect frame-to-frame registration; and, we adapt a technique due to Nagel for calculation of the parameters of sensor motion.

1.0 Introduction.

This paper, first, supplies background on terminology and on notation. Following these preliminaries, the transform employed in our research is described, and, results with synthetic data are cited. Next, in section 4, the paper exhibits the problems of applying the transform in real life contexts and gives our solutions to these problems. Section 4 also develops this use of the transform in formation of displacement fields. Finally, the last two sections incorporate the displacement field computations into our chosen applications: spatial registration and computation of sensor motion parameters. The first of these two sections provides results on FLIR data.

2.0 Background.

This section clarifies our terminology as it applies to the current topic.

'Dynamic imagery' usually refers to a time sequence of photographs. However, we limit our study to dynamic imagery generated by a single moving optical sensor (such as a camera mounted on an RPV). All subsequent discussion references imagery gathered by a monocular optical system in motion unless otherwise specified.

When a camera takes photographs as it moves past a non-homogeneous terrain, the picture plane (2D) projection of a

three-dimensional texture point changes picture plane position between frames. If we draw a vector from the projection of every texture point in frame i to its new position in frame (i+1) we generate a displacement field. Thus the displacement field shows how the image of each texture point moves in the picture plane between frames.

Registration of two temporally adjacent frames of a dynamic sequence aligns new and old positions for all (motionless) texture points. The absolute frame difference ($|f2[i,j] - f1[i,j]|$ for all i,j) of two perfectly registered frames of a dynamic sequence contains zeros everywhere except in the vicinity of contrasting objects moving relative to the background. We refer to the pixels in (relative) motion as target pixels. In actuality, the system encounters noise and changes in the intensity-to-grey scale mapping. Additionally, perfect registration of two frames of a dynamic sequence is a yet unsolved problem, especially in the absence of range data. We adopt conventional terminology and term high intensity pixels not due to moving objects as false targets. (Any dc (constant) pixel intensity can be discounted as due to change in grey scale mapping.)

2.1 Assumptions.

This section defines out assumptions as to data content and character.

We assume our data has the following characteristics:

It is passively sensed data, either optical or thermal.

It represents a time sequence produced by a moving sensor whose motion can be characterized, at least for short periods of time, as having a constant velocity.

A high enough frame rate is used, ie, sufficient "inter- frame overlap" exists so that registration is a meaningful process (with "sufficient" as yet undefined).

(Projections of) objects possessing velocities (with respect to the background) occupy a small portion of the image plane.

We assume no fore-knowledge of specific image contents. Additionally, we assume that the sensor viewing angle is low oblique (ie, the horizon does not appear in the images) thus we have no cues from which to calculate viewing angle.

In addition, we wish to be able to tolerate poor image quality. For example, the real-world data we employ is notably undersampled FLIR with a low SNR.

Given these conditions, feature matching approaches to registration are impractical, as are any other techniques dependent on local image characteristics.

2.2 Notation.

Our notation is similar to that in Nagel (N1,N2,N3). The following text summarizes this notation.

World coordinates appear, herein, as (X,Y,Z). Lower case qualifiers signify a particular texture point. For example, (Xm,Ym,Zm) designates the mth point in world coordinates. The camera system coordinates appear as (XC,f,ZC) (C - modifier). The term "focal length" refers to the so-called effective focal length of the imaging system; and, f is used to represent this value. Camera coordinate system has its origin located at the lens center and the point (0,f,0) at the base of the focal axis. We use a camera coordinate system with YC directed along the focal axis, ZC pointing "up" and XC pointing to the "right", defining a plane parallel to the image plane. The letter 'P' indicates picture plane coordinates, such as (XP,f,ZP). Image plane coordinates are closely related to camera coordinates. The relationship between camera coordinates and picture plane coordinates for the ith point is

$$(XCi,YCi,ZCi) = si(XPi,f,ZPi),$$

where si is a scale factor such that f*si = YCi. Note that camera coordinates retain relative depth information for each texture point, that is, they yield 3D structure. Picture plane coordinates, however, lose relative depth information, retaining only the visual direction to each texture

point. The world coordinate system is static throughout the image sequence while the camera coordinate system changes, with respect to the world coordinate system, from frame to frame. Device coordinates, or pixel addresses, appear as (XD,ZD) (D -device- modifier).

Lower case coordinates (x,y) are employed as a generic 2D system.

We choose central projection as our model of the image formation process.

3.0 The Motion Detection Transform.

The motion detection transform (MDT) combines the Exponential Area Transform (ExpAT - defined below) with an FFT (fast Fourier transform) to extract, from a dynamic sequence, the component of image plane velocity (in pixels per frame) in either spatial dimension. See (Figure 1) for the mathematical formulation of the MDT. Two clarifications of the MDT are in order. First, the transform computes movement parallel to some axis. Secondly, two applications of the MDT supply the two components of the 2D velocity, one application for each of the coordinate axes. References (R1 and R2) provide the complete theoretical basis for the transform. The following text presents a somewhat intuitive explanation.

Let t be the number of frames in a time sequence generated by a stationary camera. For frame one, imagine an homogeneous mxn frame containing a single contrasting two-dimensional object. For simplicity, assume the background intensity is zero, and that the object occupies a single pixel and has unit intensity. The steps of the MDT for the sequence follow. Project the plane onto (say) the x axis (ie. sum across the columns). This yields a vector whose entries are, identically, zero excepting at the projection of the object. Multiply the ith component of this vector by an exponential, exp(-2*j*k*i/t), and sum the m products.

The resulting sum equals exp(-2*j*k*i/t). Let the (2D) object move unit distance with respect to the y axis between frames one and two, and repeat the three steps (above: project, multiply by the appropriate exponential, and sum) for frame two. The result is exp(-2*j*k*(i+1)). If the object continues to move one pixel per frame, and if the three steps are repeated for each frame of the sequence, the resulting vector of sums traces out a complex sinusoid with frequency k. If, however, the object moves p pixels (where the sign of p implies direction) between frames, the sinusoid has frequency p*k. The Fourier transform of the result vector consists of a single peak at frequency k*p. The result vector is the ExpAT of the dynamic sequence for the x dimension; and, its Fourier transform is the MDT. A peak search in the MDT domain, followed by division by k divulges the component of object velocity in the x direction.

Alternatively, if each mxn frame contains some arbitrary pattern; but no motion occurs during the sequence, all frames of the sequence are the same. Consequently, all entries of the result vector (ExpAT) are identical; and, the MDT (Fourier transform of the result vector) consists of a single dc (zero frequency) peak.

The explanation for the more general case of an object moving across an arbitrary background combines the two cases

given above. The sequence generated by pointwise summation of the two sequences described above closely models an object in motion across an arbitrary background. Since the steps of the MDT constitute a linear operator, their application to the sequence representing the object moving across an arbitrary terrain is equivalent to summing the two previously derived MDTs. Thus the MDT of the sequence representing the general case consists of a peak at frequency k*p and a dc peak.

Next consider the case of moving sensor, immobile terrain. When the sensor moves across a textured terrain with velocity p pixels per frame, the image of (stationary) texture points appear to move at -p pixels per frame. Reference (R2) shows that the MDT "maps pixels moving at the same velocity into the same location in transform space." So, while performance is demonstrably better in the stationary camera case, extrapolation to the translating sensor model is straight-forward. The degradation in performance in the moving camera case stems from the difference between new pixels entering the transformation window and the old pixels leaving the window.

Figure 1: MOTION DETECTION TRANSFORM EQUATIONS *

Exponential Area Transform

$$(1)\quad Fx(kx,t) = \sum_{x=0}^{a-1} \sum_{y=0}^{b-1} f(x,y,t)\, \exp(-j*2*kx*x/c)$$

$$kx = 0, 1, 2,\ldots$$

$$0<=x<a \quad 0<=y<b \quad 0<=t<c$$

$$(2)\quad Fy(ky,t) = \sum_{y=0}^{b-1} \sum_{x=0}^{a-1} f(x,y,t)\, \exp(-j*2*ky*y/c)$$

$$ky = 0, 1, 2,\ldots$$

$$(3)\quad Gx(kx,f) = 1/c \sum_{t=0}^{c-1} F(kx,t)\, \exp(-j*2*f*t/c)$$

$$f = 0, 1, 2,\ldots$$

$$(4)\quad Gy(ky,f) = 1/c \sum_{t=0}^{c-1} F(ky,t)\, \exp(-j*2*f*t/c)$$

where kx and ky are weighting factors which determine both the maximum velocity detectable (without aliasing) and the resolution with which velocity may be detected.

* from (R1) and (R2).

3.1 Preliminary Results.

This section exhibits the highlights of our research to date on synthetic imagery involving translating sensors and translating objects.

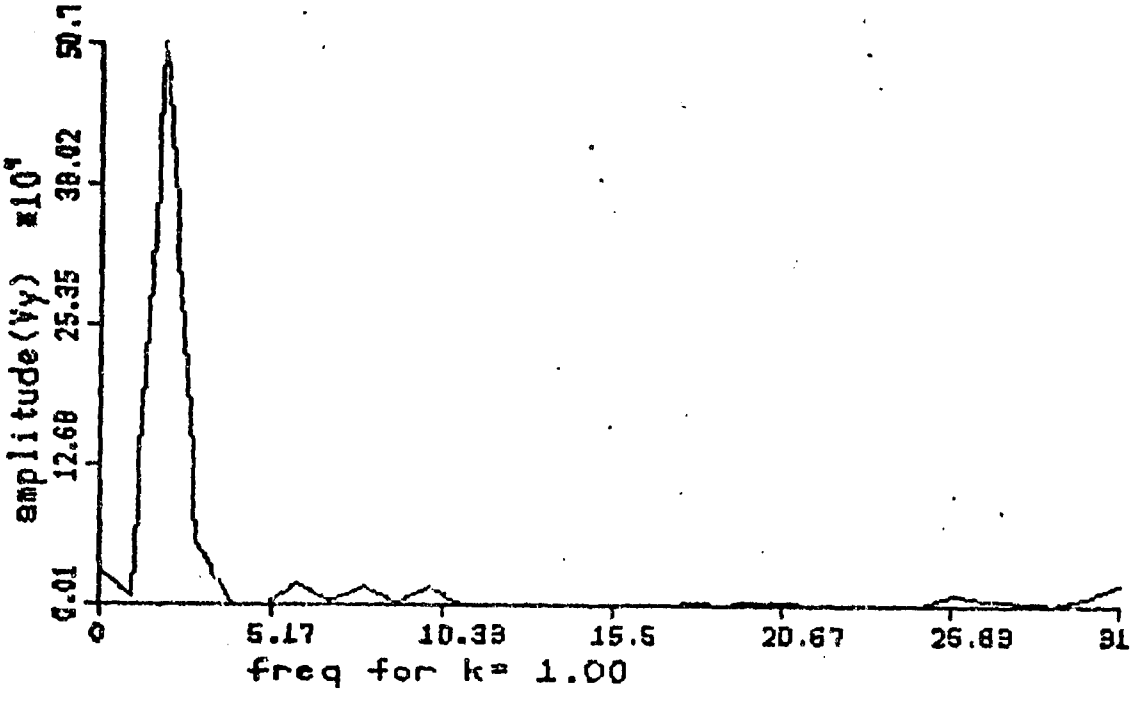
Results on synthetic images demonstrate that the technique is effective both in pinpointing sensor velocity and in separation of relative sensor-object velocity in cases where moving targets occurred in the images (See Figures 2, 3).

Synthetic images were produced with intensity ranges of [0-50] and [0-255] (Figures 2,4). The procedure was applied to several set of images of both categories. These simulations indicate that results depend on the pixel intensity range, as well as on such parameters as SNR, image resolution, and frame-to-frame grey scale changes. There is a simple reason for this dependency. The smaller intensity differences between pixels entering and leaving the scene create smaller disturbances in the sinusoid generated in the first stage of the procedure. Evidently, the procedure may benefit from image preprocessing by some function, such as log, which reduces the intensity range (and variance).

The simulations were run with various numbers of frames

of several sizes. (16x16, 32x32, 64x64 with 8, 16, 32 frames). As might be expected, among the tested variations, larger frame sizes and fewer frames produced the best results. Again, less disturbance was introduced into the sinusoid (generated in step one of the motion detection transform) by the smaller ratio of new pixels to old pixels within the transform window. (we note that perspective distortion in real images complicates this issue)

Figure 2. Plots: MDT results for synthetic image sequence. Random background intensities [0,50]. Block size (64x64) in pixels. Sensor velocity (in pixels per frame) (Vx,Vy) = (-2,3). No object motion occurred in the sequence. Upper plot for Vy, lower plot for Vx. Note: since kx=ky=1, the frequency showing the peak directly gives the velocity.



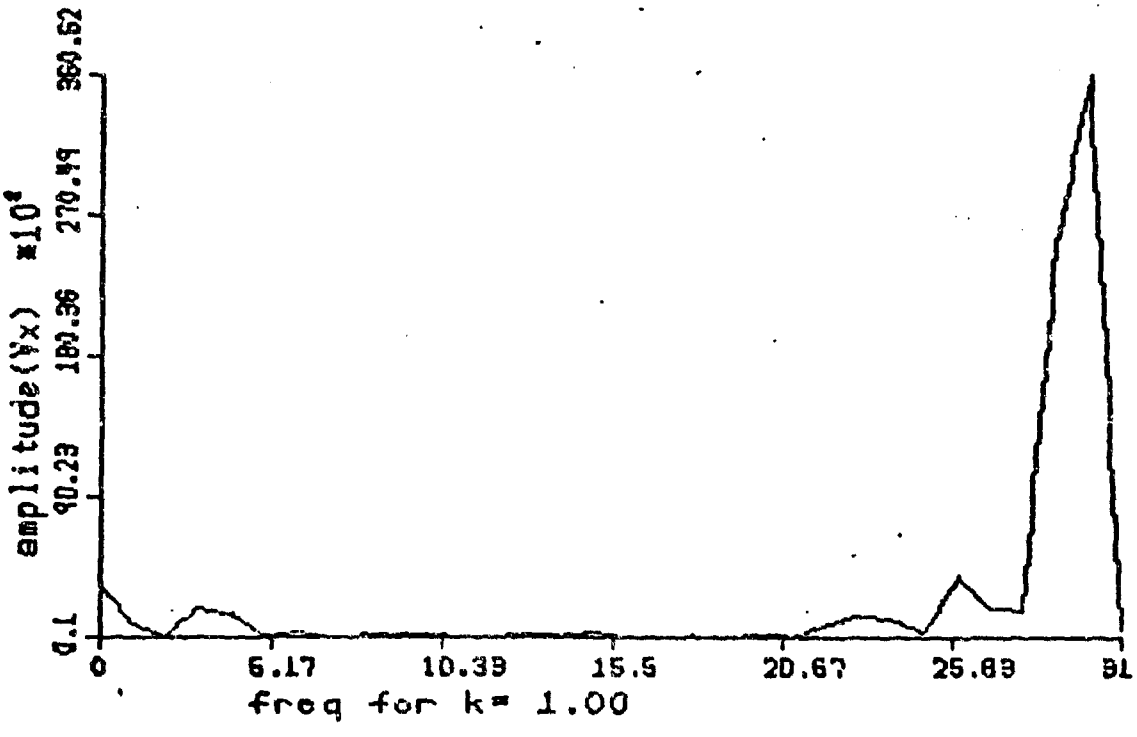
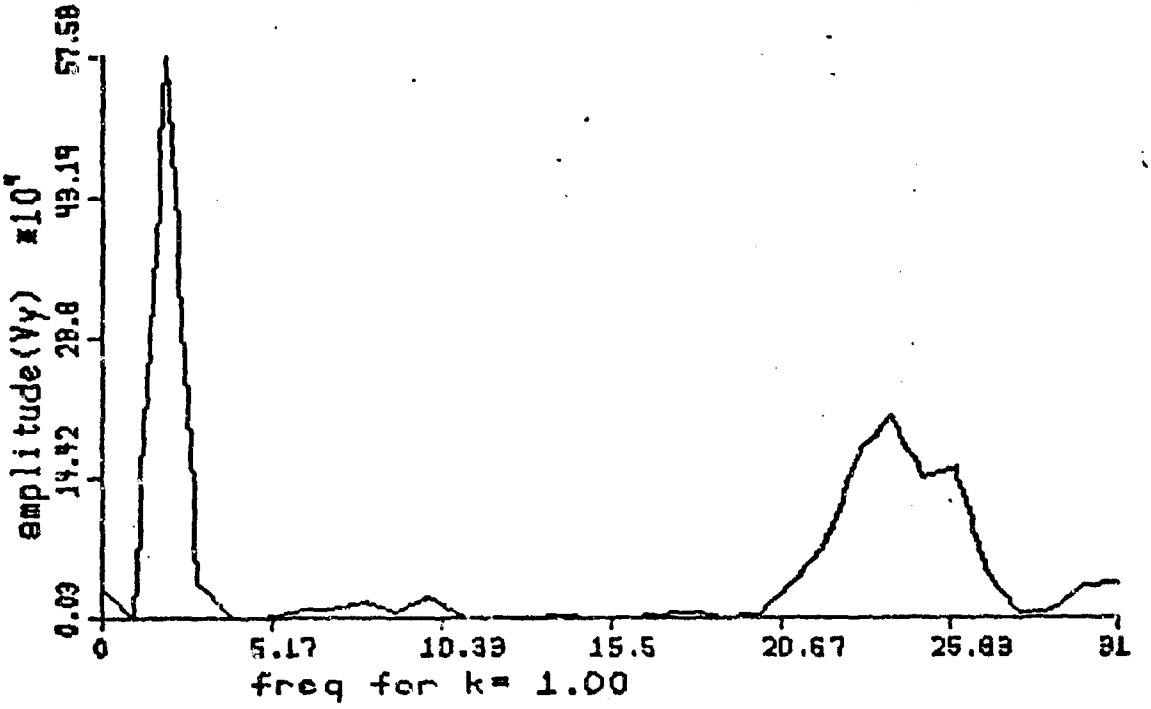


Figure 3. Plots: MDT results for synthetic image sequence. Random background intensities [0,50]. Block size (64x64) in pixels. Sensor velocity (in pixels per frame) (Vx,Vy) = (-2,3). Object intensity 100.
Object velocity (7,8). Object size (6x6).
Upper plot for Vy, lower plot for Vx.



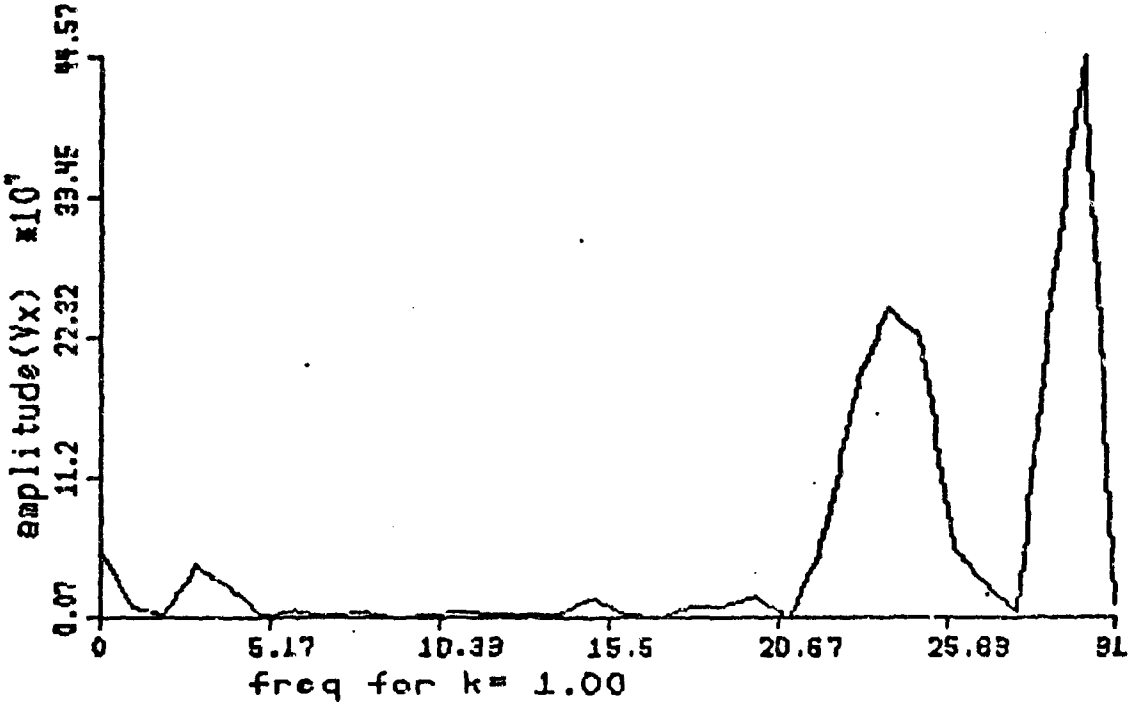
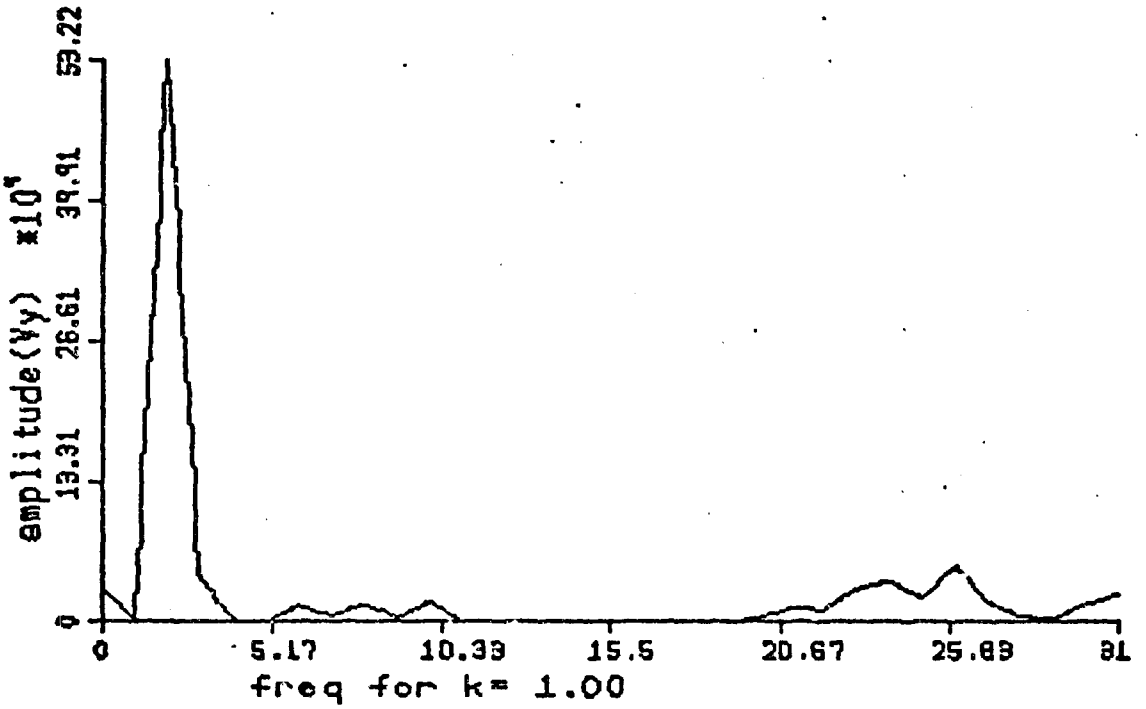
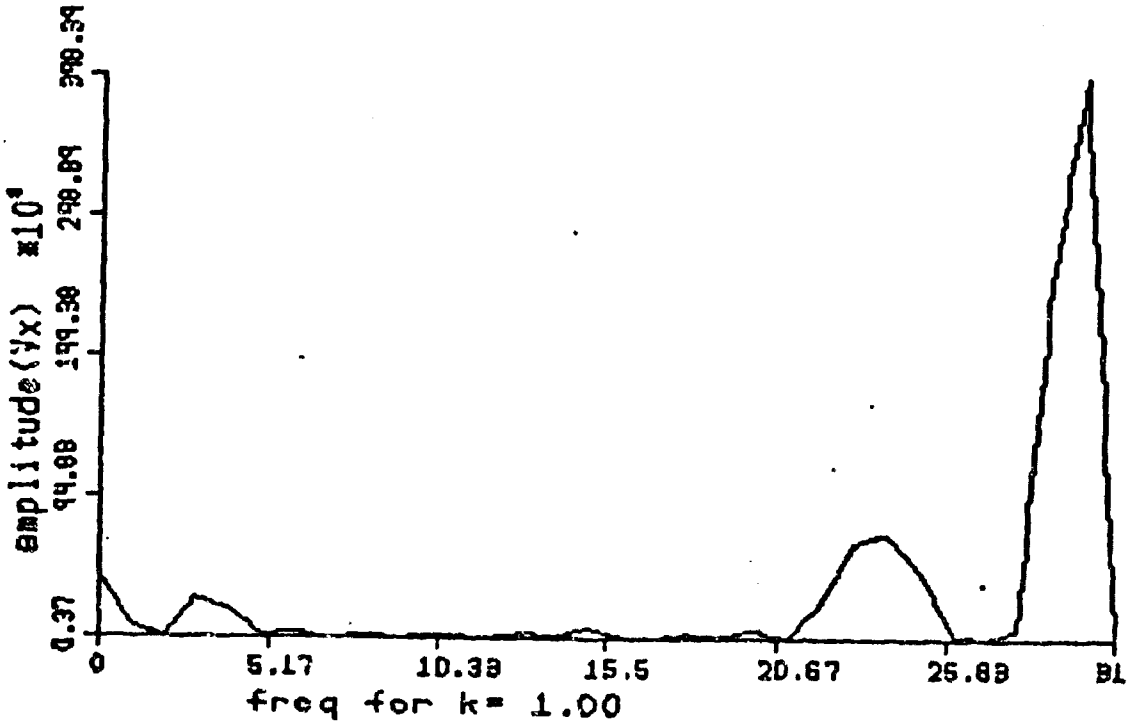


Figure 4. Plots: MDT results for synthetic image sequence. Random background intensities [0,50]. Block size (64x64) Sensor velocity (in pixels per frame) (Vx,Vy) = (-2,3). Object intensity 50. Object velocity (7,8). Object size 6x6. Note that peak due to object not much above the extraneous peaks. Upper plot for Vy, lower plot for Vx.





4.0 Use of the Motion Transform in Cases of General Motion.

This section enumerates some of the inherent difficulties in applying the motion detection transform to analysis of general motion and presents our solution to these problems. Our solution culminates in computation of a sort of displacement field for the image sequence.

Mathematically speaking, the direct application of the motion detection transform is valid only when all motion is restricted to translation in planes parallel to the image plane. With motion unrestricted, the sensor can rotate as well as translate in depth. Furthermore, the transform will detect different apparent velocities for texture points at different depths (perspective or motion parallax effects). However, we can, over small regions, approximate effects of general sensor motion by the effects due to a translating sensor. In addition, we choose to ignore the perspective effects. We partition frames into small blocks. We assume that perspective effects are minimal over a single block, and can be dealt with more easily at the next (higher) level of processing. Thus spatially dividing the images into blocks simultaneously defers the perspective distortion problems and permits characterization of a wide variety of sensor motions.

4.1 Problems with FLIR Data.

Additional difficulties stem from the real-world data we employ in our testing. We are using FLIR (forward-looking infra red) data to test our approach. The data demonstrates many corruptions. For example, it suffers from severe salt and pepper noise. In addition, the images were preprocessed so that the intensities span the complete range [0-255] in each frame. This introduces spurious frame-to-frame grey-scale changes.

We enhanced transform performance by preprocessing our images. To reduce salt-and-pepper noise, we applied a non-linear noise-cleaning algorithm (from P1) to each frame of the sequence. The rule used is: replace any pixel some threshold above the average of its 4- connected neighbors by that average. Next, we ameliorated the grey scale changes due to self- normalization of the images via an histogram correction technique (G1). We arbitrarily chose one image of the sequence as "ideal." We computed the cumulative frequency distribution (CFD) of this ideal image, and forced the CFD's of all the other images in the sequence to approximate this ideal CFD. (See Figures 5 a., b.)

Also generating significant problems, we limited ourselves to 8 or 16 frames of the sequence as a temporal window. We noted that the (remaining) salt-and-pepper noise in

the FLIR imagery looks like small objects moving with high velocities to the motion detection transform. It is obvious from the mathematical formulation of the transform that the higher the weighting factor, (kx and ky) the more sensitive the transform becomes to noise. Therefore, we found it expedient to limit our weighting factors to values of one or two. This left us with very low velocity resolution; since the precision with which a velocity can be located is higher for larger weighting factors. We needed higher resolution. We located the peak frequency, as discussed in section 3.0, and then interpolated (computed a first moment about the peak with the frequencies on either side of the actual peak) to enhance velocity resolution.

Another problem still remained: if the terrain were virtually homogeneous, the dc component (zero velocity) could mask the true velocity-related frequency. We instituted an ad hoc rule stating that no zero-velocity peak would be accepted unless it exceeded the next highest peak by more than some adjustable percentage.

Figure 5 a. Cumulative Frequency Distribution (CFD) for "ideal" image.

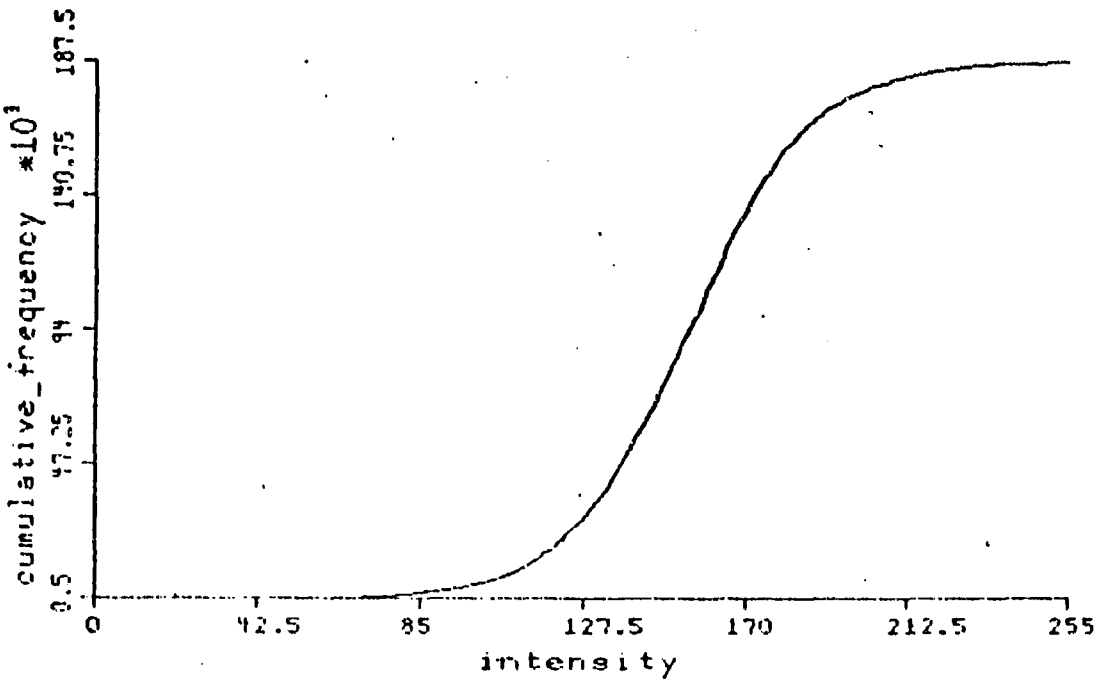
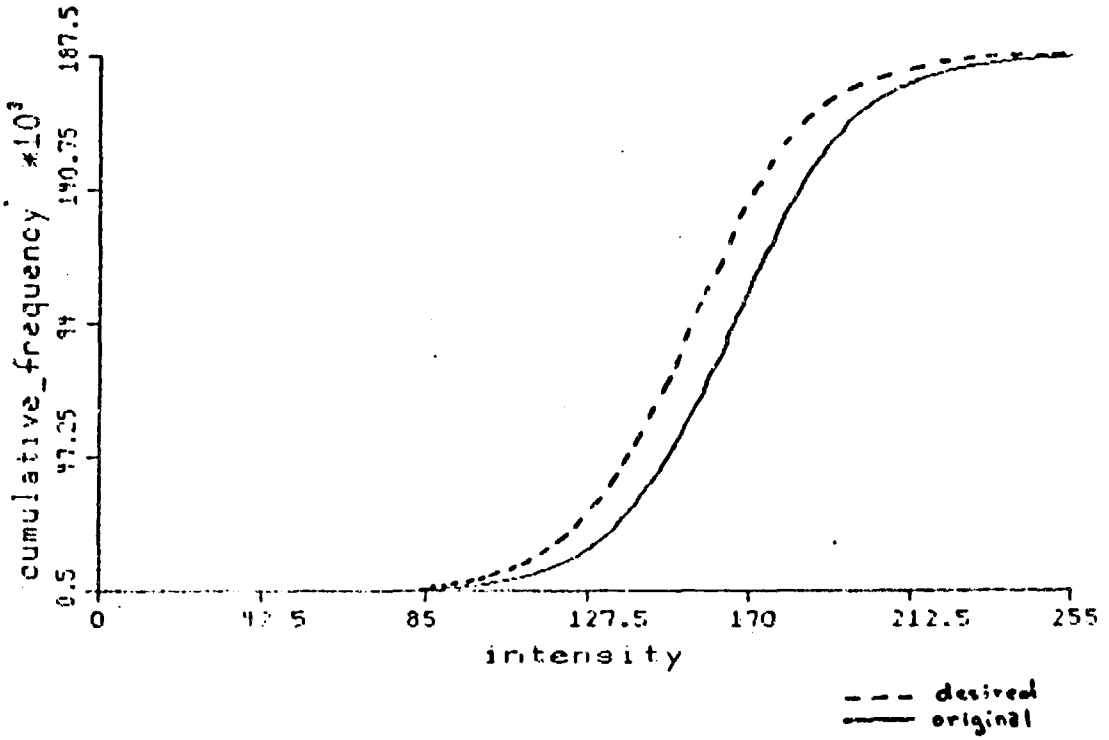


Figure 5 b. Original CFD and CFD obtained through histogram correction for frame number n of sequence.



5.0 Use of the MDT for Frame-to-Frame Registration.

This section overviews the MTI algorithm (H2) then describes our general approach of spatial registration which is based on MTI.

Given the conditions and goals stated above, we adapt the basic MTI algorithm for out purposes. The MTI algorithm, as defined by (H2) is as follows:

1) partition an image into blocks.

2) compute a displacement vector (frame[i] to frame[i+1]) for each block of the image by maximizing a grey level cross correlation function (for each block) between two frames of the sequence.

3) form motion model based on least squares fitting a quadratic equation (in XD and ZD) to the displacement vectors, with the Chi Square criterion employed to cull out erroneous displacement vectors, in a potentially multi-pass process.

4) use the motion model from 3) and frame[i] of the sequence to predict frame[i+j], P(frame[i+j]).

5) compute $|\text{frame}[i+j] - P(\text{frame}[i+j])|$ (as a pixel operation)

See (H1) for complete details.

Our adaptation of the MTI algorithm is as follows:

1) partition the image into blocks.

2) compute a displacement vector for each block of the image by means of the MDT.

3) apply a multi-pass non-linear noise-cleaning

algorithm separately to the x and y components of the displacement vectors to eliminate vectors due to noise and those due to objects moving within the scene.

4) use the set of displacement vectors from 3) to compute the coefficients of a quadratic (in XD and ZD) motion model.

5) MTI step 4).

6) MTI step 5).

We exploit the motion detection transform to compute our displacement field. In the manner of (H1), we partition frames into blocks and compute a displacement vector for each block. However, the motion detection transform, rather than cross correlation, produces these vectors.

5.1 Results on FLIR Data.

This section describes our success in application of this technique to FLIR imagery, and compares the results with simple frame differencing.

Figure 6 a. shows the initial set of displacement vectors computed in step two of our procedure. Figure 6 b. shows the same set of vectors after several passes of the noise cleaning process. Our system applies the motion model derived from the displacement vectors to one frame of the sequence and predicts some subsequent frame, frame n.

Figure 7 a. shows the absolute frame difference image for actual and predicted frames. Figure 7 b. gives the histogram of this image. Figures 8 a. and 8 b. represent analogous data for the frame difference image for (actual) frame one versus (actual) frame n. As the target in these images occupies a small percentage of the total image, the spread of the histogram is a good indicator of how well the frames have been registered, and thus, how well the sensor motion has been characterized by the motion model. The histogram for the predicted case compares favorably with the second, simple difference, case. As the Figure 7 a. demonstrates, a simple thresholding operation would virtually eliminate false targets from the difference picture generated by our system. This is certainly not so for the simple difference picture, Figure 8 a.

Figure 6 a. Displacement Vectors. (Unprocessed) superimposed on framel of the sequence.

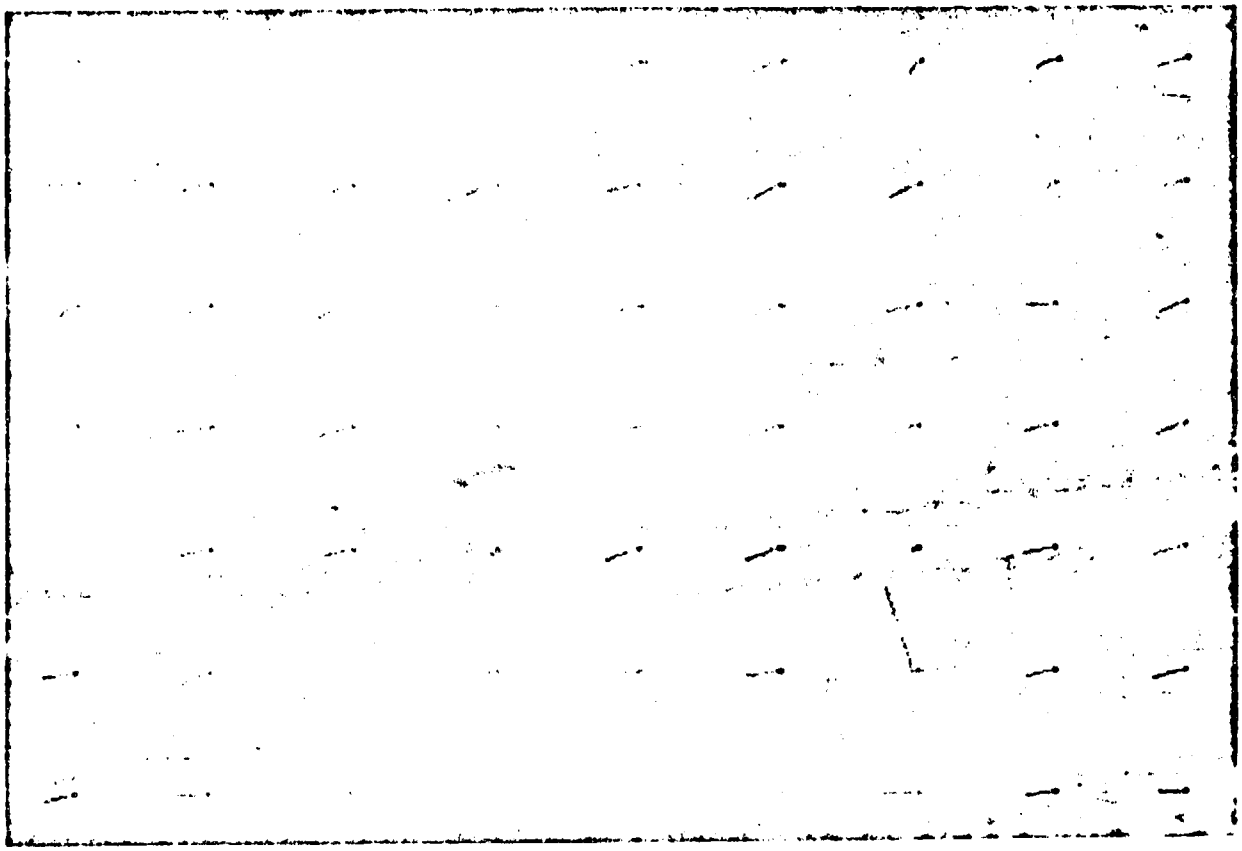


Figure 6 b. Displacement Vectors after four passes of a nonlinear noise cleaning algorithm.

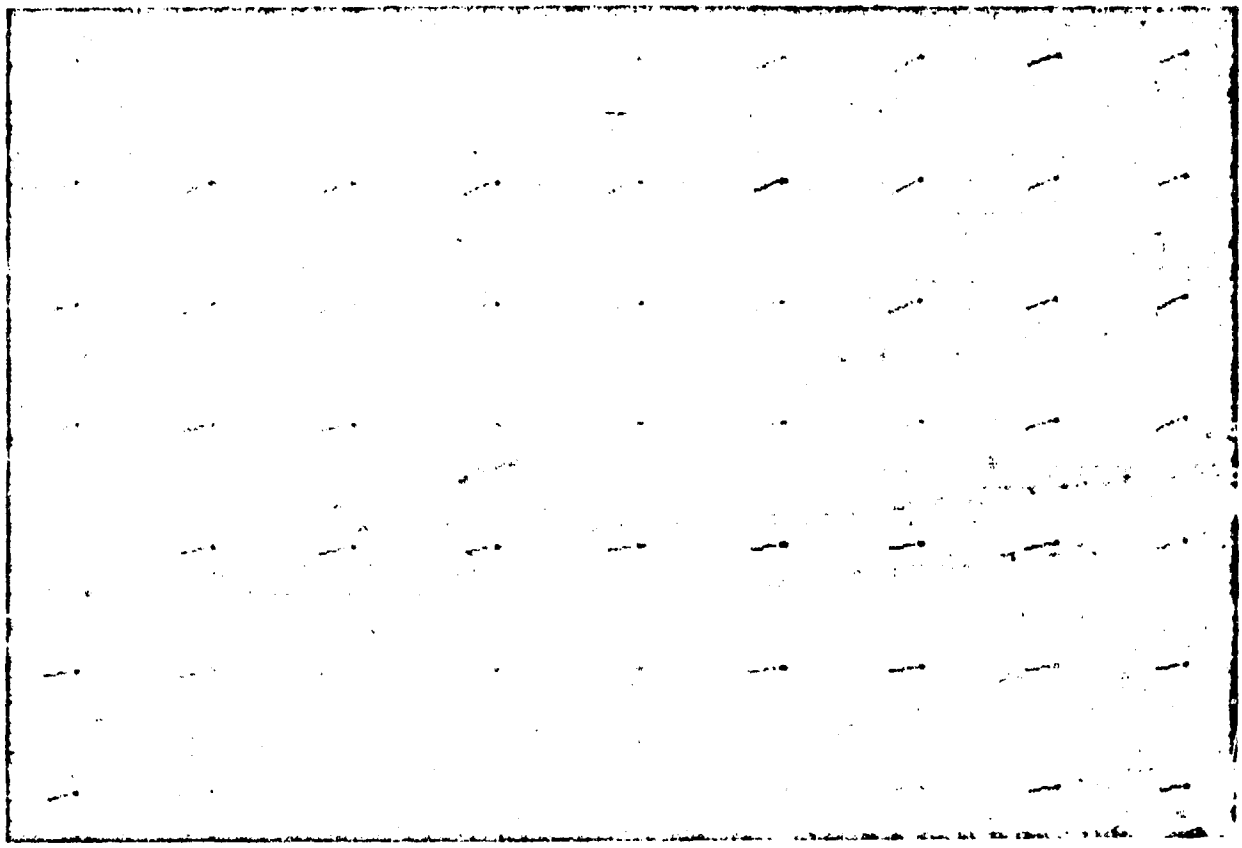


Figure 7 a. Frame difference for motion model:
d[i,j] = abs(frame#1[i,j] - PREDICTED$frame#n[i,j]†)

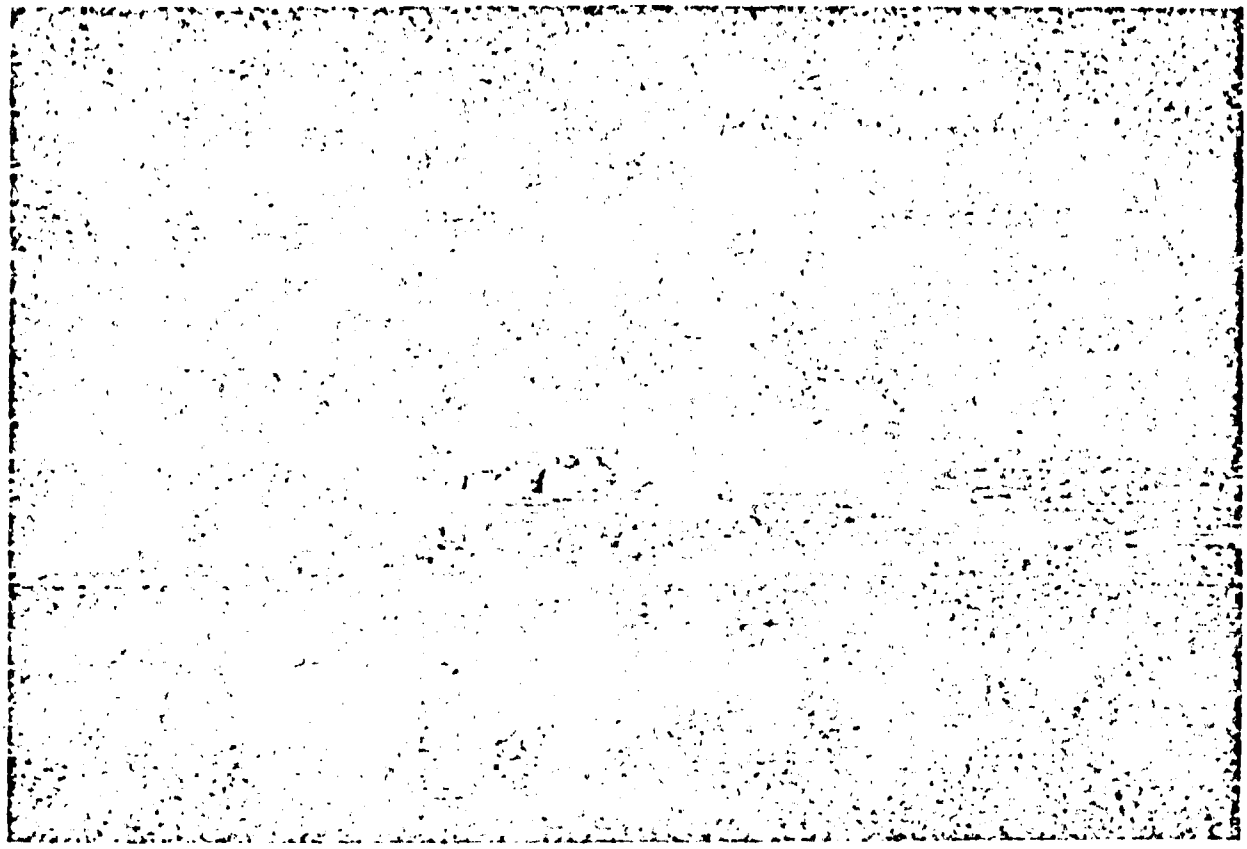


Figure 7 b. Histogram for motion model frame difference picture. (see figure 7a.)

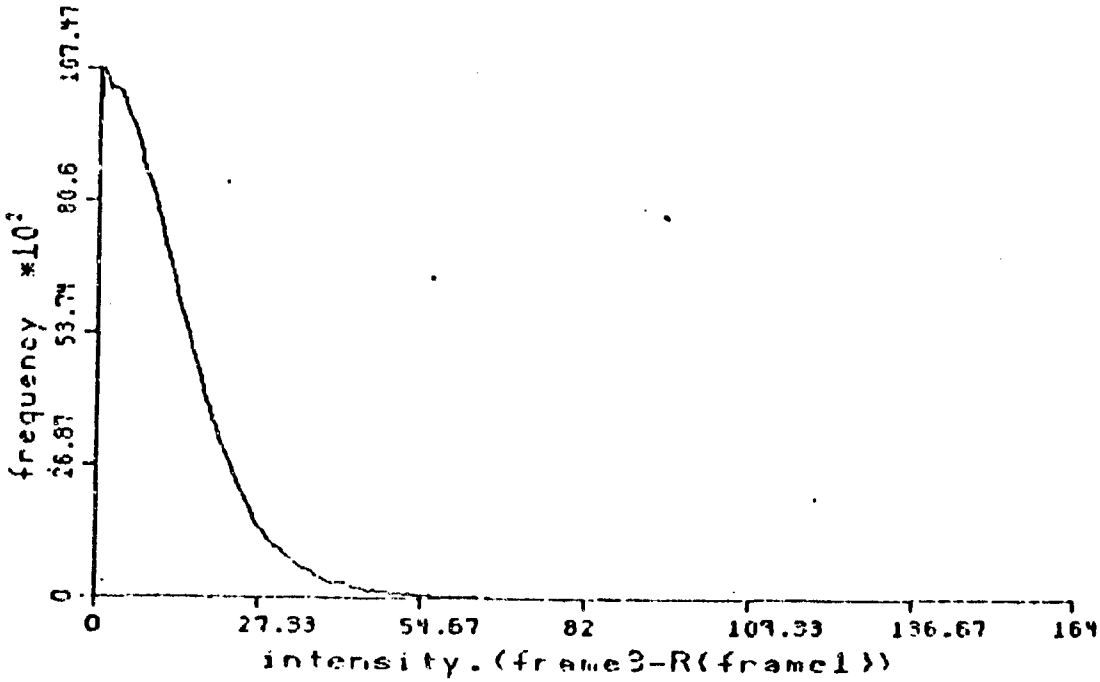


Figure 8 a. Simple difference picture:
d[i,j] = abs(frame#1[i,j] - frame#n[i,j])

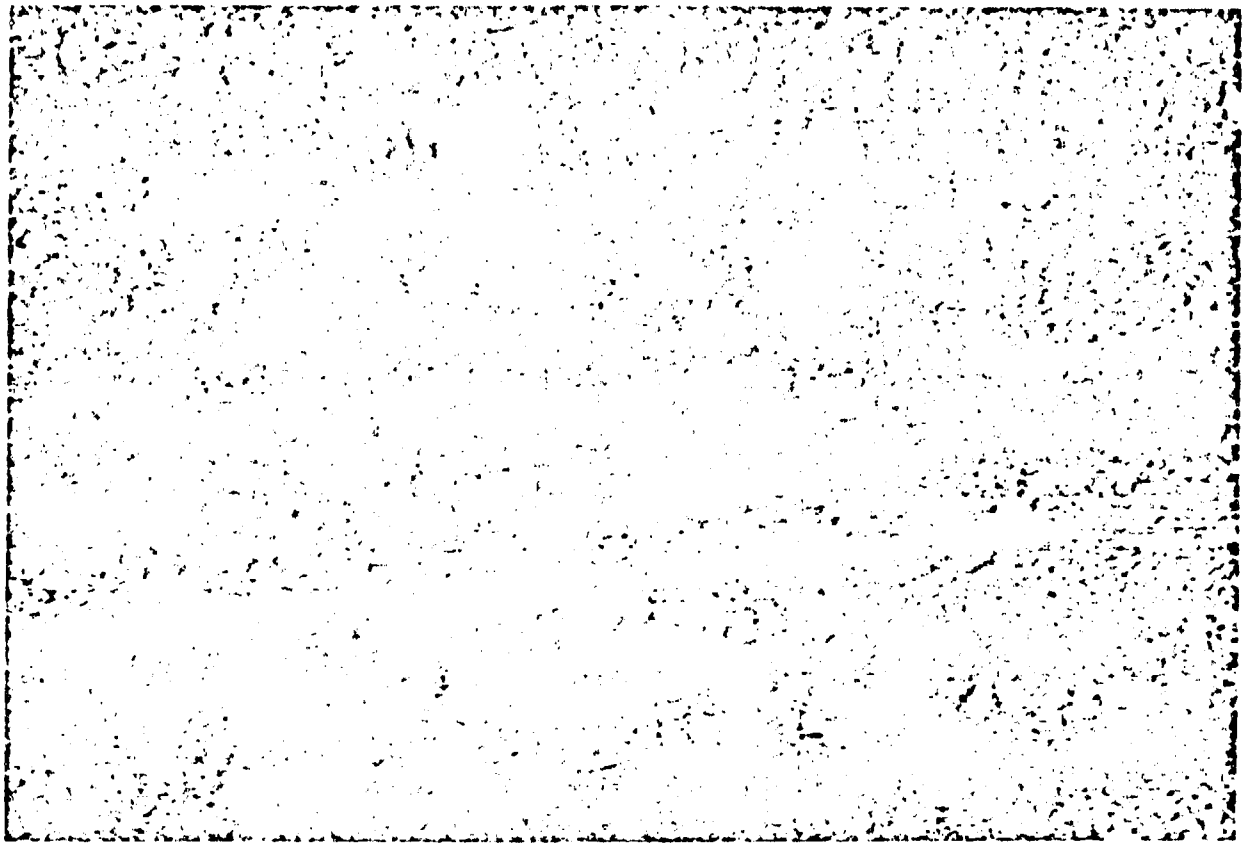
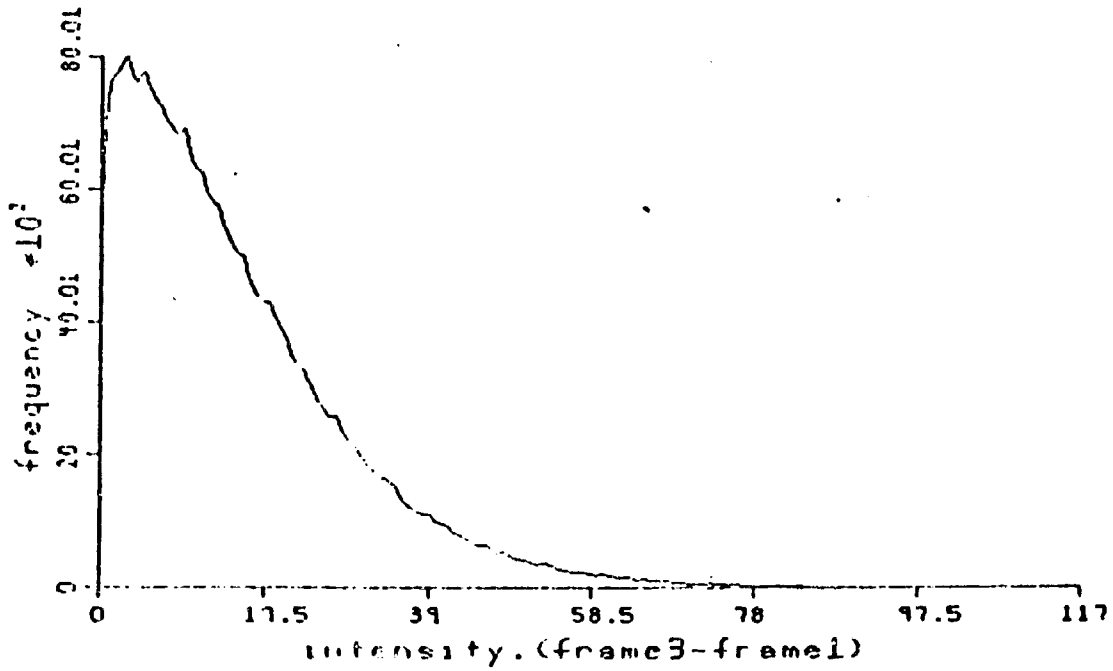


Figure 8 b. Histogram of simple difference picture.
(see figure 8 a.)



6.0 Ongoing Work.

The following text discusses extraction of sensor motion parameters in general and describes our adaptation of Nagel's technique for calculating sensor motion parameters (N1, N2, N3) in particular.

Six parameters completely define the motion of a moving sensor between frames of an dynamic sequence (D1,N1,A1). These may be stated as three angles of rotation and three components (X,Y,Z) of a translation vector. Application of the translation vector moves the lens center in frame 2 into coincidence with its frame 1 position. Borrowing notation from computer graphics/aerial photogrammetry (D1,A1), the three angles express three sequentially applied rotations necessary to align each axis of the "new" camera coordinate system to its position in the previous frame (D1, A1). Other specifications of the information in these parameters exist (eg P2,P3).

It is a well known fact that the translation vector can be recovered only to within a scale factor, given a monocular sensor (P2, N1, N2, N3, D1). (In fact, no absolute distances can be recovered, only relative distances). The rotation angles can be unambiguously recovered[1], however,

[1]Nagel's technique requires the data satisfy conditions given in N3.

as long as the relationship XP:f:ZP is maintained (ie, in a digital environment, we need the focal length of the optical system in device coordinate system units, as well as the positional and temporal intensity information.) Without the focal length of the system, we essentially have the parallel projection case, from which translation in depth cannot be recovered and which renders indistinguishable "an object rotating by some angle, a, and its mirror image rotating by -a." (U1). In any case, perspective (central) projection more closely approximates the photographic process.

Generally speaking, solving for sensor motion parameters implies solution of a system of nonlinear equations in five unknowns, or equivalently, a search through a five-dimensional space for the correct set of parameters (T1, N1). Since the system is nonlinear, there is no guarantee of a unique solution (T1). In addition, two parameters (the translation) occupy infinite axes in this five-dimensional space. Nagel has separated the calculation of the translation from that of the rotation by means of algebraic manipulation (Figure 9). He uses a minimization approach to solve for the rotation parameters. Notice that his translation vector calculations are linear and that the search space is reduced to a three-dimensional search space.

We use a priori knowledge to further reduce the search space. We know our imagery was generated by an IR system

side-mounted on an RPV. Significant rotation about the focal axis is, therefore, deemed unlikely. Also, given a high enough frame rate, the rotation angles between frames should be small. Therefore, we need only search a portion of the remaining search space for rotation angles.

We compute the sum of the absolute value of Nagel's equation (2 Figure 9) over the entire image for each point in our reduced search space. The minimum over the entire 2D search space is chosen as the correct set of rotation angles. (Of course, we use the SSDA concepts of B1 to speed up calculations.) Then Nagel's equations (3 Figure 9) and (1 Figure 9) deliver the translation vector (to within a scale factor). The technique will be applied to displacement fields extracted from our IR imagery as soon as camera data (ie, focal length in device units) becomes available.

Figure 9: Nagel's Equations *
(for extraction of sensor motion parameters)

The world coordinate system is static. The camera coordinate system travels with the camera, and thus, changes with respect to the world system from frame to frame. We want to know how the sensor moves between frames. This motion is called relative sensor motion. In formulating the solution to the relative sensor motion problem, common practice (N1,N2,N3,A2,D1) is as follows: in frame one establish the world coordinate system as coinciding with the (current) camera coordinate system, ie. (Xi,Yi,Zi) = (XCi1,YCi1,ZCi1) for all i. Then

Am = (Xm,Ym,Zm) = (XCm1,YCm1,ZCm1) gives the mth point in world coordinates

Cm2 = (XCm2,YCm2,ZCm2) gives camera coordinates for the point in the 2nd time frame.

= (Am + T)D

where T is the translation (vector) between world system origin and camera system origin in frame 2. Cm2' = Am + T
and D is a 3 by 3 rotation matrix which aligns the (XC',YC',ZC') axes with the (X,Y,Z) axes.

Bmn = (XPmn,f,ZPmn) picture plane coordinates

so Cm2 = (sm2)Bm2 where (sm2)f = YCm2.

---

1) T = (sm2)Cm2D' - (sm1)Cm1

2) (Cm1 x Cm2D') * ((C11 x C12D') x (C21 x C22D')) = 0

3) (C21 x C22D') * ((sm2)Cm2D' - (sm1)Cm1)

Note: First, equation (2) solved for the parameters of D then equation (3) used to solve for sm2 in terms of sm1 finally, T is derived from (1).
D represents an 3 by 3 matrix, all other upper case entities represent 3D vectors. Lower case items are scalers. Scaler multiplication is represented by concatenation while "*" and "x" indicate vector dot and cross product, respectively.
The single quote, "'", designates the transpose operator.

* from (N1) and (N3)

REFERENCES

A1. American Society of Photogrammetry, Manual of Photogrammetry, 4th ed.

B1. Daniel I. Barnea and Harvey F. Silverman, "A Class of Algorithms for Fast Digital Image Registration," IEEE trans. on Computers C-21, No 2, pp. 179-186, Feb 1972.

D1. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, New York, NY: John Wiley and Sons, 1973.

G1. R. C. Gonzalez and P. Wintz, Digital Image Processing, Reading, MA: Addison-Wesley Publishing Company, 1977.

H1. Ernest L. Hall, Computer Image Processing and Recognition, New York, NY: Academic Press, Inc., 1979.

H2. Richard Holben, "Moving Target Identification (MTI) Algorithm for Passive Sensors," SPIE Vol 219, pp. 165-172, 1980.

L1. J. M. Lloyd, Thermal Imaging Systems, New York, NY: Plenum Press, 1975.

N1. H. -H. Nagel, "Representation of Moving Rigid Objects Based on Visual Observations," Computer, Vol 14, No 8, PP. 29-39., Aug 1981.

N2. ibid., "On the Derivation of 3D Rigid Point Configurations from Image Sequences," PRIP 81, pp. 103-108, Aug 1981.

N3. H. -H. Nagel and B. Neumann, "On 3D Reconstruction from Two Perspective Views," Proc 7IJCAI, Vol 2, pp. 661-663, Aug 1981.

P1. W. K. Pratt, Digital Image Processing, New York, NY: Wiley, 1978.

P2. K. Prazdny, "Determining the Instantaneous Direction of Motion From Optical Flow Generated by a Curvilinearly Moving Observer," Computer Vision Lab, C.S. Center, Univ Maryland.

P3. ibid., "Egomotion and Relative Depth Map from Optic Flow," Biological Cybernetics, Vol 102, 1980.

R1. Sarah A. Rajala, Alfy M. Riddle, and Wesley E. Synder, "Application of the One-dimensional Fourier Transform for Tracking Moving Objects in Noisy Environments," Raleigh, NC: NCSU Computer Science Technical Report, 1982.

REFERENCES (continued)

R2. A. N. Riddle and S. A. Rajala, "A Method for Simultaneous Velocity Detection and Object Identification Using the Fourier Transformation," 15th Asilomar Conference on Circuits Systems and Computers, Nov 1981.

R3. J. W. Roach and J. K. Aggarwql, "Determining the Movement of Objects from a Sequence of Images," PAMI-2 No 6, pp. 554-562, Nov 1980.

T4. R. Y. Tsai and T. S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," Proc PRIP 82, pp. 112-118, June 1982.

U1. S. Ullman, The Interpretation of Visual Motion, Cambridge, MA: MIT Press, 1979.