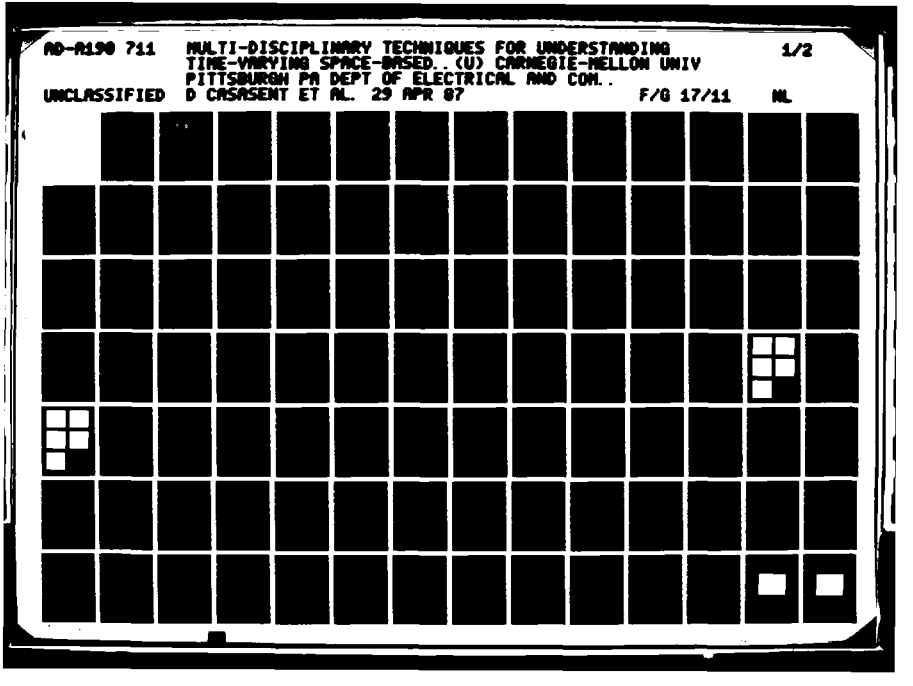
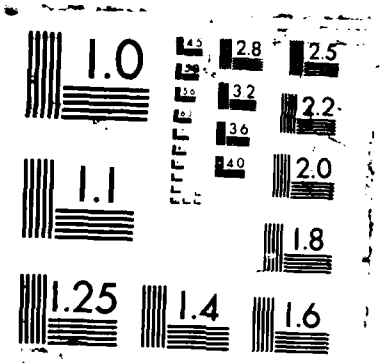

AD-A190 711
MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL AND COM..
1/2
UNCLASSIFIED
D CASASENT ET AL. 29 APR 87
F/G 17/11
NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE

1. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release: distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 87-1756
6a. NAME OF PERFORMING ORGANIZATION: Carnegie Mellon University	6b. OFFICE SYMBOL (If applicable):
7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research	
6c. ADDRESS (City, State, and ZIP Code): Department of Electrical & Computer Engrg. Pittsburgh, PA 15213	7b. ADDRESS (City, State, and ZIP Code): Directorate of Mathematical & Information Sciences, Bolling AFB, DC 20332-6448 BLDG 410
8a. NAME OF FUNDING / SPONSORING ORGANIZATION: AFOSR	8b. OFFICE SYMBOL (If applicable): NE
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-84-0239	
8c. ADDRESS (City, State, and ZIP Code): BLDG 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102 F	2305	B4	

11. TITLE (Include Security Classification): Multi-Disciplinary Techniques for Understanding Time-Varying Space-Based Imagery

12. PERSONAL AUTHOR(S): David Casasent, Arthur Sanderson, Takeo Kanade and B.V.K. Vijaya Kumar

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
UNCLASSIFIED	FROM 5/85 TO 3/87	87/04/29	147

16. SUPPLEMENTARY NOTATION: NONE

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	NONE

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

This project is a multidisciplinary effort between three departments and principal investigators. It combined pattern recognition, image understanding and artificial intelligence techniques for space-based image processing. A special feature of this effort is the attempt to use both optical and digital processing methods. Subpixel target detection and tracking algorithms are analyzed and conclusions are presented regarding their suitability for this application. We also present an adaptive subpixel delay estimation method using Group-Delay Functions. Image understanding techniques for 3D (three dimensional) scene interpretation are also discussed.

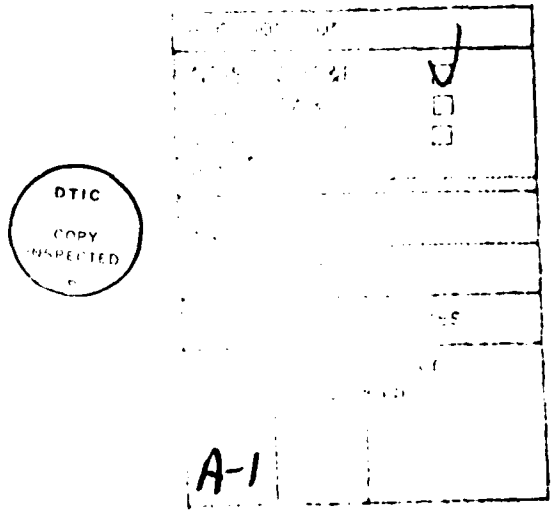
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Dr Giles	202-767 4984	NL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Table of Contents



List of Figures

List of Tables

ABSTRACT

This project is a multidisciplinary effort between three departments and principal investigators. It combined pattern recognition, image understanding and artificial intelligence techniques for space-based image processing. A special feature of this effort is the attempt to use both optical an digital processing methods. Subpixel target detection and tracking algorithms are analyzed and conclusions are presented regarding their suitability for this application. We also present an adaptive subpixel delay estimation method using Group-Delay Functions. Image understanding techniques for 3D scene interpretation are also discussed.

KEY WORDS

3D scene interpretation, Artificial Intelligence, Hybrid Processor, Image Understanding, Space-based Imagery, Optical Processing, Subpixel Delay Estimation, Group-Delay Function, Time-Change Imagery, Tracking.

Chapter 1

Introduction

1.1. Overview

This project is a multidisciplinary effort intended to combine methodologies for image analysis and interpretation, and evaluate the application of this integrated approach to problems of space-based imagery. The project has brought together research teams from within the Department of Electrical and Computer Engineering, Computer Science, Robotics, and Biomedical Engineering of CMU.

We have chosen *time-varying space-based imagery* as the applications domain in which to evaluate our integrated approach. The two aspects of this domain are described below:

- *Space-based imagery* involves large amounts of information and incorporates both structural and textural properties of a scene. Efficient detection and representation of information in scene are essential not only to interpretation but also to the storage and transmission of information. Scenes are predominantly two-dimensional although light and shadows affect imaging of both structures and texture, and interpretation of scenes at increasingly high optical resolution will require three-dimensional models.

- Interpretation of *time-varying data* is a primary goal of space-based image analysis and adds an additional dimension of complexity to the problem. We have chosen to look at three time-frame scenarios which

require somewhat different analysis tools. High speed tracking is viewed as primarily a feature extraction problem and has been approached using optical methods. Medium and long-term time change detection must be based on a more abstract description of the scene and methods of representation and model-based interpretation must be brought to bear.

Within the context of the applications domain, we have addressed the following methodological research issues:

- Optical feature extraction and detection
- Structural and textural representation and matching
- Model-based image interpretation
- Hybrid optical/digital computer architectures

These issues are fundamental to implementation and performance of analysis tools which could embed the inherently fast and parallel preprocessing power of optical techniques into a system which develops and tests hypotheses about scene representations and scene models.

In Chapter 1 of this report, we provide a more detailed overview of the conceptual framework of our proposed hybrid optical/digital system, define the space-based image processing problem, and discuss the importance of this work to Air Force technology and to related Air Force programs. Section 1.5 provides a summary of our research up to this year. Section 1.6 provides a summary of our current year of research, with details in Chapters 2-5.

1.2. Conceptual Framework for Hybrid Optical/Digital Image Processing

In Figure 1.1. we show the general structure for our proposed hybrid optical/digital system using multiple methodologies for understanding space-based images. As shown in Figure 1.1, input images are preprocessed and then fed to parallel optical and digital channels in which multiple features are extracted. A parallel image modeling system is also shown which extracts structural descriptions of the image. These data plus image registration and target detection information obtained from an optical correlator channel are then used by an AI/IU system to modify the parallel input processing channels, to assemble and interpret a time-history track file on objects of interest in the image and to provide the necessary textural and graphic output reports.

1.3. Problem Definition

Advanced space-based sensor systems will provide us with high-resolution real-time multisensor data acquisition in the near future. This will totally pollute present processors unless we address how to intelligently and timely process and handle the projected data rates. NASA and others have already verified that the United States is capable of collecting more data than we can intelligently process (less than 1% of all NASA data has even been looked at).

The key issue in Space-Based Image Understanding (SBIU) is not to transmit every frame of data (with 5000 x 5000 sensor elements in three bands with ten bits of data per pixel, and a 30 frame/sec rate, this is a data collection

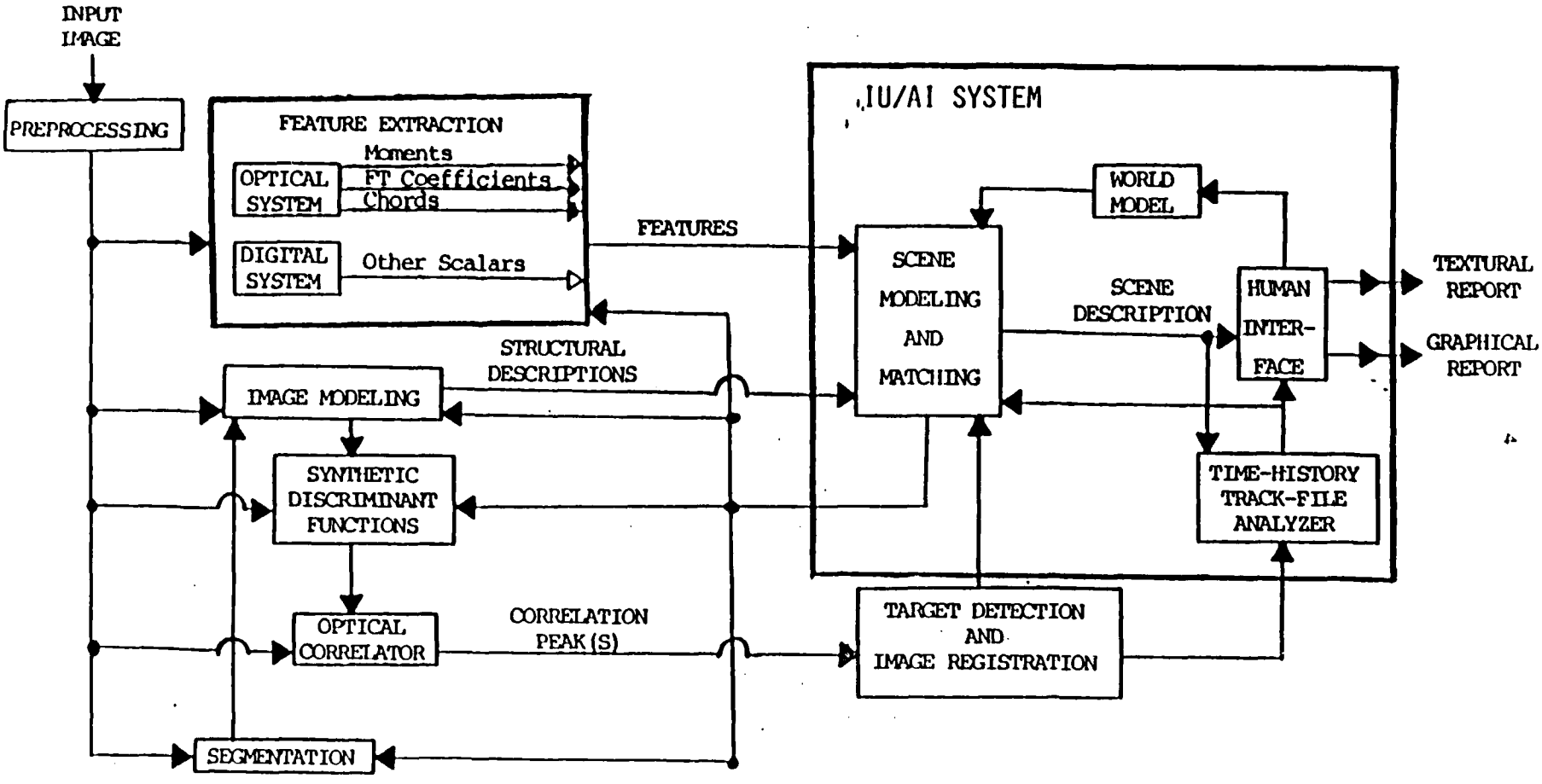


Figure 1-1 : Hybrid optical/digital multi-disciplinary (pattern recognition/image understanding and artificial intelligence) processor.

rate of over 10^{10} bits/sec). No existing technology can accommodate such a high data collection rate. Therefore, attention should be given to the algorithms required to achieve this. But first, here are several facts about SBIU problems:

1. In space-based image acquisition, we are monitoring certain areas and regions for diverse well-defined missions. We are only concerned with changes and do not need to know that nothing new has occurred in the image being looked at. When we transmit only the associted *change information*, we achieve a quite significant *bandwidth reduction*. Thus, we should process the data from space-based sensors on-board the platforms, determine image changes on-line, interpret the results and transmit only textural and graphic output reports.

2. We know rather well where the satellite is and where it is looking and we know that the scene being imaged correlates with the prior image frame or with our stored reference. The problem is thus different from the often discussed unbounded and unsupervised target recognition problem. We can and must utilize this *a priori information* that the frame we are investigating correlates with a previous one in our processing algorithms.

3. To provide better *image registration* accuracy and to facilitate pointing of secondary sensors at given areas of interest, it is often necessary to *locate key landmarks* in the image. This is also useful in determining *geometrical corrections* needed.

4. It is also useful and necessary to register two successive image frames for *inter-frame integration* to decrease the variance of the noise and to improve the image quality. This is essential to accommodate platform variations with time and background drift. Often, *subpixel image registration* is necessary.

5. It is obviously essential to *subtract successive frames* since this provides the necessary change detection or time-varying target data.

6. However, in most cases, the image registration in (4) is subpixel and thus before performing (5), we must *interpolate* the images.

7. Once *time-history track files* of candidate objects of interest in the field-of-view of the sensors have been obtained, a multitude of discrimination analysis techniques, AI, IU, pattern recognition and human perception algorithms are necessary to classify, understand and interpret the time-change activity noted.

8. In advanced sensor systems, 3D information on the scenes will be available from stereo satellites or other techniques. In such cases, we can fully capitalize on the available image information only by the use of advanced *3D scene modeling and interpretation*. The key point is the extraction of scene information (3D) from time-histories of 2D images.

9. To detect and describe detailed changes in the 3D structure of scenes, it is useful to first *generate 3D scene descriptions from the 2D images*, and then to compare the descriptions for changes. Conventional 2D change detection approaches are not as useful for high resolution images of complex scenes since they do not take into account factors such as different viewpoints and different lighting conditions for the different images of the scene. In order to detect changes over successive images of a given scene obtained over time, it is useful to maintain a 3D model of the scene and automatically update the model as changes occur. This requires the ability to match the model with each new view of the scene. *Matching in 3D* is more desirable than matching in 2D since the 3D information is represented in a manner that is independent of viewpoint and lighting conditions.

10. The *3D scene model* is a useful central component for many aspects of the change detection task. Not only is it useful for determining whether changes have occurred, but it also permits *model-based interpretation* of new images and serves as a central representation for accumulating 3D scene information from various low-level experts. Our new research addresses these aspects of time-history 3D scene information.

Items 1-6 address the high throughput signal processing aspects of SBIU, whereas items 7-10 address the advanced image understanding aspects of this problem. Table 1-1 summarized objectives which must be attained to achieve the overall goal of SBIU. In Table 1-2, techniques required to attain these objectives are listed, and Table 1-3 lists the disciplines which will contribute to the achievement of our goals. As well as image processing *per se*, we must study the importance of efficient database organization and manipulation since storage or transmission of a very large database will be required for SBIU.

To properly address understanding of time-varying space-based images, we fell that three different SBIU time-varying image processing scenarios (Table 1-4) must be separately addressed. We distinguish the three cases by the change rate and the domain of analysis. In the first case (rapid time-variations), we can consider a missile launch. In this application, the objective is to track the time-history of the missile and to transmit the information that a missile has been launched (from subsequent sensors, the missile's trajectory etc. can be obtained from our system techniques and algorithms). The second case (medium time-variations) can concern monitoring of key sites such as airports, railroads and harbors and known areas of anticipated concentrations of troops or armor. In this case, troop or armor movement and air, land and sea activity can be obtained from time-varying image data. This second scenario is typical of a case in which extensive AI and IU techniques are appropriate (i.e., the use of information on the locations of hangers, runways, railroad tracks, terminals, switching yards, harbor channels, docks, piers, etc.). This also requires the locations and registration of these items in sequential image frames. The third case (slow time-variations) addresses urban development and agricultural or land use activity (as in Landsat and ERTS case-studies).

The three scenarios noted in Table 1-4 constitute our definition of the SBIU problem. All cases require the techniques and disciplines noted in Tables 1-2 and 1-3. The first case (rapid time-variations) requires primarily subpixel image registration, frame integration, frame interpolation, and image differencing. The second case requires techniques involving image interpretation, 3D scene

- Detection of image changes
- Use of *a priori* knowledge
- Location of key landmarks
- Time-history track file acquisition
- Interpretation of time-history data
- 3D *scene* interpretation
- Efficient storage and retrieval of information from database

Table 1-1: Objectives of Space-Based Image Processing

- Image enhancement and preprocessing
    - Image registration (subpixel) for frame integration
    - Image subtraction for time-history extraction
    - image interpolation for image subtraction
- Image segmentation
- Feature extraction
- Image modeling
- 3D scene modeling and interpretation
- Hierarchical database design

Table 1-2: Image Processing Techniques Required for SBIU

modeling, 3D matching and comparison, plus knowledge-based geometric reasoning. The third case needs more statistical techniques and statistical image models, more so than do the others. All cases require object and scene modeling, image preprocessing and enhancement plus segmentation, feature extraction and

- Pattern recognition
- Image understanding
- Human perception
- Artificial Intelligence
- Optical Processing
- Digital Processing

Table 1-3: Disciplines Required to Achieve Real-Time Space-Based Image Processing

TIME CHANGE	EXAMPLES	DOMAIN OF ANALYSIS
Rapid	Missile Launch	Image Pixels
Medium	Railroad, Airport, Harbor, Troops, Armor	Scene Structure
Slow	Agricultural, Land-use, Urban Development	Statistical Image Modeling

Table 1-4: Time-Change Scenarios

classification. Figure 1.1 depicts these aspects and the interactive multidisciplinary feedback required to solve these SBIU problems.

1.4. Benefit to Air Force Technology

With our three scenario problem definition (Table 1-4), we now consider the myriad of Air Force programs and technology that can benefit from our proposed research. First, we note that our research is directed toward the development of new algorithms and their realization in a hybrid optical/digital

architecture. However, devices and architectures being developed in related Air Force programs in VHSIC and VLSI, systolic array processors, Josephson junction devices, etc. can also be used for implementation of these algorithms. Our work will thus provide problem definition and direction regarding algorithms for such parallel processor architectures and technology programs. Large data storage requirements and studies of what constitutes a valid database are also integral parts of this program. Similar Air Force efforts toward data storage and database acquisition are thus of direct concern to this program. The Air Force programs in: intelligent sensors, intelligent task automation, automated manufacturing, image understanding, human perception and visual psychophysics will directly benefit from the inter-disciplinary nature of our research. The large Air Force effort in optical data processing will directly benefit since real-time spatial light modulators and holographic optical elements will be needed for implementation of our algorithms in real-time. The Air Force programs in missile guidance require a new set of algorithms and attention to the database requirements and performance measures used and thus they will likewise benefit extensively from the program. DARPA/AF programs such as HALO and HICAMP will clearly benefit from our chosen time-varying SBIU tasks.

The monitoring of changes and developments at cultural sites, such as urban areas military bases, is a very useful application of space-based sensors. The techniques we develop will aid in detecting and describing both large-scale an detailed changes. Furthermore, the techniques dealing with 3D matching and comparison, and knowledge-based geometric reasoning will enhance Air Force programs in sensing and robotics.

1.5. Summary of Research Done So Far

1.5.1. Year One Research

In our first year of research, we focused on the development and evaluation of methods which yield representations of structural and textural information in an image and relate these representations to object and surface contour properties of the scene. The techniques studied included *Probabilistic Graph Matching*, *Multiple Resolution Structural Basis Functions*, and *Textural Surface Models*. The structural basis function and textural models were found to be particularly well suited to parallel or optical processor implementation. Two digital processing facilities for use in this program were also assembled: the RAPIDBus architecture, and an Optical Data Processing, Digital Processing and Simulation Facility.

We also achieved a major effort on the extraction of time-varying subpixel target in noise. This time-change scenario concerns applications such as the detection of missile launches or aircraft in flight. In the first year, we successfully demonstrated the conceptual ability to detect and track subpixel targets.

In the low-level processing, we have described techniques for extracting building structures from high resolution aerial images of urban scenes. Edge points are first extracted from an image, and then straight line segments are fitted to them. Junctions are then formed from the line segments. These junctions are used to assign the segments to a structural model of buildings. A search using a Hough transform is then performed to look for new line segments predicted by the model.

A fundamental problem in interpreting complex images is to relate image features to scene features. In our context, this involved distinguishing two classes of image line segments, those arising from building boundaries and those arising from texture or shadow boundaries. We handle this problem by utilizing task specific knowledge. We assume that lines forming junctions arise from building corners only if one of the lines is vertical in the scene, i.e., is directed toward the vertical vanishing point. These lines are then labeled as part of a building model that consists of an arbitrary number of connected vertical faces covered by a roof. Lines that are not consistent with this building model are assumed to arise from texture or shadow boundaries.

In the low-level processing, we have also described experiments which determine how to efficiently search a line image in order to form junctions. Each line segment in the image is represented as a unique unit containing the x,y coordinates of the two end points. The set of line segments are stored as a list. A simple but inefficient way to determine the lines that lie within a small window in the image is to test each line in the list. The access time can be improved by dividing the image into a number of small areas called sectors. Each sector has a list of the line in its area. The search now requires only that the lists of the sectors containing the window be searched. We have empirically determined that the fastest access time is obtained when the image is divided into sectored areas forming from 6 to 8 rows and columns.

In the high-level processing we have described, techniques for representing,

constructing, and updating the scene model. The scene model is a surface-based description of an urban scene, and is incrementally acquired from a sequence of images. Each view of the scene undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of building. The initial model, constructed from the wire frames obtained from the first view, represents an initial approximation of the scene. As each successive view is processed, the model is incrementally updated and gradually becomes more accurate and complete. Task-specific knowledge is used to construct and update the model from the wire frames.

The model is represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. This permits the representation of partially complete, planar-faced objects. Because incremental modifications to the model must be easy to perform, the model contains mechanisms to (1) add primitives in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and (2) modify and delete primitives if discrepancies arise between newly derived and current information. The model also contains mechanisms that permit the generation, addition, and deletion of hypotheses for parts of the scene for which there is little data.

1.5.2. Year Two Research

This optical feature extraction effort in year two included attention to moment, chord and other optically-generated feature spaces. Architectures for each of these methods were devised and initial results were obtained. These showed: the ability to optically implement various feature extractors: the architecture for a hybrid optical/digital moment processor, successful initial tests of this architecture on a ship image data base and a robotic pipe part data base; new results on the accuracy of distortion parameter estimation with this processor, an advanced correlation SDF synthesis method and most successful initial test results of it on ATR vehicles. Our time-change detection work has achieved various significant results and demonstrations of the ability to detect subpixel target; the development of new single differencing methods that prove promising for clutter suppression; the initial formulation of general space/time filtering for target enhancement and background suppression; the investigation of detector limitation effects. Our efforts have also pointed towards more sophisticated space/time processing methods for better clutter suppression.

The hybrid optical/digital representation and matching effort of the project in year two has focused on the development and evaluation of methods which yield representation of structural and textural information in an image, and may be used for matching image to scene models. For *Probabilistic Graph Matching*, we have investigated methods of subgraph decomposition which permit branch-and-bound search of the matching tree and provide efficient pruning of the possible matches. The MRI (multiresolution Rotation Invariant) operator and the

MRD (Multiresolution Difference) transform have been introduced to extract structural and textural features of images for use in matching and interpretation phases of analysis. The MRI is a complex operator derived from derivative expansions of Gaussian kernels and has magnitude of response independent of feature orientation and phase angle of response which provides information about orientation. The spatial and frequency domain properties of these operators have been studied and an approximation MRI operator which uses difference of shifted Gaussian kernels has been derived and shown to be computationally efficient due to the scaling and shift properties of the Gaussian kernel. The MRI operator have been applied to aerial images of objects and textures. The MRI operators have been used to characterize and classify textures from aerial images. This set of multiresolution operators permits classification of texture independent of the size and orientation of the texture pattern itself. The statistical distribution provides information on the relative scale and the relative orientation. Experiments on textures from aerial images and textures from simple patterns have been carried out and compared to previous texture energy operators.

Our effort of year two has also resulted in techniques dealing with two levels of processing required for the task of describing 3D scene: the 2D image level detecting features, such as edges, lines, and corners in images and the 3D scene level representing, constructing, and updating the 3D scene model.

In the low-level of processing, we have determined a set of 3-D line segments in the scene which correspond to building boundaries. For this

purpose, we have developed a stereo algorithm using the technique of dynamic programming. We have explored a method to match the epipolar line pairs in a stereo pair and determine a rather dense depth map of the scene, using intra- and inter-scanline search.

Intra-scanline search determines the correspondence between edges in the same scanline of the left and right images. This search can be treated as the problem of finding a matching path on a 2D search plane whose axes are the right and left scanlines. Vertically connected edges in the images provide consistency constraints across the 2D search planes. Inter-scanline search in a 3D search space, which is a stack of the 2D search planes, finds the vertically connected edges and applies the constraints. By considering both intra- and inter-scanline searches, the correspondence problem can be cast as that of finding in a three-dimensional search space the matching surface that has the best match scores from intra-scanline search and also satisfies the consistency constraints from inter-scanline search. This problem is solved using dynamic programming for both searches.

In the high-level of processing, we have investigated model building using rangefinder data, which is already three dimensional, bypassing the problem of generating a 3D description from 2D data. We have developed techniques for representing, constructing, and updating the scene model. The model is in the form of 3D faces, edges, vertices, and their topology and geometry. A range image is segmented into edge points to which linear segments are fit. The original

line segments are refined to eliminate gaps. Faces are then fit to the line drawing. The final model is represented as a graph in terms of the symbolic primitives line, face, edge, and vertex. Although the final description is three-dimensional, most of the processing is done in the two-dimensional image space.

1.6. Research Progress in Year Three

The reduced level of effort of each portion of this contract allowed only limited support. Thus we have chosen to emphasize only the following research issues in our year three effort.

1.6.1. Subpixel Target Detection and Tracking

The objective in processing time-sequential imagery obtained from a staring mosaic sensor is to detect and track dim and small-area targets in the presence of additive noise (due to sensors) and background (e.g., cloud) movement between the frames. We investigated the use of six algorithms (Single Differencing, Double Differencing, Linear Interpolated Differencing, Parabolic Interpolated Differencing, Spatial Differencing and Spatial Filtering) using a well-controlled set of synthetic imagery. These results are detailed in Chapter 2 of the report.

1.6.2. Subpixel Delay Estimation Using Group Delay Functions

An important aspect of the space-based image processing is the estimation of background movement so that it can be effectively compensated for in the processing. This background shift between successive frames is typically *subpixel*, i.e., the shift between the image will be a fraction of a pixel. We have investigated the use of the recently introduced Group-Delay Function(GDFS) for

this problem. The GDFS provide the delay information as well as the Signal to Noise Ratio (SNR) information simultaneously. Based on this, we have come up with a new *adaptive* delay estimation procedure. The details are provided in Chapter 3 of this report.

1.6.3. Detection of Target Trajectories using the Hough Transform

Once the time-sequential space-based imagery are processed on a frame by frame basis, we can identify the target movement from frame to frame. In the ideal scenario of constant-velocity, unoccluded single targets, the target locations from frame to frame form a straight line in 3D. But because of changes in the target velocities and because of occlusions and multiple targets, it is important to track curved trajectories. We present a new technique for this in Chapter 4. This involves a straight-line Hough Transform (explained in detail in Chapter 4), thresholding and simple transformation in the *Hough* space and an *inverse HT*. The transformation are easily achieved by merely shifting the Hough space along one of the axes. The peaks in the Hough space identify the type of trajectory and provide its location. Experimental results are presented.

1.6.4. Image Understanding Techniques for 3D Scene Interpretation

The problem of detecting three-dimensional change in a complex urban scene is a very difficult one, particularly since any information extracted from the complex images is highly incomplete and contains many errors. Therefore, we have thus far concentrated mainly on the problems of extracting information from such images and accumulating the information in a 3D scene model.

In this report, we describe results in two aspects of these problems: low level image analysis and high level model maintenance. The goal of low level image analysis is to generate a set of reliable line segments in the scene which correspond to building boundaries.

In order to purse the problem of high level model maintenance independent of the current state of the low level image analysis research, we have chosen to investigate model building using rangefinder data, which is already three dimensional. Specifically, we have developed a method of generating a recognition strategy of an object from its 3D model. The recognition strategy uses only the 3D boundaries which are fairly easy to extract, thereby reducing the segmentation time. Thus, an object description to be used at run time is first precompiled from its model. The description includes the possible aspects of the occluding boundaries of the object, which we call 3D-profiles, when it is observed from all the possible viewing directions. In addition, it contains all explicit description of the order in which the search tree must be explored at run time.

Chapter 2

Subpixel Target Detection and Tracking

2.1. Introduction

The objective in processing time-sequential imagery obtained from a staring mosaic sensor is to detect and track dim and small-area targets in the presence of additive noise (due to sensors) and background (e.g., cloud) movement between the frames. Several algorithms have been suggested in the literature for detecting moving targets. In this chapter, we compare six of these algorithms using a well-controlled set of synthetic imagery generated by a computer.

Before presenting the algorithms, a few remarks about the peculiarities of this problem are needed. (i) The targets are small in size (occupying a few pixels at most) and have radiances comparable to the background radiances. (ii) The targets move relatively fast compared to the background (iii) The background movement between successive frames is usually less than a pixel. (iv) The sensor noise level is usually low (20 to 30 dB below) compared to the target intensities. Because of these size peculiarities, our algorithms must be capable of estimating subpixel shifts and compensating for them. Thus our numerical experiments pay special attention to the generation and processing of subpixel shifts in images.

This chapter is organized as follows. In the next section, we outline our procedure to generate the desired imagery on the computer. This also helps set up the notation used. Section 2.3 provides the equations and explanations connected to the six algorithms (Single Differencing, Double Differencing, Linear Interpolated Differencing, Parabolic Interpolated Differencing, Spatial Differencing, and Spatial Filtering) we will investigate. Section 2.4 introduces the measures of effectiveness we will use to carry out this performance evaluation. Extensive numerical results are then presented in Section 2.5 in a condensed form. This is followed by our conclusions and comments in Section 2.6.

2.2. Generation of Synthetic Imagery

We can evaluate the algorithms properly only if we can generate extensive imagery that represents various possible parameters. Towards this purpose, the generation of images is carried out at two levels, namely, the ground (high resolution) level and the detector (low resolution) level.

2.2.1. Ground-Level Imagery

This represents the high-resolution imagery and consists of two parts. Let $\mathbf{H}_k(\mathbf{x})$ denote the hight-resolution image at time k where $\mathbf{x}$ denotes the 2-D space variable (x,y). The image $\mathbf{H}_k(\mathbf{x})$ consists of two parts, namely, the target and the background.

$$\mathbf{H}_k(\mathbf{x}) = \mathbf{T}_k(\mathbf{x}) + \mathbf{B}_k(\mathbf{x}), \tag{2.1}$$

where $\mathbf{T}_k(\mathbf{x})$ and $\mathbf{B}_k(\mathbf{x})$ denote the target image and the background image.

respectively at time k. In the high resolution image, the target $\mathbf{T}_k(\mathbf{x})$ occupies a few pixels of area and is assumed, without loss of generality, to have magnitude 1. Let Δt denote the time interval between the snapshots. If $\mathbf{v}_T$ denotes the speed of the target, the target moves by $(\mathbf{v}_T \Delta t)$ between successive snapshots. This target movement is easily simulated by shifting $\mathbf{T}_k(\mathbf{x})$ to $\mathbf{T}_k(\mathbf{x}-\mathbf{v}_T \Delta t)$ between successive frames.

We model the background $\mathbf{B}_k(\mathbf{x})$ in the high resolution image as a sample realization of a 2-D random process with zero mean and the following non-isotropic covariance function.

$$\mathbf{R}_B(\tau_x,\tau_y) = \sigma_B{}^2 \cdot \rho_x{}^{|\tau_x|} \cdot \rho_y{}^{|\tau_y|} \ , \tag{2.2}$$

where $|\rho_x| < 1$ and $|\rho_y| < 1$. In the above, $\sigma_B{}^2$ denotes the background variance and ρ_x and ρ_y denote the correlation coefficients in x and y directions. Small ρ_x and ρ_y values imply sharper covariance functions, larger spectral bandwidths and more fluctuations in the background. Prior work [1, 2, 3] indicates that the infrared images of the earth's background in 3-5 μm and 8-12 μm can be characterized by the above 2-D Markov model. While the zero mean assumption is not entirely accurate, we can justify its use because the optical processing systems (that we plan to use) do perform an automatic DC removal [4]. Movement of the background between successive frames is usually small compared to the target movement and can be simulated in the same way. However, this background movement appears as a *subpixel* shift in the detector (low resolution) image thus forcing us to use special algorithms capable of

subpixel shift estimation in the processing stages. The background image $\mathbf{B}_k(\mathbf{x})$ is obtained by passing the 2-D white noise (available through standard subroutines) through first-order Infinite Impulse Response (IIR) filters in both x and y directions. By changing the coefficients of these IIR filters, we can obtain different ρ_x and ρ_y values.

2.2.2. Detector Imagery

The imagery available for processing is not the high resolution imagery discussed above, but rather is the low resolution output of the mosaic sensor array. We assume this to be an array of 60x60 pixels whereas the high resolution image contains 512x512 pixels. We assume the detector footprint to cover a region equivalent to an 8x8 region in the high resolution image. However, the atmospheric effects cause overlap between adjacent detector footprints. This overlap is modeled by the Gaussian blur function shown in Figure 2.1. Thus, the 60x60 detector image is obtained by convolving the high resolution image with the blur function in Figure 2.1 and then summing the pixel values in non-overlapping 8x8 regions. This summing operation has many effects. The first is that the target now appears to be of a subpixel size in the detector image. The second effect is that the background movement between successive snapshots will be subpixel. Finally, the correlation coefficient of the background noise changes because of the summing. In fact, if the background noise has a correlation coefficient ρ_h in the high resolution image, the correlation coefficient ρ_l in the low resolution image (obtained by adding d pixels) is given by

$$\rho_l = \frac{\rho_h (1 - \rho_h^d)^2}{d(1 - \rho_h^2) - 2\rho_h (1 - \rho_h^d)}, \tag{2.3}$$

where we have omitted the intermediate steps due to length constraints. To illustrate the reduction of the correlation coefficient, consider $\rho_h = 0.99$. After 8x8 pixel detector integration, we obtain $\rho_l = 0.90$. Similarly $\rho_h = 0.95$ becomes $\rho_l = 0.76$ after detector integration. Thus, the conversion from the high resolution to the low resolution image causes a decrease in the correlation coefficient, or equivalently, an increase in the spectral bandwidth.

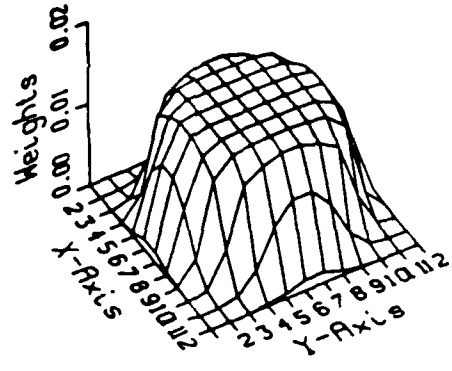


Figure 2-1: Detector Blur Function

Finally, the sensor noise in the mosaic sensor array is simulated as additive, white, zero mean Gaussian noise with variance σ_u^2. This noise is uncorrelated from one snapshot to another unlike the background noise which is related through simple shifts between image frames. The uncorrelated noise (UCN) level

is typically 20 to 30 dB below that of the background or correlated noise (CN). This UCN is easily generated on the digital computer and added to the detector image. This procedure for imagery generation is summarized in the block diagram in Figure 2.2.

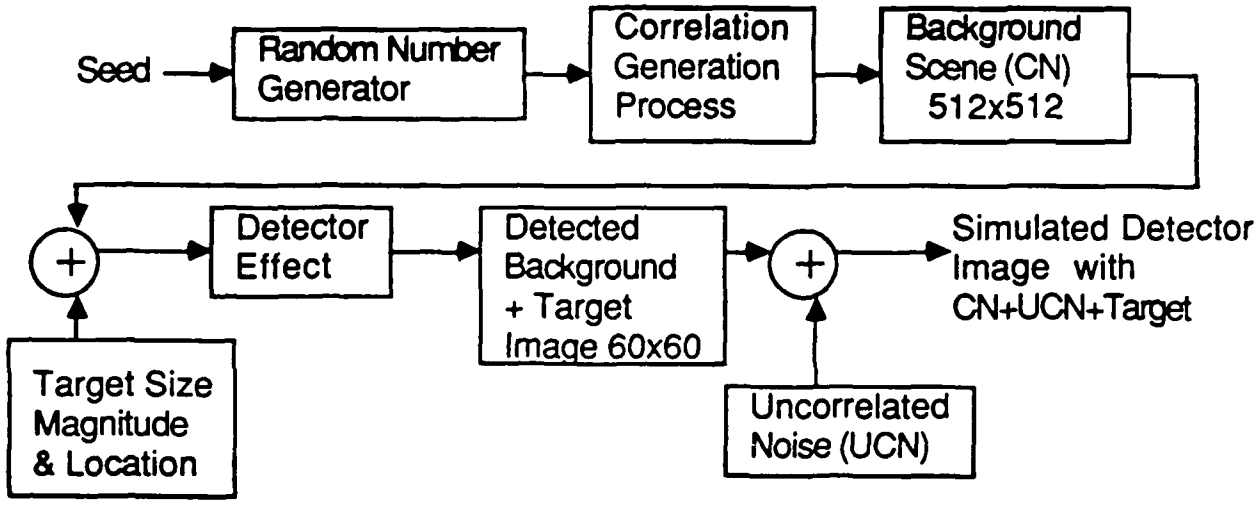


Figure 2-2: Block Diagram Representing the Synthesis of the Imagery

2.3. Processing Algorithms

We consider six different algorithms designed to extract the point targets from the slowly moving background and uncorrelated noise. These algorithms are characterized by the fact that only a few snapshots are used for processing. This is necessary because of the memory constrains on the processor. Each snapshot represents 512^2 bytes of data and thus it is desirable to keep the number of snapshots required at any time to as low a number as possible. These algorithms also make use of the fact that the target is moving faster compared to the

background. Let $\mathbf{I}_k(x)$ denote the detector image at the time instant k. We consider only 1-D analysis for simplicity.

2.3.1. Single Differencing

This is the simplest of the algorithms and subtracts one snapshot from the next one to produce the output image $\mathbf{D}_k(x)$ as below.

$$\mathbf{D}_k(x) = [\mathbf{I}_k(x) - \mathbf{I}_{k-1}(x)] / 2. \tag{2.4}$$

It is easy to see from the above equation that the output $\mathbf{D}_k(x)$ is zero if the input $\mathbf{I}_k(x)$ is constant. Thus slower backgrounds are attenuated more than the faster targets. The uncorrelated noise is obviously doubled in variance by this algorithm. This simple algorithm is an approximation to the time derivative operation and thus enhances images changing faster in time. This is very simple and thus easy to implement optically.

One can also consider the above temporal filtering operation as spatial filtering, i.e., Equation (2.4) can be viewed as the description of a linear, time-invariant system with input $\mathbf{I}_k(x)$ and output $\mathbf{D}_k(x)$. If Δt denotes the sampling interval between the snapshots and v denotes the speed of the background or the object of interest, the snapshot at time k is related to the image at time $(k-1)$ by

$$\mathbf{I}_{k-1}(x) = \mathbf{I}_k(x - \Delta x), \tag{2.5}$$

where

$$\Delta x = v \cdot \Delta t. \tag{2.6}$$

Using the results in (2.5), we can view the single differencing algorithm as a spatial filter with the following transfer function

$$\mathbf{H}_1(w_s) = (1 - e^{-jw_s \Delta x}) / 2, \tag{2.7}$$

where w_s is the spatial frequency. The magnitude response can be easily shown to be

$$|\mathbf{H}_1(w_s)| = \{(1 - cosw_s \Delta x) / 2\}^{1/2}. \tag{2.8}$$

This transfer function clearly demonstrates that as $(w_s \Delta x)$ decreases, $|\mathbf{H}_1(w_s)|$ also decreases. Thus, images with low w_s spatial frequencies (broad extents) and slow movements (Δx or v) will be attenuated more than smaller and faster objects. This is what makes this algorithm distinguish slower backgrounds from faster targets. One disadvantage that must be pointed out is that this algorithm and filter have the fixed structure in (2.4), (2.5), and (2.7) and is thus not capable of adaptively changing its transfer function.

2.3.2. Double Differencing

If single differencing is an approximation to the first time derivative, the double differencing algorithm,

$$\mathbf{D}_k(x) = -\frac{1}{2}\mathbf{I}_k(x) + \mathbf{I}_{k-1}(x) - \frac{1}{2}\mathbf{I}_{k-2}(x), \tag{2.9}$$

is an approximation to the second derivative in time. Once again, the output is zero for stationary objects and the output increases with the movement of the object. A frequency domain analysis of the above transfer function yields the following transfer function

$$\mathbf{H}_2(w_s) = -\frac{1}{2} + e^{-jw_s\Delta x} - \frac{1}{2}e^{-j2w_s\Delta x}. \tag{2.10}$$

For the magnitude of the above transfer function, we obtain

$$|\mathbf{H}_2(w_s)| = \{\frac{1}{2}(3 - 4cosw_s\Delta x + cos2w_s\Delta x)\}^{1/2}. \tag{2.11}$$

The above magnitude response clearly shows that the gain is very small for small $(w_s\Delta x)$ values as derived. The computational requirements of this algorithm are slightly more than for the single differencing algorithms in that it requires three image frames for processing. But it has the same drawback as the single differencing algorithm in that it is not adaptive.

2.3.3. Linear Interpolated Differencing

The two previous algorithms do not adapt to different background shifts. A simple algorithm that seems to have been overlooked in most literature is based on the block diagram in Figure 2.3. In the first step, the subpixel shift Δ_B between the two successive image frames is estimated as $\hat{\Delta}_B$. This shift is mainly due to the large background and thus requires subpixel shift estimation. Several methods [5, 6, 7, 8, 9] have been suggested for this purpose. A careful study of these had indicated that simple parabolic subpixel delay estimator outlined in references [5] and [9] is the best compromise between the estimation accuracy and ease of implementation. In this method, the two images of interest, namely $\mathbf{I}_k(x)$ and $\mathbf{I}_{k-1}(x)$ are cross-correlated to obtain $\mathbf{C}_k(x)$. This correlation surface is then searched for the peak value. A second-order polynomial is then filtered to the correlation values near the peak. The coefficients of this second order polynomial

can then be used in a straight forward manner to obtain the subpixel shift estimates.

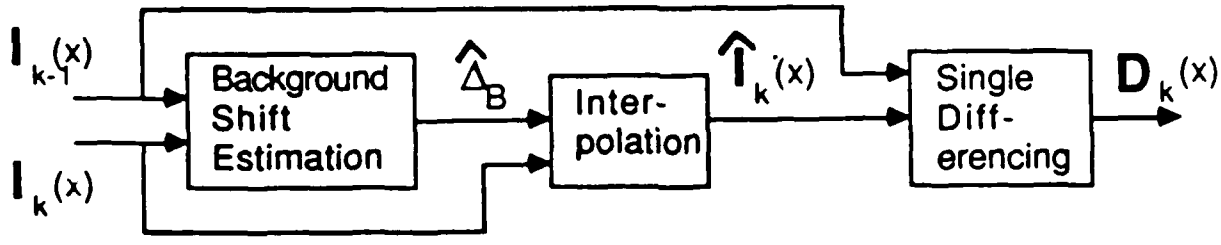


Figure 2-3: Block Diagram Indicating the Various Steps in Interpolated Differencing

The subpixel shift estimate $\hat{\Delta}_B$ is then used in an interpolator to obtain a shifted version of $\mathbf{I}_k(x)$. If the interpolation and estimation are carried out accurately, the background in the interpolated image will be identical to that in $\mathbf{I}_{k-1}(x)$. The differencing operation indicated in the last block of Figure 2.3 will then force the background term to zero while retaining most of the target information. In reality, neither the subpixel delay estimator nor the interpolator works perfectly. Thus, it is of interest to compare the performance of different interpolation schemes.

In the linear interpolated differencing method, the output $\mathbf{D}_k(x)$ is given by

$$\mathbf{D}_k(x) = [\hat{\mathbf{I}}_k(x) - \mathbf{I}_{k-1}(x)] / 2 , \tag{2.12}$$

where $\hat{\mathbf{I}}_k(x)$ is obtained from $\mathbf{I}_k(x)$ and $\mathbf{I}_k(x-1)$ according to the following linear interpolation rule.

$$\hat{\mathbf{I}}_k(x) = (1 - \Delta_B)\,\mathbf{I}_k(x) + \Delta_B \mathbf{I}_k(x-1). \tag{2.13}$$

where Δ_B denotes the shift of the background between two successive frames. The cascading of the interpolator and the single differencing represents a linear time-invariant filter with the following transfer function.

$$\mathbf{H}_3(w_s) = \{[1 + \Delta_B(cosw_s - 1) - cos(w_s\Delta_B)] + j\,[sin(w_s\Delta_B) - \Delta_B sinw_s]\} / 2\,. \tag{2.14}$$

where we assumed that Δ_B is estimated perfectly. Once again, we see that $\mathbf{H}_3(w_s)$ is small for small values of $(w_s\Delta_B)$. If Δ_B or w_s is zero, we find $\mathbf{H}_3(w_s)$ to be identically zero. Thus the background will be suppressed by the above method. The above shift estimate $\hat{\Delta}_B$ is accurate only for the background and not for the target. Thus the target which moves by more than Δ_B will not be suppressed by the linear interpolated differencing method.

2.3.4. Parabolic Interpolated Differencing

Obviously, the effectiveness of the previous algorithms depends not only on the subpixel shift estimation method, but also on the interpolation. In order to address this issue, we have also included the following parabolic interpolation scheme in our algorithms. This method uses three neighboring image pixels for interpolation

$$\hat{\mathbf{I}}_k(x) = \frac{\Delta_B(\Delta_B+1)}{2}\mathbf{I}_k(x-1) + (1 - \Delta_B^2)\,\mathbf{I}_k(x) + \frac{\Delta_B(\Delta_B-1)}{2}\mathbf{I}_k(x+1).$$

Once again, the cascade of the interpolator and the single differencing can be viewed as a linear time-invariant filter with the following transfer function

$$\mathbf{H}_4(w_s) = \{[1 + \Delta_B{}^2(cosw_s - 1) - cos(w_s\Delta_B)] + j[sin(w_s\Delta_B) - \Delta_B sinw_s]\} / 2 . \quad (2.16)$$

This transfer function becomes zero if either Δ_B or w_s is zero. Thus, the background will be suppressed more than the target. A complete analysis based on the spectral characteristics of the target and the background are necessary to decide which interpolated differencing method work better and when.

Both interpolated differencing methods are *adaptive* in the sense that they attempt to align the backgrounds before subtraction. While this may be computationally more involved, it should yield more consistent (as well as better) results over a range of background shifts. We believe that the required subpixel shift estimation as well as the interpolation-shift operation can be carried out using optical processing methods. We did not consider higher-order interpolation such as cubic splines because they cannot be easily implemented in optical processors.

The interpolated differencing methods have not received much attention previously because of the amount of computation required for interpolation before shifting. But optical processors are inherently continuous and the point spread functions associated with them cause automatic interpolation using the sinc function kernel (for square apertures).

2.3.5. Spatial Filtering

In this method proposed by Wang [3], we consider only one image frame at a time and use the fact that the background has adjacent pixels highly correlated, whereas the target pixels exhibit less spatial correlation. This is used to design an optimal spatial filter that yields minimum mean square error. This is achieved by estimating an average 3x3 covariance matrix estimated for each image snapshot and using this to solve for 9 coefficients that are used in a 3x3 filter mask. While this is mathematically elegant, it has a few practical drawbacks. First, this method requires the estimation of the covariance matrix which can be quite time consuming and not easily amenable to optical implementation. Second, this method requires that we specify a desired vector before the filter coefficients are obtained. Specifying the desired vector requires that we know whether the target is in the region of interest or not. This is not always feasible. Finally, this method does not make use of the temporal information at all. It ignores the fact that the targets move faster than the background. We included this algorithm in our study to see if the image parameters of interest are sufficiently tolerant that this algorithm can perform successfully.

2.3.6. Spatial Differencing

The last algorithm we consider is a nonlinear algorithm recently proposed [10]. This method works by considering 3x3 windows centered at the same point in successive image frames. The center pixel value from the current frame is subtracted from the nine pixel values of the same window in the previous frame. The output of this algorithm is the magnitude of the smallest of these nine differences. Because of the sorting operation inherent in this, it is a nonlinear operation. In that sense, this algorithm is very different from the previous five.

To understand how this method works, we note that the targets move faster than the backgrounds. Thus if the 3x3 observation window does not contain the target, there should be little difference between successive frames thus yielding very small values for the difference. On the other hand, if the window in one frame contains a target, it is probably not seen in the same window in the next frame. Thus simple subtraction will thus not eliminate this target. While this method is attractive because of its robustness to small amounts of noise, it also has two drawbacks. First, because of the nonlinear ranking operation needed, an optical implementation is difficult. The second problem is that targets that do not move by more than 1 pixel between frames (essentially those that do not move out of the window) will be suppressed by this algorithm.

2.4. Measures of Effectiveness

It is necessary to quantify the performance of the six algorithms proposed in the previous section before we select among them. To enable us to do this, we define the following measures of effectiveness. As always $\mathbf{T}_k(x)$ denotes the target image, $\mathbf{B}_k(x)$ denotes the Background image and $\mathbf{N}_k(x)$ denotes the sensor noise image at time instant k. While the measures we will use do not convey all the nuances of the algorithms, they can serve as useful measures of the average performance of these algorithms.

2.4.1. Background Suppression Factor

A common goal of all six algorithms is to suppress the background while retaining the target. To evaluate the ability of an algorithm to suppress the background, we define the Background Suppression Factor (BSF) as the ratio of the variance of the background-only image after processing to its variance before processing. Obviously, BSF depends on many variables other than the algorithm itself. These include the background size, shift and its spectral content. This measure has been used before [2] for this purpose. A word of caution in the use of this measure is that it must always be accompanied by a measure of target detectability. Otherwise, we can obtain an infinite BSF by simply setting all the filter coefficients to zero. This, of course, destroys any target that may be present.

2.4.2. Target Detectability

As discussed above, we must pay attention to what happens to the targets as various algorithms are applied. While several possibilities exist for characterizing the target effects, a simple and useful measure is the ratio of target radiance after processing to the radiance before processing. Good algorithms should keep this ratio large while yielding large BSF values. Another way we can evaluate how targets are affected is by plotting the receiver operating characteristics such as the probability of detection versus the probability of false alarm for various target detection thresholds. While the latter approach may provide more complete information, it is difficult to use such information as it is not a single number.

2.4.3. Uncorrelated Noise Variance

A third quantity usually ignored in the analysis of the algorithms is the variance in the output image due to input uncorrelated noise. While the uncorrelated noise in the input is 20 to 30 dB below the correlated noise, algorithms based on differencing increase the variance of uncorrelated noise while decreasing that of the correlated background. Thus it is important to measure the variance at the output due to uncorrelated noise also.

Of the six algorithms considered, only spatial differencing method is not linear. Thus, we can analyze the effects of the five linear algorithms on the target, the background and the uncorrelated noise separately and combine the results. This is not the case for the nonlinear spatial differencing schemes. It is

also better to consider ratios of the above measures rather than the absolute measure. This prevents the occurrence of rather unrealistic measures when some of the coefficients are arbitrary. For example, the single differencing in (2.4) uses the coefficients -1/2 and +1/2. By using coefficients -1, +1 instead, we will still obtain single differenced images, but with both the background and the uncorrelated noise variance going up by a factor of 4. Thus, use of ratios is more appropriate.

2.5. Experimental Results

In this section, we will summarize the results of an extensive effort to evaluate the performance of the six algorithms. First, a few words about the choice of the experimental parameters are needed. The high resolution image was chosen to be 512x512 pixels. With an effective detector footprint used of size 8x8 pixels, we obtained a low resolution detector image of 60x60 pixels (after edge pixel effects were ignored). The target size was chosen to be 8x8 pixels in the high resolution imagery (or equivalently 1 pixel or spread over 2 pixels in the low resolution image). The target radiance was assumed to be 1 in the high resolution image and it averaged out to be small (0.3706) in the low resolution image because of the detector effect. The correlation coefficients for the background noise were assumed to be the same in x and y and either 0.99 or 0.95. Substituting these ρ_h values in (2.3) yield ρ_l values of 0.90 and 0.76, respectively. We considered these two correlation coefficients, as they were reported [6] to model real IR images well. The uncorrelated noise level was kept at a level of 20

or 30 dB below the target and the background noise level. Various subpixel background shift values for Δ_B were considered. We next provide the observed results in a concise form. All values (shifts, ρ, etc) are given for the final low resolution detected image.

2.5.1. Background Suppression Factor (CN only present)

Table 2.1 shows the BSFs obtained by the six algorithms for the two choices of ρ_l (namely 0.90 and 0.76) and for three different shifts (all subpixel) of the background.

Back-ground ρ	Sub-Pixel Shift (horiz.,Vert.)	Linear Processing					Nonlinear Processing
		Simple Differencing		Interpolated Differencing		Spatial Filtering	Spatial Differencing
		Single	Double	Linear	Parabolic		
0.90	(0 0.125)	1209	15193	2435	3645	25416	1249
	(0.25 0.25)	160	546	397	515		441
	(0.5 0.5)	44	52	226	234		282
0.76	(0 0.125)	415	4126	709	1060	539898	446
	(0.25 0.25)	50	124	119	143		156
	(0.5 0.5)	15	13	61	64		120

Table 2-1: Background Suppression Factor for Only Correlated Noise Background (No Target or UCN Detector Noise)

The most prominent observation from this table is the large BSF values obtained by spatial filtering. The origin of this large BSF values is based on the fact that the coefficients of the spatial filter are rather small (of the order of 10^{-2} or less, depends on the background correlation coefficient), thus making the

outputs very small. In fact, these filter coefficients can all be scaled up or down by the same factor without affecting its optimality. Thus it is very important that we consider that ratio of this BSF to the variance of the uncorrelated noise or the target intensity. We note that this algorithm is independent of background shift and depends only on ρ as expected. We also note that a small ρ value broadens the spectrum and thus more suppression is observed for the smaller $\rho = 0.76$ value (as expected).

It can also be seen from Table 2.1 that decreasing ρ causes a general decrease (expect for the anomalous spatial filtering case) in the BSFs observed. This is consistent with our understanding that it is more difficult to suppress backgrounds varying more (smaller ρ values correspond to more white-noise like situations, i.e., containing higher spatial frequencies w_s). Another obvious and expected general trend is that larger background shifts (for the same reasons as above) also causes the achieved BSF to decrease.

All methods seem to yield very good BSFs for small background shifts. But as this background subpixel shift approached 1/2 pixel, interpolated differencing performed nearly five times better than the uninterpolated simple differencing methods. This is expected since the interpolated differencing used the correct background shift whereas no shifting is included in the simple differencing.

The nonlinear spatial differencing algorithm slightly outperformed the interpolated differencing schemes for the case of both smaller ρ values and larger background shifts. But the performance of the spatial differencing method is

much inferior to that of the interpolated differencing for smaller background shifts (for both small and large ρ). Recall that for target shifts above one pixel, spatial differencing will suppress the target also.

Since we do not know a priori the background shifts to be expected, we should use the algorithm that performs best for all background shifts, thus use of parabolic interpolated differencing is recommended on the basis of the fact that it yields reasonably large BSF value at very small background shifts and yields better BSF values than other methods (except for the anomalous spatial filtering algorithm) at larger background shifts. Thus, the consistency of the parabolic interpolated differencing method makes it very attractive.

2.5.2. Target Effects (Target only present)

While the six algorithms are designed to suppress the background correlated noise, it is desirable that they do not attenuate the target. To understand the effect of the various algorithms on the target, we show in Table 2.2 the "target amplification" (actually a loss, since all values are less than one) of various methods for different shifts. This quantity is simply the ratio of the target intensity after processing to the intensity before processing. We do not include the spatial differencing methods results in this table, because the nonlinear nature of this algorithm prevents us from analyzing the target and the noise separately. The spatial filter coefficients, of course, depend on the background correlation coefficient and thus we consider the performance for $\rho_l =$ 0.90 and 0.76. The target (in the final detected image) is about one pixel in these

data. As can be expected, the larger the target size, the better the target gain is expected to be. The results for subpixel targets may give preference to the interpolated methods.

Target movement (in Pixels) Between Frames (Horiz., Vert.)	Simple Differencing		Interpolated Differencing		Spatial Filtering	
	Single	Double	Linear	Parabolic	$\rho = 0.90$	$\rho = 0.76$
(0 , 0.5)	0.0265	0.0217	0.0301	0.0344	0.1157	0.0028
(0 , 1.0)	0.0695	0.1207	0.0634	0.0777	0.1157	0.0028
(0 , 1.5)	0.0928	0.1835	0.0810	0.0971	0.1157	0.0028
(0 , 3.0)	0.0972	0.1943	0.0841	0.0945	0.1157	0.0028

Table 2-2: Target Amplification Due to Various Algorithms

The first trend to note in this table is that increasing target movement results in less target degradation (indicated by larger target amplification values). This is as expected because faster targets are more easily detected by these algorithm than slower targets. We also note that the spatial filtering results are independent of target movement. This is expected because the spatial filtering procedure considers only one image frame at a time and ignores target movement between frames. This also reiterates our earlier observation that the very large BSFs shown in Table 2.1 for spatial filtering are not very meaningful. This is seen from the fact for $\rho_l = 0.76$, the target intensity after processing is 0.0028 times its intensity before processing.

In spatial filtering, the target intensity is reduced 50 times more than by other methods, but the background intensity variance is reduced 100 times more. Thus the (BSR/Target gain) ratio is better for the spatial filtering algorithm for many cases. To employ the same units, the target gain should be squared. In this case, spatial filtering performs poorly. The large reduction in the target strength is expected to cause practical problems when the target strength and size are varied and when limited dynamic range and various error source effects are included. The practical problems: estimating the covariance matrix (with its large computational requirements) and not knowing if the target is present (and its effect on specifying the desired vector) seem to be the dominant reasons to consider this algorithm only in specific cases. This merits further attention. The ease of implementing each algorithm (in optical technology, because the high computational load of each algorithm) is another issue of concern.

Once again, both double differencing and parabolic interpolated differencing seem to outperform the others for larger target shifts. Then parabolic interpolated differencing seems to yield consistent (low target degradation and high BSF) results over a wide range of parameter variations (although simple double differencing may perform best for some background shifts, target shifts and ρ values). Note that for smaller target movements (below ≈ 0.75 pixels), the target amplification achieved with interpolated differencing is better than with simple differencing.

We carried out a similar evaluation of the nonlinear spatial differencing

algorithm effects on the target. However, because this is a nonlinear algorithm, we introduced the target and the background together. The target amplifications obtained for different ρ_l values and different target movements are shown in Table 2.3. These values are one order smaller than those of the interpolated differencing methods. Whereas the BSF values for this algorithm are similar or less. Thus, spatial differencing does not seem to be preferable to the interpolated differencing methods.

Target 's velocity, (h,v) pixels/frame	Target Amplification	
	ρ =0.90	ρ =0.76
(0 , 0.5)	0.0060	0.0082
(0 , 1.0)	0.0072	0.0120
(0 , 1.5)	0.0402	0.0240
(0 , 3.0)	0.1038	0.0678

Table 2-3: Target Amplification for the Spatial Differencing Algorithm. (Target and CN present)

2.5.3. Uncorrelated Noise Effects

One aspect overlooked by the previous simulation is the effect of the algorithms on uncorrelated noise. We now consider the normalized ratio of the variance of the UCN to the variance of the CN background for the algorithms. Let β denote this ratio in dB. A large β value thus denotes more UCN than CN. For the images used in these tests, β = -30 and -20 dB for the original images. Table 2.4 contains the difference (in dB) of the β value after processing to that before processing. Larger entries in this table denote higher background CN

suppression factors, thus indicating more suitable algorithms. (i.e. less CN after processing and thus a larger β after processing, from which when we subtract the original negative β we will obtain a larger and more positive β difference). If the UCN value (before and after processing) remains unchanged, the above remarks hold (and large values in Table 2.4 denote better suppression of both CN and UCN). All algorithms generally increase UCN. This will lower β after processing (poorer performance considering UCN alone) and lower the values in Table 2.4 (poorer performance). Thus, in all cases, larger values in Table 2.4 correspond to more suppression of both CN and UCN and hence better results (ignoring target effects).

A few trends are obvious in this table. Decreasing ρ_l causes a decrease in the β improvement factor for all methods. Similarly, larger background shifts result in smaller improvement. These results are expected. A surprising result is that the double differencing algorithm yields numbers comparable to the interpolated differencing schemes. (This may be due to quantization issues, such as the fact that our subpixel shifts are limited to the multiples of 1/8 pixel). More tests and analysis are necessary to answer this. For $\rho_l = 0.90$, small background shifts yield β improvement factors of about 30 dB and larger background shifts yield a 20 dB improvement. Once again, the parabolic interpolated differencing algorithm and others perform well.

Back-ground ρ	Sub-Pixel Shift (hori.Vert.)	Linear Processing					Nonlinear Processing	
		Simple Differencing		Interpolated Differencing		Spatial Filtering	Spatial Differencing	
		Single	Double	Linear	Parabolic		-20	-30
0.90	(0 0.125)	28	44	30	33	30	25	30
	(0.25 0.25)	19	29	21	24		22	26
	(0.5 0.5)	13	19	18	19		20	25
0.76	(0 0.125)	23	38	25	27	18	23	27
	(0.25 0.25)	14	23	16	18		20	23
	(0.5 0.5)	9	13	13	14		19	22

Table 2-4: Improvement (in dB) in the Ratio of the Uncorrelated Noise Variance to the Correlated Noise Variance with Processing

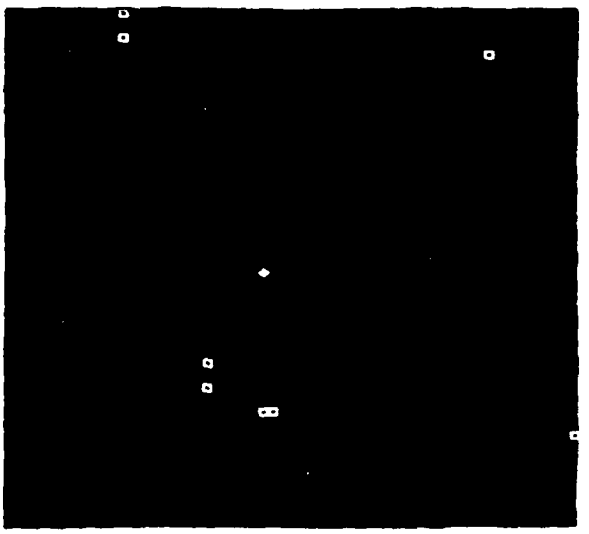
2.5.4. Case Study

While the previous methods summarize the performance of several aspects of the various algorithms, they do not present the total picture, i.e. will targets be detected. Towards this purpose, we carried out the following simulation. In the first case study, the CN background was modeled as having $\rho_l = 0.90$ and a variance of 1. The subpixel movement of the target between successive snapshots was taken to be (0.5, 0.5) pixels. The targets were assumed to be of size 1x1 in the detector image with radiance of 1. The targets strength was the same compared to the strength of the background. Eight targets were randomly dispersed in the detector image of size 60x60. All targets were assumed to move by (3, 4) pixels between successive image frames. In each image frame, uncorrelated noise with strength -30 dB below that of the correlated noise was added.

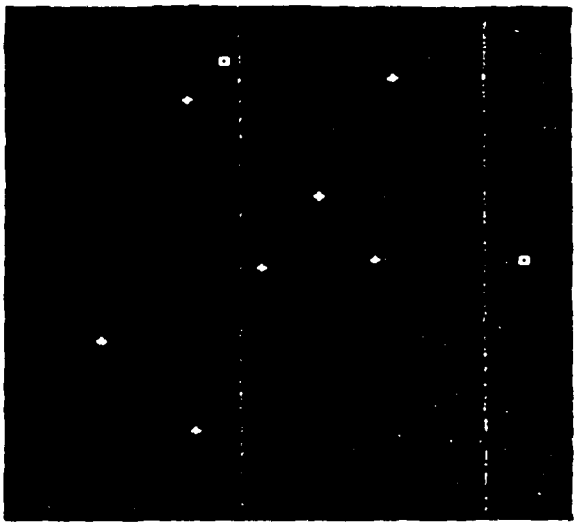
Differencing Method	Number of Hits
Single	1
Double	6
Linear Interpolated	8
Parabolic Interpolated	7
Spatial	2

Table 2-5: Number of Hits (Out of 8 Targets) for the Five Methods

These images were thus input to the five algorithms (excluding spatial filtering which does not use time information) and the resulting processed output images were then searched for the 8 largest output values. If the algorithm worked perfectly, the 8 outputs should correspond to the 8 targets and should yield the correct target locations. In reality, this does not happen. Since we know a priori the correct location of the targets, we can decide how many of the targets have been correctly located. The 8 largest target output values obtained are shown in Figure 2.4. In this figure, the crosses denote the targets correctly located whereas the squares denote output peak values that do not correspond to correct locations of targets. The number of correct targets for the five algorithms is listed in Table 2.5. This table reveals the inadequacy of the single differencing and the spatial differencing methods. On the other hand, the other 3 methods seem to perform equally well. We were surprised to see that linear interpolated differencing yielded 8 correct targets (1 better than the more exact parabolic interpolated differencing). These results must be investigated further for more scenarios.

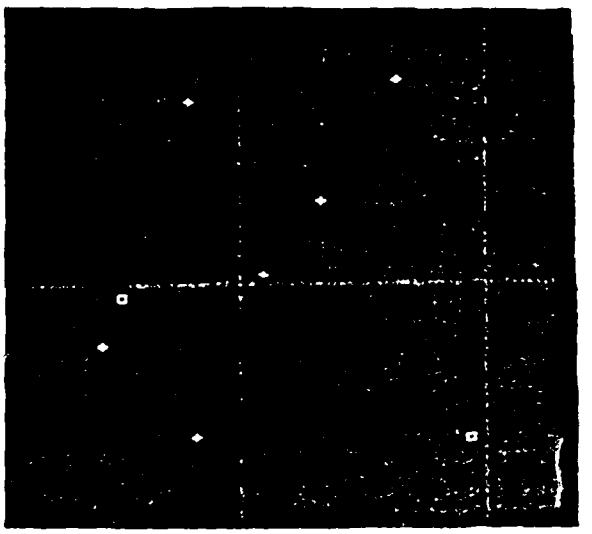


(a) Single Differencing

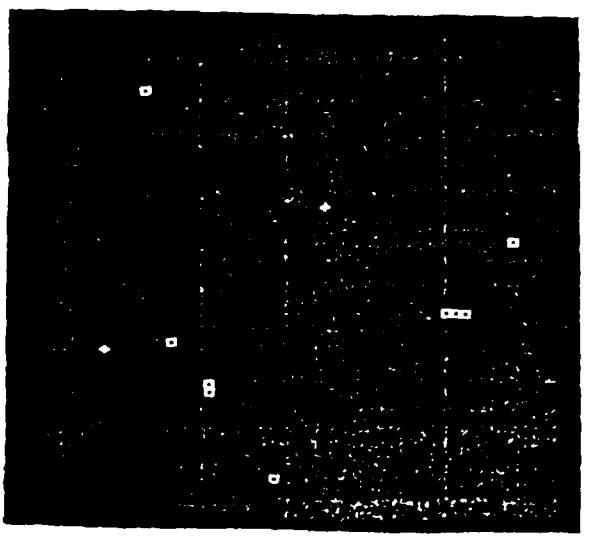


(b) Double Differencing

(c) Linear Interpolated Differencing



(d) Parabolic Interpolated Differencing



(e) Spatial Differencing

Figure 2-4: The Thresholded Outputs of the Five Algorithms for CN with $\rho_l = 0.90$ and UCN = -30 dB lower. Crosses denote correct targets and squares denote false peaks

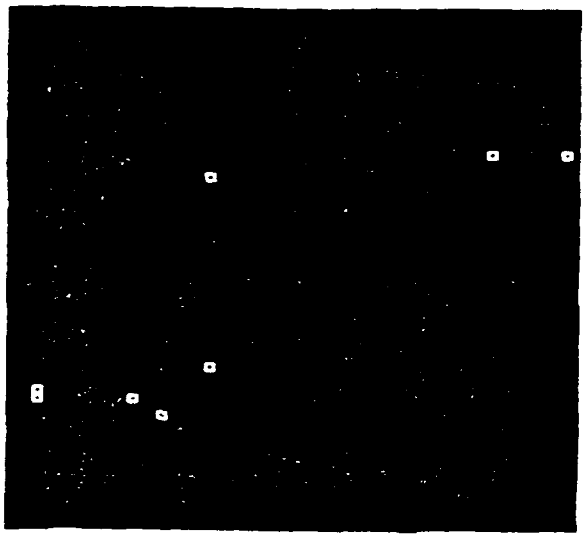
For the second case study, we removed the uncorrelated noise and changed the background correlation coefficients ρ_l to 0.76. All other parameters remained the same. The corresponding thresholded outputs are shown in Figure 2.5. All algorithms fail. However, the parabolic interpolated differencing method is able to locate 4 of the 8 targets. Only double differencing is able to locate a single target correctly. All other algorithms cannot locate even one target correctly. This case study clearly demonstrates that the parabolic interpolated differencing algorithms yields more consistent results for a wide range of scenario parameters. It thus appears to be preferable (and quite suitable for optical realization).

2.6. Conclusions

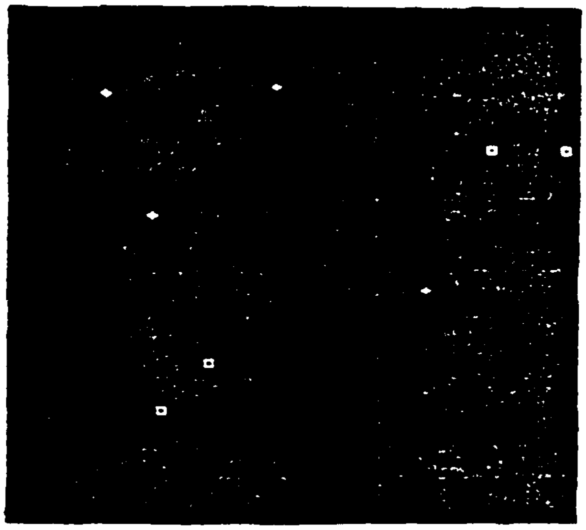
We have investigated six different algorithms (five linear ones and one nonlinear one) for their ability to detect small targets in slowly moving backgrounds. Analysis and simulation results are presented. Overall, parabolic interpolated differencing seems to outperform all others and is suitable for parallel real-time realizations.

(a) Single Differencing

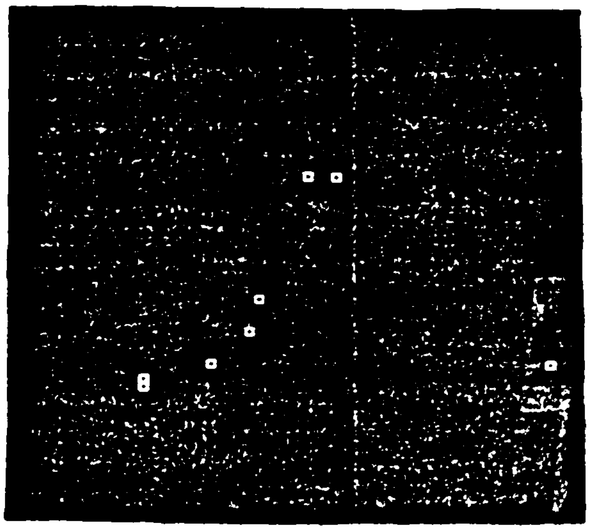
(b) Double Differencing



(c) Linear Interpolated Differencing



(d) Parabolic Interpolated Differencing



(e) Spatial Differencing

Figure 2-5: The thresholded outputs for the Five Al[illegible] [illegible] 0.75. Crosses denote correct targets and squares denote false peaks.

References

1. N. Ben-Yosef, B. Rahat, and G. Feigin, "Simulation of IR images of Natural Backgrounds", *Applied Optics*,Vol. 22No. 1January 1983, pp. 190-193.

2. A.T. Maksymowicz, R.A. Bankus, W.W.Davis, V.J. Pecora and L.H. Wald, "Statistical Modeling of Scene Variability", *Real Time Signal Processing IV*, Proc. SPIE Vol. 304, August 1981.

3. C. David Wang, "Adaptive Spatial/Temporal/Spectral Filters for Background Clutter Suppression and Target Detection", *Optical Engineering*,Vol. 21December 1982, pp. 1033-1038.

4. B.V.K. Vijaya Kumar, "Error in Optical Computation of Correlation Coefficients", *Applied Optics*,Vol. 22No. 2January 1983, pp. 209-211.

5. B.V.K. Vijaya Kumar, D. Casasent and A. Goutzoulis, "Fine Delay Estimation with Time Integrating Correlators", *Applied Optics*,Vol. 21No. 21November 1982, pp. 3855-3863.

6. B.V.K. Vijaya Kumar and Srikanth Rajan, "Subpixel Delay Estimation Using Group Delay Functions", *Real-Time Signal Processing VII*, Proc. SPIE Vol. 697, 1986.

7. B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application Stereo Vision", *Proceeding of the Workshop on Image Understanding*, IEEE Press, 1981.

8. Q.Tian, and M.N. Huhns, "Algorithms for Subpixel Registration", *Computer Vision, Graphics, and Image Processing*,Vol. 35 1986, pp. 220-233.

9. R. Voles, "Interpolating Sampled Cross-Correlation Surfaces of Images for Fine Registration", *IEE PROC.*,Vol. 127No. 5March 1980, pp. 401-404.

10. T.J. Patterson, D.M. Chabries, and R.W. Christiansen, "Image Processing for Target Detection Using Data from A Staring Mosaic IR Sensor in Geosynchronous Orbit", *Optical Engineering*,Vol. 25January 1986, pp. 166-172.

Chapter 3

Subpixel Delay Estimation Using Group Delay Functions

3.1. Introduction

Time Delay Estimation (TDE) has received much recent attention as evidenced by a journal special issue [1] devoted to this topic. TDE involves the determination of time delays between signals or coordinate shifts between images and finds applications in many diverse areas including Radar and Sonar signal processing and image sequence processing. In image sequence processing, the typical objective is to extract the desired information from a sequence of snapshots of a dynamic scene. An important application of image sequence processing is the extraction of target tracks in the presence of clutter in staring mosaic sensor imagery [2,3]. Another application involves the registration of two successive images [4] prior to their subtraction in Digital Subtraction Angiography.

An important feature of the above delay estimation problems is that the shifts to be estimated are typically *subpixel*, i.e., the delay between the two observed digital signals may not correspond to an integer multiple of the sampling interval. Accurate determination of this subpixel shift is essential for

proper alignment leading to good background suppression. A popular approach to subpixel delay estimation is based on fitting a second-order polynomial to the cross-correlation between the two digital signals. The coefficients of the second-order polynomial are then used to obtain an estimate of the subpixel delay. This method uses the quadratic interpolation inherently and hence its delay estimation accuracy is limited by the accuracy of the second-order interpolator [5,6]. The bias and variance associated with such an estimate were analyzed earlier [7]. In this chapter, we investigate the use of recently proposed [8] *Group Delay Functions* (GDFs) for the subpixel delay estimation problem.

The GDF of a signal can be viewed as the derivative of its Fourier phase with respect to frequency. We will show in the next section that two types of GDFs can be defined. $G_p(\omega)$ is the GDF based on the Fourier phase and $G_m(\omega)$ is the GDF based on the Fourier magnitude. The GDF representation allows us to consider the phase and the magnitude of the Fourier transform on an *equal* basis. $G_p(\omega)$ encodes the delay information whereas $G_m(\omega)$ contains the Signal-to-Noise Ratio (SNR) information. We present in this chapter an *adaptive* subpixel delay estimation technique using these features. This adaptive procedure weights the delay estimates from the high-SNR regions more heavily than those from the low-SNR regions.

After defining the GDFs in the next section, we outline the procedure for their computation in Section 3.3. This is followed by our presentation of the adaptive delay estimation algorithm in Section 3.4. A comparison of the

computational effort of the proposed method with that of the cross-correlation based method is provided in Section 3.5. Simulation results are then presented in Section 3.6 to illustrate the adaptive delay estimation algorithm and concluding remarks are provided in the last section.

3.2. Definition of GDFs

A discrete-time signal $x(n)$ can be represented in many different ways. One of the most popular methods uses the Discrete Time Fourier Transform (DTFT) [9] defined as below.

$$\begin{aligned} X(\omega) &= \sum_{n=-\infty}^{\infty} x(n) exp[-jn\omega] \\ &= |X(\omega)| \ exp[-j\theta(\omega)] \end{aligned} \tag{3.1}$$

where $|X(\omega)|$ and $\theta(\omega)$ are the magnitude and the phase of the DTFT, respectively. It is very easy to verify that $X(\omega)$, $|X(\omega)|$ and $\theta(\omega)$ are all *periodic* functions in ω with a period of 2π. An alternate description of $x(n)$ is through its GDFs $G_p(\omega)$ and $G_m(\omega)$. The phase-based GDF $G_p(\omega)$ is simply the derivative of the phase function $\theta(\omega)$ with respect to frequency ω. On the other hand, the magnitude-based GDF $G_m(\omega)$ is the derivative with respect to ω of the phase of a *minimum phase* signal whose Fourier transform magnitude equals $|X(\omega)|$. A minimum phase signal [9] has a z-transform with all its poles and zeroes inside the unit circle in the z-plane.

While the above definitions of GDFs are correct, certain subtleties are not apparent. Since the phase function $\theta(\omega)$ exhibits jumps of 2π, a *phase*

unwrapping procedure [10] must usually be carried out prior to the derivative operation in the definition of the GDFs. Let $\theta'(\omega)$ be the unwrapped phase corresponding to $\theta(\omega)$. Then the phase-based GDF is defined as

$$G_p(\omega) = - \frac{d\theta'(\omega)}{d\omega} . \tag{3.2}$$

Note that one can obtain the unwrapped phase $\theta'(\omega)$ completely (except for an additive constant) from $G_p(\omega)$ by integration. The definition of $G_m(\omega)$ is prompted by the result that for minimum phase signals, $\ln|X(\omega)|$ and $\theta(\omega)$ form a Hilbert transform pair [9]. Let $\beta'(\omega)$ be the unwrapped phase function of a minimum phase signal with magnitude transform $|X(\omega)|$. Then

$$G_m(\omega) = - \frac{d\beta'(\omega)}{d\omega} . \tag{3.3}$$

If the signal $x(n)$ is a minimum phase signal, then $G_p(\omega)$ and $G_m(\omega)$ are *identical*. For a maximum phase signal (all poles and zeroes outside the unit circle), $G_p(\omega)$ and $G_m(\omega)$ are negatives of each other. But for a general signal, no simple relations exist between $G_m(\omega)$ and $G_p(\omega)$. If $G_m(\omega)$ is known, then $\beta'(\omega)$ can be obtained except for an additive constant. Then the Hilbert transform relations [9] can be used to determine $|X(\omega)|$ except for a multiplicative constant. Thus $X(\omega)$ can be reconstructed completely (except for a multiplicative complex scalar) from the knowledge of the two GDFs. Of course, the two GDFs do not contain any more information than is available in the conventional Fourier transform-based representation. The GDFs simply represent another possible method of describing the signal, and seem to be of potential benefit to the delay estimation problem.

Before we consider the computational procedures for the GDFs in the next section, we introduce the GDFs for two-dimensional signals. The 2-D DTFT of a 2-D signal $x(n_1,n_2)$ can be characterized by the Fourier Transform (FT) magnitude function $|X(\omega_1,\omega_2)|$ and the FT phase function $\theta(\omega_1,\omega_2)$. Then two phase-based GDFs are defined as below.

$$G_{p1}(\omega_1,\omega_2) = - \frac{\partial\theta'(\omega_1,\omega_2)}{\partial\omega_1} . \tag{3.4}$$

and

$$G_{p2}(\omega_1,\omega_2) = - \frac{\partial\theta'(\omega_1,\omega_2)}{\partial\omega_2} . \tag{3.5}$$

Similarly, two magnitude-based GDFs $G_{m1}(\omega_1,\omega_2)$ and $G_{m2}(\omega_1,\omega_2)$ can be defined as the appropriate partial derivatives with respect to ω_1 and ω_2 of $\beta'(\omega_1,\omega_2)$. Thus we have four GDFs associated with the 2-D signal $x(n_1,n_2)$. Throughout the rest of this chapter, we confine our attention only to 1-D signals for the sake of simplicity. Our conclusions can be easily generalized to the 2-D case.

3.3. GDF Computation

The GDF of a sequence $x(n)$ can be obtained by taking the derivative of its Fourier phase. In practice, this approach suffers from two major problems. The first is that it requires the computationally cumbersome phase-unwrapping. The second problem is that the derivative operation can only be approximated on a digital computer. We can circumvent these problems by exploiting the

relationships between the GDFs and cepstral coefficients [11]. Since $\ln|X(\omega)|$ and $\theta'(\omega)$ are periodic in ω with a period of 2π, their Fourier series decomposition can be written as below.

$$\ln|X(\omega)| = \sum_{k=-\infty}^{\infty} a_m(k)e^{jk\omega}, \tag{3.6}$$

and

$$j\theta'(\omega) = \sum_{k=-\infty}^{\infty} a_p(k)e^{-jk\omega}, \tag{3.7}$$

where $a_m(k)$ and $a_p(k)$ are the Fourier series coefficients.

If the signal $x(n)$ is real, then $\ln|X(\omega)|$ is *even* in ω and $\theta'(\omega)$ is *odd* in ω. Thus Equations (3.6) and (3.7) can be rewritten as

$$\ln|X(\omega)| = a_m(0) + 2\sum_{k=1}^{\infty} a_m(k)cos(\omega k), \tag{3.8}$$

and

$$\theta'(\omega) = -2\sum_{k=1}^{\infty} a_p(k)sin(\omega k). \tag{3.9}$$

Substituting Equation (3.9) in Equation (3.2), we obtain

$$G_p(\omega) = 2\sum_{k=1}^{\infty} ka_p(k)cos(\omega k). \tag{3.10}$$

For the log magnitude function in (3.8), the minimum phase equivalent phase is given by

$$\theta'(\omega) = -2 \sum_{k=1}^{\infty} a_m(k) sin(\omega k). \tag{3.11}$$

and substituting Equation (3.11) in Equation (3.3), we obtain the following expression for $G_m(\omega)$.

$$G_m(\omega) = 2 \sum_{k=1}^{\infty} k a_m(k) cos(\omega k). \tag{3.12}$$

Above equations provide the basis for the method to compute the GDFs of a given signal $x(n)$. To determine $G_m(\omega)$, we first obtain $\ln|X(\omega)|$ from $x(n)$ by Fourier transform. Then the Fourier series coefficients $a_m(k)$ of the periodic function $\ln|X(\omega)|$ are obtained through inverse Fourier transform. These coefficients are then multiplied by k and the resulting sequence is Fourier transformed to yield the desired $G_m(\omega)$. The computation of $G_p(\omega)$ also proceeds in a similar manner. The phase function $\theta'(\omega)$ is first obtained from the Fourier transform of $x(n)$. Then an inverse Fourier transform is carried out on $j\theta'(\omega)$ to obtain the coefficients $a_p(k)$. These are then multiplied by k and the resulting sequence is Fourier transformed as in Equation (3.10) to yield $G_p(\omega)$. The coefficients $a_m(k)$ and $a_p(k)$ are obtained by the inverse Fourier transform of the logarithm of $X(\omega)$. These are known as *cepstral* coefficients. Thus $G_p(\omega)$ and $G_m(\omega)$ can be obtained from the complex cepstral coefficients of $x(n)$.

The various Fourier transform operations indicated can be carried out using the Fast Fourier Transform (FFT) algorithms. However, we must remember that the cepstrum of a finite-duration signal is not usually of finite

duration. Thus, certain amount of aliasing will always be present. This aliasing can be reduced by increasing the FFT size at the expense of increasing computational complexity. The complete computational algorithm for $G_p(\omega)$ and $G_m(\omega)$ was presented elsewhere [8].

3.4. Adaptive Delay Estimation

To understand how GDFs can be used for sub-pixel delay estimation, let us consider the GDFs of two signals $x_1(n)$ and $x_2(n)$, where

$$x_2(n) = x_1(n-n_0) \tag{3.13}$$

and n_0 is the delay to be estimated. Strictly speaking, n_0 must be an integer for Equation (3.13) to be valid. But we will allow n_0 to be non-integer and interpret Equation (3.13) as below. Let $x_{1a}(t)$ and $x_{2a}(t)$ be two continuous-time signals related through

$$x_{2a}(t) = x_{1a}(t-t_0), \tag{3.14}$$

where t_0 is the time delay between the two continuous signals. Assume that $x_1(n)$ and $x_2(n)$ are obtained from $x_{1a}(t)$ and $x_{2a}(t)$ by uniformly sampling them at intervals of Δt. Then $n_0=(t_0/\Delta t)$ can be an non-integer. With this interpretation, the DTFTs are related as below.

$$X_2(\omega) = X_1(\omega)exp\{-j\omega n_0\}. \tag{3.15}$$

This indicates that $|X_1(\omega)|$ and $|X_2(\omega)|$ are equal, resulting in identical magnitude-based GDFs. Thus the $G_m(\omega)$ does not carry the delay information. On the other hand, the phase-based GDFs contain the delay information. To

illustrate this, let $\theta^1(\omega)$ and $\theta^2(\omega)$ denote the phase of $X_1(\omega)$ and $X_2(\omega)$, respectively. From Equation (3.15), we have

$$\theta^2(\omega) = \theta^1(\omega) - \omega n_0. \tag{3.16}$$

Remembering that $G_p(\omega)$ is the negative *derivative of the phase function* with respect to ω, we obtain the important result that

$$G_p^2(\omega) = G_p^1(\omega) + n_0, \tag{3.17}$$

where $G_p^1(\omega)$ and $G_p^2(\omega)$ are the phase-based GDFs for $x_1(n)$ and $x_2(n)$, respectively. Thus the delay n_0 between $x_1(n)$ and $x_2(n)$ can be obtained as the constant difference between their phase-based GDFs. In practice, the difference between $G_p^2(\omega)$ and $G_p^1(\omega)$ will not exactly be a constant and n_0 can be estimated by the following average.

$$n_0^e = \frac{1}{2\pi} \int_{-\pi}^{\pi} [G_p^2(\omega) - G_p^1(\omega)] d\omega, \tag{3.18}$$

where we used the fact that the GDFs are periodic in ω with a period of 2π. Another departure from ideality in practice is that we can only compute $G_p(l)$ and $G_m(l)$ which are *sampled* versions of $G_p(\omega)$ and $G_m(\omega)$, respectively. Then the integral in Equation (3.18) can be replaced by a summation and realizing that $G_p(\omega)$ in Equation (3.10) and $G_m(\omega)$ in Equation (3.12) are *even* functions of ω, we can estimate the delay n_0 as below.

$$n_0^e = \frac{1}{1+N/2} \sum_{l=0}^{(N/2)} [G_p^2(l) - G_p^1(l)], \tag{3.19}$$

where N denotes the size of the FFT.

An interesting feature of the proposed method is that the delay estimate is obtained as the averaged difference between the GDFs of the two signals. Since the GDFs can take on a continuum of values, the n_0^e in Equation (3.19) can take on all possible values including non-integer values. Thus, this method treats both integer and non-integer shifts on an equal basis and requires no special effort to obtain subpixel delays. On the other hand, the cross-correlation based methods require the interpolation of the digital cross-correlation function to obtain the subpixel delays. The accuracy of the interpolator thus determines the accuracy of the delay estimates.

The delay estimates discussed so far were based only on the phase-based GDFs. We now explore how $G_m(\omega)$ can help the estimation procedure. To see this, let us assume that the delay estimation problem is to estimate n_0 from $x(n)$ and $y(n)$, where

$$y(n) \;=\; x(n-n_0) \;+\; w(n). \tag{3.20}$$

In the above equation, $w(n)$ is a sample realization of a random process with zero mean and represents the noise always present in any measurement. When $w(n)$ is zero for all n (no noise), the situation is the same as discussed before and $G_m(\omega)$ does not help the estimation procedure in Equation (3.19). But the magnitude-based GDFs can be used to advantage in the presence of noise.

It has been shown elsewhere [12] that the negative derivative of the phase spectrum and the Fourier magnitude are very similar to each other for a minimum phase signal. Recalling that $G_m(\omega)$ is indeed the negative derivative of

the phase spectrum of a minimum phase signal with magnitude spectrum $|X(\omega)|$, we see that $G_m(\omega)$ tracks the FT magnitude. Let $G_m^x(\omega)$ and $G_m^y(\omega)$ denote the magnitude-based GDFs of $x(n)$ and $y(n)$, respectively. Then the following can be written.

$$G_m^x(\omega) \approx |X(\omega)| \quad , \tag{3.21}$$

and

$$G_m^y(\omega) \approx |Y(\omega)| \approx |X(\omega)| + \sqrt{P(\omega)}, \tag{3.22}$$

where $P(\omega)$ is the spectral density [13] of the random process $w(n)$. In writing Equation (3.22), we assumed that the noise process is independent of the signal $x(n)$. From Equations (3.21) and (3.22), a crude estimate of the SNR can be obtained in terms of the magnitude-based GDFs as below.

$$\text{SNR} = \frac{|X(\omega)|}{\sqrt{P(\omega)}} \approx \frac{G_m^x(\omega)}{G_m^y(\omega) - G_m^x(\omega)} \quad . \tag{3.23}$$

Thus the delay between the signals $x(n)$ and $y(n)$ can be estimated as below.

$$n_0^e = \sum_{l=0}^{N/2} \{ [G_p^y(l) - G_p^x(l)] \, T\left[\frac{G_m^x(l)}{G_m^y(l) - G_m^x(l)}\right] \} \Bigg/ \sum_{l=0}^{N/2} T\left[\frac{G_m^x(l)}{G_m^y(l) - G_m^x(l)}\right]$$

where $T[.]$ is a *weighting function* designed to emphasize the high-SNR regions. Of course, we also require that the denominator in Equation (3.24) be non-zero.

A good choice for the weighting function $T[\,]$ is the single *threshold function*

$$T[\,\mathrm{SNR}\,] = \begin{cases} 1 & \text{if SNR} \geq T_0 \\ 0 & \text{if SNR} < T_0 \end{cases} \tag{3.25}$$

where T_0 is a preselected threshold value. With this weighting function, n_0^e in Equation (3.24) can be seen to a simple average of the phase-based GDF difference *only* over those frequencies for which the SNR exceeds T_0. If T_0 is chosen too small, then essentially all frequencies will be considered leading to a delay estimate similar to that in Equation (3.19). On the other hand, if T_0 is chosen too high, then very few frequencies will contribute to n_0^e and the resulting estimate may be inaccurate due to insufficient averaging. Thus the choice of T_0 is important. In practice, one may want to obtain a sequence of n_0^e estimates based on a sequence of choices for T_0 and select the most consistent n_0 from this sequence. We will discuss this issue some more when presenting the simulation results. Of course, other positive functions can also be used for the weighting function $T[\,]$. A desirable property of this weighting function is that it takes on values close to 1 when its argument is large and takes on values close to 0 when the argument is small. The adaptive delay estimation of n_0 given the two signals $x(n)$ and $y(n)$ can be summarized as below.

Step 1: Determine $G_p^x(l)$, $G_m^x(l)$ from $x(n)$ and $G_p^y(l)$, $G_m^y(l)$ from $y(n)$ for l=0,1, . . . ,(N−1) according to the GDF computational procedure outlined in Section 3.3.

Step 2: For a preselected SNR threshold T_0, use $G_p^x(l)$, $G_m^x(l)$, $G_p^y(l)$ and $G_m^y(l)$ in Equations (3.24) and (3.25) to obtain the subpixel delay estimate n_0^e.

The proposed approach has two advantages over conventional methods to estimate subpixel shifts. The first is its natural ability to accommodate subpixel delays without any explicit interpolation. The second advantage is the ability to emphasize the high-SNR regions. We will show in the next section that the price paid for these two advantages is the increased computational complexity.

3.5. Computational Considerations

In this section, we compare the computational complexity of our proposed method with that of the conventional cross-correlation method [5]. Let us first consider the number of operations involved in estimating the delay n_0 between $x(n)$ and $y(n)$ using the adaptive delay estimation procedure. As discussed in Section 3.3, following operations are needed to compute $G_p^x(l)$ and $G_m^x(l)$ from $x(n)$.

(i) An N-point FFT to obtain $X(\omega)$.

(ii) Complex logarithm to obtain $\ln|X(\omega)| + j\theta(\omega)$.

(iii) An N-point inverse FFT to obtain the cepstral coefficients $a_m(k)$ and $a_p(k)$.

(iv) An N-point FFT of $ka_m(k)$ and $ka_p(k)$ to obtain the two GDFs.

The major computational load in the above steps belongs to the three N-point FFTs. Thus the number of operations needed to produce $G_p^x(l)$ and $G_m^x(l)$ from $x(n)$ is approximately $3N \log_2 N$ (assuming, of course, that N is an integer

power of 2). Looking back at the adaptive delay estimation procedure in the previous section, we note that the main computational burden is approximately $6N \log_2 N$ (for the computation of the GDFs of $x(n)$ and $y(n)$).

The cross-correlation based subpixel delay estimation method [5] first computes $c(k)$, the cross-correlation between $x(n)$ and $y(n)$. This correlation sequence $c(k)$ is then searched and its peak is located. The three correlation values centered at the peak are fitted with a quadratic polynomial and the subpixel delay is estimated from the coefficients of this polynomial. The main computational load of this method is the determination of the cross-correlation between $x(n)$ and $y(n)$. This can be obtained by taking the inverse FFT of the product of the FFTs of $x(n)$ and $y(n)$. This amounts to three N-point FFTs resulting in a computational load of $3N \log_2 N$ operations. We did not include the computations needed for locating the correlation peak in this analysis.

Our simple computational analysis seems to indicate that the adaptive delay estimation procedure requires roughly *twice* as many operations as the conventional cross-correlation based delay estimation. Offsetting this computational disadvantage are the two advantages mentioned earlier. These are (i) the natural manner in which subpixel delays are handled, and (ii) the ability to emphasize high-SNR regions.

3.6. Simulation Results

In this section, we present our initial simulation results to illustrate the functioning of the adaptive delay estimation algorithm. Towards this end, we selected the discrete-time linear frequency modulated (LFM) signal (known also as the chirp signal) as the reference signal. The mathematical expression for this is

$$x(n) = \sin[2\pi n(f_0 + an)], \quad 0 \leq n \leq (N_0-1) \tag{3.26}$$

where f_0 and a are the initial frequency and the chirp rate of the chirp signal and N_0 is its length. The spectrum of the above signal extends from f_0 to $f_0 + 2aN_0$ in digital frequencies. Since the sampling interval is 1, the parameters f_0, a and N_0 must be chosen such that the spectrum does not extend beyond a digital frequency of 0.5. Otherwise, aliasing would result. We chose $f_0 = 0.05$, $a = 0.0005$ and $N_0 = 200$. For this choice, the spectrum is nonzero in the interval (0.05, 0.25), well within the allowed limits. The chirp signal was selected because of the simplicity with which it can be generated and with which subpixel delays can be simulated.

The second signal $y(n)$ is obtained by delaying $x(n)$ in Equation (3.26) by a desired amount n_0 and adding a sample noise realization $w(n)$ to it. The random noise is assumed to be of zero mean and variance σ^2. We also consider only Gaussian noise. Two different types of noise correlations are considered. The first is the *white* noise which has successive noise samples that are statistically independent. This white noise is generated by standard random number

generators [14]. The second type of noise is the *colored* noise $z(n)$ obtained by passing the white noise $w(n)$ of unit variance through a digital filter. We selected the following first-order recursive digital filter.

$$z(n) = \rho\ z(n-1) + \sqrt{1-\rho^2}\ w(n) \quad . \tag{3.27}$$

where $-1 \leq \rho \leq 1$ denotes the correlation coefficient between adjacent samples in the correlated noise. It can be seen from Equation (3.27) that $\rho=0$ yields white noise. Increasing ρ values correspond to increasingly low pass spectra for colored noise. Since $w(n)$ is of unit variance, it is easy to show that $z(n)$ is also of unit variance. In our simulation we chose ρ =0.8.

For the first simulation study, we employed $n_0 = 20$, $N_0 = 200$ and $N = 512$. Recalling that N denotes the FFT size used for GDF computation, we note that the signals are padded with the appropriate number of zeroes to yield length N. We show in Table 3.1 the delay estimates for various choices of threshold T_0 and for various *white* input noise variances. Since the chirp signal is of length 200 and amplitude 1, its energy is approximately 100. Noise of variance σ^2 and length N results in noise energy of $N\sigma^2$. Thus the input SNR in dB is given by

$$\text{SNR}_{\text{I}} = 10 \log_{10} \left(\frac{100}{512\sigma^2}\right) \tag{3.28}$$

Thus a noise variance of 10^{-6} corresponds to approximately 53 dB input SNR. From Table 3.1, we see the obvious trend that the estimation accuracy decreases as the input noise variance increases. We also see from this table that the choice of the threshold affects the obtained accuracies. While no firm relation seems to

exist between the optimum T_0 and the input SNR, we can observe the general trend that lower input SNR values require lower threshold values. This makes sense because by setting a high threshold value for low input SNR case, we will be including very few frequencies in the estimation process. Thus when we are sure that the input noise is very low, we can use a high threshold value. Note from Table 3.1 that the possibility of obtaining very accurate estimates (19.971 instead of 20) exists for even large input noise variances (10^{-3}) if a proper threshold (60) is chosen. Of course, selection of proper threshold depends on *a priori* knowledge of the input SNR.

The above experiment was repeated next with identical parameters except that a *colored* noise with $\rho = 0.8$ was used in place of white noise. The corresponding results are shown in Table 3.2. Note once again that increasing noise variance leads to generally poorer delay estimates. We also note the general trend that lower threshold values are better for lower input SNRs. We observe very good estimation accuracy once again (19.617 instead of 20)even for large input noise variance with proper threshold (35) selection. Comparing the results in Tables 3.1 and 3.2, we see that the estimation accuracies seem to be somewhat better for colored noise case than for white noise case. This may be a consequence of the ability of the weighting function to emphasize the high SNR regions available when the noise is of low pass type.

The two simulation examples considered so far used n_0=20, an integer delay. To demonstrate the subpixel delay estimation capability of the proposed

algorithm, we next simulated n_0=5.8. This is easily achieved by replacing n by (n-5.8) in the right hand side of the equality in Equation (3.26). The resulting estimates are shown in Table 3.3 for the case of white noise. While the results are not as good as expected, we see the ability of our algorithm to estimate the subpixel delays for high-SNR situations. The algorithm exhibits poor estimation when given low input SNR signals. But in all four cases, the best delay estimates (5.785, 5.773, 5.766 and 5.604) were very close to the correct value of 5.8. These results demonstrate the capability of the proposed adaptive delay estimation procedure to estimate the subpixel delay between two signals.

3.7. Conclusions

In this chapter, we have presented an adaptive delay estimation procedure using the Group Delay Functions of the signals. While it seems to be computationally twice as burdensome as the conventional cross-correlation based methods, this has two advantages. The first is the natural way it accommodates the fractional or subpixel delays. The next advantage is that the GDFs provide estimates of SNR as a function of frequency and thus delay estimates from high-SNR regions can be emphasized. We have provided initial simulation results to illustrate this idea.

DELAY ESTIMATES AS A FUNCTION OF WHITE NOISE VARIANCE

TRUE DELAY = 20

THRESHOLD T_0	NOISE VARIANCE			
	10^{-6} (1)	10^{-5} (2)	10^{-4} (3)	10^{-3} (4)
0	19.758	19.559	16.580	-0.697
5	19.710	19.869	20.601	15.320
10	19.859	20.007	20.429	19.553
20	19.901	20.067	20.485	18.945
50	19.895	20.222	19.854	18.982
100	19.896	20.192	19.725	ND
200	19.910	20.852	19.341	ND
500	19.458	20.040	ND	ND
1000	19.370	20.115	ND	ND
2000	19.028	ND	ND	ND
5000	ND	ND	ND	ND

ND: Nondeterminable because no terms existed above the threshold.

(1) The best delay estimate (20.007) occurred for a T_0 of 120.

(2) The best delay estimate (20.007) occurred for a T_0 of 10.

(3) The best delay estimate (19.980) occurred for a T_0 of 45.

(4) The best delay estimate (19.971) occurred for a T_0 of 60.

Table 3-1: Delay estimates as a function of white noise variance

DELAY ESTIMATES AS A FUNCTION OF COLORED NOISE VARIANCE

TRUE DELAY = 20

THRESHOLD T_0	NOISE VARIANCE			
	10^{-6} (1)	10^{-5} (2)	10^{-4} (3)	10^{-3} (4)
0	20.041	19.875	19.549	19.401
5	20.029	19.815	19.978	18.076
10	20.056	19.819	19.650	17.718
20	20.041	19.641	19.491	19.040
50	20.050	19.736	19.090	15.377
100	20.040	19.753	18.328	20.761
200	20.125	19.345	20.839	20.761
500	20.199	20.195	ND	ND
1000	20.006	20.195	ND	ND
2000	20.095	ND	ND	ND
5000	ND	ND	ND	ND

ND: Nondeterminable because the threshold T_0 is too high.

(1) The best delay estimate (20.004) occurred for a T_0 of 110.

(2) The best delay estimate (19.875) occurred for a T_0 of 0.

(3) The best delay estimate (19.978) occurred for a T_0 of 5.

(4) The best delay estimate (19.617) occurred for a T_0 of 35.

Table 3-2: Delay estimates as a function of colored noise variance

SUBPIXEL DELAY ESTIMATES IN WHITE NOISE

TRUE DELAY = 5.8

THRESHOLD T_0	NOISE VARIANCE			
	10^{-6} (1)	10^{-5} (2)	10^{-4} (3)	10^{-3} (4)
0	5.731	5.664	2.600	-22.738
5	5.424	5.429	5.766	- 2.160
10	5.837	5.491	5.158	0.657
20	5.868	5.773	7.085	1.932
50	6.527	5.333	6.237	3.938
100	6.955	4.112	6.830	3.938
200	7.621	2.949	ND	3.938
500	5.909	6.036	ND	ND
1000	ND	ND	ND	ND
2000	ND	ND	ND	ND
5000	ND	ND	ND	ND

ND: Nondeterminable due to T_0 being too high.

(1) The best delay estimate (5.785) occurred for a T_0 of 15.

(2) The best delay estimate (5.773) occurred for a T_0 of 20.

(3) The best delay estimate (5.766) occurred for a T_0 of 10.

(4) The best delay estimate (5.604) occurred for a T_0 of 15.

Table 3-3: Subpixel delay estimates in white noise

References

1. Special Issue on Time Delay Estimation, *IEEE Trans. on Acou. Sp. Sig. Proc.*, Vol. 29, No. 3 (1981).

2. H.E. Rauch, W. I. Futterman and D. B. Kemmer, "Background Suppression and Tracking with a Staring Mosaic Sensor," in *Modern Utilization of Infrared Technology*, Proc. of SPIE, Vol. 197, 19 (1979).

3. T. J. Patterson, D. M. Chabries and R. M. Christiansen, "Image Processing for Target Detection Using Data from a Staring Mosaic Infrared Sensor in the Geosynchronous Orbit," *Optical Engineering*, Vol. 25, 166 (1986).

4. A. Venot and V. Leclerc, "Automated Correction of Patient Motion and Gray Values Prior to Subtraction in Digitized Angiography," *IEEE Trans. on Medical Imaging*, Vol. 3, 179 (1984).

5. R. Voles, "Interpolating Sampled Cross-Correlation Surfaces of images for Fine Registration," *IEE Proceedings*, Vol. 127, 401 (1980).

6. R. E. Boucher and J. C. Hassab, "Analysis of Discrete Implementation of Generalized Cross-Correlators," *IEEE Trans. on Acou. Sp. Sig. Proc.*, Vol. 29, 609 (1984).

7. B.V.K. Vijaya Kumar, D. Casasent and A. Goutzoulis, "Fine Delay Estimation with Time Integrating Correlators," *Applied Optics*, Vol. 21, 3855 (1982).

8. B. Yegnanarayana, D. K. Saikia and T. R. Krishnan, "Significance of Group Delay Functions in Signal Reconstruction from Spectral Magnitude or Phase," *IEEE Trans. on Acou. Sp. Sig. Proc.*, Vol. 32, 610 (1984).

9. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey (1975).

10. J. M. Tribolet, "A new Phase Unwrapping Algorithm," *IEEE Trans. on Acou. Sp. Sig. Proc.*, Vol. 25, 170 (1977).

11. D. G. Childers, D. P. Skinner and R. C. Kemerait, "The Cepstrum: A Guide to Processing," *Proc. of IEEE*, Vol. 65, 1428 (1977).

12. B. Yegnanarayana, "Formant Extraction from Linear-Prediction Phase Spectra," *Journ. of Acou. Soc. Amer.*, Vol 63, 1638 (1978).

13. A. Papoulis, *Probability, Random Variables and Stochastic processes*, McGraw-Hill, New York (1965).

14. International Mathematical and Statistical Library (IMSL), 7500 Bellaire Blvd., Houston, Texas 77036.

Chapter 4

Detection of Target Trajectories Using the Hough Transform

4.1. Introduction

The Hough transform [hough] (HT) was originally proposed as a means of detecting straight lines in the input image. The transformation can be easily implemented both digitally [Duda, Merlin] and optically. [Eichman, Gindi] It has been later modified so that it can be used to locate objects of other shapes (e.g. circles, [Kimme] ellipses [Tsuji] and parabolas. [Wechsler] The so-called generalized Hough transform can theoretically handle objects of any shape i.e., objects of both analytical and non-analytical shapes. [Ballard] However, these techniques require much pre-processing such as gradient detection and edge-following. They also require that the list of the positions and orientations of the small segments that compose the object be stored. Moreover, the generalized Hough transforms require that one deal with problems of high dimensionality, which makes peak detection very difficult. [Brown]

The Hough transform and the generalized Hough transform have also been previously used for the detection of moving target tracks. [Cowart, Padgett] These applications, however, also suffer from the drawbacks mentioned above.

One way of overcoming some of these problems is to consider only subspaces of the multi-dimensional Hough space. [Ballard83] One can also make use of some of the properties of the particular type of curve to mitigate the problem. [vision] Yet another approach [Krishna1] uses certain transformations in the 2-D Hough space to achieve dimensionality reduction. It has also been shown that this method can be very efficiently extended for the detection of curved objects of any arbitrary shape. [Casasent1] In this chapter, we show that this particular approach can be used as a new technique for the detection of moving target tracks.

In Section 4.2, we review the straight-line HT and simple transformations that one can apply to this space to describe input translations and rotations. We also provide the general theoretical basis for our new approach to the detection of target tracks. Section 4.3 describes how missile trajectories can be parameterized so that our technique can be applied for their detection. Section 4.4 provides the experimental results of the performance of our technique on simulated missile target track images.

4.2. Use of HT for Locating Maneuvering Object Tracks

As has been pointed out, [Cowart, Padgett] third-order differencing can produce reasonably good track images of a moving target. Aircraft and missiles (in certain phases) very often follow straight-line trajectories. In other words, if the target is a non-maneuvering type, then the track can quite often be approximated by a set of straight-line segments. Moreover, multiple targets

produce multiple straight lines in the track image, which can be easily detected by a simple Hough transform. The position and heading can also be accurately determined. [Cowart] It has been suggested [Cowart] and illustrated [Padgett] that the same technique can be extended to the detection of maneuvering (non-straight-line) tracks. These present techniques for curved tracks, however, suffer from the disadvantages pointed out in Section 4.1. An alternative way of parameterizing the Hough space in order to overcome some of these problems is presented in this section. In what follows, we assume that the target trajectory can be described by a second-degree curve. It must be noted, however, that the same technique is valid for more general trajectories. [Casasent1]

The HT maps the points (x,y) in the input image to a sinusoid in the transformed (θ,p) domain ($p \leq 0$, $0 \leq \theta < 2\pi$) given by

$$p = x\cos\theta + y\sin\theta. \tag{4.1}$$

The above equation can also be written as

$$p = (x^2+y^2)^{1/2}\cos(\theta-\tan^{-1}y/x). \tag{4.2}$$

Let the trajectory of the target in the reference position be described by the second-degree equation

$$y^2 + ax^2 + bx + cy + d = 0. \tag{4.3}$$

The reference position is usually chosen such that the curve is symmetric with respect to the origin (if possible). The values of the parameters of the curve (i.e., a,b,c and d) are determined by the reference position and the type of curve. For

example, if the curve is a circle centered at the origin, then $a=1$, $b=c=0$ and $d=-r^2$, where r is the radius of the circle. The equation of the curve can also be expressed in terms of the perpendicular diatance p from the origin to the tangent of the curve at any point (x,y), and the angle θ the perpendicular makes with the x-axis. (See Figure 4.1). We thus write (4.3) as

$$p = T(a,b,c,d,\theta) \qquad \text{or} \qquad p - T(a,b,c,d,\theta) = 0. \tag{4.4}$$

where the function T describes the curve and its parameters.

It can be shown that this description of the curve can be obtained by Hough transforming the input image and thresholding it. [Casasent1] Thus, the thresholded Hough space of the curve can be described by (4.4). Given this description of the Hough space $H(\theta,p)$ of the curve, it can be shown [Casasent1] that if the curve is rotated by an angle ϕ about the origin and then translated to a new origin point (x_0,y_0), the resulting Hough space $H(\theta',p')$ (after thresholding) can be described either by

$$p' = t\cos(\theta'-\alpha) + T(a,b,c,d,\theta'-\phi)$$

or by

$$\begin{aligned} p' &= -t\cos(\theta'-\alpha-\pi) - T(a,b,c,d,\theta'-\phi-\pi) \\ &= t\cos(\theta'-\alpha) - T(a,b,c,d,\theta'-\phi-\pi), \end{aligned} \tag{4.5}$$

where

$$t = (x_0^2+y_0^2)^{1/2}; \quad \alpha = \tan^{-1}(y_0/x_0). \tag{4.6}$$

We choose the equation in (4.5) that gives a positive value for p'.

If a rotated and translated version of the curve in (4.3) appears as the input image trajectory of a target, then the Hough space of the trajectory is described by (4.5). It can be shown [Casasent1] that at least one of following transformations

$$p = p' - T(a,b,c,d,\theta'-\phi)$$

$$p = p' + T(a,b,c,d,\theta'-\phi-\pi) \tag{4.7}$$

from the $H(\theta',p')$ to a new $H(\theta,p)$ space will yield a sinusoidal pattern in this new $H(\theta,p)$ space. The inverse Hough transform of this new $H(\theta,p)$ space (with a sinusoidal HT plane pattern) will then give a peak at (x_0,y_0) in the inverse HT space. (If both equations in (4.5) give positive values for p', then both transformations in (4.7) will yield a sinusoidal pattern in the new Hough space. In that case, we apply both the transformations in (4.7) simultaneously to the Hough space, add the transformed spaces together, and invert the resulting HT space). This determines the (x_0,y_0) parameters for the curve. The parameters (a,b,c,d) of the transformation in (4.7) that yield a sinusoid in the new $H(\theta,p)$ space (or a peak in the inverse HT space) define the parameters of the input curve. The transformations in (4.7) are easily achieved [Krishna1, Casasent1] by shifting the Hough space along the p-axis with the amount of shift being, in general, a function of θ. If the values of a,b,c,d and ϕ are known, the transformation in (4.7) can be applied and the location of the trajectory is easily determined by searching for peaks in the inverse HT space.

If the curve parameter values are not known, then the transformations and

inversions are carried out for several values of a,b,c,d and ϕ over the range of expected values for ϕ. Organized search procedures for this have been detailed. [Krishna1, Casasent1] The values of ϕ and other curve parameters that give the best peak in the inverse Hough space are taken to be the input curve parameters. The height of the peak in the inverse Hough space (compared to a threshold) determines if the curve is present. It is to be noted that the transformations are completely specified by the thresholded HT of the trajectory $T(a,b,c,d,\theta)$ in the reference position. If several translated trajectories with the same parameters are present in the input, then these would appear as several peaks in the inverse Hough space at locations corresponding to the (x_0,y_0) parameters of the centers of each curve. The peaks will also occur in the inverse Hough space even if only parts of the trajectories are present. (The strengths of the peaks are proportional to the amount of each trajectory present in the input). Examples of this are provided in Section 4.4.

4.3. Parameterization of Missile Trajectories

As a specific case study, we consider the trajectories of ballistic missiles. These can be divided into several phases. The missile is initially expelled from its storage canister by a steam generator. The missile then ignites its first stage motor which burns out at an altitude of about 22 km. The second and third stage motors operate until the missile reaches an altitude of approximately 200 km at the end of the 3-minute boost phase. Since this boost phase part of the missile trajectory is short in comparison with the range of the missile ($\approx$10,000 km), it can be approximated by a straight line.

The second phase lasts from the end of boost phase until the missile reaches its apogee. During this time, the missile uses its thrusters to make small adjustments and after each adjustment, it releases a re-entry vehicle. These re-entry vehicles have different trajectories and travel independently to different destinations. All of these trajectories (as well as the missile's trajectory in this second phase) can be approximated by second-degree curves. If we know ϕ and the parameters of one or more of these second-degree curves, we can apply the transformation in (4.7) to the HT of the track image and then invert it to determine the existence (and the parameters and location) of that part of the trajectory. If ϕ and the curve parameters are not known, several educated guesses at these values are used and from the results in the inverse HT space we can determine the final parameters after several iterations. In practical situations, the range of values that the missile's parameters can take is limited by the various geometrical and aerodynamic constraints on the missile path. In addition, the detection of some parts of the trajectory places constraints on the parameters of the other parts of the trajectory. These facts are used to reduce the search space.

For example, we consider three common apogees used for ICBM trajectories: (i) a depressed trajectory which has an apogee of about 900 km and a re-entry angle of 15°, (ii) the normal minimum energy trajectory with an apogee of about 1200 km and a re-entry angle of 23° and (iii) a lofted trajectory with an apogee of 2300 km and a re-entry angle of 35°. The trajectory of the missile after the boost phase can be adequately modeled as a circle. In this case, the radius of the circle is determined by the type of apogee and re-entry angle.

The type of apogee itself is constrained by the parameters of the trajectory in the boost phase. In fact, the constraints may be such that we need to compute only a part of the HT and/or its inverse. This saves additional computational time and is a significant advantage and feature of a HT space.

4.4. Experimental Results

A computer program was used to simulate track images of missile trajectories. The program is capable of producing straight-line and second-degree curve images with any given translation, rotation and any desired curve parameters. The sampling rate of the pixels comprising the curve can also be changed. These 128×128 images were used to demonstrate the techniques in Sections 4.2 and 4.3. For all Hough spaces, a sampling interval of 1 was used for both p and θ and all peak values in the Hough space and the inverse Hough space were computed as the sum of the values in a 3×3 window.

Figure 4.2 shows data for three straight-line trajectories during the boost phase. Missiles are launched from right to left and the three tracks are at launch angles of 165°, 145° and 150° with respect to the positive x-axis. In order to demonstrate the effectiveness of the technique in the presence of breaks in the tracks, a sampling rate of 2 (i.e., every other pixel) was used for tracks 1 and 3. The central 145° track has a sampling rate of 1. Figure 4.2(b) shows the Hough transform of the original missile track images in Figure 4.2(a). Figure 4.2(c) shows the thresholded version of Figure 4.2(b) with the threshold set at 10. The number of points in a trajectory dictate the threshold chosen. The peaks in the

thresholded HT space yield the associated curve parameters. Three peaks are clearly visible in the final output pattern of Figure 4.2(c), corresponding to the three tracks in the image. The first track produced a peak with a strength of 69 at θ=75° and p=35 in the Hough space. The θ and p values agree with the theoretical values. The second track produced a peak with a strength of 137 at θ=54° and p=61. The theoretical values are θ=54° and p=61. This peak strength is approximately twice that of track 1, which agrees with the fact that there are twice as many pixel points or samples on track 2 as compared to track 1. The third track produced a peak with a strength of 66 at θ=60° and p=64. The theoretical values are θ=60° and p=64. The peak strength for this track 3 peak is about the same as that for track 1, as expected.

Figure 4.3(a) shows three circular trajectories corresponding to the second phase of flight with three different apogees. The radii of the tracks are 80, 100 and 120, their centers are located at (64,-20), (64,-55) and (64,-85) respectively and all tracks are approximately of equal length. Figure 4.3(b) shows the Hough transform of the input. It can be seen that we need to compute the Hough transform only for θ values between 0° and 180°. (The values in the HT between 180° and 360° are practically zero, because only the top parts of the circles were present in the input). Since the values of the radii of the trajectories can be predicted from the apogees and the re-entry angles, the values of the radii were assumed to be known. (The apogees and the re-entry angles are usually known in advance, as noted in the previous section). Thus, only three different values for the radii were tried in the transformation given by (4.7). In the case of a circle,

$T(a,b,c,d,\theta)$ is a straight horizontal line [Casasent1] at $p=r$ (where $r=(-d)^{1/2}$ is the radius of the circle) and is thus independent of θ. Therefore, in the case of a circle, (4.7) becomes

$$p = p' \mp T(r,\theta') = p' \mp r. \tag{4.8}$$

These results (Figures 4.2 and 4.3) clearly show the ability of the system to process and distinguish target tracks of different r and apogees. The transformations required in HT space involve shifting the Hough space vertically (uniformly for all θ) by a distance equal to the r being searched for. To carry out the transformation for several possibilities for r, we merely shift the HT by different amounts. The maximum value in the Hough space was 19 in Figure 4.3(b). The Hough space was thresholded at 10 and the transformations in (4.8) were applied for different values of r (i.e., r=80, 100 and 120). (The threshold in the Hough space is usually selected at about 50% of the expected peak value, but it is increased if noise is present). The results of the inverse HT processing are shown in Figures 4.3(c), 4.3(d) and 4.3(e). One dominant peak is observed in each of the inverse Hough spaces, indicating that the r value selected for that space was correct. The location of the peak is within one pixel of the actual (x_0,y_0) values. The strengths of the three dominant peaks are, as expected, approximately the same, since all three track lengths are roughly the same. The inverse space was computed only for $0 \leq x \leq 128$ and $-128 \leq y \leq 0$, since this is the range in which we expect the peaks to lie.

Figure 4.4(a) shows three circular arcs, slightly displaced with respect to

one another. These are typical of the tracks of three re-entry vehicles released by the same missile. All three tracks have the same radius(=80), but the centers are located at different points ((64,-20), (84,-20) and (124,-20)). Figure 4.4(b) shows the Hough transform of the image in Figure 4.4(a). The maximum value in the Hough space was 33. This is much higher than that of the previous case, because parts of all three curves lie on the same horizontal line. The Hough space was again thresholded at 10, the transformations in (4.8) were applied with r=80 and then inverse Hough transformed. The final result is shown in Figure 4.4(c). Three peaks are clearly visible and their (x_0, y_0) parameter locations are within 1 pixel of the actual values.

To see how the technique performs in the presence of noise, the image in Figure 4.4(a) was corrupted with noise, as shown in Figure 4.5(a). A Gaussian random noise generator was used to generate noise with a zero mean. The noise values were added to each pixel in the original binary image and the resulting image was rebinarized by thresholding at 0.5. Figure 4.5(a) shows the result when the variance of the noise was σ=0.3. (Noise with variances of σ=0.1 and σ=0.2 were used with no noticeable difference in the performance of the technique). Figure 4.5(b) shows the HT of the image in Figure 4.5(a). The maximum value in the Hough space was 44. Figure 4.5(c) shows the result of thresholding the HT at 15, (a threshold higher than 10 is used when noise is present), transforming as in (4.8) and inverse transforming. The transformation used in (4.8) was again a simple vertical shift and the amount of shift (= radius of circle) used was 80. Figure 4.5(d) shows the result of thresholding the output at 60. It can be seen

that the peaks are still discernible. The locations of the peaks are also unchanged. Thus, this technique appears to be quite robust in the presence of considerable noise.

4.5. Summary and Conclusions

A new technique for the detection of trajectories of targets has been presented. The technique involves a straight-line Hough transform (HT), thresholding and simple transformations in the Hough space, and an inverse HT. The transformations are easily achieved by merely shifting the Hough space along the p-axis. The amount of shift is in general a function of θ and is given by a control function $T(a,b,c,d,\theta)$ which is simply the thresholded HT of the curve in the reference position. This new technique circumvents the problems of storing the curve as a list of line segments along with their orientations and also problems of high dimensionality.

In the experimental results provided, the trajectory was assumed to be either a straight-line or a circle. It must be noted that the technique is much more general and is valid for any type of curve. If the trajectory consists of several piece-wise continuous curves, each part can be detected separately and the detection of one part used to place constraints on the parameters of the other parts, thus reducing the search space. Performance of the technique in the presence of noise was also demonstrated quite successfully.

List of Figures

(c)The Inverse HT space when a trajectory radius of 80 was used in the transformation.

(d)The Inverse Hough space thresholded at 60.

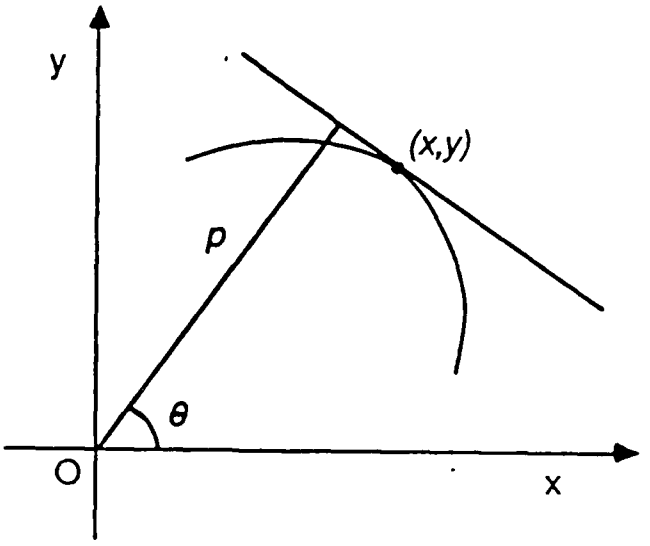


Figure 4-1

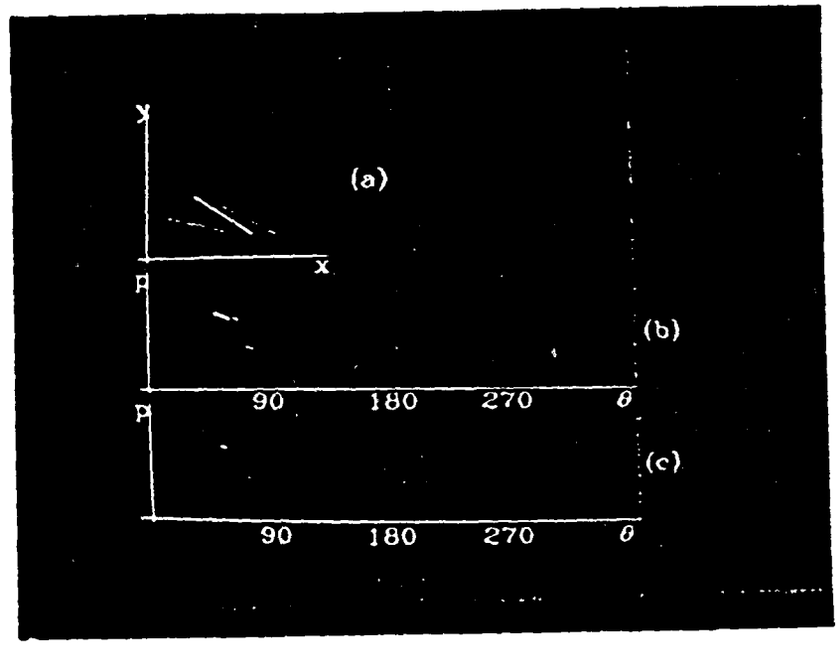


Figure 4-2

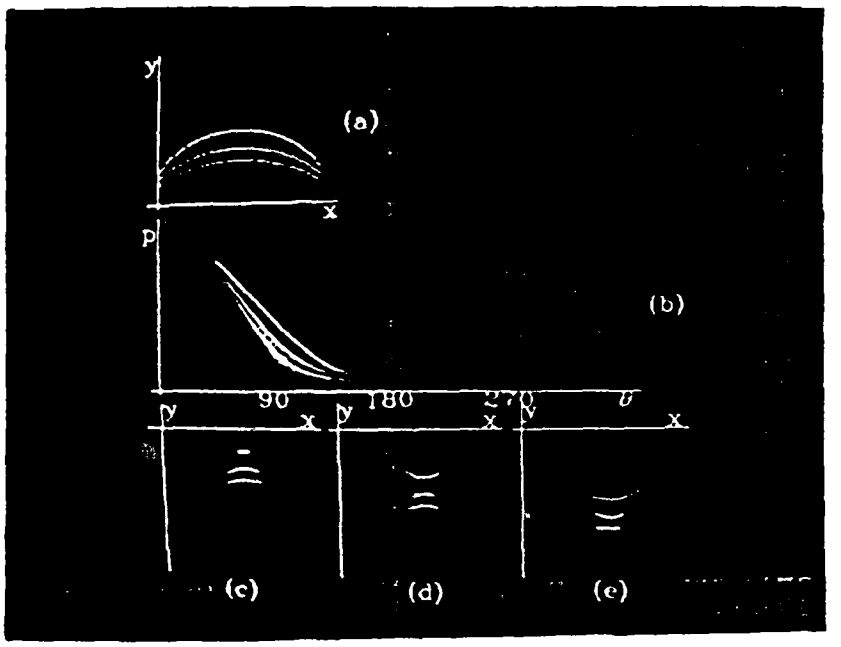
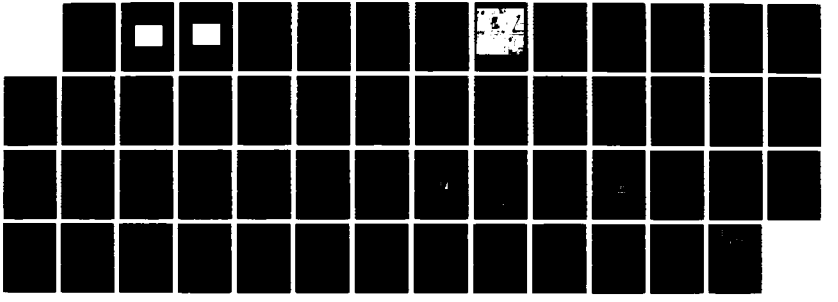
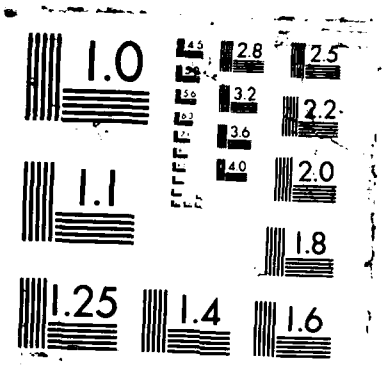


Figure 4-3

AD-A190 711 MULTI-DISCIPLINARY TECHNIQUES FOR UNDERSTANDING TIME-VARYING SPACE-BASED.. (U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF ELECTRICAL AND COM.. 2/2

UNCLASSIFIED D CASASENT ET AL. 29 APR 87 F/G 17/11 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

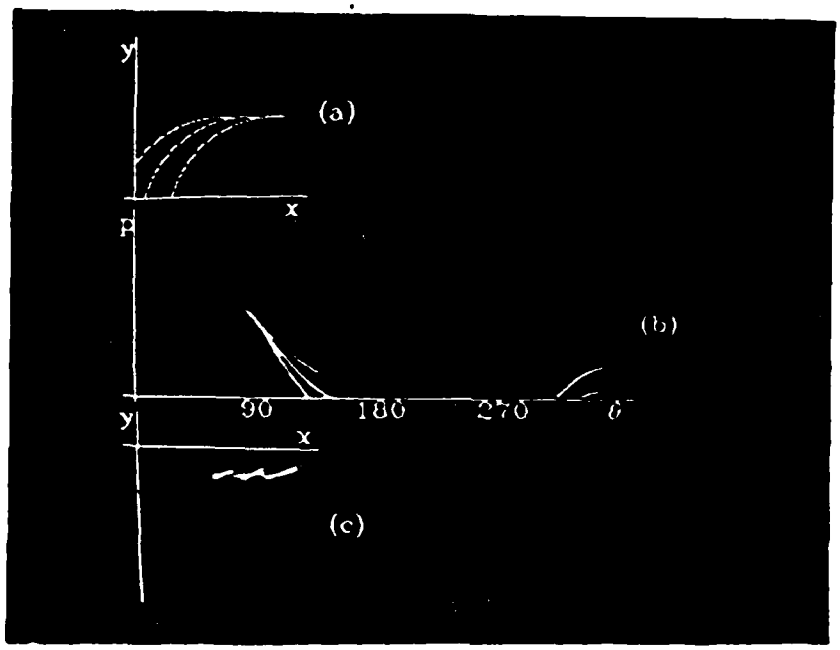


Figure 4-4

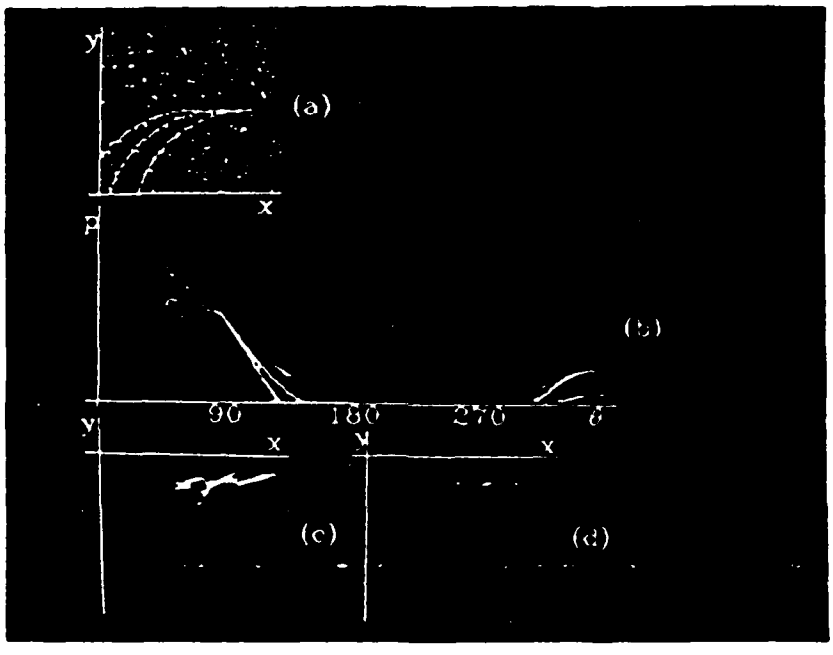


Figure 4-5

References

1. P.V.C.Hough, "Method and Means for Recognizing Complex Patterns", *U.S. Patent 3 069 654,* , 1962.

2. R.O.Duda and P.E.Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures", *Communications, Association for Computing Machinery*,Vol. 15Jan 1972, pp. 11-15.

3. P.M.Merlin and D.J.Farber, "A Parallel Mechanism for Detecting Curves in Pictures", *IEEE Transactions on Computers*,Vol. 24No. 1Jan 1975, pp. 96-98.

4. G.Eichman and B.Z.Dong, "Coherent Optical Production of the Hough Transform", *Applied Optics*,Vol. 22No. 615 Mar 1983, pp. 830-834.

5. G.R.Gindi and A.F.Gmitro, "Optical Feature Extraction via the Radon Transform", *Optical Engineering*,Vol. 23No. 5Sept/Oct 1984, pp. 499-506.

6. C.Kimme, D.Ballard and J.Sklansky, "Finding Circles by an Array of Accumulators", *Communications, Association for Computing Machinery*,Vol. 18No. 2Feb 1975, pp. 120-122.

7. S.Tsuji and F.Matsumoto, "Detection of Ellipses by a Modified Hough Transformation", *IEEE Transactions on Computers*,Vol. 27No. 8Aug 1978, pp. 777-781.

8. H.Wechsler and J.Sklansky, "Finding the Rib Cage in Chest Radiographs", *Pattern Recognition*,Vol. 9No. 1Jan 1977, pp. 21-30.

9. D.H.Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes", *Pattern Recognition*,Vol. 13No. 2 1981, pp. 111-122.

10. C.M.Brown, "Inherent Bias and Noise in Hough Transform", *IEEE Transaction of Pattern Analysis and Machine Intelligence*,Vol. 5No. 5Sept 1983, pp. 493-505.

11. A.E.Cowart and W.H.Ruedger, "The Detection of Unresolved Targets Using the Hough Transform", *Computer Vision, Graphics and Image Processing*,Vol. 21No. 2Feb 1983, pp. 222-238.

12. M.L.Padgett, S.A.Rajala, W.E.Snyder and W.H.Ruedger, "Detection of Maneuvering Target Tracks", *SPIE Proceedings*,Aug 1985.

13. D.H.Ballard and D.Sabbah, "Viewer Independent Shape Recognition", *IEEE Transactions of Pattern Analysis and Machine Intelligence*,Vol. 5No. 6Nov 1983, pp. 653-660.

14. D.H.Ballard and C.M.Brown, *Computer Vision*, Prentice Hall, Englewood, New Jersey, 1982.

15. R.Krishnapuram and D.Casasent, "Hough Space Transformations for Discrimination and Distortion Estimation", *Computer Vision, Graphics and Image Processing*,Feb 1987.

16. D.Casasent and R.Krishnapuram, "Curved Object Location by Hough Transformations and Inversions", *Pattern Recognition*,Vol. 20No. 2 1987.

Chapter 5

Image Understanding Techniques for 3D Scene Interpretation

5.1. Introduction

In this chapter, we present results in two aspects of the 3D change detection task: the low-level problem of analyzing images, and the high-level problem of developing an optimal recognition strategy with using the description. For the low-level processing, we describe a new method of generating edge description. For the high-level processing, we describe a new method of generating a recognition strategy of a object. The basic idea is to reduce the complexity of the observed scene description by designing a model which includes a complete description of the geometry of the 3D object to be recognized. The description of the scene uses only the 3D boundaries which are fairly easy to extract, thereby reducing the segmentation time. This model includes the description of the possible aspects of the occluding boundaries of the object, which we call 3D-profiles, when it is observed from all the possible viewing directions. In addition, it contains all explicit description of the order in which the search tree must be explored at run time.

5.2. Improvement in Edge and Line Extraction from Images

Extraction of lines and edges is one of the most fundamental techniques for image understanding. Previously, we have been using a technique based on modified Nevatia and Babu algorithm. During the last year, we have improved the technique for more reliable extraction of detailed structures from images. Instead of Nevatia and Babu operator, the Canny operator is used to detect edge points, namely, abrupt intensity changes. The operator generates pixel sequences where the intensity change are steep. Figure 5.1 shows the original intensity distribution. Figure 5.2 shows the result applying the modified Nevatia and Babu edge operator to the distribution. Figure 5.3 shows the result applying the Canny edge operator to the same distribution. The Canny operator generates the only important edge segments and less noise elements than the Nevatia-Babu operator does. Thus, the result is much easier to handle by later processing modules.

To generate line segments, we have to track the edge pixels and to detect corners so that we can obtain line segments connecting two corners. Corners may be found using the angle between three adjacents pixels. Let us define Direction-before as a vector from Pixel-now to Pixel-before and Direction-next as a vector from Pixel-now to Pixel-next over an edge pixel sequence. If the angle between Direction-before and Direction-next is larger than a certain threshold, then Pixel-now is considered as a corner point. This method is simple and easy to calculate, but unfortunately, generates too many corner points due to noise. On the other hand, if we increase the distance from Pixel-now to Pixel-before

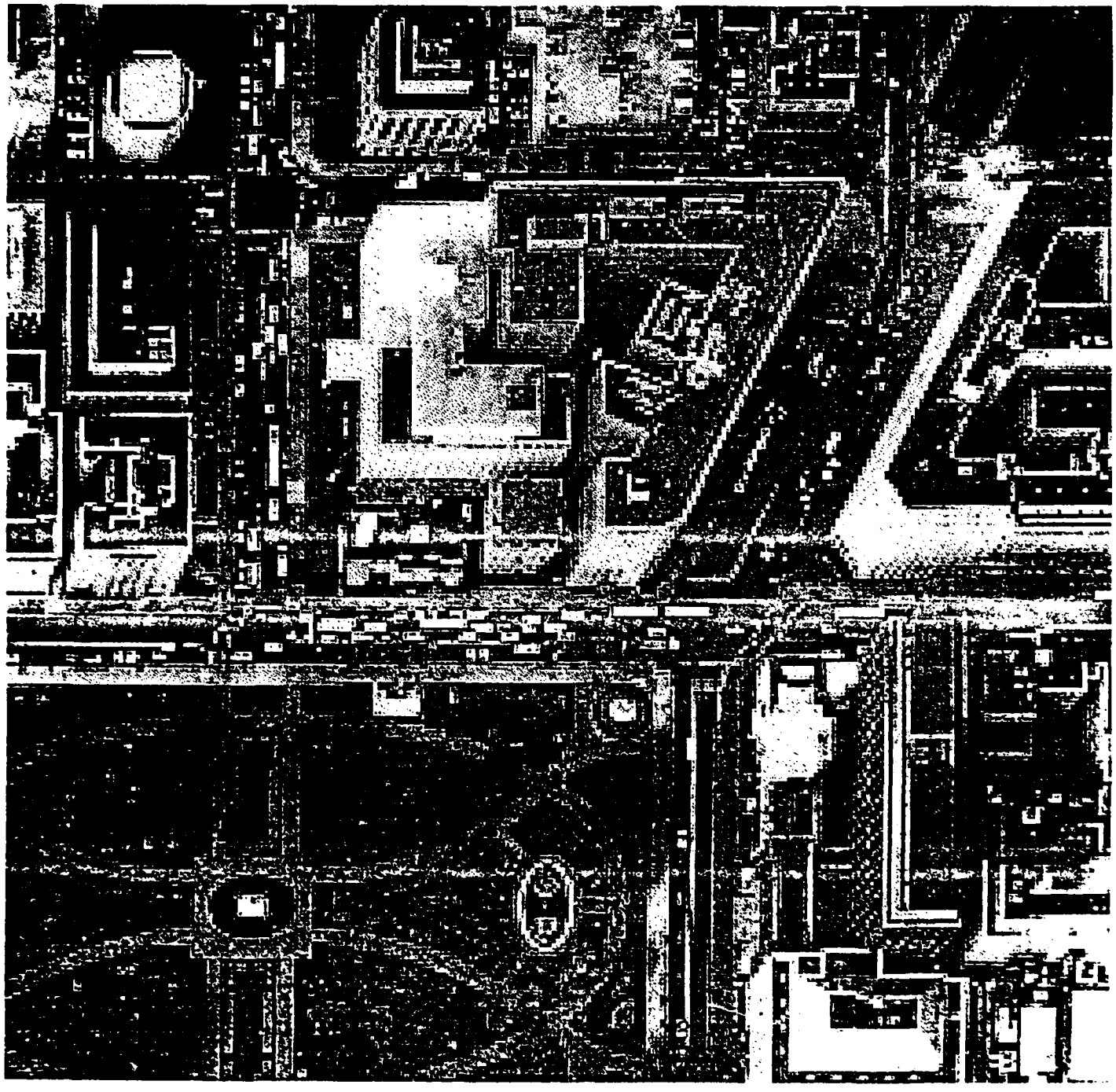


Figure 5-1: Original intensity distribution

and Pixel-next, the method becomes stable, but generates different positions of the corner points.

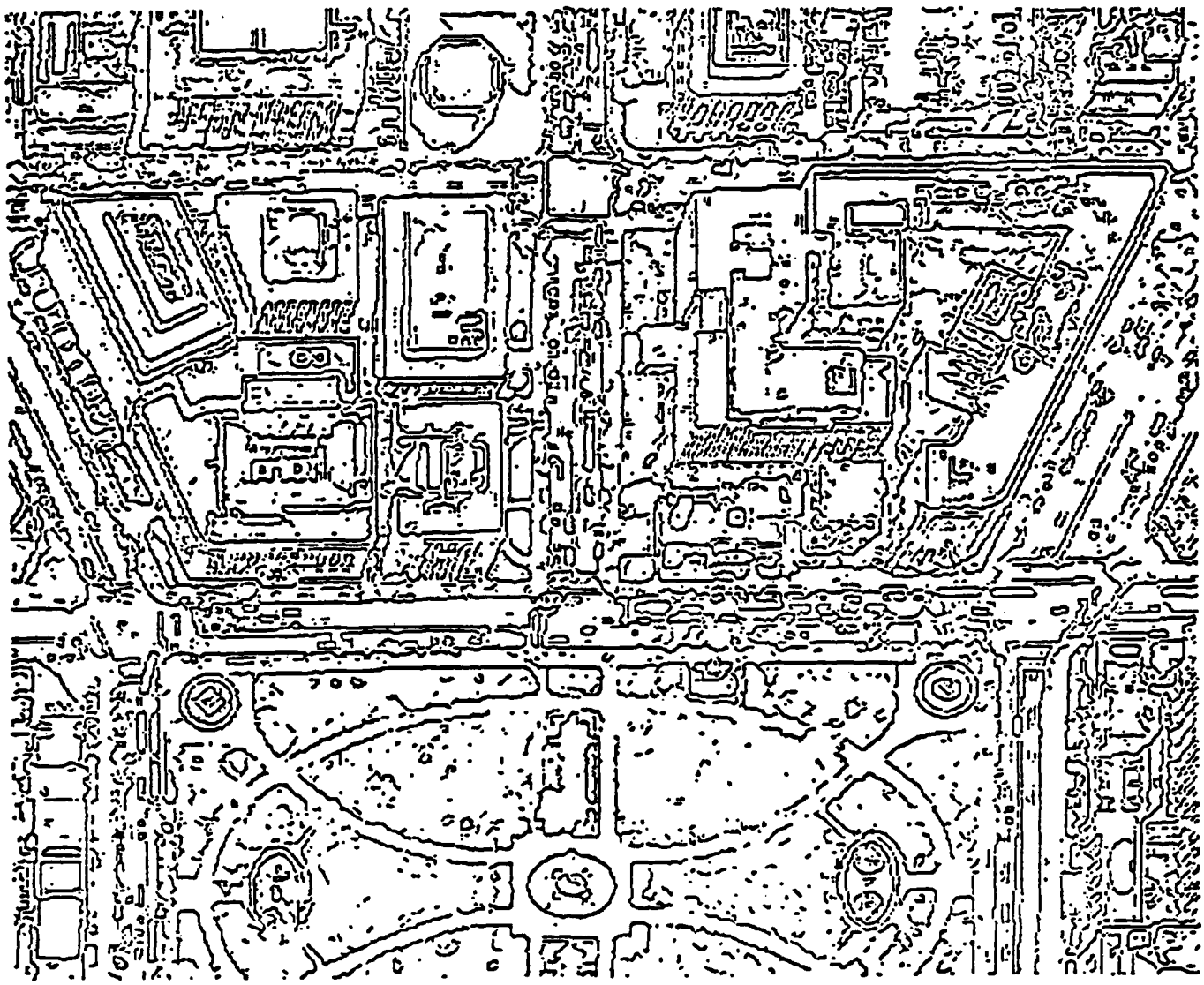


Figure 5-2: Result by the Nevatia-Babu operator

In order to get stable but exact corner points, we propose a multi-level detection method. We will generates corner points using various size of the detection distance. The coarsest level gives the search area of corners, while the

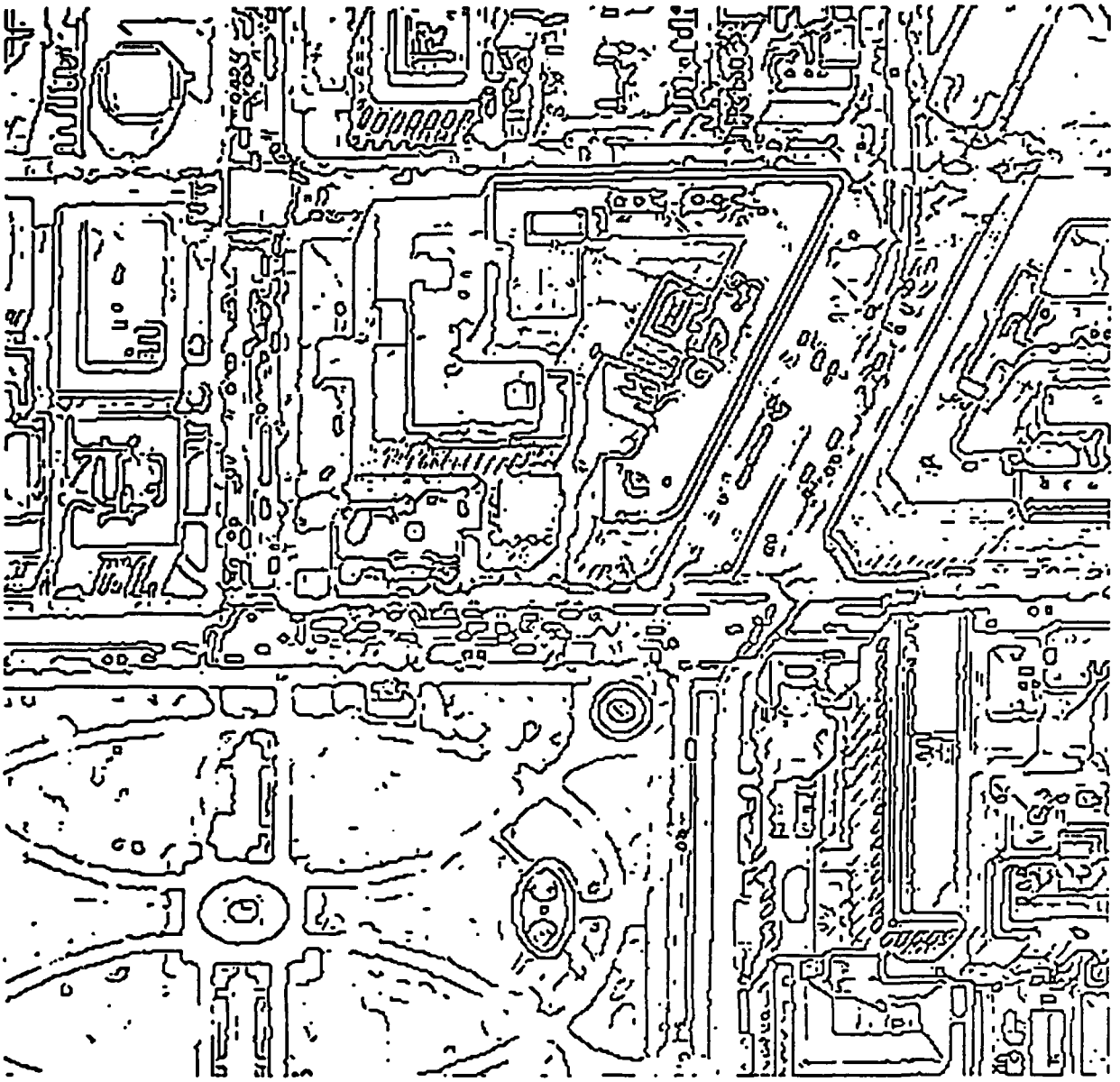


Figure 5-3: Result by the Canny operator

finest level gives candidate positions of corners. Several rules are prepared to interpret the finest level of description based on the description of the coarsest level.

Figure 5.4 shows the angle distributions over the edge sequence using four levels of the detection distances. The upper one is the result by the largest distance (the coarsest level). The bottom one is the result by the smallest distance (the finest level). The dotted lines indicate the threshold level. Extreme points over that value are considered as corner points. Dots indicate indicates candidate locations of corners at each level. Figure 5.5 shows obtained line segments, where the bold lines indicate the confident lines and the thin lines indicate the less confident lines.

5.3. Extraction and matching 3D structures in range images

5.3.1. Introduction

Once 3D scene information, either in the form of a depth map from stereo or range data from an active radar device, is obtained, the next step of scene understanding is to extract 3D structures from it. In this section we explore the problem of efficiently recognizing and positioning objects with a known 3D model in a range image. This problem is important for such tasks as 3D change detection and target recognition: in target recognition 3D structures which match with the target model must be detected, and in 3D change detection 3D structure which were previously identified must be located in the current image. Several solutions to this problem have been proposed: the general approach is to describe the objects in terms of simple primitives such as 3D edges, surface patches or isolated points in the case of sparse data, and then match the sets of primitives describing the model and the scene.

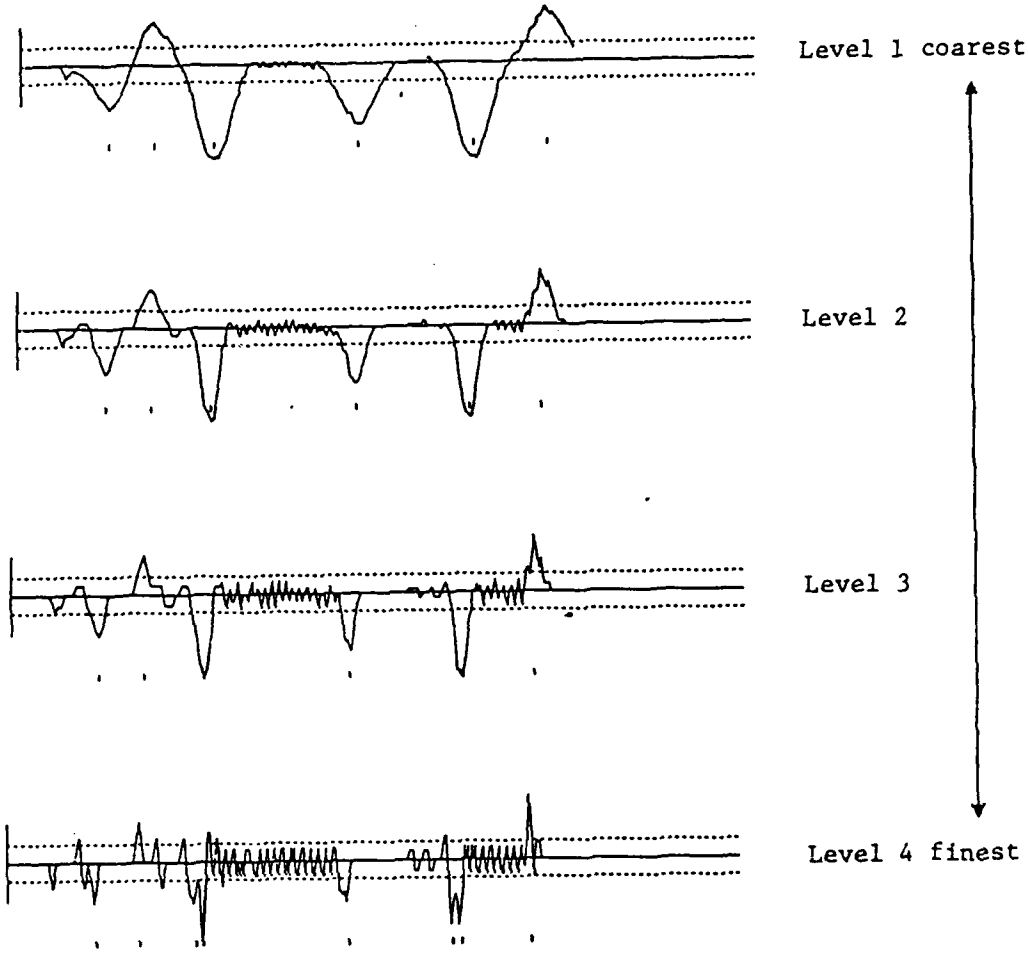


Figure 5-4: Angle distributions using four levels of corner detectors

One critical issue is to extract reliable primitives from the section in a reasonable time and to ensure that the primitives contain enough information for the identification of the object. In this section we propose the 3D-Profile method which allows the recognition of 3D objects by using a very simple processing of the scene, namely the extraction of the 3D occluding edges. The basic idea is that only a small amount of information is needed for identifying and positioning an

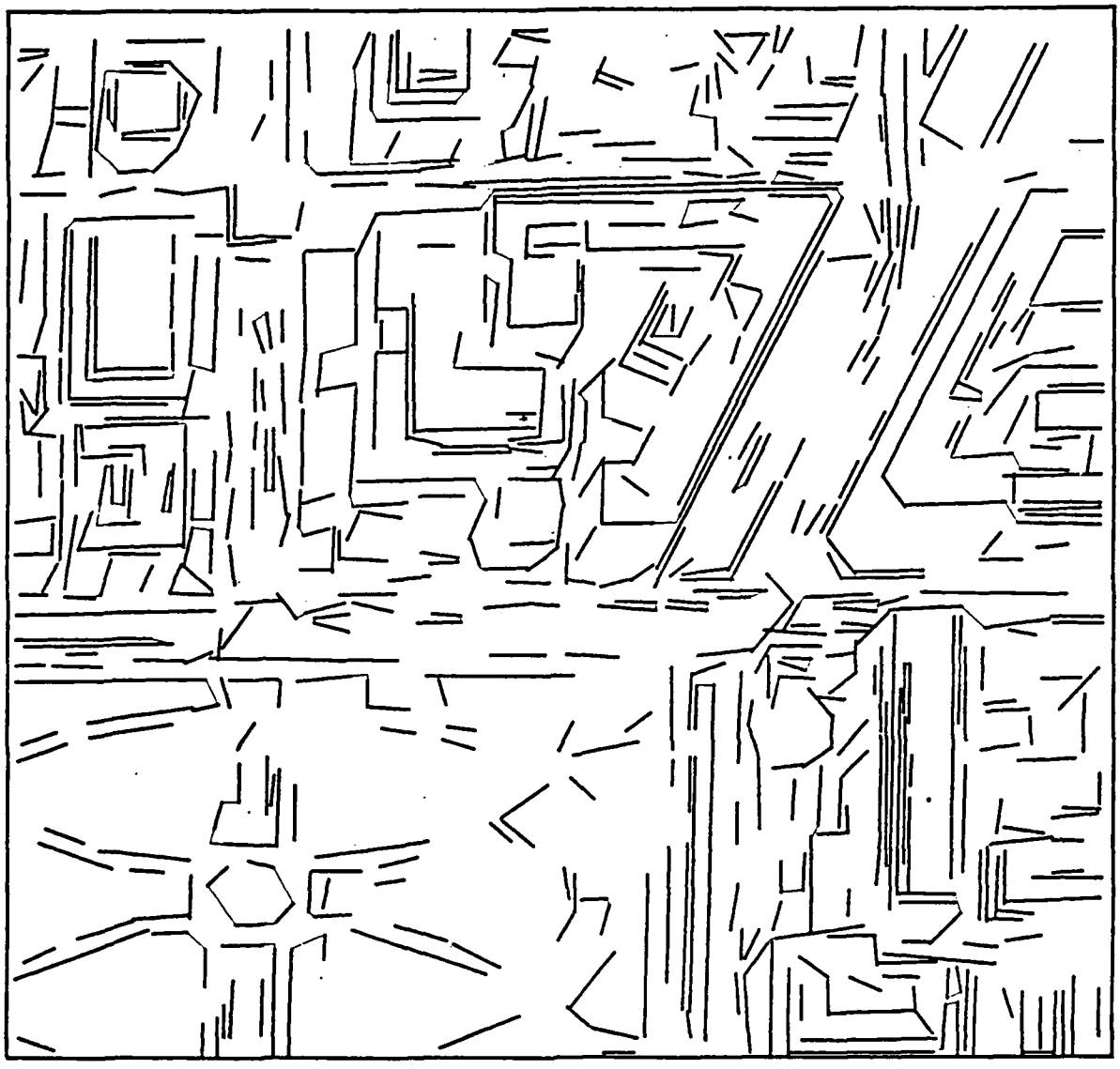


Figure 5-5: Obainted line segments

object provided that the model contains a complete description of the geometry of the object. So, our goal is to "compile" the object in order to produce a model which includes all the information that can be extracted off-line. It includes the possible aspects of the object when observed from different viewing directions and the order in which the search tree should be explored at runtime.

5.3.2. Search Strategies

We denote by $\mathbf{e}_i$ and $\mathbf{e'}_j$ by the primitives (jump edges) of the model and the scene respectively. $\mathbf{w}$ is the current viewing direction. $\mathbf{w}_1$ and $\mathbf{w}_2$ are the canonic x- and y-axis associated with a direction $\mathbf{w}$. α is an inner product of a model edge and the viewing direction $\mathbf{w}$, r is the inner product of an edge and the canonic x-axis of a viewing direction $\mathbf{w}$, $\mathbf{l}(\mathbf{e})$ is the length of e.

Most of the 2-d and 3-d vision algorithms dealing with rigid objects can be described by a "hypothesis/prediction/verification" scheme. This general type of algorithm can summarized as follows:

- Hypothesis:
  Select a primitive **e** of the model and a compatible one **e'** of the scene, the pair (**e**,**e'**) is the initial hypothesis.

- Prediction:
  Try to derive a estimation of the location of the object based upon the hypothesis. Use it to predict the remaining scene primitives that could be instances of model primitives.

- Verification:
  Explore the solutions generated by the Prediction step in order to find the best one according to some criterion.

Since the initial hypothesis may not provide enough information for the prediction of solutions, the last step may require another hypothesis/prediction step. This situation arises typically when the position of the object can be only partially predicted from a single hypothesis (e.g. only the orientation), the prediction phase provides only a guide for the choice of a second hypothesis but generates too many possible solutions. In any case, the prediction phase provides a search region for the remaining primitives in the scene space (See Figure 5.6).

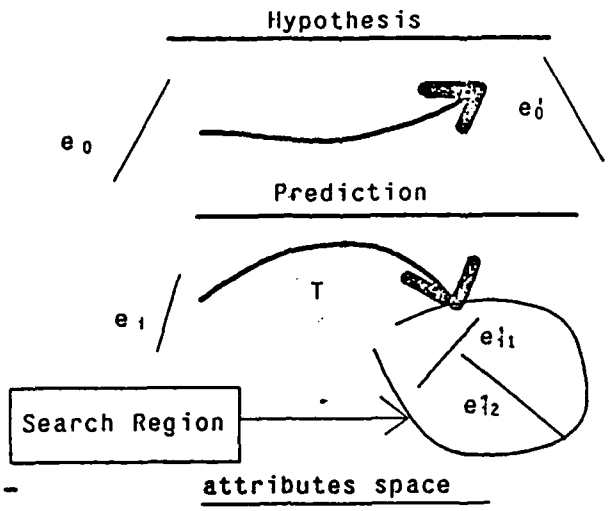


Figure 5-6: The prediction step

The exploration and classification of the viable solutions in the Verification step can be achieved by various methods like tree-search, relaxation, dynamic programmic..etc All these methods try to find an optimum of a criterion among a (expected) small set of possible solutions.

The HPV scheme as described above can be efficient only if two conditions are verified by the scene and model representations:

- The model contains detailed informations about the order of determination of hypotheses and numerical data allowing to reduce the cost of computation of the "search region".

- The representation of the scene allows a fast retrieval of the primitives lying inside the "search region" during the prediction step. This condition is important because a naive implementation of this step requires the exploration of the whole set of scene primitives for

each model primitive even if no compatible matching exists. In other words, the cost of the prediction step should be related to the number of solutions compatible with the first hypothesis and not to the total number of primitives.

The H/P/V scheme is quite general and does not make use of the properties of the actual problem. In the following section, we try to find a representation of 3-d objects suitable for this kind of algorithm.

5.3.2.1. Using low-level features

The idea of using an elaborate description of 3-d contour edges of a solid comes from two observations:

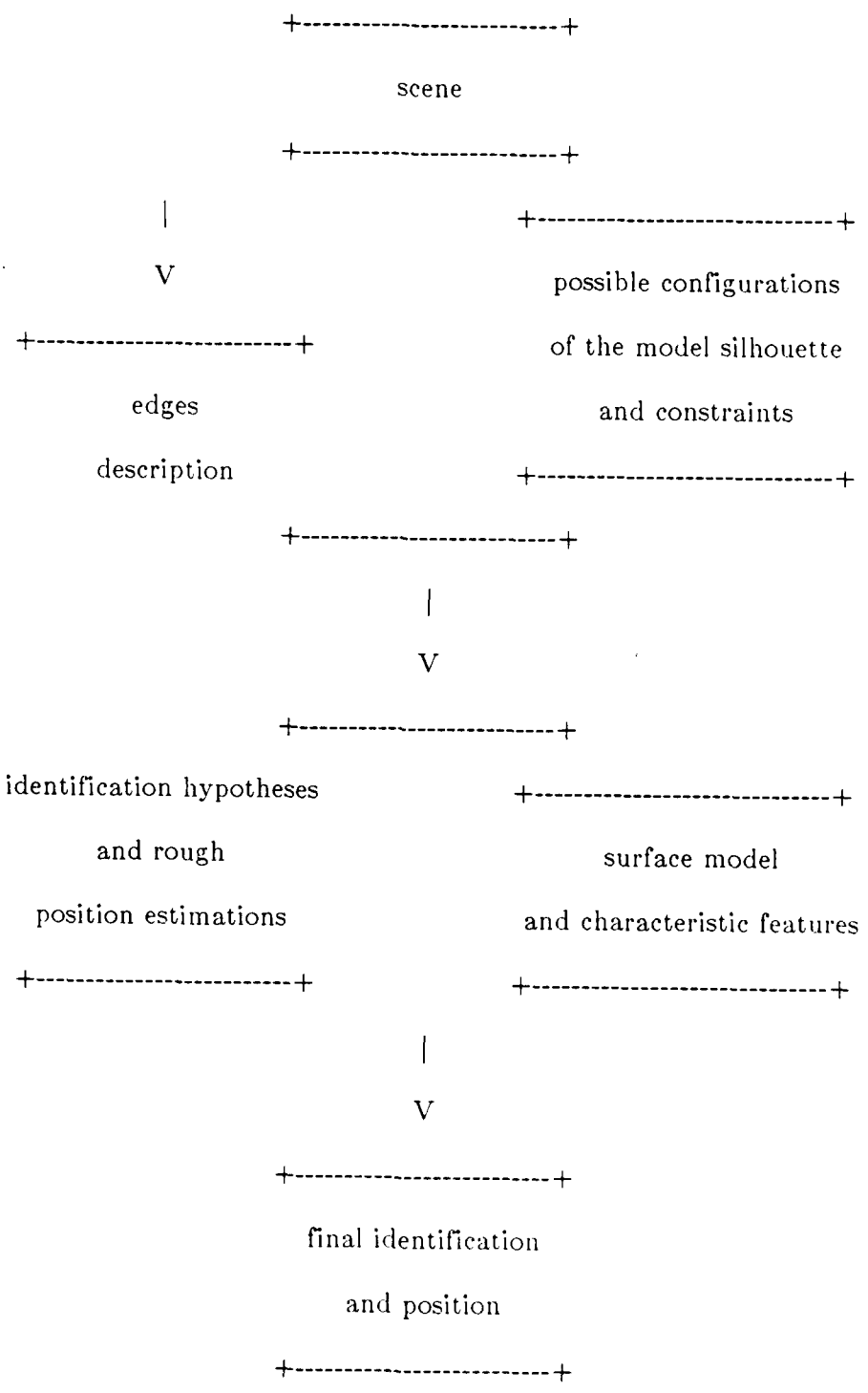
- In most of the 3-d recognition problems, the recognition part can be made relatively efficient by using a small number of reliable high-level primitives. Unfortunately, the segmentation of the observed data into this kind of primitives is time consuming and the reliability of the resulting description is discutable when using higher level primitives (think of the segmentation in quadric patches).

  An alternative is to use low-level features which are easy to extract and to manipulate. These features cannot provide a complete interpretation of the data because of their poor quality and the ambiguities they cannot resolve.

  When 3-d data is available, the simplest features are the occluding (or jump, or silhouette) edges. They are simple to extract and can be used to perform a first analysis of the scene data. So, the idea is to reduce drastically the time of segmentation while dividing the recognition process into several processes instead of having only one tree-search-like process working with the unstructured sets of model and scene features.

- The number of possible configurations of contour edges is generally small and the number of edges in each configuration is also small. The idea is to decompose the whole matching problem into smaller ones by using the fact that some combinations of object features cannot be observed at the same time which reduces the combinatorial complexity of the problem.

Generally, a solid cannot be described entirely by its possible silhouette appearances. So, the contour edges cannot provide an identification of the object and a verification procedure using a surface description but without any combinatorial search should be used:


scene
possible configurations
of the model silhouette
and constraints
edges
description
identification hypotheses
and rough
position estimations
surface model
and characteristic features
final identification
and position

5.3.2.2. Dividing the recognition process

The search strategy can be further decomposed by observing that the geometrical aspects of the model can be divided in four categories:

Attributes	characterized by:
connectivity and length of the silhouette edges	configuration
angle with the viewing direction and projected lengths and mutual angles	viewing dir.
3-d orientations of edges	rotation around the viewing dir.
3-d position of edges	spatial location of the viewer

The problem is to chose which geometrical feature should be used at the various steps of the search. A highly invariant feature requires less complete hypothesis to be applied but induces a weaker constraint, on the other hand a highly constrained one (e.g. the 3-d position and orientation) induces a strong constraint which might reduces the search but requires more evidence to be applied.

So, the idea is to use the various levels of constraints as their evaluation becomes possible. Actually, this is a very natural approach since it means that we recognize first an "aspect" of the object (i.e. a particular silhouette of the object), we check the 2-d consistency (i.e. rotation around the hypothesized viewing direction), then the 3-d consistency (i.e. 3-d position and orientation), and at last we resolve possible ambiguities by checking the predicted surface configuration (e.g. discrimination between a circular coin and a sphere.).

5.3.3. Model Formation

5.3.3.1. Compilation of the Model

One of the problem of 3-d vision is the weakness of the model description. More precisely, the model should contain an explicit description of the search strategy for the particular shape to be recognized. In other words, the model represents not only *what* is the shape in terms of a list of features but also *how* to recognize it. Following the general hypothesis/verification scheme, the model should include:

- The order of exploration of the search-tree. For example, the model could include a piece of knowledge like:

  "if the feature **e** is identified, then the best one to try next if feature **e'**."

  This kind of knowledge avoids searching for irrelevant features or finding the relevant one at recognition time.

- The numerical constraints at the different levels of the search. For instance, a condition like:
  if we e_i belongs to a configuration j then the angle of the next edge should be between α_1 and α_2.

  This kind of knowledge is beneficial in the prediction phase.

The term "model compilation" comes from the original Goad's work who described a way of generating a special-purpose Lisp program from a 3-d object. This program allows to recognize efficiently the "compiled" object by using an optimal search strategy.

In the next section, I try to build a model representation of 3-d objects based upon the decomposition sketched above.

5.3.4. Computing the Configurations

5.3.4.1. Occluding edges and silhouette configurations

We are interested in the simplest 3-d features, the occluding edges. An occluding edge is defined as an edge between the object and the background or another object in the frontal plane. We will call a configuration of edges a set of edges which can be occluding edges at the same time (i.e. for at least one view direction). Notice that this definition is not exactly the definition of a silhouette in a strict sense since a possible occluding edge could be hidden by another part of the object. Besides, the strict definition of the silhouette increases dramatically the number of configurations while not reducing significantly the number of edges in each configuration.

5.3.4.2. Computing the occluding edges of a planar-faced object

In the case of a planar-faced object, the characterization of the occluding edges in fairly simple:

An edge **e** bounding faces **f** and **f'** of normals **n** and **n'** is an occluding one if:

- **e** is not a concave edge.

And

-
  - $(\mathbf{n}.\mathbf{w})(\mathbf{n}'.\mathbf{w}) < 0$

  Or

  - $\mathbf{n}.\mathbf{w} = 0$ and the viewpoint is in front of **f'**.

(The normals are oriented from the interior to the exterior of the surface as usual)

These simple rules allows to compute the possible occluding edges corresponding to a viewing direction (Recall that we do not consider possible self-occlusion for the moment).

5.3.4.3. Determination of the configurations

Two methods can be considered for computing the configurations:

- Analytic method:

  The condition for an edge to be occluded defines a portion of the unit sphere of possible directions which is bounded by curves of known equations. The possible configurations are obtained by tracing all these curves on the sphere which bound a set of regions, each of which corresponds to a configuration.

  The advantage of this method is to ensure that all the configurations will be found, moreover it allows to remove degenerate configurations corresponding to degenerate regions on the sphere (curves or vertices).

  The major drawback is that this algorithm requires the manipulation of analytical curves and patches on the sphere which is difficult to implement. Moreover, we don't need a precise description of the region of the viewing sphere corresponding to a configuration.

- Enumeration method:

The simplest method is to discretize the sphere of viewing directions as regularly as possible and then compute the occluding edges for each directions. The final description is obtained by enumerating the different configurations.

The only drawback is that some configurations can be missed if the digitization is not fine enough.

The second algorithm has been implemented using the *icos* package of the university of Rochester.

5.3.5. Hypotheses Determination

We consider first the case of a description of the edges by line segments. The analysis of curved objects is similar except some transformations computation.

Following the division of the recognition process described above, we have two levels of hypotheses:

- level 0: Select a configuration, **current_config**
  (This level provides bounds on the position of the viewing direction)
- level 1: Select a first edge, $\mathbf{e_{viewdir}}$
  (given a configuration hypothesis, this level gives a partial viewing direction estimation)
- level 2: Select a second edge, $\mathbf{e_{rottrans}}$
  (given a configuration, a first edge, a partial viewing direction estimation. This level provides a viewing direction, a rotation and a translation estimation)

Remark:

If we used the endpoints of the segments, less levels would be required, unfortunately the endpoints of the segments are not reliable because of the occlusion and the possible measurements errors.

We describe now the kind of information that has to be included in the model in order to make these "select" steps as efficient as possible.

5.3.5.1. Configuration selection

There seems to be no guide for the choice of configuration. The only information that could be carried by the model is to sort the configurations in decreasing order of probability.

When **current_config** is selected, the order of selection of edges for the following levels must be determined. In other words, the model contains a set of lists:

$$(e^i_1,...,e^i_n)_{i=1..k}$$

These edges are only those appearing in **current_config**. These lists gives the order of search to the recognition program which tries to match with:

$$e_{viewdir}=e^i_1,\ e_{rottrans}=e^i_2,\ \text{for } i=1..k.$$

The choice of the first three primitives during the model construction should obey the rules:

- The edges are the most reliable ones (e.g. the longest)
- The edges are linearly independent.

The current configuration corresponds to a region on the sphere of directions, the boundary of this region cannot be used directly, but bounds can be computed on the angle between an edge and the viewing direction. Therefore an interval $[\alpha_{min},\alpha_{max}]$ is attached to each edge. At recognition time, only the scene edges such that:

$$\alpha_{\mathbf{min}}^{\mathbf{viewdir}} < \alpha < \alpha_{\mathbf{max}}^{\mathbf{viewdir}}$$

are considered for the level one hypothesis.

5.3.5.2. Level one

The selection of an edge $\mathbf{e}_{\mathbf{viewdir}}$ determines an interval of search for the possible scene edges $\mathbf{e'}_{\mathbf{viewdir}}$, once this edge is selected, the viewing direction can be predicted up to one degree of freedom.

More precisely, the viewing direction **w** is in a cone:

$$\mathbf{w}.\mathbf{e}_{\mathbf{viewdir}} = \mathbf{e'}^{\mathbf{z}}_{\mathbf{viewdir}} = \alpha'_{\mathbf{viewdir}}$$

This cone provides the prediction condition for the second edge $\mathbf{e'}_{\mathbf{rottrans}}$, this edge must verify:

$$\alpha'_{\mathbf{min}}(\alpha'_{\mathbf{viewdir}}) < \alpha' < \alpha_{\mathbf{max}}(\alpha'_{\mathbf{viewdir}})$$

The bounds must be stored in the model representation, which means that for each $\mathbf{e}_{\mathbf{viewdir}}$ and α, the bounds on $\alpha'_{\mathbf{rottrans}}$ are precomputed. The values of α must be discretized and a data structure must be designed to allow a fast access to those values.

This solution implies that $\mathbf{N}_{\mathbf{config}}.\alpha_{\mathbf{step}}.\mathbf{N}^{\mathbf{2}}_{\mathbf{prims}}$ intervals are stored, where $\mathbf{g(a)}_{\mathbf{step}}$ is the discretization level of the angles and $\mathbf{N}_{\mathbf{prims}}$ is the average number of potential **viewdir** and **rottrans** primitives in each configuration. An alternative is to derive these intervals by simple computations from a smaller set of stored data (see Figure 5.7).

We can associate a coordinate frame $(\mathbf{e}_{\mathbf{viewdir}}.\mathbf{a_0}.\mathbf{a_1})$ to the current edge, such that $\mathbf{a_0}.\mathbf{e}_{\mathbf{rottrans}} = 0$, in this case we have:

$$\mathbf{w} = \cos(\alpha)\mathbf{e}_{\mathbf{viewdir}} + \sin(\alpha)\cos(\phi)\mathbf{a}_1 + \sin(\alpha)\sin(\phi)\mathbf{a}_0$$
$$\mathbf{e}_{\mathbf{rottrans}} = \cos(\psi)\mathbf{e}_{\mathbf{viewdir}} + \sin(\psi)\mathbf{a}_1$$

So, the inner product $\mathbf{w}.\mathbf{e}_{\mathbf{rottrans}}$ is bounded by:

$$\mathbf{c}_1 = \cos(\psi - \alpha) \text{ and } \mathbf{c}_2 = \cos(\psi + \alpha)$$

And the angle $\alpha(\mathbf{e}_{\mathbf{rottrans}})$ is bounded by:

$$\alpha^{\mathbf{rottrans}}_{\mathbf{min}} = \mathrm{Max}(\alpha^{\mathbf{viewdir}}_{\mathbf{min}}, \mathrm{Min}(\mathrm{acos}(\mathbf{c}_1), \mathrm{acos}(\mathbf{c}_2)))$$
$$\alpha^{\mathbf{rottrans}}_{\mathbf{max}} = \mathrm{Min}(\alpha^{\mathbf{viewdir}}_{\mathbf{max}}, \mathrm{Max}(\mathrm{acos}(\mathbf{c}_1), \mathrm{acos}(\mathbf{c}_2)))$$

The "acos" function used in these formula is simply an abbreviation for:

$$\mathrm{acos}(x) = \text{if } 0<x<\pi \text{ then } x \text{ else if } x>\pi \text{ then } 2\pi\text{-}x \text{ else } -x$$

These calculations are quite simple and provides the orientation prediction for $\mathbf{e}_{\mathbf{rottrans}}$. The information stored in the model is the set of angles $(\mathbf{e}_i,\mathbf{e}_j)$ for all relevant edges i and j.

5.3.5.3. Level two

- Orientation After having selected a scene edge such that $\alpha'_{\mathbf{rottrans}}$, we are able to compute a first estimate of the object orientation which is decomposed in a first estimate of the viewing direction and of the rotation around this direction.

  The direction is computed by solving the equations:

$$\mathbf{w}.\mathbf{e}_{\mathbf{viewdir}} = \mathbf{e}'^{\mathbf{z}}_{\mathbf{viewdir}} = \alpha'_{\mathbf{viewdir}}$$
and
$$\mathbf{w}.\mathbf{e}_{\mathbf{rottrans}} = \mathbf{e}'^{\mathbf{z}}_{\mathbf{rottrans}} = \alpha'_{\mathbf{rottrans}}$$

  We can associate with $\mathbf{w}$ a canonical frame $(\mathbf{w},\mathbf{w}_1,\mathbf{w}_2)$, then the rotation around $\mathbf{w}$ is entirely determined by the angular offset $\Delta\theta$:

$$\mathbf{w}_1.\mathbf{e}_{\mathbf{viewdir}} = \mathbf{e}'^{\mathbf{x}}_{\mathbf{viewdir}} = \alpha'_{\mathbf{viewdir}}$$
and
$$\mathbf{w}_1.\mathbf{e}_{\mathbf{rottrans}} = \mathbf{e}'^{\mathbf{x}}_{\mathbf{rottrans}} = \alpha'_{\mathbf{rottrans}}$$

  This information is used for the third level prediction by associating with each potential $\mathbf{e}_{\mathbf{trans}}$ the angular intervals:

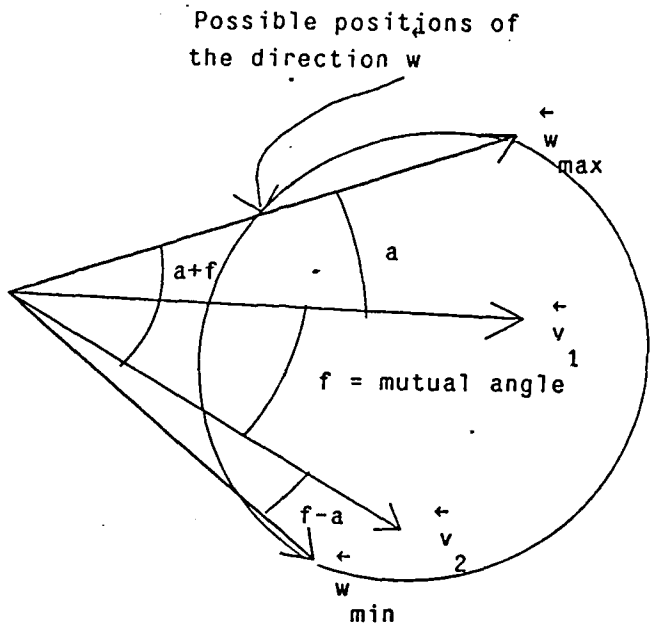


Figure 5-7: Angular bounds computation for the second level prediction

$$[\alpha(\mathbf{e_{rottrans}}) - \epsilon\ ,\ \alpha(\mathbf{e_{rottrans}}) + \epsilon]$$
$$[\theta(\mathbf{e_{rottrans}}) + \Delta\theta - \epsilon\ ,\ \theta(\mathbf{e_{rottrans}}) + \Delta\theta + \epsilon]$$

Where ϵ is a tolerancy factor used to take into account the discretisation level and the measurements precision.

The viewing direction and the offset angle provide only a rough estimate of the orientation, but they can be easily computed and they can be combined with the model structure without any operations (only the addition of $\Delta\theta$).

In summary, the model must contain a representation of the discretized sphere with for each cell:

- A list of occluding edges (Which is the same one for the whole configuration)
- The canonic frame corresponding to the discretized direction.
- The angle $\mathbf{g}[\mathbf{a}]$ and θ for every edge.

- position

The position of an edge is entirely defined by the "normal" vector:

$$\mathbf{n} = \mathbf{OP} - (\mathbf{OP}.\mathbf{e})\mathbf{e}$$

If we apply a rotation and a translation to the edge, the new normal vector $\mathbf{T(n)}$ is:

$$\mathbf{T(n)} = \mathbf{R.OP} + \mathbf{t} - ((\mathbf{R.OP} + \mathbf{t}).\mathbf{e})\mathbf{e}$$

In a simpler way:

$$\mathbf{T(n)} = \mathbf{n} + (\mathbf{t} - (\mathbf{t}.\mathbf{Re})\mathbf{Re})$$

($\mathbf{T(n)}$ is only a notation for the normal vector of the transformed edge, it's not the transformed normal vector.) Therefore, the translation can be estimated from the second level hypothesis by solving the set of six equations:

$$\mathbf{n'_{viewdir}} = \mathbf{n_{viewdir}} - (\mathbf{t}.\mathbf{Re_{viewdir}})\mathbf{Re_{viewdir}}$$
$$\mathbf{n'_{rottrans}} = \mathbf{n_{rottrans}} - (\mathbf{t}.\mathbf{Re_{rottrans}})\mathbf{Re_{rottrans}}$$

The resolution of these equations requires the calculation and application of rotation $\mathbf{R}$. Fortunately, we can express the coordinates of $\mathbf{n}$ and $\mathbf{e}$ in the current local coordinate system $(\mathbf{w},\mathbf{w_1},\mathbf{w_2})$ which means that $\mathbf{R}$ is simply a rotation of angle $\Delta\theta$ and axis x. So, the estimation of $\mathbf{t}$ requires only a few computations.

Similarly, we can use $\mathbf{t}$ to refine the search region already built from the rotation:

For every edge $\mathbf{e_{remaining}}$, the search region in the scene is the set of edges such that the above relation between $\mathbf{n'}$ and $\mathbf{n_{remaining}}$ is verified. Since we must add a tolerancy factor $\epsilon_{\mathbf{trans}}$, the search region for $\mathbf{e_{remaining}}$ is defined by:

$$|\mathbf{T(n)^{axis}_{remaining}} - \mathbf{n'^{axis}}| < \epsilon_{\mathbf{trans}}, \text{ for } \mathbf{axis} = \mathbf{x.y.z.}$$

Notice that the normal vector $\mathbf{T(n)}$ must be computed for every remaining edge but the rotation is very simple thanks to the local coordinate system.

Remark

It might seem strange that we need two levels for estimating the translation because the relation between $\mathbf{n}$ and $\mathbf{T(n)}$ provides three equations with three unknowns but, unfortunately, the linear system is singular. More precisely, the matrix of the system is $\mathbf{A}$-$\mathbf{Id}$, where $\mathbf{A}$

is the matrix: $\mathbf{A}_{ij} = \mathbf{e}^i\mathbf{e}^j$. A simple verification shows that $\mathbf{A}$ has an eigenvalue 1 and thus the system is singular.

5.3.5.4. Summary of model structure

The model structure as sketched above can be summarized as: (see Figure 5.8).

Configurations : $\mathbf{C}_1,..,\mathbf{C}_n$

current_config

|

V

Order of hypotheses

$\mathbf{e}^1_{\mathbf{viewdir}}$	.	$\mathbf{e}^i_{\mathbf{viewdir}}$
$\mathbf{e}^1_{\mathbf{rottrans}}$	.	$\mathbf{e}^i_{\mathbf{rottrans}}$
.	Or	.
$\mathbf{e}^1_{\mathbf{rot}}$	.	$\mathbf{e}^i_{\mathbf{rot}}$
$\mathbf{e}^1_{\mathbf{trans}}$	.	$\mathbf{e}^i_{\mathbf{trans}}$
.	.	.
$\mathbf{e}^1_{\mathbf{remaining}_1}$	.	$\mathbf{e}^i_{\mathbf{remaining}_1}$
.	.	.
$\mathbf{e}^1_{\mathbf{remaining}_j}$	.	$\mathbf{e}^i_{\mathbf{remaining}_j}$

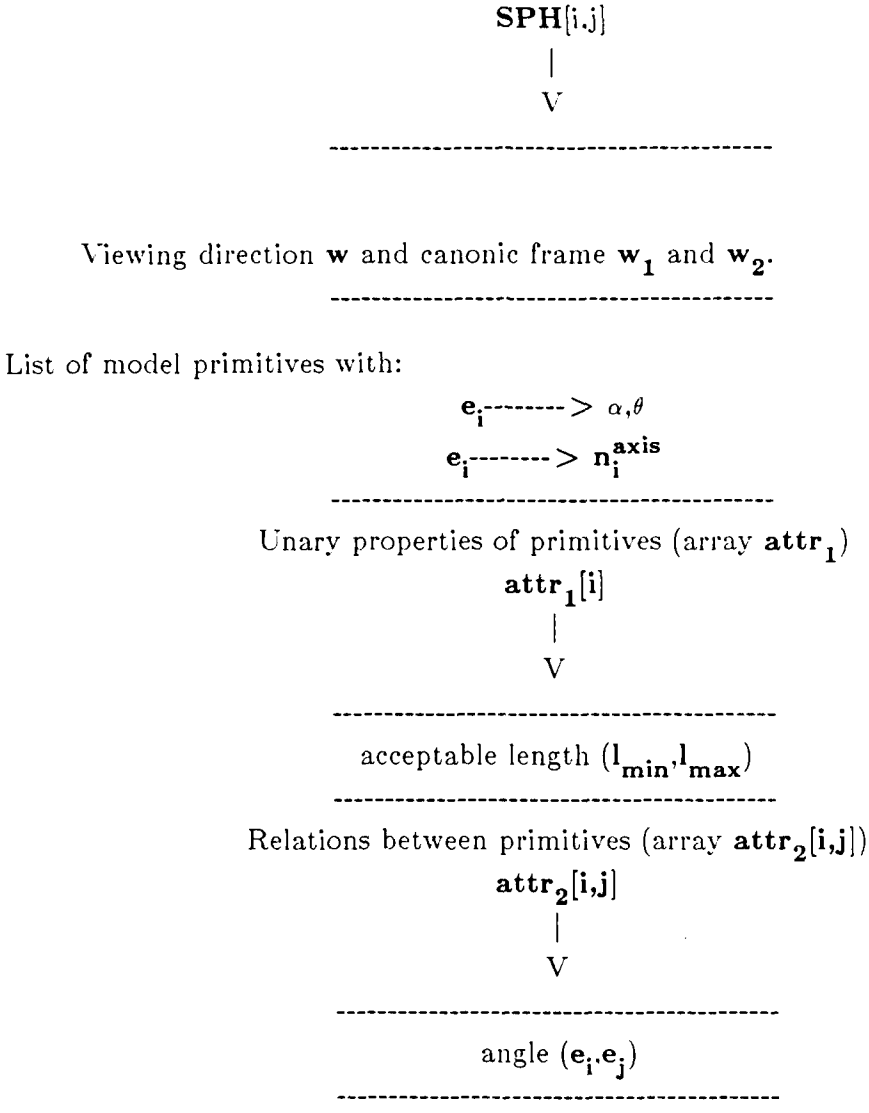
Angular interval for $\mathbf{e'}_{\mathbf{viewdir}}$ selection:

$\mathbf{e}_1$ ---------> $[\alpha(\mathbf{e}_1)_{\mathbf{min}}, \alpha(\mathbf{e}_1)_{\mathbf{max}}]$

.....

$\mathbf{e}_i$ ---------> $[\alpha(\mathbf{e}_i)_{\mathbf{min}}, \alpha(\mathbf{e}_i)_{\mathbf{max}}]$

Discretized sphere **SPH**[i,j]:



5.3.6. Implementation

The major drawback of this model representation is the size of the data structure which is related to the level of discretization of the sphere and the resolution of the angle tables. The physical representation of the model should be carefully designed in order to avoid wasting time just for initializing the internal representation in the recognition program. Two implementations can be designed:

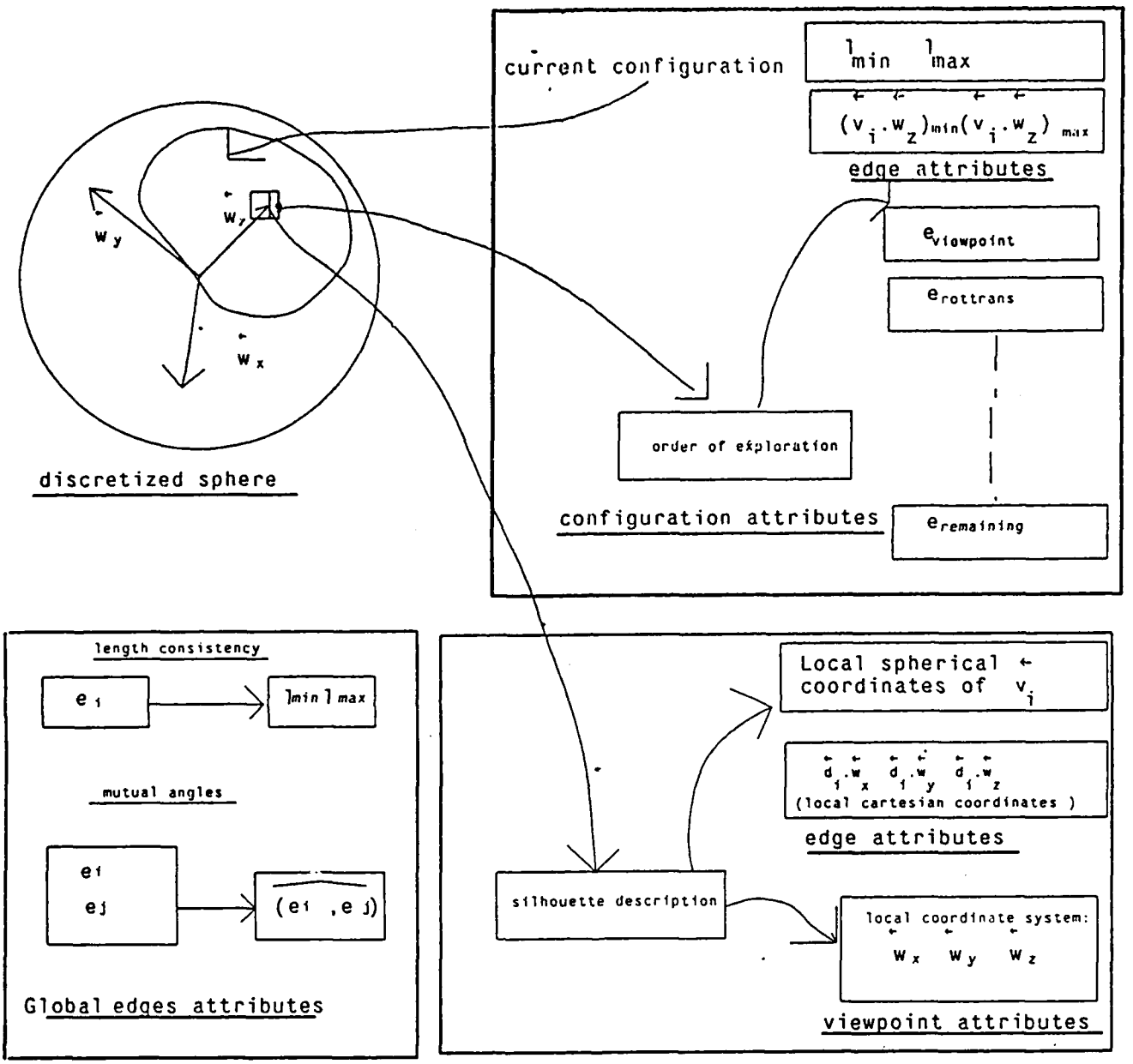


Figure 5-8: Model data structure

- Program generation:
  The first way is to represent the set of configurations and the corresponding orders of exploration as a program by replacing the generic model edges by the actual compiled edges, $e_{viewpoint}$,

e_{rot}..etc, in the general recognition program. The viewpoints attributes and the sphere are stored in arrays initialized at compilation time.

In that case, no file reading in required but the size of the resulting program could be quite large.

- File representation:
Another possibility is to store the entire data structure in a file which is read at recognition time. The worst way for doing that is to dump on a file the structure described above and read the entire data when needed because the time required for reading would be much higher than the recognition time and we would have no benefit in designing a detailed model. Moreover, only a small part of the representation is needed for a particular instance of the observed scene. For example, only a few set of viewpoint attributes ($w, w_1, w_2, e_1, ..., e_n$) out of $N_{SizeOfSphere}$ are needed. Therefore we must divide the elements of the representation in two sets: "Minimum Description" and "On Request", the former being loaded by the program at the beginning, the latter being read when required by the actual arrangement of scene primitives:

  - Minimum Description:
  -The possible configurations with the order of exploration and the min and max angles for $e_{viewpoint}$.
  -The discretized sphere, each cell of which contains the address (in the file) of the viewpoint attributes. -The unary constraints (length consistency..)

  - On Request:
  -The viewpoint description.

5.3.7. Scene Representation

5.3.7.1. The range search problem

The previous discussion shows that the general operation in the various prediction step is the following:

- The scene primitives are described by a set of numerical values $(a_1,...,a_n)$.
- The hypothesis step provides a set of intervals, $([a^{min}_1,a^{max}_1],...,[a^{min}_p,a^{max}_p])$
- A scene primitive **i** lies in the search region if:

$$a^{min}_j < a_i < a^{max}_j$$

This problem is known as the multidimensional range search. In fact, our problem is much simpler than the general range search because we are dealing with a *static* set of data, i.e. neither insertions nor deletions are performed because all the primitives are known at the beginning of the process.

Several structures were proposed for solving the range searching problem: *k*-d trees, range trees, super b-tree..etc. Three costs functions must be considered when selecting a particular structure, the *preprocessing* time, the *storage* requirements and the time required to answer a *query*. It can be proved that the optimal structure requires an untractable storage size ($N^{dimension}$) and preprocessing time. In that respect, the best solution seems to be the *range tree* which allows a worst-case behavior very near the optimal and a modest preprocessing time. The range tree is described in Appendix 1.

Therefore, a efficient scene represensation can be built for computing quickly the set of primitives lying in a predicted search region by answering queries like:

"Find the edges such that the length is between l_{min} and l_{max} and the angle with the Z-axis is between α_{min} and α_{max}"

5.3.8. Results

The program has been tested on several objects in order to validate the method. The models have been generated by the procedures described above, and the geometric description of the object was entered by hand. The output of the modeling program is a set of C files that are linked with the standard model-independent recognition program. By doing this we avoid the problem of fast access to a large file containing the model description. The size of the model for the previous examples is about 200KB.

Example 1

The first example has been obtained by using synthesized range images to which uniform noise is added. The two other examples use range data obtained from a laser range finder (The White Scanner).

This examples uses the image of Figure 5.10 which was synthetized from the object of Figure 5.9. A noise was added to the depth image. The result of the recognition program is displayed in Figure 5.11. It shows the superimposition of the identified scene edges (solid lines), the recognized edges of the model (dashed lines), and the other edges of the model (dotted lines). The same display convention is used for the next two examples.

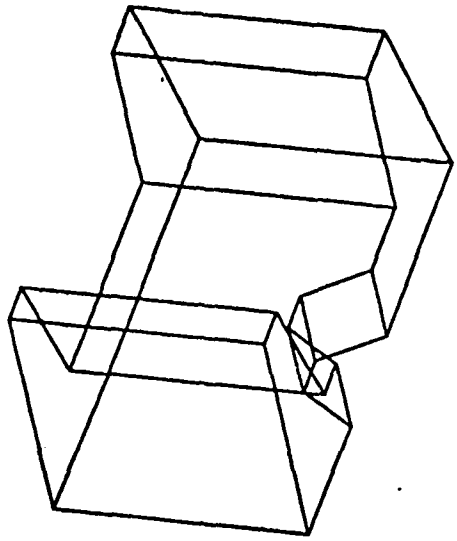


Figure 5-9: Example of a 3D object

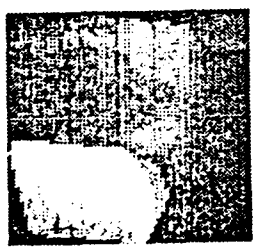


Figure 5-10: Example of a range image

Example 2

Figure 5.12 shows the range image of a planar-faced object, Figure 5.13 shows the polygonal segmentation of the occluding boundary. The measuring device uses a light-stripe technique with one laser and one camera. The range image is observed from a direction which is halfway between the directions of the laser and the camera. The points are then transformed into the laser coordinate

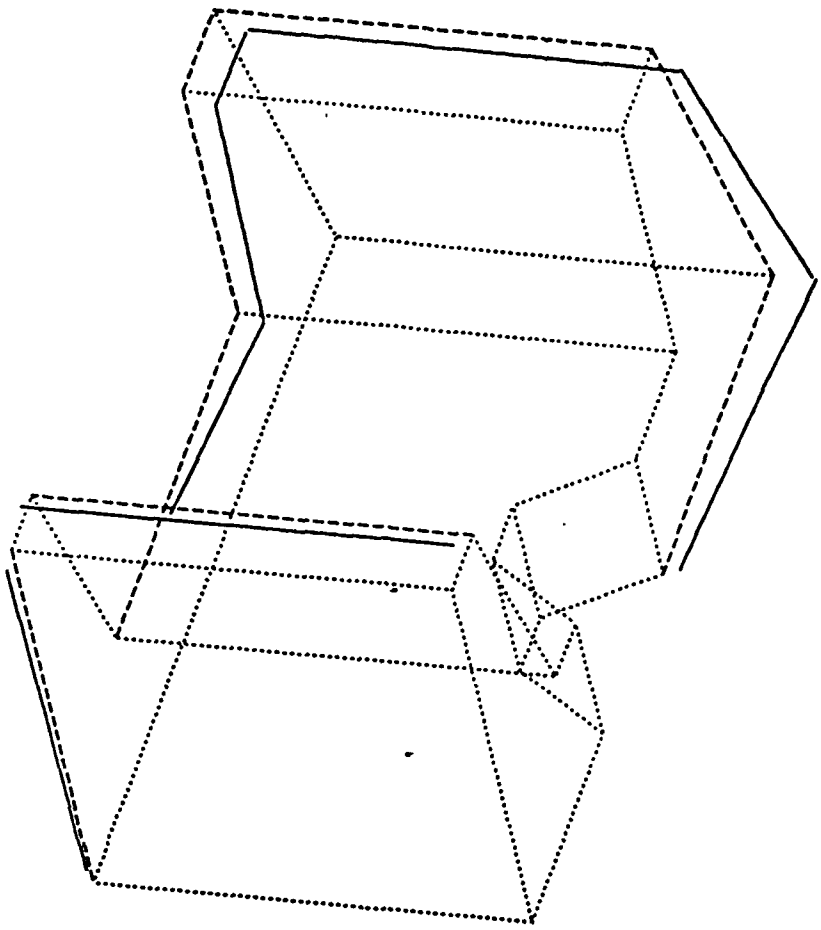


Figure 5-11: Result of the recognition program

system in order to obtain meaningful occluding boundaries (i.e. the z-axis used in the segmentation and recognition is the direction of the laser).

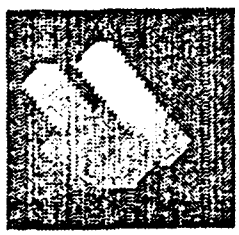


Figure 5-12: Range image

The solutions produced by the recognition algorithm are displayed on Figure 5.14. This result leads to several observations. First, the correct solution (solution 1 of Figure 5.14) is found despite the poor quality of the data - the scanner introduced a systematic geometrical distortion-. Second, only a small number of solutions is produced even though the thresholds are large, the number of primitives is small and there are several parallel edges. This result indicates that the occluding edges contain enough information for recognizing a 3D object and also that the structure of the model provides a good control of the search.

The recognition time is 7 sec. cpu time on a VAX. The model contains 40 edges and twenty configurations (all the configurations were examined by the recognition program).

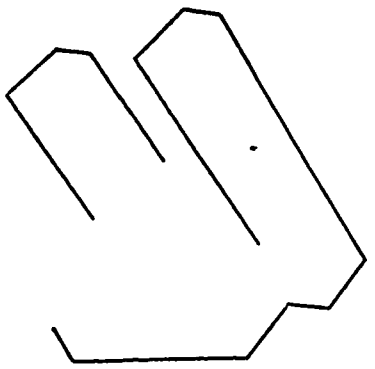


Figure 5-13: Occluding boundary

Example 3

This example uses another view of the previous object. The polygonal

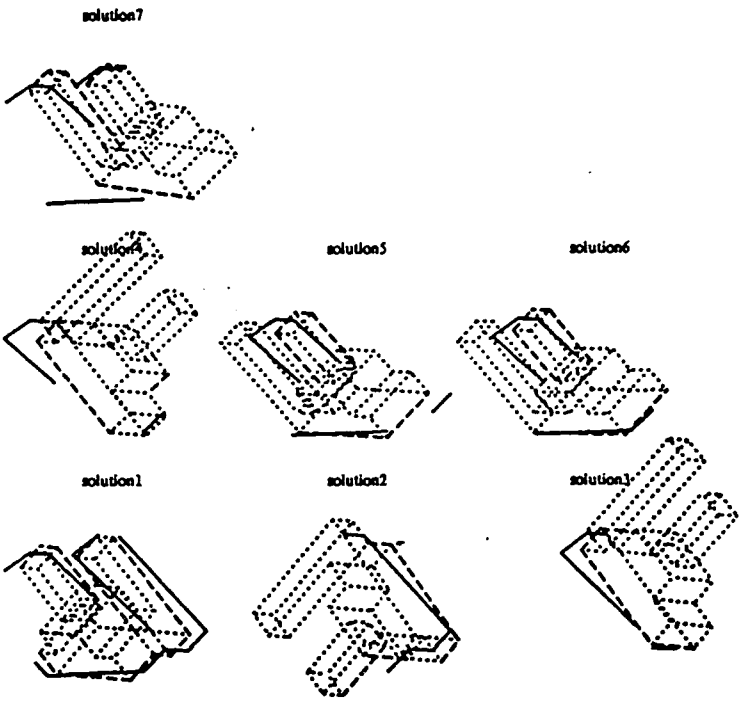


Figure 5-14: Result of the recognition program

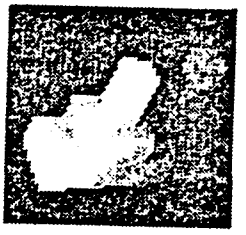


Figure 5-15: Range image

segmentation (Figure 5.16) presents a strange edge which is due to the erroneous measurement by the range finder. The solutions are presented in Figure 5.17, the correct one is the first one.

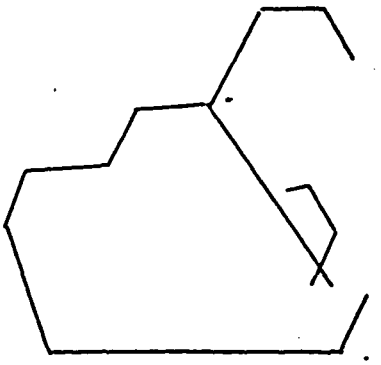


Figure 5-16: Occluding boundary

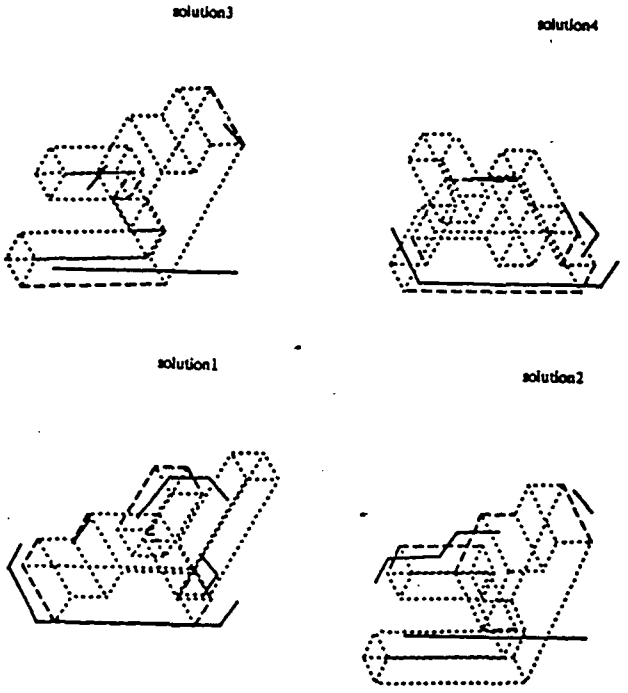


Figure 5-17: Result of the recognition program

5.3.9. Conclusions

We have presented a new method for building recognition-oriented models of 3D objects by representing explicitly the viewer-dependent geometry of the object and by including explicit guidelines for the tree-search procedure. The structure of the model leads to a fast recognition program which requires only a simple preprocessing of the measured data. The method has been proved feasible on a sample set of moderately complicated objects. We are now in the process of testing the program on more complicated scenes and objects.

Future work includes the design of an even more complete model by adding information about the symmetries of the objects, the automatic generation of verification procedures for removing possible extra solutions produced by the recognition program (this is usually closely related to the symmetries of the object), the extension to objects with curved surfaces by using polygonal approximations of the 3D edges and the profile edges.

References

1. Brady, M. et al., "Describing surfaces," Proc. Second Int. Symp. on Robotics Research, Kyoto, Japan, 1984, pp.434-445.

2. Canny, J.F.,"Finding lines and edges in images," Massachusetts Institute of Technology, Artificial Intelligence Laboratory, AI-TR-720, 1983.

3. Faugeras O.D., Hebert, M. "A 3D recognition and positioning algorithm using geometrical constraints between primitive surfaces," Proc. Eight Int. Joint Conf. on Artificial Intelligence, Karlsruhe, August, 1983, pp.996-1002.

4. Faugeras O.D., Hebert, M., Pauchon, E."Segmentation of range data into planar and quadratic patches," Proc. CVPR83, Washington, June, 1983, pp.57-70.

5. Goad, C. "Special purpose automatic programming for 3D model-based vision," Proc. DARPA Image Understanding Workshop, June, 1983, pp.94-104.

6. Grimson, W.E.L. Lozano-Perez, T. "Model-based recognition and localization from sparse three-dimensional data," J. Robotics Research 3, 3 1983, pp.3-35.

7. Nevatia, R. and Babu, K.R., "Linear feature extraction and description," Computer Graphics and Image Processing, 13, July, 1980, pp.257-269.

8. Hebert, M. Ponce, J. "A new method for segmenting 3-D scenes into primitives," Proc. ICPR82, Munich, October, 1983, pp.836-839.

9. Bolles,R.C. Cain,R.A. "Recognizing and locating partially visible objects; The local-feature-focus method," J. Robtics Research 1.3 1982, pp.57-82.

10. Jarvis, R.A. "A perspective on range finding techniques for computer vision," IEEE Trans PAMI, PAMI-5.2, 1983.

11. Oshima, M. Shirai, Y. "Object recognition using three-dimensional information," IEEE Trans PAMI, PAMI-5.4,1983,353-361.

12. Tomita,F. Kanade,T. "A 3D vision system: generating and matching

scene descriptions in range images," Proc Second Int Symposuim of Robotics Research, Kyoto, August, 1984, pp.9-16.

Chapter 6

Summary

In Chapters 2-5, we have described our progress towards achieving a combination of pattern recognition, image understanding and artificial intelligence techniques for space-based image processing, using both optical and digital processing methods. The various results achieved in the past year are summarized below.

We investigated the use of six different algorithms (five linear and one nonlinear) for their ability to detect small targets in slowly moving backgrounds. Analysis and simulation results were presented. Overall, parabolic interpolated differencing seems to outperform all others and is suitable for parallel real-time realization.

We proposed a new method for adaptive subpixel shift estimation using Group Delay Function. This method has two advantages over the conventional (cross-correlation based methods). The first is that it accommodates the subpixel shifts naturally. The second is that GDFs provide an estimate of the Signal-to-Noise Ratios as a function of frequency and thus high-SNR region can be emphasized. Thus this new delay estimation procedure is *adaptive*.

A new technique for the detection of trajectories of targets has been presented. This involves a straight-line Hough Transform (HT), thresholding and simple transformations in the Hough space and an inverse HT. This new technique circumvents the problem of storing the curve as a list of line segments along with their orientations and also problems of high dimensionality. Performance of this technique in the presence of noise was also demonstrated quite successfully.

Our effort this year has also resulted in techniques dealing with two levels of processing required for the task of describing 3D scenes: the 2D image level detecting features such as edges, lines and corners in images, and the 3D scene level extracting and matching 3D structures in range images. Our principle results include:

1. *Description from edge information (2D Image Level)* The method to generate a scene description from edge information is explored in order to find line segments. The Canny operator is used to obtain abrupt intensity changes instead of the Nevatia-Babu operator. To generate stable and exact line segments, we have tracked the edge pixels to detect corners using various size of detectors. We will generates corner points using various size of the detection distance in order to get stable and exact corner points. The coarsest level gives the search area of corners, while the finest level gives candidate positions of corners.

2. *Extraction and matching 3D structures (3D Scene Level)* We have investigated a method for building recognition-oriented models of 3D objects by representing explicitly the viewer-dependent geometry of the object and by including explicit guidelines for the tree-search procedure. The structure of the model leads to a fast recognition program which requires only a simple preprocessing of the measured data. The method has been proven feasible on a sample set of moderately complicated objects.

Chapter 7

Publications, Presentations, and Staff Supported

7.1. Staff Supported

Electrical and Computer Engineering:

David Casasent (Professor), Principal Investigator

B.V.K. Vijaya Kumar (Assistant Professor), Associate Principal Investigator

Yeou-Lin Lin (Graduate Student)

R. Krishnapuram (Graduate Student)

The Robotics Institute:

Arthur C. Sanderson (Professor), Principal Investigator

John Willis (Graduate Student)

Nanda Alapati (Graduate Student)

Computer Science

Takeo Kanade (Professor), Principal Investigator

In So Kweon (Graduate Student)

Ellen Walker (Graduate Student)

7.2. Publications

Electrical and Computer Engineering.. (from start of contract)

1. B.V.K. Vijaya Kumar and C. Carroll. "Loss of Optimality in Cross Correlators", JOSA-A, Vol. 1, 1984, pp. 392-397.

2. D. Casasent and V. Sharma. "Feature Extractors for Distortion-Invariant Robot Vision", Optical Engineering, Vol. 23, September/October, 1984, pp. 492-498.

3. B.V.K. Vijaya Kumar. "Lower Bound for the Suboptimality of Cross-Correlators", Applied Optics, Vol.23, July 1984, pp. 2048-2049.

4. D. Casasent, A. Goutzoulis and B.V.K. Vijaya Kumar, "Time-Intergrating Acousto-Optic Correlator: Error Source Modeling", Applied Optics, Vol.23, September 1984, pp.32430-3237.

5. R.L. Cheatham and D. Casasent, "Hierarchical Fisher and Moment-Based Pattern Recognition", Proc. SPIE, Vol. 504, August 1984, pp. 19-26.

6. D. Casasent and R.L. Cheatham, "Hierarchical Feature-Based Object Identification", OSA Topical Meeting on Machine Vision, March 1985.

7. D. Casasent, "A Recent Review of Holography in Coherent Optical Pattern Recognition", Proc. SPIE, Vol. 532, January 1985.

8. D. Casasent, "Hybrid Optical/Digital Image Pattern Recognition: A Review", Proc. SPIE, Vol. 528, January 1985.

9. W.T. Chang and D. Casasent, "Chord Distributions in Pattern Recognition: Distortion-Invariance and Parameter Estimation", Proc. SPIE, Vol. 521, November 1984, pp. 2-6.

10. W.T. Chang and D. Casasent and D. Fetterly, "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition", Proc. SPIE, Vol. 507, August 1984, pp. 9-18.

11. D. Casasent and R.L. Cheatham, "Image Segmentation and Real-Image Tests for an Optical Moment-Based Feature Extractor", Optics Communications, 51, September, 1984, pp. 227-230.

12. D. Casasent, "Coherent Optical Pattern Recognition: A Review", Optical Engineering, 24, Special Issue, January 1985, pp. 25-32.

13. D. Casasent and V. Sharma, "Feature Extractors for Distortion-Invariant Robot Vision", Optical Engineering, 23, September/October, 1984, pp. 492-498.

14. D. Casasent, B.V.K. Vijaya Kumar and Yeou-Lin Lin, "Subpixel Target Detection and Tracking", SPIE, Vol. 726, October, 1986.

15. B.V.K. Vijaya Kumar and Srikanth Rajan, "Subpixel Delay Estimation Using Group Delay Functions", SPIE, Vol. 697, October, 1986.

16. D. Casasent and R. Krishnapuram, "Detection of Target Trajectories Using the Hough Transform", Applied Optics, Vol.26, Janurary 1987, pp.247.

The Robotics Institute

1. J.L. Crowley and A.C. Sanderson, "Multiple resolution representation and probabilistic matching of 2-D grey-scale shape," Proc. 2nd IEEE Soc. Workshop on Computer Vision, Representation, and Control, May, 1984, pp. 95-105.

2. J. Willis, "RAPIDbus: Design of an Extensible Multiprocessor Structure", Master's thesis, Carnegie-Mellon University, May, 1984.

3. J.C. Willis, A.C. Sanderson, N.K. Alapati, "Rapidbus: Design of an Extensible Multiprocessor Structure", Technical Report 84-13, Carnegie-Mellon Robotics Institute.

Computer Science

1. M. Herman and T. Kanade, "The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images", Technical Report CMU-CS-84-102, Carnegie-Mellon University, February 1984.

2. M. Herman, T. Kanade and S. Kuroe, "Incremental Acquisition of a 3-D Scene Model From Images", IEEE Trans. PAMI, PAMI-6, 1984.

3. T. Kanade(ed.), "Three-Dimensional Vision Systems", Kluwer, Boston 1985.

4. M Herman, "Representation and Incremental Construction of a 3-D Scene Model", Technical REport CMU-CS-85-103, Carnegie-Mellon University, January 1985.

5. Y. Ohta and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7:2, 1985, pp. 139-154.

6. F. Tomita and T. Kanade, "A 3D Vision System: Generating and Matching Shape Description in Range Images", in *Robotics Research 2*, The MIT Press, 1985, pp 35-42.

7. D. Smith and T. Kanade, "Autonomous scene description with range imagery", CVGIP, 31, 1985, pp. 322-334.

8. M. Herman and T. Kanade, "The 3D MODAIC Scene Understanding System", *From Pixels to Predicates*, A. Pentland (ed.), Ablex Publishing Corp, 1986, pp., 322-358.

7.3. Conference Presentations and Seminars

Electrical and Computer Engineering--

1. D. Casasent, "Fourier Transfer Feature-Space Studies", Presented at SPIE, November, 1983, Cambridge, MA.

2. D. Casasent, "Synthetic Discriminant Functions", Presented at DARPA, February 1984.

3. D. Casasent, "Robotics Applications of Optical Data Processing", Presented at Polytechnic Institute of New York, February 1984.

4. D. Casasent, "Optical Information Recognition", Presented at the Air Force Office of Scientific Research, May 1984.

5. D. Casasent, "Parallel Coherent Optical Processor Architectures and Algorithms for ATR", Presented at the Workshop on Algorithm Guided Parallel Architectures for Automatic Target Recognition, Leesburg, Virginia, July 1984.

6. D. Casasent, "Hierarchical Fisher and Moment-Based Pattern Recognition", Presented at the SPIE Conference in San Diego, California, August 1984.

7. D. Casasent, "SDF Control of Correlation Plane Structure for 3-D Object Representation and Recognition", Presented at the SPIE Conference in San Diego, California, August 1984.

8. D. Casasent, "Research in the Center for Optical Data Processing", Presented at Carnegie-Mellon University, ECE Sophomore Seminar, October 1984.

9. D. Casasent, "Advanced Multi-Class Distortion-Invariant Pattern Recognition", Presented at the University of Pittsburgh, Center for Multivariate Analysis, Pittsburgh, PA, October 1984.

10. D. Casasent, "Optical Information Processing", Presented at George Mason University, Washignton, D.C., October 1984.

11. D. Casasent, "Chord Distributions in Pattern Recognition", Presented at the SPIE, Cambridge, MA, November 1984.

12. D. Casasent, "A Recent Review of Holography in Coherent Optical Pattern Recognition", Presented at SPIE, January 1985.

13. D. Casasent, "Hybrid Optical/Digital Image Pattern Recognition: A Review", SPIE, January 1985.

14. D. Casasent, "Optical Pattern Recognition and Optical PRocessing", Presented at Fairchild Weston, Long Island, New York, January 1985.

15. D. Casasent, "Hierarchical Feature-Based Object Identification", Presented at OSA Topical Meeting on Machine Vision, Lake Tahoe, NV., March 1985.

16. B.V.K. Vijaya Kumar, "Subpixel Delay Estimation Using Group Delay Functions", presented at SPIE, San Diego, August 1986

The Robotics Institute

1. J.C. Willis, A.C. Sanderson, "Segmented Crossbar Switching: Design for a Hybrid Message Passing Structure", ICCD 1985.

Computer Science

1. M Herman, "Representation and Incremental Construction of a 3-D Scene Model", Presented at the Workshop on Sensors and Algorithms for 3-D Machine Perception, Washington, D.C., August 1983.

2. F. Tomita and T. Kanade, "A 3D Vision System: Generating and Matching Shape Description in Range Images", presented at the 2nd International Symposium of Robotics Research, Kyoto, Japan, August 1984.

3. D. Smith and T. Kanade, "Autonomous Scene Description with Range Imagery", presented at the 15th DARPA Image Understanding Workshop, October 1984.

4. M. Herman, "Generating Detailed Scene Descriptions from Range Images", presented at the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, March 1985

5. Y. Ohta and T. Kanade, "Stereo by Intra- and Inter-Scanline Search

Using Dynamic Programming", presented at the Ninth Internation Joint Conference on Artificial Intelligence, Los Angels, Aug. 1985.

6. M. Hebert and T. Kanade, "The 3-D Profile Method for Object Recognition", Presented at IEEE International Conference on Computer Vision and Pattern Recognition, San Francisco, June 1985.

7. M. Hebert and T. Kanade, "Outdoor Scene Analysis Using Range Data", presented at International Conference on Robotics and Automation, San Francisco, April, 1985.

END

DATE

FILMED

4-88

DTIC