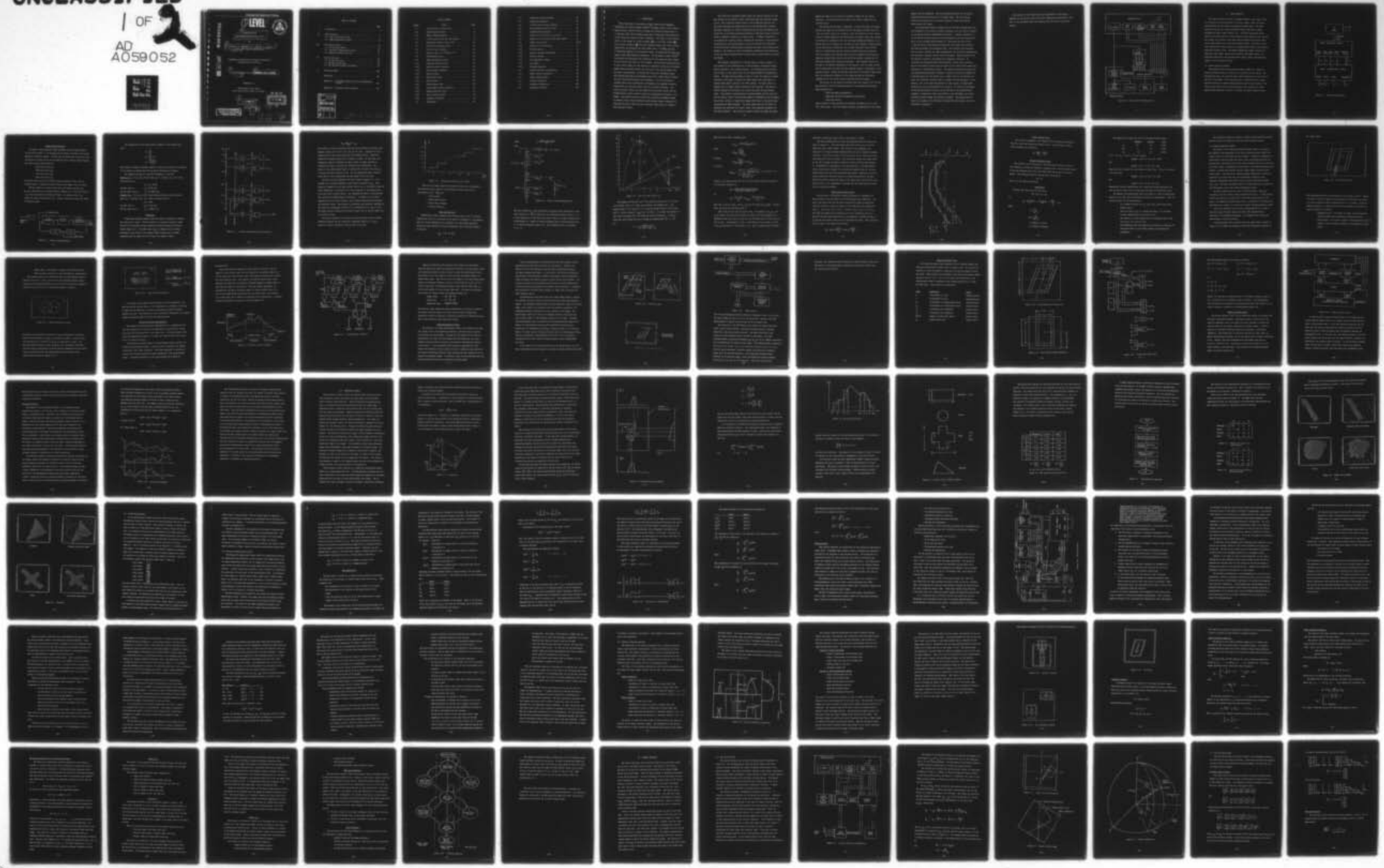
AD-A059 052   NEW MEXICO STATE UNIV LAS CRUCES ENGINEERING EXPERIM--ETC   F/G 17/8
A PRE-PROTOTYPE REAL-TIME VIDEO TRACKING SYSTEM.(U)
JAN 77   G M FLACHS, W E THOMPSON, R J BLACK   DAAD07-76-C-0024

UNCLASSIFIED   NL

1 OF

AD
A059052



Engineering Experiment Station

1 LEVEL II

NEW MEXICO STATE UNIVERSITY

6

A PRE-PROTOTYPE REAL-TIME VIDEO TRACKING SYSTEM.

by

10

G. M. Flachs, W. E. Thompson, R. J. Black, J. M. Taylor
W. Cannon, R. Rogers, and Yee Hsun U

Department of Electrical and Computer Engineering
New Mexico State University

9 Final Report for Contract 15 DAAD07-76-C-0024

Submitted to:

White Sands Missile Range
White Sands Missile Range, New Mexico

11 17 January 1977

12 162p.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

I. INTRODUCTION

This final report is presented to White Sands Missile Range ~~in~~ fulfillment of Contract DAAD07-76-C0024. New The major thrust of the contract ~~is~~ to develop a state-of-the-art hardware and software configuration for a real-time video (RTV) tracking system and to investigate the feasibility of the system by testing its major components. The result of the research effort is the design of a video processor ~~[2]~~, a structural tracker [redacted], and a control processor ~~[3]~~ for a RTV tracking system. The video processor synchronizes and digitizes the video signal from a TV camera, performs a statistical analysis of the digitized image, and separates the image into target, plume, and background components. ABSTRACT The structural tracker locates the position and determines the orientation of the target and plume images in the field-of-view (FOV) of the TV camera. The structural characteristics of the located target are then analyzed to establish a structural confidence weight which describes how well the structure of the located target fits the object being tracked. By having this structural confidence weight, the tracking algorithm can discriminate and track a large class of targets [5] with increased performance and reliability. It is the structural tracking feature that significantly distinguishes the proposed structural tracker from the contrast trackers that are currently available. The control processor closes the loop between the structural tracker and the tracking optics control system to keep the tracking optics pointed at the target. The control processor utilizes the structural confidence weight to combine current target coordinates with previous target coordinates to orientate the optics toward the next expected target position, forming a fully automatic system.

The structural confidence weight allows the control processor to rely more heavily on the previous target coordinates when the structural weight is low. This structural weight feature is very important when the target passes through noisy backgrounds such as the mountain/sky interface. The major components of the RTV tracking system have been breadboarded and tested [2] in the image processing laboratory at New Mexico State University using a standard TV camera as the sensor input. A dynamic simulation of the complete RTV tracking system, including the Contraves F Cinetheodolite tracking mount, has been developed to test and evaluate the performance of the entire RTV tracking system, including the target and tracking optics dynamics. The results of these investigations demonstrate the feasibility of the structural tracking method at frame rates of sixty frames per second and higher.

The hardware configuration of the RTV system is shown in Figure 1.1 and consists of the tracking optics, video processor, structural tracker, control processor, and a control monitor. The tracking optics interfaces the TV camera to the optics track of the Contraves Model F Cinetheodolite mounts. An image rotating element is used to rotate the image for V-angle tracking, a zoom lens to enlarge the target image, and a CCD camera to provide video signal outputs to the video tracker. Position control is required for the image rotating element and the zoom lens. The position control signals are provided by the control processor during automatic track. The problem of interfacing the rotating element and the zoom lens into the optics track of the Contraves Model F Cinetheodolite and designing the control system is a significant design effort that is currently being investigated by WSMR personnel. The video signal from the TV camera is digitized and separated into target, plume, and background components by the video processor. The structural tracker locates the target and plume images and computes the structural confidence weight for the control processor. The control processor computes the control signals for the tracking optics.

The system has two modes of operation. In the first mode, the system locates the target in the field-of-view of the camera, provides the boresight correction signals for automatic track, and records the video output of the TV camera along with the tracking data by using a video tape recorder. A TV monitor is provided to show the real-time tracking performance, and a manual joystick is available to override the fully automatic tracking system. In the second mode of operation, the recorded video data can be played back at a reduced frame rate for further detailed study and evaluation. After the mission, the recorded video tape can be played back through the system at reduced frame rates to provide better quality tracking data by using more sophisticated analysis techniques. This playback feature will be used in the initial development of the system to test the video tracker and to characterize and match the RTV tracking system to the dynamics of the control systems. Having the recorded video tape of the target flight along with the estimated coordinates of the target provides an interesting augmentation of the current optical film processing techniques.

The results of the research performed under this contract have focused the solution of the RTV tracking problem on three significant problems. These problems are:

- Real-time Image Decomposition
- Real-time Structural Recognition and Location
- Real-time Control

Some solutions of these problems are discussed in Chapters II, III, and IV of this report. The first problem concerns the separation of the target images from the background. The second problem concerns the structural characterization and location of the target image. The last problem concerns the prediction of the control signals to point the tracking optics toward the target.

A pre-prototype hardware configuration for the RTV tracking system has been developed in the light of present technology, and the critical elements of the system have been breadboarded and tested. A dynamic simulation of the entire RTV tracking system has been developed to test and evaluate all parts of the tracking system (Chapter V). Resulting from this effort, a state-of-the-art RTV tracking configuration has evolved which utilizes the high performance LSI processing chips. Four separate LSI processors will be used to perform the image decomposition, the projection computations, the structural tracking, and the control processor. These processors will be running in parallel, performing their designated functions. By utilizing the programmed logic design method to realize these functions, much of the complex control logic is stored in the control store memory and executed by a microprocessor system. The hardware structure resulting from the programmed design method resembles a computer architecture composed of a microprocessor for the control unit, a memory unit for its control program, and peripheral interface units to connect the microprocessor to user devices. Due to the speed requirements of this project, the high speed Schottky bit slice microprocessors are required. To assist in the design of the microprocessors structure and the writing of the required microprograms, a significant effort was devoted to the development of a microprocessor oriented design and simulation program. This program will be used in the design of the processors required for this project, and it is described in Appendix A.

The results of this research have been published in four papers (Appendix B) and two more papers have been submitted for publication. [31], [32] It is anticipated that other publications will result from these studies.

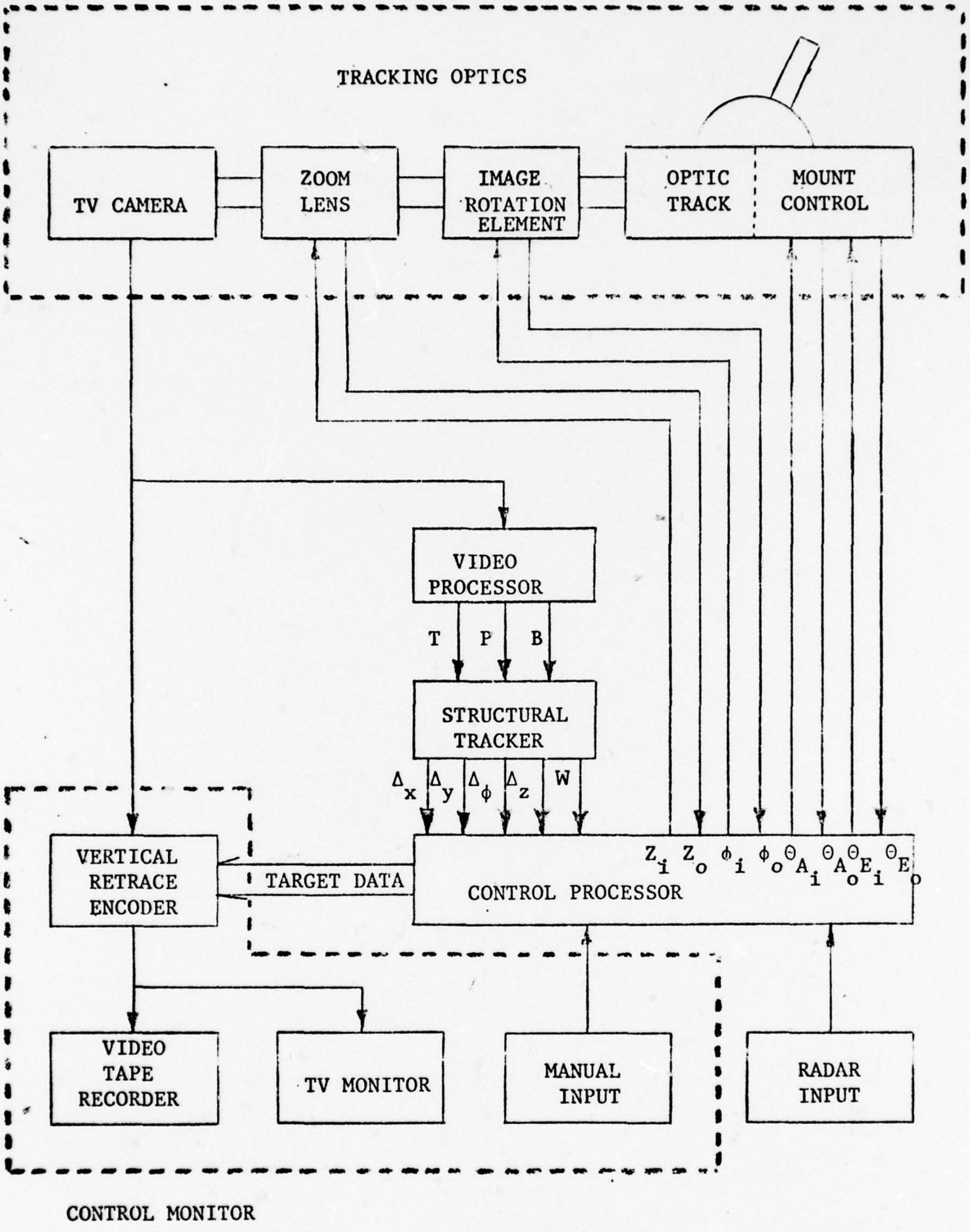


Figure 1.1 Prototype RTV tracking system

II VIDEO PROCESSOR

The video processor receives a standard composite video signal from the TV camera and provides binary (TTL level) target, plume, and background signals for locating and tracking the target image. The video processor consists of a video acquisition module (VAM) and an image decomposition module (IDM) (Figure 2.1). The video acquisition module receives the video output of the TV camera by phase locking to the vertical sync signal and performs a high speed analog-to-digital conversion (ADC) of the video signal into L discrete intensity levels. The output of the video acquisition system consists of a timing strobe and the corresponding digitized intensity level. The image decomposition module receives the digitized video signal; statistically anslyzes the target, plume, and background components; and provides binary target, plume, and background signals to the structural tracker.

2.1 Video Acquisition Module

The purpose of the video acquisition module (VAM) is to generate a precise digitized picture of the scene in the FOV of the TV camera. The VAM generates an $N_3 \times N_2$ intensity pixel grid representation of the scene in the FOV of the camera. Each intensity pixel is digitized into L levels. The VAM consists of a master timing generator that is phase locked to the vertical sync signal for providing the basic pixel timing, and a high speed analog-to-digital converter to provide the pixel intensity levels.

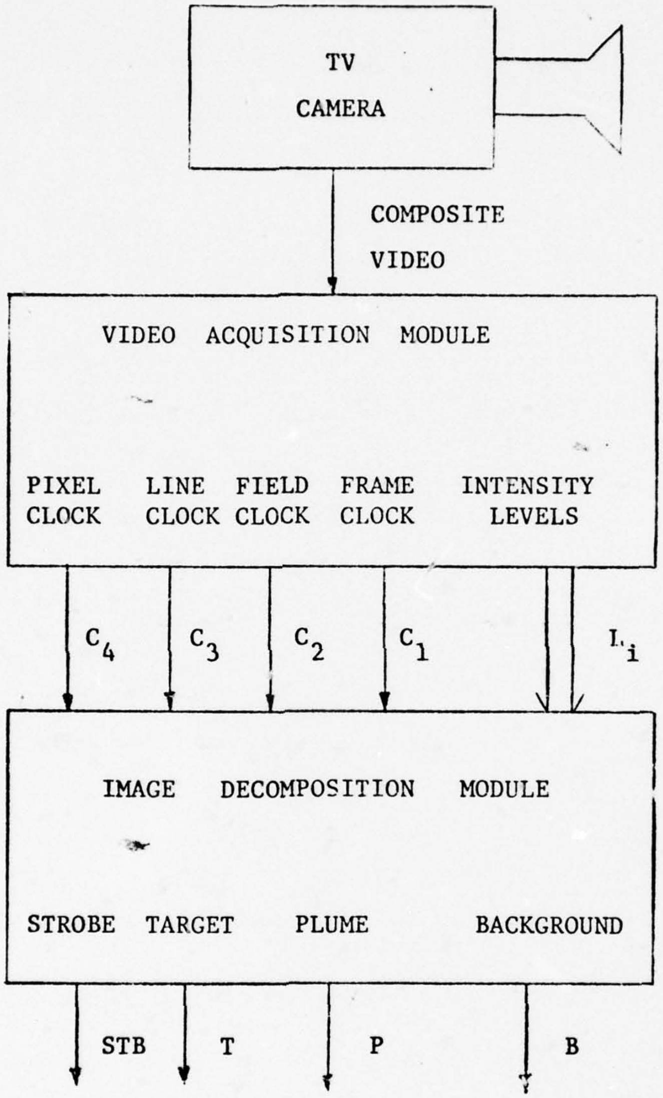


Figure 2.1. Video processor module

Master Timing Generator

The master timing generator (MTG) provides the basic timing signals for the RTV tracking. It is designed with a modular structure to be easily adapted to different cameras. The MTG uses the field rate clock (C_1) of the TV picture as a master clock for deriving all other required timing signals. The output clocks from the MTG are:

- field rate clock (C_1)
- frame rate clock (C_2)
- line rate clock (C_3)
- pixel rate clock (C_4)

The master clock (C_1) can be derived from the composite video, from an external clock, or directly from the vertical sync signal from the camera.

The MTG consists of a phase locked loop with three counters, (N_1, N_2, N_3), operating at the desired harmonic (Figure 2.2). The first counter , N_1, counts the number of fields-per-frame. The second counter, N_2, counts the number of pixels-per-line. Finally, counter N_3 counts the number of lines-per-frame.

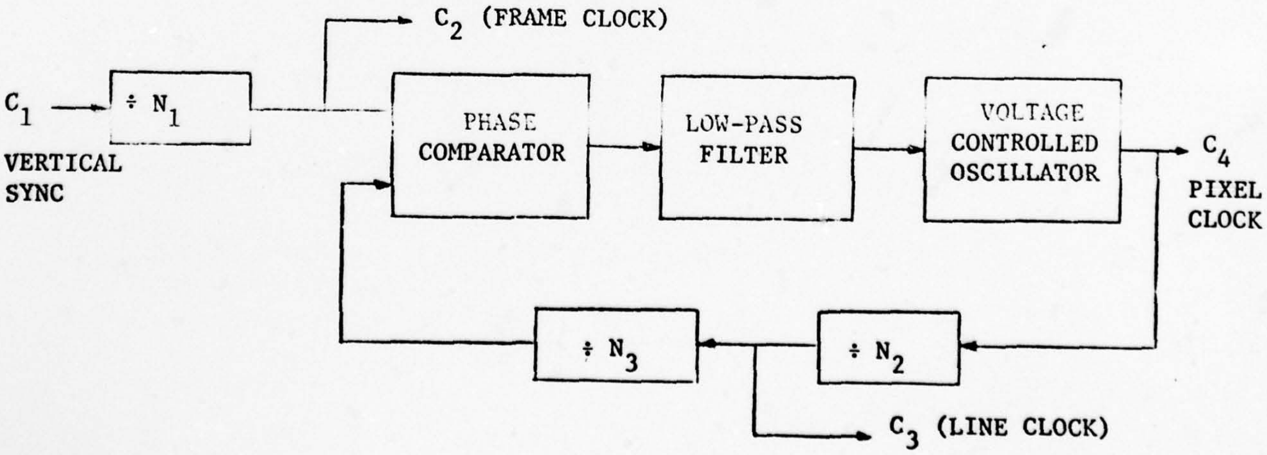


Figure 2.2. Master timing generator

The frequencies of the clocks shown in Figure 2.2 are derived from f_1 by

$$f_2 = \frac{f_1}{N_1}$$

$$f_3 = \frac{N_3}{N_1} f_1$$

$$f_4 = \frac{N_2 N_3 f_1}{N_1}$$

Since there is always an integral number of lines in one frame, the inclusion of the counter N_1 insures that the N_3 counter divides by an integer.

Two examples are given to show the flexibility of the MTG.

<u>Example 2.1.1</u>: Let C_1 be a 60 Hz clock, $N_1 = 1$, and $N_2 = N_3 = 512$. Then the frame rate is

$$f_2 = f_1 = 60 \text{ Hz}$$

the line rate is $f_3 = 30.72$ K Hz

and the pixel rate is $f_4 = 15.72864$ M Hz

<u>Example 2.1.2</u>: Let C_1 be a 60 Hz clock derived from the camera vertical sync, $N_1 = 2$, and $N_2 = N_3 = 512$. Then the frame ratio is

$$f_2 = 30 \text{ Hz}$$

the line rate is $f_8 = 15.36$ K Hz

and the pixel rate is $f_4 = 7.86432$ M Hz

ADC Design

A high speed analog-to-digital converter (ADC) is designed to digitize the video into L levels. The ADC consists of a precision resistor ladder network with high speed voltage comparators differentiating the intensity levels (Figure 2.3). The video input (V_{in}) is compared to the voltage developed at each level of the resistor ladder network with a voltage comparator and the output of the n^{th} level is a logical 1 when

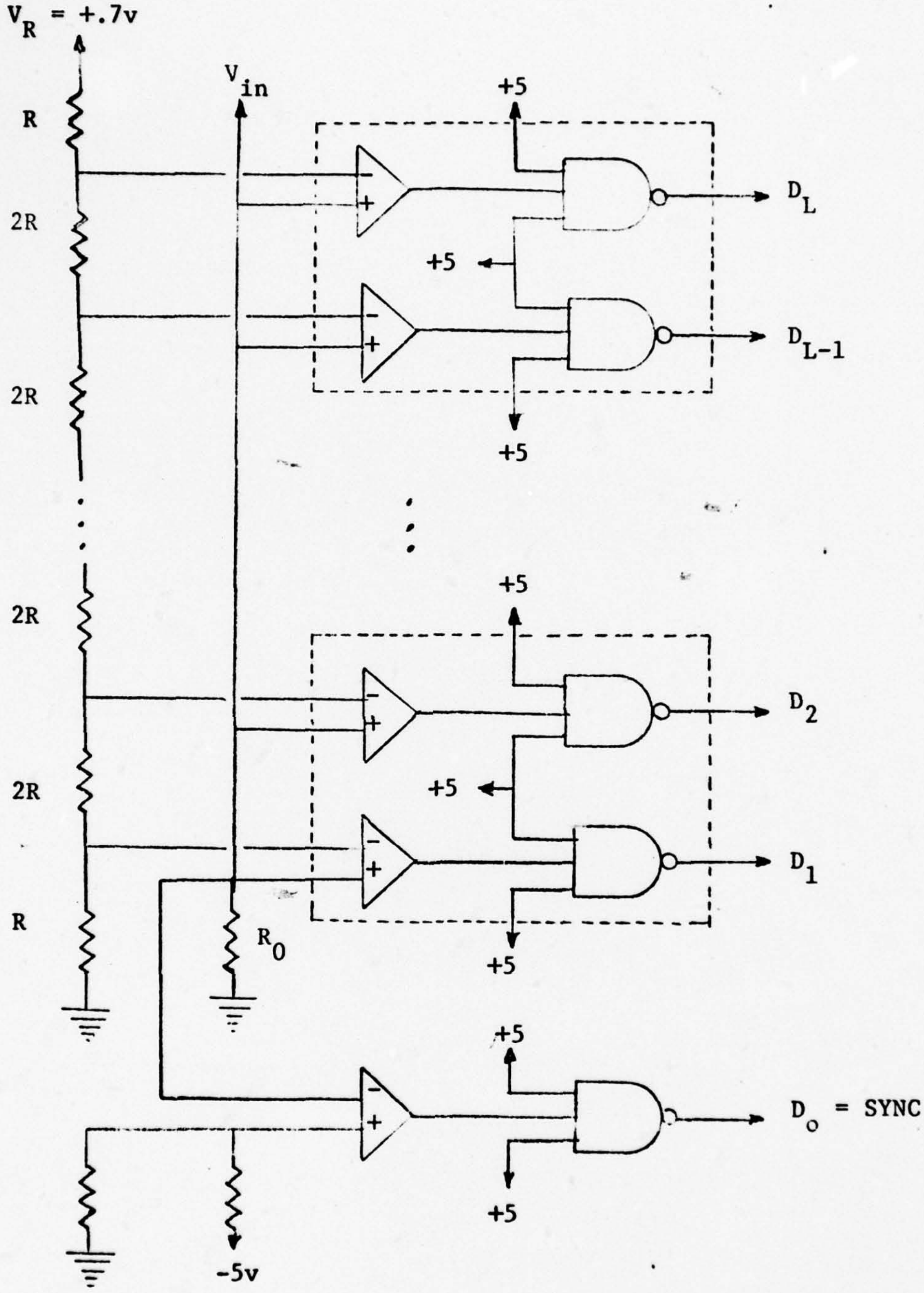


Figure 2.3. L Level parallel ADC and sync detector

$$V_{in} \geq \frac{2n - 1}{2L} V_{ref}$$

This yields an L-level conversion with the level detection occurring at the midpoint between the (n-1)th level and the nth level. Signetics 521 dual comparators are used to differentiate the intensity levels. These comparators can operate either with or without a strobe. At this time, the comparators operate unstrobed, and their output is simply analyzed by combinational logic to provide the intensity level information. The output of the combinational logic is strobed with the pixel rate clock C_4 to define the pixel intensity level. The 521 comparators have a maximum delay time of twelve nanoseconds and are usable at up to 40 M Hz.

The basic circuit diagram of the ADC is shown in Figure 2.3. It is assumed that a standard composite video signal is applied at V_{in}. This signal is normally one volt peak-to-peak, and it is d.c. restored so that the video information is contained in the top 70 percent of the signal, which is the part that is above ground. The remaining 30 percent is below ground and contains the synchronization information. In Figure 2.3, the lower comparator acts as a sync stripper, and the upper comparators differentiate the intensity levels. The resistor R_o represents the equivalent impedance of the source and terminating resistor on the transmission line ($R_o \approx 37.5\Omega$). The ladder network divides the reference voltage into the desired number (L) of intensity levels.

The total resistance of the lader network is R_T=2LR, and the voltage difference between any two consecutive comparator inputs is $V = V_R/L$. Figure 2.4 shows a normalized transfer curve of the ADC .

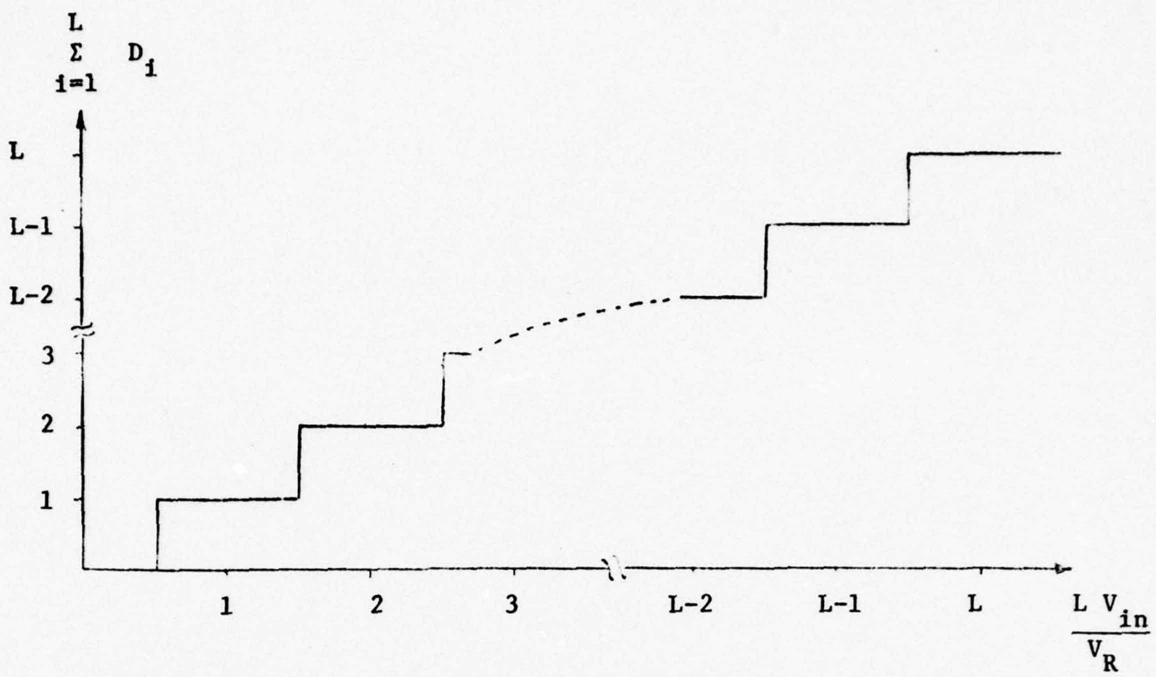


Figure 2.4. Normalized transfer curve of ADC

There are four major sources of errors that need to be considered in selecting the ladder resistor R and defining an upper bound on the digitization error. These are:

- bias currents
- offset bias currents
- offset input voltages
- resistor tolerance

Bias Current Error

Experimental studies indicate that the bias current error is the most significant, and it is a function of the value of resistance R. By assuming that n-1 of the comparators are on (Figure 2.5), the voltage developed across the input terminals of the nth comparator due to the bias currents is given by

$$V_{E_n} = -(Z - Q) I_B R$$

where

$$Z = \frac{(2n-1)\ [n-(L+1)]^2}{2L}$$

and

$$Q = \frac{(n-1)}{R} R_o = \frac{n-1}{K}\ ,\ (K = R/R_o)$$

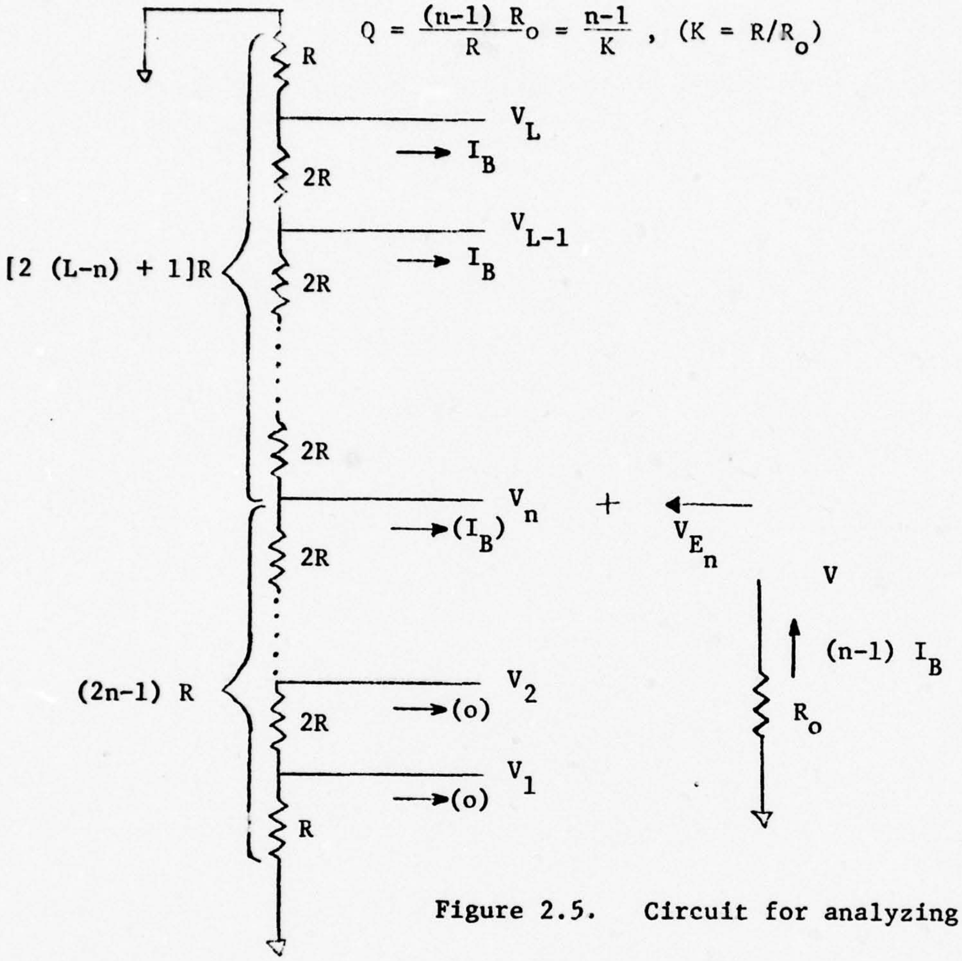


Figure 2.5. Circuit for analyzing errors

The error voltage V_{E_n} which represents an apparent input voltage due to the bias currents is a direct function of the difference between Z and Q. Z is a cubic equation in n with three real roots, and Q is a linear equation in n with parameter K. Figure 2.6 shows the general form of these equations for L = 8 and two different values of K. The normalized error y is defined Y = Z - Q.

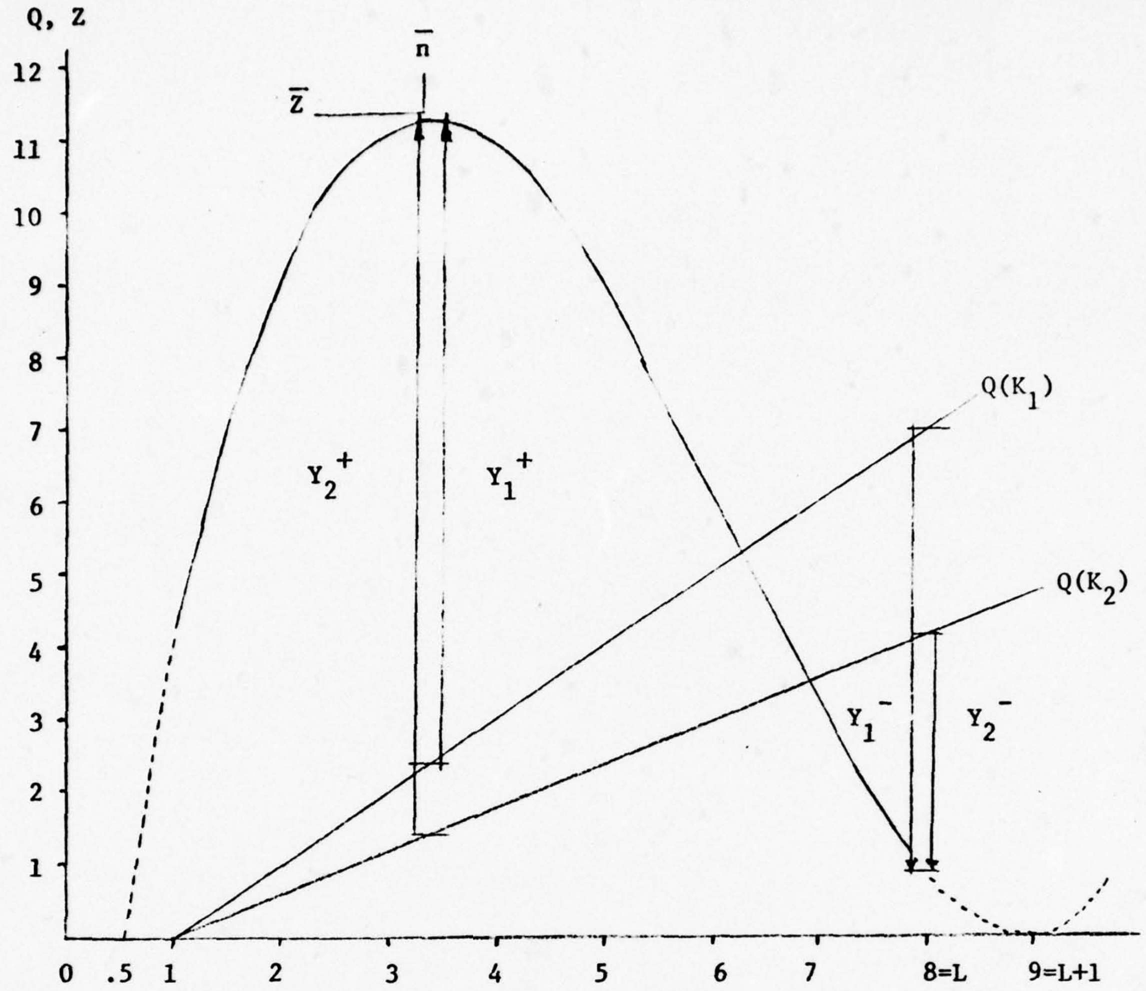


Figure 2.6. Plot of Q and Z $(K_2 > K_1)$

The maximum differences Y^+ and Y^- are clearly functions of K, and there is an optimal value of $K = R/R_o$ that minimizes the maximum error. The maximum value of $|Y^-|$ occurs at $n = L$, and the maximum value of $|Y^+|$ occurs near $\bar{n} = (L+2)/3$, where $\bar{Z} = Z_{max} = (2L + 1)^3/54L$. It is clear from Figure 2.6 that when K increases $|Y^+|$ also increases and $|Y^-|$ decreases. It can be shown that the minimum bias error voltage is obtained when $|Y^+| = |Y^-|$ and

$$K = \frac{36L\ (L - 1)}{4L^3 + 6L^2 + 30L - 13}$$

This value of K gives a maximum error

$$|Y|_{max} = \frac{4L^3 + 6L^2 - 6L + 5}{36L}$$

where

$$|Y|_{max} = \left|\frac{V_{E_n}}{I_B R}\right|_{max}$$

Letting

$$|X|_{max} = \left|\frac{V_{E_n}}{I_B R_o}\right| = K\ |Y|_{max}$$

we obtain

$$|X|_{max} = \frac{(L-1)\ [1 - \dfrac{6L - 5}{2L^2\ (2L + 3)}\ ]}{[1 + \dfrac{30L - 13}{2L^2\ (2L + 3)}\ ]} \lesssim L-1, \text{ for } L > 8$$

Finally, the relative step error due to bais currents when the resistance R is selected optimally is

$$|E| = \frac{\text{Bias Current Error Voltage}}{\text{One Half Voltage Step}}$$

or

$$|E| = \frac{2\ L\ I_B\ R_o}{V_R}\ |X|_{max} \lesssim \frac{2L\ (L-1)\ I_B\ R_o}{V_R}$$

Note that if $|E|$ is small, then $I_B \cdot R_o$ must be small and V_R large. Furthermore, the error $|E|$ increases with L^2.

This result may be used in several ways. For example, if I_B, R_o, and V_R are known, then the maximum number of levels L can be determined for a normalized bias error $|E| \leq 1$, that is, for an actual error of less than one half a step. In particular, if $V_R = 1$ volt and $R_o = 37.5\ \Omega$ then $L \leq 25.8$. If V_R is increased to 3 volts then $L \leq 44$. Thus, it appears that 32 levels represent a theoretical upper bound on the number of levels.

The above calculations are tested experimentally and the results are shown in Figure 2.7. The experimental procedure consists of a 32 level ADC driven with a ramp voltage. The outputs of the comparators are connected to a simple resistor summing network to form a digital-to-analog converter. The circuit is then run at slow speed and the input and output are connected to an x-y recorder. The difference between the actual value of the input voltage for which transitions occurred and the ideal values is established as a percentage of the half-step size. The experimental error curves given in Figure 2.7 show that the maximum error is, in fact, significantly less than the theoretical upper bound. Obviously, the error curves include all the other sources of errors attributable to the ADC. Since the curves do follow the shape of the theoretical curves for the bias current error, it is reasonable to conclude that the major source of error is due to the bias currents.

Offset Bias Current Errors

The error due to the offset currents is caused by a difference in bias currents in the two halves of the differential input amplifier. The worst case occurs when all the offset currents are added to (subtracted from) the current at node V of Figure 2.5 and subtracted from (added to) node V_n. This model can be handled. However, since ΔI_B is less than I_B and since the maximum error produced occurs for a different value of n than the one previously determined, it is reasonable to simply add ΔI_B to I_B and use the previous calculation to determine the error. Thus, we obtain the error due to the offset bias currents

$$E_1 = 2L(L-1) \frac{I_B R_o}{V_R} + 2L(L-1) \frac{\Delta I_B R_o}{V_R}$$

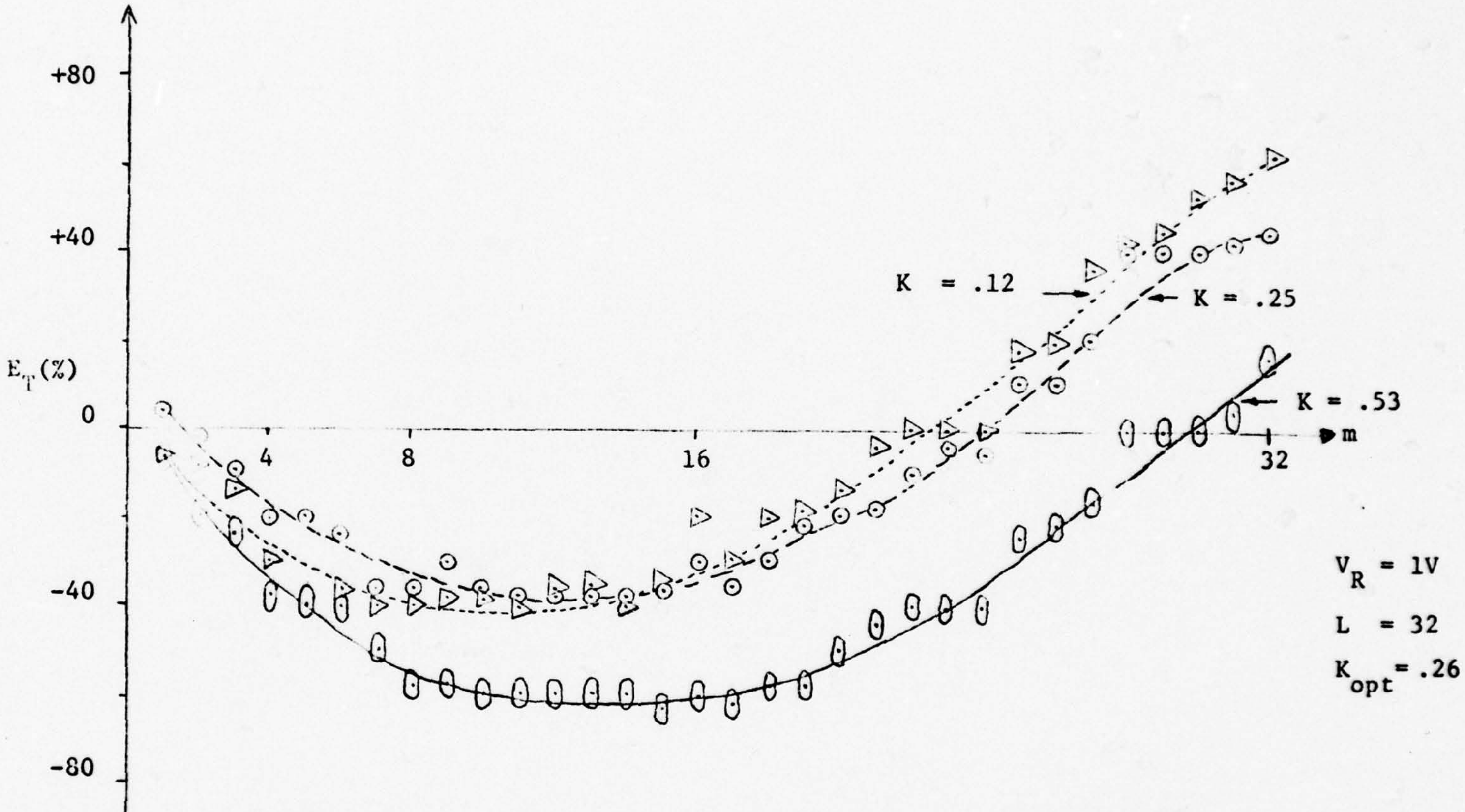


Figure 2.7. Relative error as a function of n (experimental)

Offset Voltage Error

Due to device mismatches, the comparator will not switch precisely at the point where the differential input is zero. The error due to this offset voltage (V_o) is given by

$$E_2 = \frac{2V_o}{\Delta V_n} = \frac{2\ V_o\ L}{V_R}$$

Resistor Tolerance Error

The tolerance of the resistors in the R-2R network causes an error in the voltage V_n at which the comparator switches level. The worst case occurs if all the resistors below node n are high (low) and all the rest are low (high). The resulting resistor tolerance error is given by

$$E_3 = Lx$$

where x is the resistor tolerance.

Total Error

Finally, the total error is boun'ed by

$$E_T = E_1 + E_2 + E_3$$

or

$$E_T = \frac{2\ L\ (L-1)}{M} + \frac{2\ L\ (L-1)}{\Delta M} + \frac{2V_D L}{V_R} + Lx$$

where

$$M = \frac{V_R}{I_B\ R_o}$$

$$\Delta M = \frac{V_R}{\Delta I_B\ R_o}$$

L = Number of Levels

X = Resistor Tolerance

The Signetics 521 comparator has the following specifications:

	Maximum	Typical	Unit
I_B	20	7.5	μ amps
ΔI_B	5	1.0	μ amps
V_o	7.5	6.0	m volts

For L = 32, V_R = 1, and x = .01 one obtains

$$E_T(\text{MAX}) = 149 + 37 + 48 + 32 = 266\%$$

and

$$E_T(\text{TYP}) = 56 + 7 + 38 + 32 = 133\%$$

where the error is given as a percentage of a half step. If V_R is increased to 3 volts then

$$E_T(\text{MAX}) = 50 + 12 + 16 + 32 = 110\%$$

and

$$E_T(\text{TYP}) = 19 + 2 + 13 + 32 = 66\%$$

Experimental studies indicate that these results are quire pessimistic and that the actual total error is often much less than the theoretical error.

By comparing the experimental results of Figure 2.7 with the theoretical results, it is clear that the theoretical results are pessimistic. There are several reasons for the discrepancies.

- It is highly unlikely that I_B, ΔI_B, and V_o will attain their maximum values.
- Since the sign of ΔI_B is a random occurence, it is possible to have cancellations in its effect of E_1.
- The sign of V_o is also random and it may decrease rather than increase E_2 at any given level.
- The assumptions made involving the distribution of high and low resistance values in the ladder network is certainly most pessimistic.

The theoretical bounds do, however, provide error guidelines and also provide a method to establish an optimum ladder resistor network.

2.2 Image Decomposition Module

The purpose of the Image Decomposition Module (IDM) is to separate the digitized scene generated by the video acquisition module (VAM) into binary pictures of the target and plume images. A method is developed for statistically characterizing the target, plume, and background intensity distributions and deciding for a given input intensity whether it came from the target, plume, or background distributions. The method demonstrates the ability to rapidly and reliably separate target images from noisy background scenes. The problem of separating target images from a picture is approached in several ways with varying degrees of success on the Automatic Programmable Film Reader (APFR) at White Sands Missile Range [33], [34]. The most successful method on the APFR provides the basis for the development of the video image decomposition module.

As the TV camera scans the scene, the VAM provides a digitized intensity value for M pixels across each horizontal line. A video picture with N lines results in a discrete N x M matrix representation of the scene. At standard television frame rates, a resolution of 512 pixels per horizontal line results in a pixel rate of about 100 nanoseconds per pixel.

The basic assumption for the initial implementation of the IDM is that the target and plume images have some video intensities not contained in the immediate background - an assumption that based upon APFR studies is fairly reasonable.

A parallelogram frame is placed about the target image, as shown in Figure 2.8, to sample the background intensities immediately adjacent to the target image.

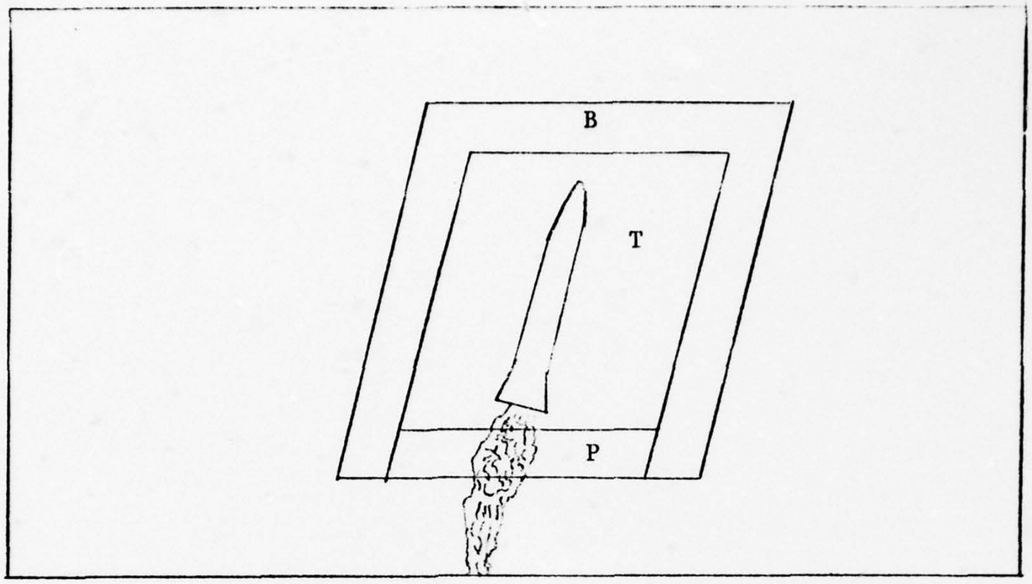


Figure 2.8. Parallelogram frame

The parallelogram frame is partitioned into two regions, B,and P. Region B is used to provide a sample of the background intensities, and region P is used to samplethe plume intensities when it is visible. Using the sampled intensities, a simple decison rule is developed to classify the pixels in region T as follows:

- Background points - All pixels in region T with intensities found in region B are classified as background points.
- Plume points - All pixels in region T with intensities found in region P, but not found in region B, are classified as plume points.

- Target points - All pixels in region T with intensities not found in either region B or P are classified as target points.

The classification can be illustrated with the Venn diagrams shown in Figures 2.8 and 2.9. Let I_S be the set of all possible intensity values occurring in the digitized picture, I_B be the set of intensities sampled in region B, and I_P be the set of intensities sampled in region P.

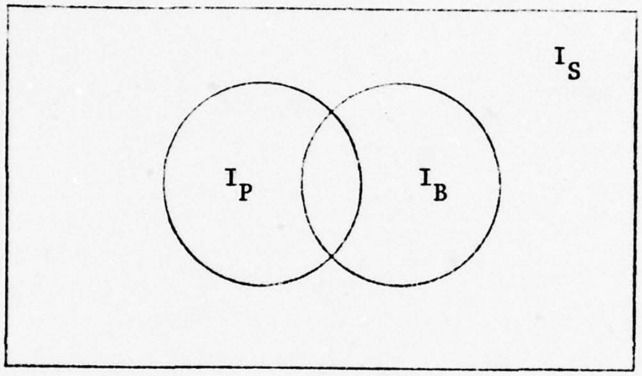


Figure 2.9. Image decomposition logic

All pixels with intensities contained in I_B are classified as background. Pixels with intensities in $I_P \cap \bar{I}_B$ are classified as plume. Finally, pixels with intensities not contained in $I_P \cup I_B$ are classified as target. The resulting classification is shown in Figure 2.10. The figure illustrates that the target classification is based upon the assumption that the target image is contained within the parallelogram frame (T) and has some intensities not found in regions P or B.

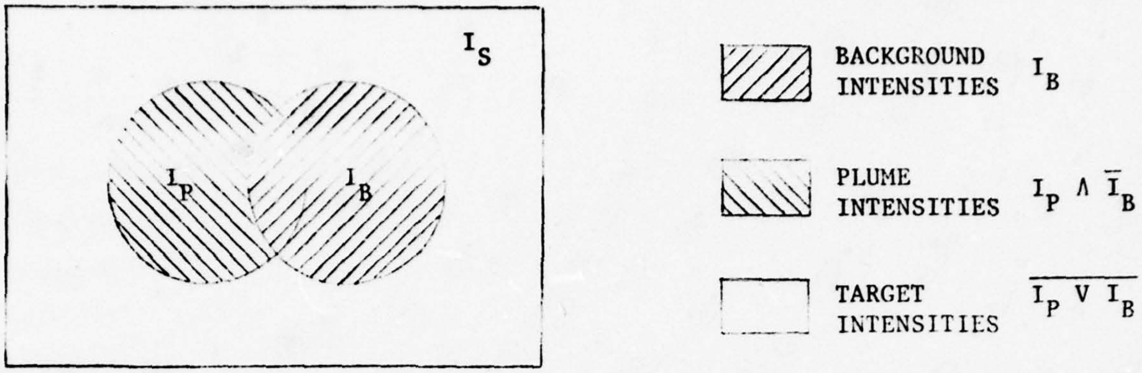

I_S
I_P
I_B
BACKGROUND INTENSITIES I_B
PLUME INTENSITIES $I_P \wedge \bar{I}_B$
TARGET INTENSITIES $\overline{I_P \vee I_B}$

Figure 2.10. Intensity classification

A variation of this classification method is used successfully on the APFR with actual mission films. It is demonsrated in a hardware realization at NMSU that the algorithm is capable of operating at speeds exceeding standard TV rates. The algorithm is also successfully implemented in a digital computer simulation model of the entire tracking system.

RTV Pre-prototype Implementation

The purpose of the pre-prototype implementation is to demonstrate and test the feasibility of the basic IDM algorithm for separating the target and plume from the background at fast pixel rates. Although standard video rates and bandwidths are used, it is shown that significantly faster input rates are easily achievable.

The pre-prototype IDM consists of a parallelogram frame controller to define the parallelogram frame, a counter array to accumulate the sampled intensities, and a pixel classifier. The frame controller is realized by an array of counters defining the region boundaries of the parallelogram frame. A detailed description of the pre-prototype IDM is given in reference [35].

The counter array is composed of three banks of counters: one for region P of the current frame, one for region B of the current frame, and one for region B of the previous frame. Each bank of counters has a counter for each intensity level. Each counter is initially loaded with a noise rejection threshold that is adjusted to provide immunity to random noise in high contrast regions of the picture. When the counter underflows, the intensity has occurred a sufficient number of times to decide that it is part of the region being characterized by the counter bank. The contents of a counter bank can be viewed as a histogram of the intensities. A typical intensity histogram for region B is shown in Figure 2.11.

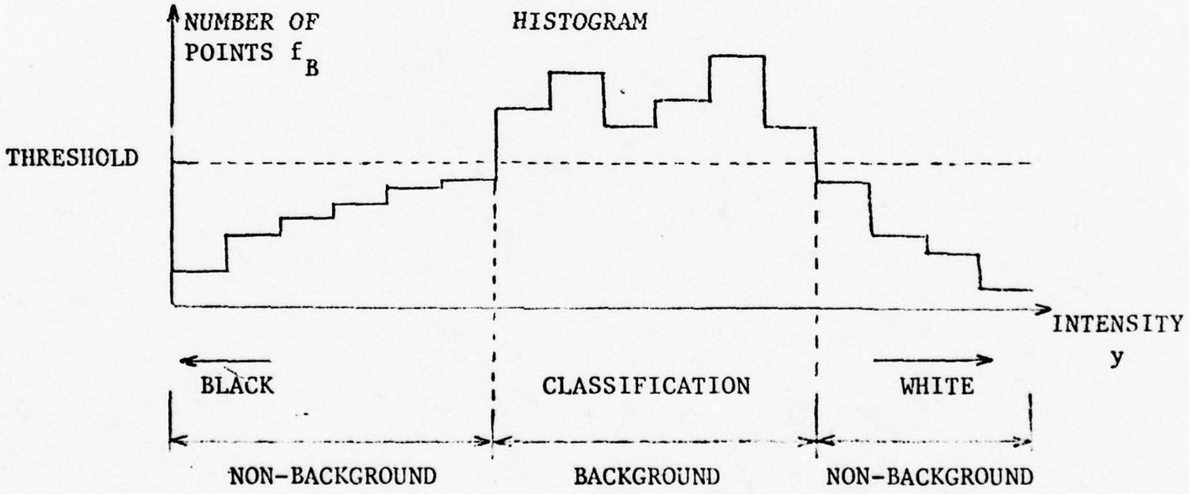


Figure 2.11. Typical counter histogram

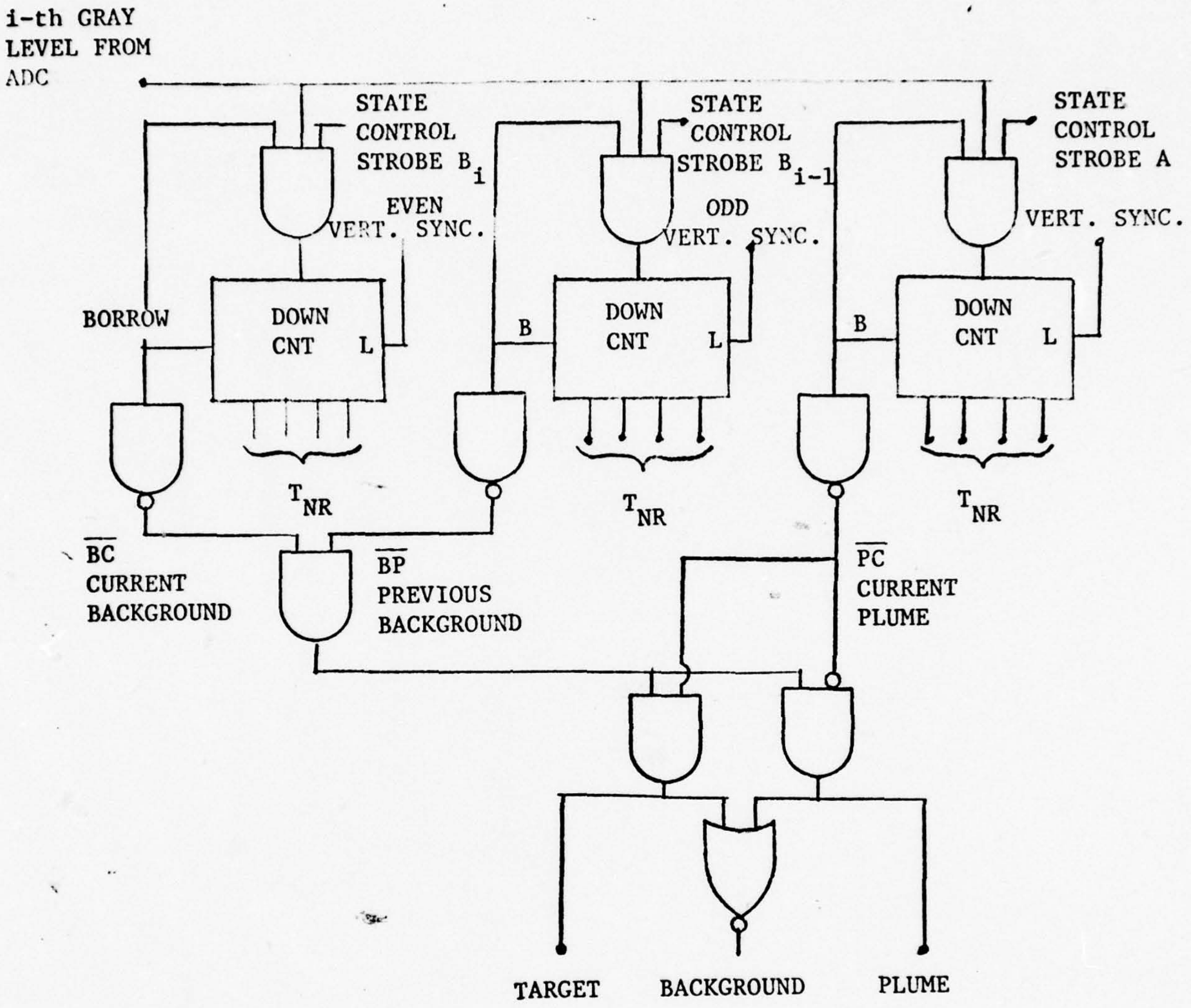


Figure 2.12. Pre-prototype classifier

Figure 2.12 shows the i^{th} intensity level counter in each counter bank and defines the logic that separates the picture into the target, plume, and background components within region T inside the parallelogram frame. The outputs of all the counters in each bank are wired-ORed together to three output buses. When the TV scan is in region T, the three buses define whether the measured intensity occurred in the background region B of the current frame (BC), the background region of the previous frame (BP), or in the plume region P of the current frame (PC). The logical expressions that define the target, plume, and background pixels are:

Target pixel = $\overline{BP} \cdot \overline{BC} \cdot \overline{PC}$

Plume pixel = $PC \cdot \overline{BC} \cdot \overline{BP}$

Background pixel = $\overline{\text{Target v Plume}}$

Although the target separation method demonstrates the ability to separate the target and plume images from noisy and low contrast backgrounds, additional research is required to optimize the method for WSMR applications and to add additional target separation powers.

Image Decomposition Module

The purpose of the image decomposition module is to separate the scene into target, plume, and background components, providing binary pictures of the target and plume images. Knowing that the purpose of the IDM is to discriminate the target and plume images from the background, the pixel features utilized should be chosen to optimize the discrimination ability of the system. A very promising choice utilizes a monochrome or color camera with optimally selected optical filters to emphasize that portion of the specturm (including infrared) which provides the best separation of the target and background images. In addition, there are many features that can be functionally derived from relationships between pixels.

Initial investigations are concerned only with the intensity levels generated by the VAM and the results are satisfactory. However, the addition of more discrimination features should significantly improve the image decomposition module. In particular, the infrared information should help in discriminating rapidly moving objects from the background. Target proximity and linearity measures should reduce noise problems. The optimal selection of the discrimination features is an open problem that is being presently investigated at New Mexico State University. Throughout the discussion of the IDM, pixel intensity is often used to describe the pixel features chosen.

A parallelogram frame placed about the target image provides a method for sampling the pixel features associated with the target and background images. The background sample should be taken relatively close to the target image, and it must be of sufficient size to accurately characterize the background intensity distribution in the vicinity of the target. The parallelogram frame also serves as a bandpass filter by restricting the target search region to the immediate vicinity of the target. Although one parallelogram frame is satisfactory for tracking missile targets with plumes, two parallelogram frames provide additional reliability and flexibility for independently tracking a target and plume, or two targets. Figure 2.13 shows typical tracking situations with two parallelogram frames. Having two independent parallelograms allows each one to be optimally configured and provides reliable tracking because either parallelogram can track.

If the object to be tracked requires only one parallelogram, then the other parallelogram can be expanded to include the entire field-of-view (FOV).

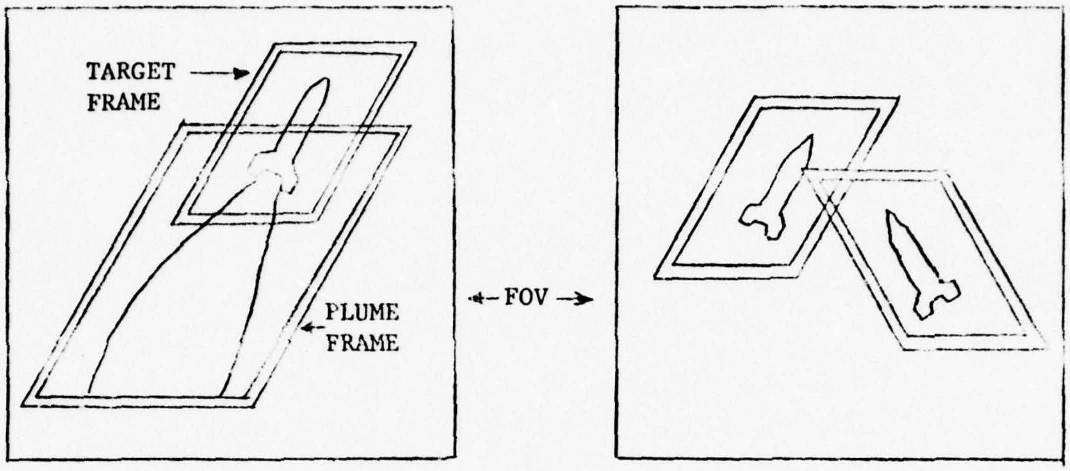


Figure 2.13. Missile target

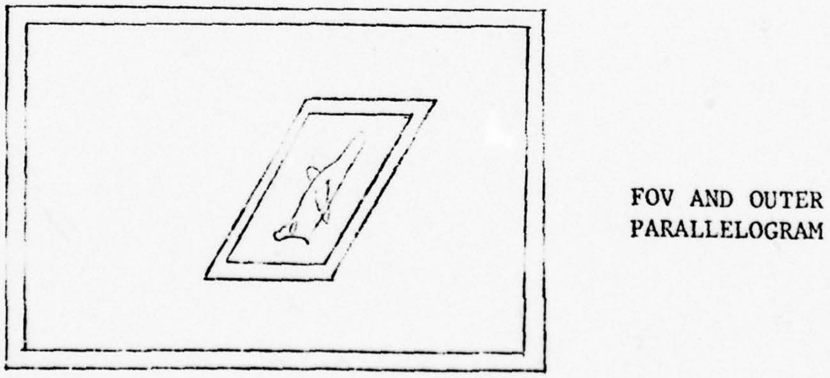


Figure 2.14. Non-missile target

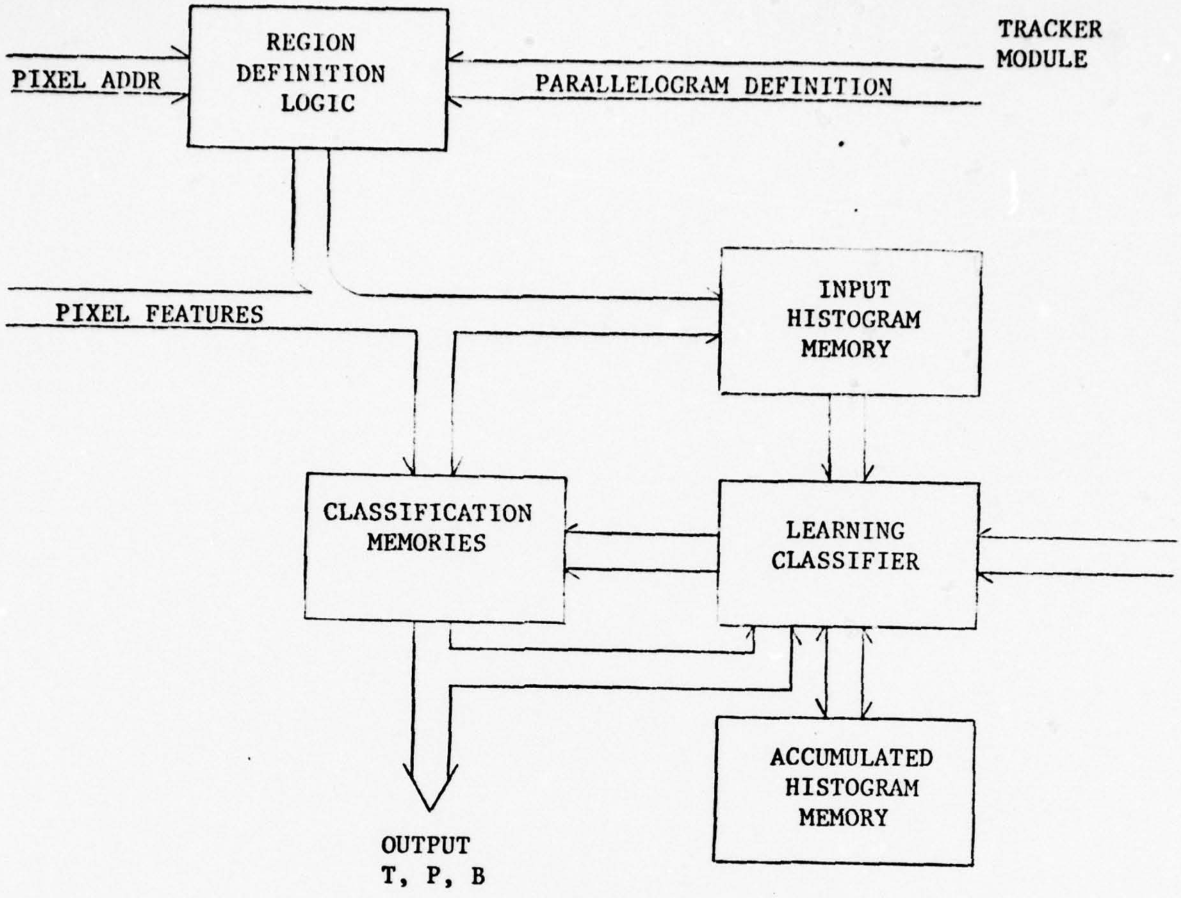


Figure 2.15. IDM Structure

The outer parallelogram provides additional reliability since it can locate the target image as long as it is in the optics FOV. However, the outer parallelogram is subject to more noise due to its larger size.

The structure of the IDM (Figure 2.15) consists of region definition logic, input histogram memory, accumulated histogram memory, learning classifier, and classification memories. The region definition logic defines the parallelogram frame in the FOV of the camera. The input histogram memory accumulates histograms (h_B, h_P, h_T) of the sampled intensities for the background, the plume and the target. The learning module accumulates knowledge on the target, plume, and background intensity density functions from the input histograms, classifier output, and structural confidence weight from the tracking algorithm. The accumulated histogram memory provides the accumulated target, plume, and background intensity density functions (f_T, f_P, f_B) for the classifier. Based upon these density functions, the classifier decides whether each pixel intensity comes from the target, or background density function and stores the results from the classification memories.

Region Definition Logic

The region definition logic generates a set of digital signals that define the parallelogram regions shown in Figure 2.16. Two counters are required to count the number of pixels per line and the number of lines per frame. These counters are available to the IDM from the master timing generator (Section 2.1). The size, position, and orientation of the parallelogram frame is specified by the tracking algorithm on a frame-by-frame basis. These inputs are given below:

Input	Description	Source
X	x-coordinate of pixel	timing generator
Y	y-coordinate of pixel	timing generator
X_c	x-coordinate of parallelogram center	tracking module
Y_c	y-coordinate of parallelogram center	tracking module
X_s	x-coordinate size dimension	tracker module
Y_s	y-coordinate size dimension	tracker module
Tan(Θ)	tangent of orientation angle Θ	tracker module
S_s	sample region width	tracker module

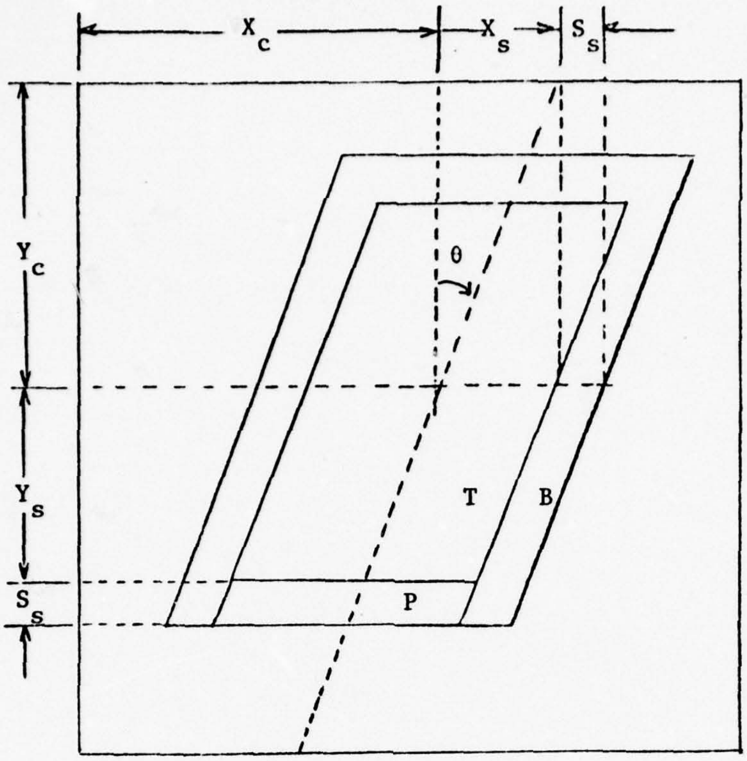


Figure 2.16. Parallelogram frame definition

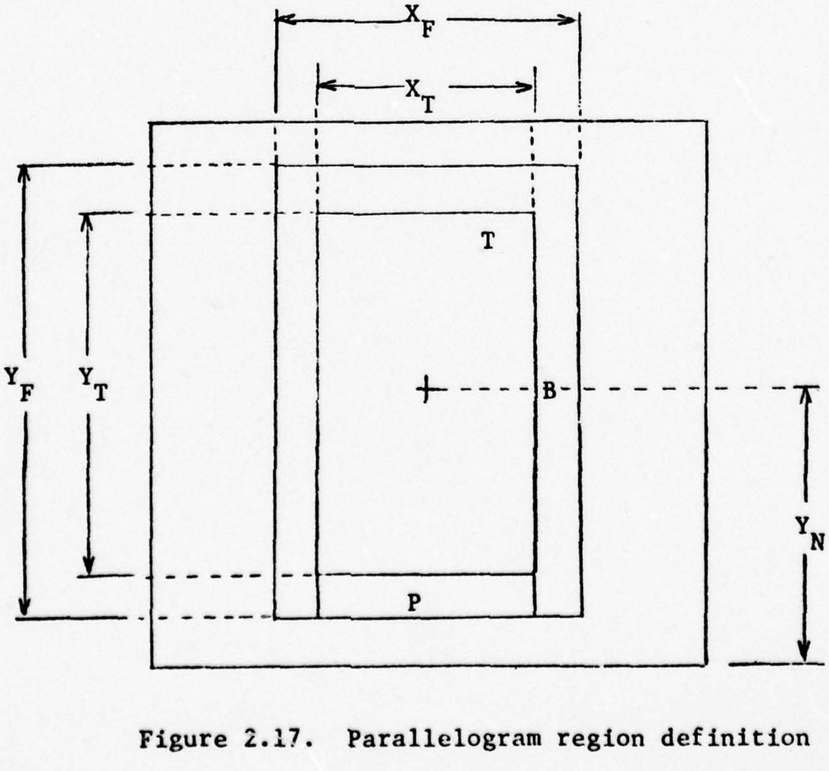


Figure 2.17. Parallelogram region definition

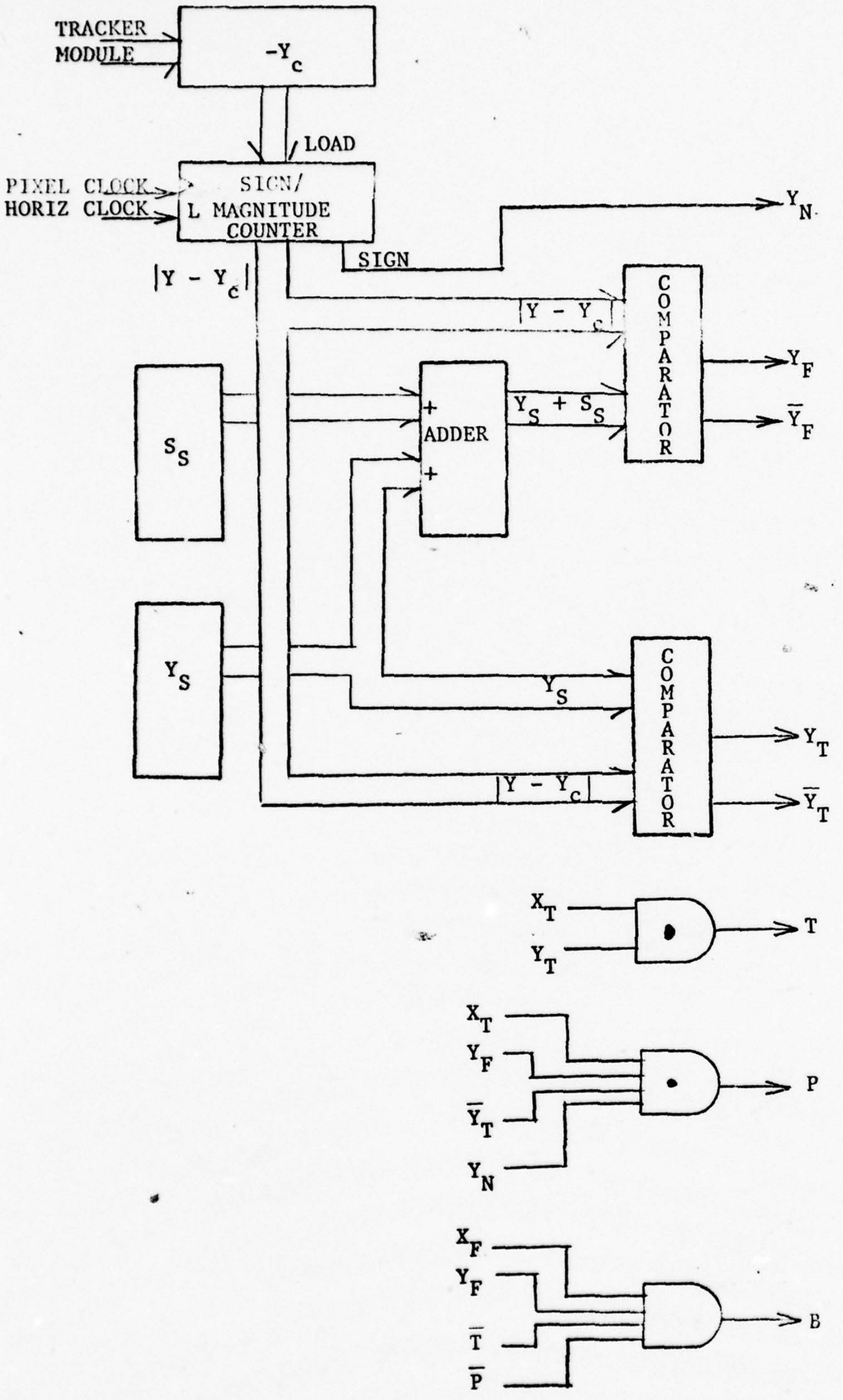


Figure 2.18. Region definition logic

The parallelogram regions are defined in terms of

$$X_T = \{X: \ |X-X_c| \leq X_s\} \qquad Y_T = \{Y: \ |Y-Y_c| \leq Y_s\}$$

$$X_F = \{X: \ |X-X_c| \leq X_s+S_s\} \qquad Y_F = \{Y: \ |Y-Y_c| \leq Y_s+S_s\}$$

$$Y_N = \{Y: \ |Y-Y_c| < 0\}$$

by

$$T = X_T \cdot Y_T$$

$$P = X_T \cdot Y_F \cdot \bar{Y}_T \cdot Y_N$$

$$B = X_F \cdot Y_F \cdot \bar{T} \cdot \bar{P}$$

Figure 2.12 describes the implementation of the region definition logic for the y coordinate; the x coordinate logic is similar. The parallelogram frame orientation is realized by adding offsets derived from tan(Θ) to X_c in order to produce a rotation of the parallelogram center line.

Region Histogram Memory

The counter approach used in the pre-prototype system to accumulate the intensity information consumes a significant amount of power and is very costly to implement, due to the large chip count required to realize the counter arrays for the target, background, and plume regions. A better approach is developed utilizing large-scale integration (LSI) memory components. The accumulation of intensity data does not lend itself to memory interleaving techniques, due to the random nature in which intensities occur. Instead, dual-port read-while-write LSI memory chips such as Signetics' 82S21 are used. Simultaneous reading and writing at the same memory address is not permitted. The structure of the region histogram memory is shown in Figure 2.19.

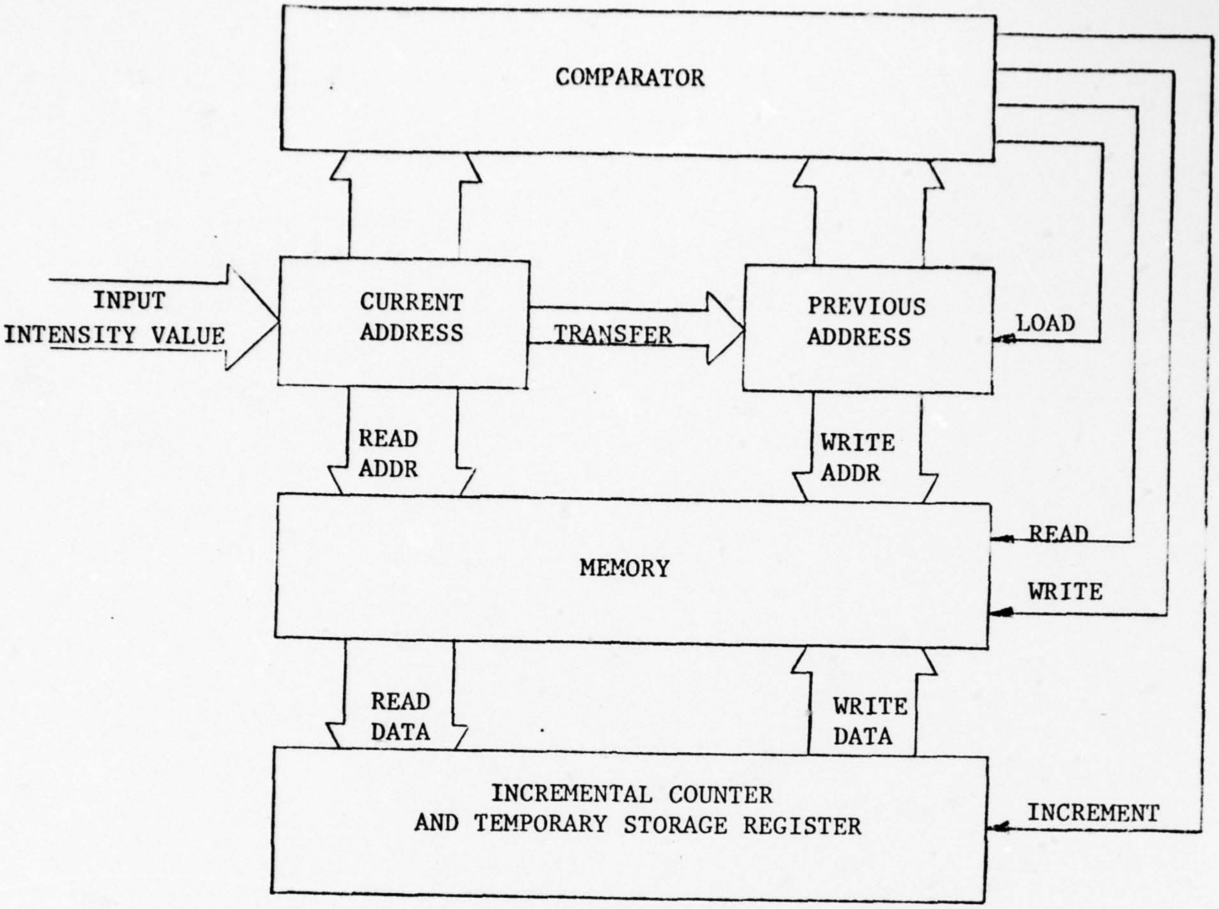


Figure 2.19. Input intensity buffer

As each new pixel intensity value arrives, it is loaded into the current address register where it is compared with the pixel intensity value of the previous pixel. If the two intensity values are not the same, then the location containing the number of occurences of that intensity value is read from memory into the temporary register and incremented by one. At the same time, the previous contents of the temporary register are stored back into its proper location in memory, as indicated by the previous address register. If the two pixel intensity values are the same, no memory read or write occurs; the temporary register contents previously read from memory are incremented by one.

This implementation can replace the current counter bank implementation with a substantial reduction in chip count and also allows considerable additional processing flexibility.

Learning Classifier

The purpose of the learning classifier is to examine each pixel intensity in region T, as it occurs, and to classify it as either a target, plume, or background pixel. Therefore, the classifier must be capable of accurately classifying the pixels at a pixel rate of 100 nanoseconds or faster. The pre-prototype classification method is quite sensitive to the assumption that the target image must have some pixel intensities not contained in the background. When much overlapping occurs between the pixel intensities of the target and background, the classification tends to be conservative - classifying actual target points as non-target points much more frequently than classifying non-target points as target points. This effect is primarily caused by the fact than no explicit knowledge of the target intensities is used by the cla-sifier. Initial studies indicate that significant improvements in the classifier can be obtained by using learning concepts to characterize the target intensities.

The Learning Classifier accumulates probability intensity histograms for the target, plume, and background components by using the frame-by-frame input intensity histograms, the classifier output, and the structural confidence weight from the tracker module. The confidence weight provides a direct feedback of the performance of the IDM in terms of how well the structure of the IDM-generated image matches the target image being tracked. Using the structural confidence weight to evaluate the classifier and to control the rate of learning provides an adaptive method to establish the intensity histograms for the target, plume, and background images. These intensity histograms are stored in the accumulated histogram memory. The classifier can then utilize direct knowledge of the target, plume, and background intensity density functions to make a simple and fast Bayesian decision [36], [37]. For example, given the intensity histograms f_T, f_P, and f_B (Figure 2.20) and equal misclassification costs, the classification rule decides that the pixel intensity y is a background pixel if

$$f_B(y) > f_T(y) \text{ and } f_B(y) > f_P(y)$$

a target pixel if

$$f_T(y) > f_B(y) \text{ and } f_T(y) > f_P(y)$$

or a plume pixel if

$$f_P(y) > f_B(y) \text{ and } f_P(y) > f_T(y)$$

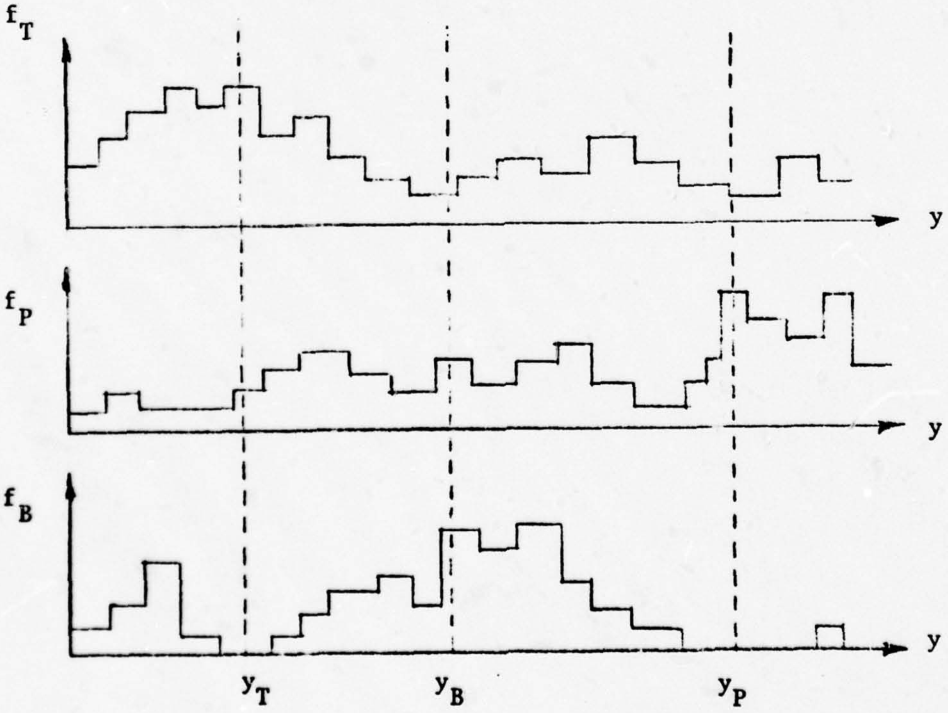


Figure 2.20. Intensity histograms

The classification results are stored in a separate classification memory so that the learning classifier can utilize the entire frame interval to update the accumulated intensity histograms and classify the pixel intensities for the next frame. Having two classification memories allows one of the memories to be used to classify the pixels for the current frame while the other is being used to store the pixel classifications for the next frame. After the pixel classification is stored in the classification memory, the real-time pixel classification is performed by simply letting the pixel-intensity address the classification memory location containing the desired classification. This process can be performed at a very rapid rate with high speed bipolar memories (under 50 nanoseconds per pixel). A 50 nanosecond pixel rate allows the processing of a 512 x 512 pixel scene at 60 frames per second or allows a coarser resolution at a higher frame rate. Storing the pixel classification in a memory allows the learning and classification to be performed simultaneously with the real-time classification. The only constraint is that the classification process must be completed and the new classifications stored in the other classification memory by the beginning of the next frame. This provides the time required to implement the learning classifier with high-speed bipolar microprocessor components. Furthermore, this allows the flexibility of programmable algorithms to implement the learning classifier.

III STRUCTURAL TRACKER

The structural tracker receives the target binary pictures from the video processor, locates the target and plume images, and establishes their structural characteristics. Binary projections are generated and used to structurally identify and precisely locate the target image. A confidence weight is generated that describes how well the located target fits the target being tracked. Adaptive algorithms continually update the structural characteristics of the target to allow the tracker to track the desired target through different spatial perspectives. The weight is a significant parameter used by the control processor to generate the control signals for the tracking optics. Having the confidence weight allows the control processor to intelligently follow the target through noisy backgrounds. The problem of locating the target in the TV window is complicated by the fact that the background is not uniform and changes initially from a desert to a mountain scene and, finally, from a mountain to a sky scene during the flight of the target. The interfaces between these background scenes have traditionally caused many problems for tracking systems. The diversity of target shapes such as missiles, helicopters, airplanes, and balloons, as well as the ability of one target to exhibit various shapes at different view angles, further complicates the problem. This requires the algorithm to adapt to the changing target structure and distinguish the target structure from other patterns in the background.

The structural tracker consists of a projection computational module (PCM) and a tracking algorithm (TA). The PCM generates horizontal and vertical projections of the target and plume images and computes the structural parameters that are used to locate and describe these images. The TA computes the target boresight correction variables, computes the confidence weight, and updates the stored structural characteristics that are used to compute the confidence weight.

The targets are structurally identified and located by using the theory of projections. A projection in the x-y plane of a picture function f(x, y) along a certain direction w onto a straight line z perpendicular to w is defined by

$$P_w(z) = \int f(x,y)\, dw$$

as shown in Figure 3.1. In general, a projection integrates the intensity levels of a picture along parallel lines through the pattern, generating a function called the projection. For binary digitized patterns, the projection gives the number of object points along parallel lines; hence, it is a distribution of the target points for a given view angle.

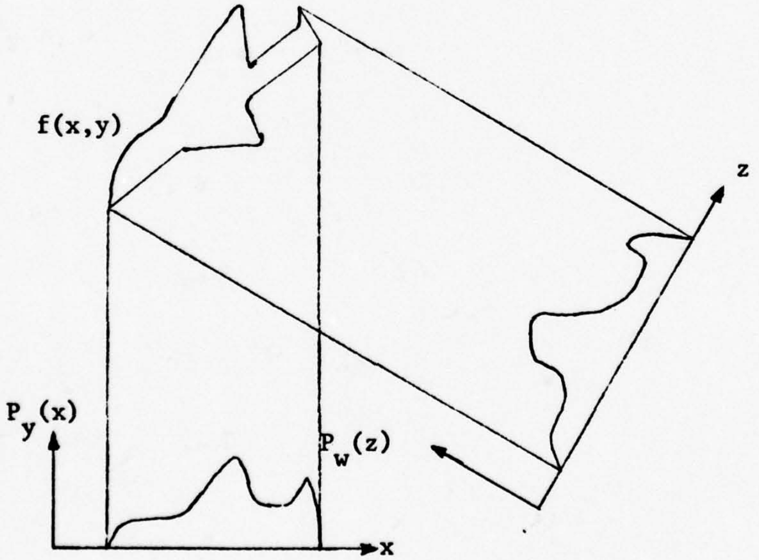


Figure 3.1 Projections

It has been shown that for sufficiently large numbers of projections a multi-gray level digitized pattern can be uniquely reconstructed [6]. This means that structural features of a pattern are contained in the projections. The video processor described earlier reduces a multi-gray level picture to a binary pattern where 0 is used for the background and 1 is used for target points. This simplifies the construction of projections and eliminates interference of structural information by intensity variation within the target pattern; consequently, fewer projections are required to extract the structural information. In fact, any convex, symmetric, binary pattern can be reconstructed by only two orthogonal projections [4], proving that the projections do contain structural information.

Much research in the projection area has been devoted to the reconstruction of binary and multi-gray level pictures from a set of projections, each with a different view angle. In the real-time tracking problem, the horizontal and vertical projections can be rapidly generated with specialized hardware circuits that can be operated at high frame rates. Although the vertical and horizontal projections characterize the target structure and locate the centroid of the target image, they do not provide sufficient information to precisely determine the orientation of the target. Consequently, the target is dissected into two equal areas and two orthogonal projections are generated for each area.

To precisely determine the target position and orientation, the target center-of-area points are computed for the top section (X_c^T , Y_c^T) and bottom section (X_c^B , Y_c^B) of the tracking parallelogram using the projections. Having these points, the target center-of-area (X_c,Y_c) and its orientation can be easily computed.

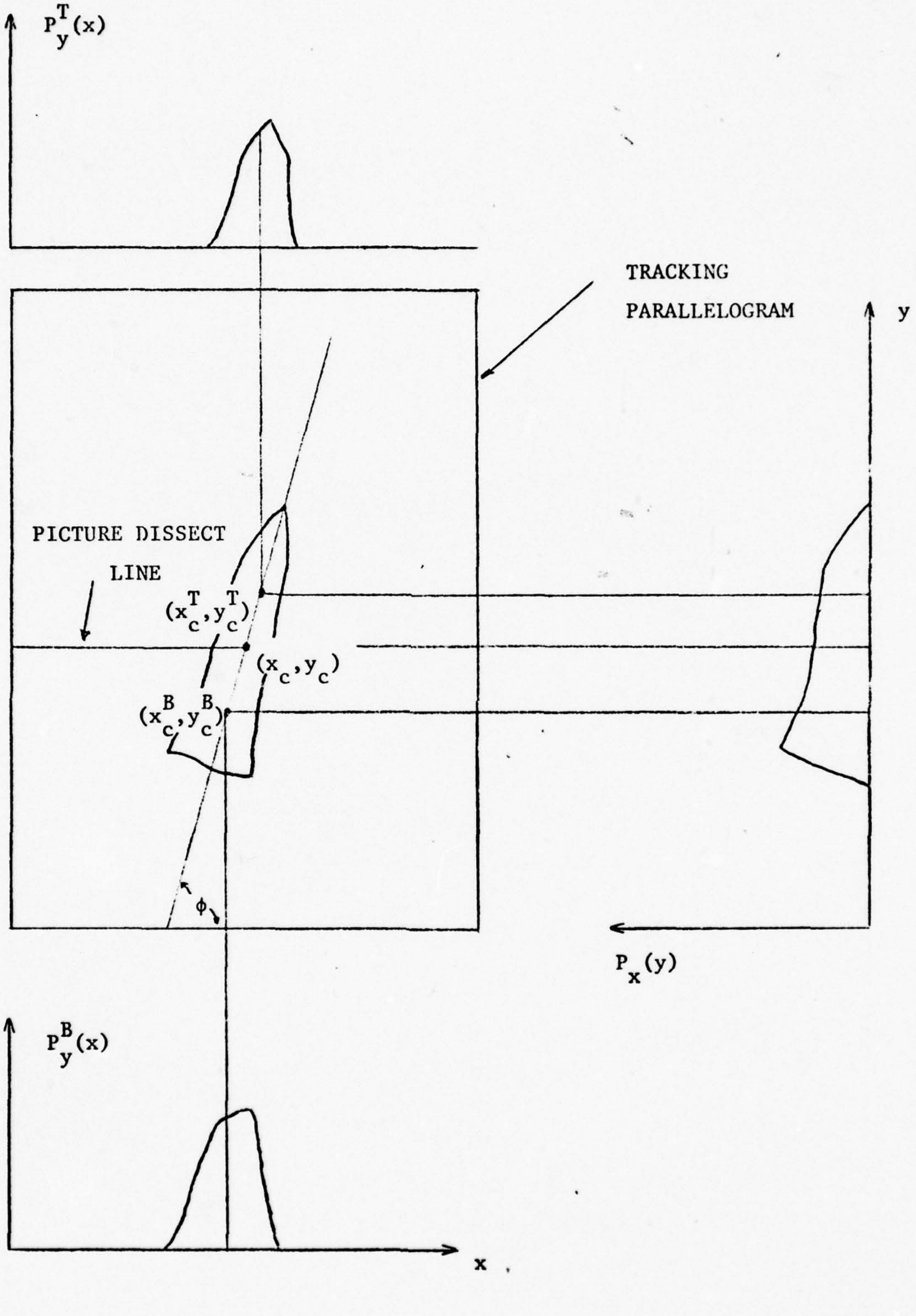


Figure 3.2. Projection location method

$$X_c = \frac{X_c^T + X_c^T}{2}$$

$$Y_c = \frac{Y_c^T + Y_c^B}{2}$$

$$\phi = \tan^{-1}\left(\frac{Y_c^T - Y_c^B}{X_c^T - X_c^B}\right)$$

The top and bottom target center-of-area points are used, rather than the target nose and tail points, since they are much easier to locate, and more importantly, they are less sensitive to noise perturbations.

It is necessary to transform the projection functions into a parametric model for structural analysis. Area quantization offers the advantage of easy implementation and high immunity to noise. This process transforms a projection function $P_w(z)$ into k rectangles of equal areas (Figure 3.3), such that

$$\int_{z_i}^{z_i + 1} P_w(z)dz = \frac{1}{k}\int_{z_1}^{z_k + 1} P_w(z)dz$$

for $i = 1,2, \ldots, k$

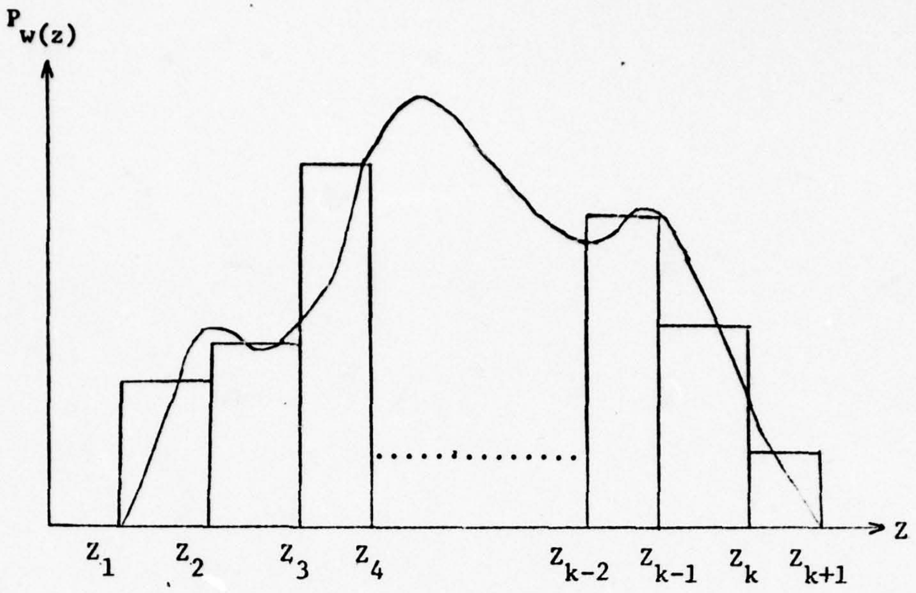


Figure 3.3 Projection parameters

Another important feature of the area quantization model for a projection function of an object is that the ratio of line segments

$$\frac{z_i\, z_j}{z_k\, z_1}, \text{ for } i \neq j,\ k \neq 1$$

is object size invariant. Consequently, these parameters provide a measure of structure of the object which is independent of size and location.

The structural model has been implemented in NMSU's image processing laboratory and used to recognize a class of basic patterns in a noisy environment. The pattern class includes triangles, crosses, circles, and rectangles with different rotation angles. These patterns are chosen because a large class of more complex shapes can be approximated with them [5].

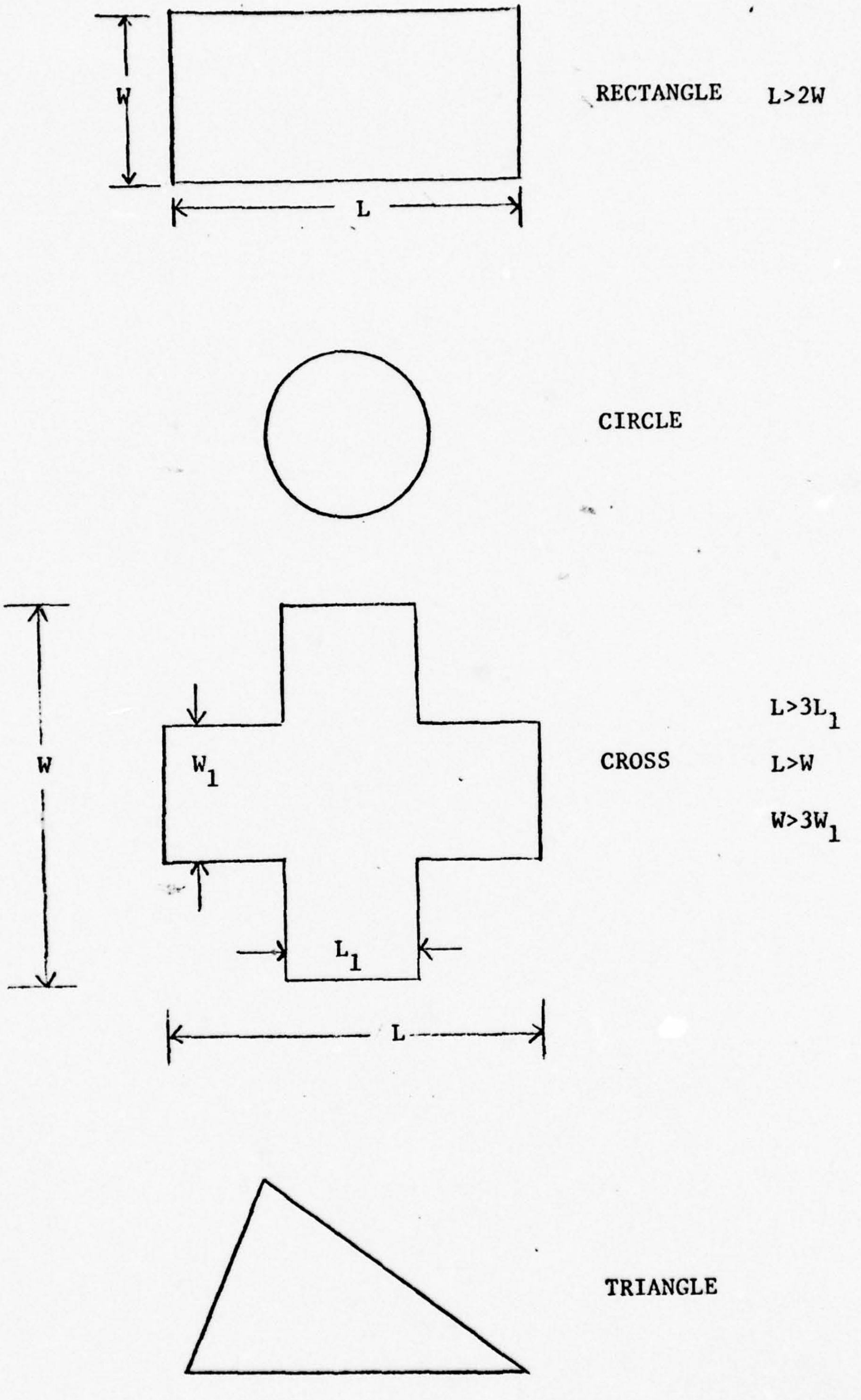


Figure 3.4. A basic class of simple patterns

Four projection functions are initially generated for each input pattern; however, only two projections are area quantized and used in the classification algorithm. The widest projection and its 45° counterclockwise neighbor are selected to reduce the rotational effect. Six parameters $A_1, \ldots, A_6$, as defined in Figure 3.5, along with a symmetry measure of the projections are utilized to distinguish between rectangles, circles, crosses, and triangles. For a given pattern, these structural parameters are computed and compared to the standard reference values for each basic pattern (Figure 3.5). The object is classified in the category, with the best match utilizing a minimal distance decision rule.

		RECTANGLE	CIRCLE	CROSS
$\theta = 135^o$	A_1	0.25	0.368	0.417
	A_2	0.5	0.599	0.722
	A_3	0.75	0.805	0.861
$\theta = 90^o$	A_4	0.434	0.368	0.380
	A_5	0.643	0.599	0.553
	A_6	0.821	0.805	0.744

$$A_1 = \frac{z_1 z_2}{z_1 z_5} \qquad A_2 = \frac{z_1 z_3}{z_1 z_5} \qquad A_3 = \frac{z_1 z_4}{z_1 z_5}$$

A_4, A_5, and A_6 are similarly defined

Figure 3.5. Area quantization data for k=8

A simple minimal distance classification algorithm is used to classify a set of test patterns. No attempt is made to design a sophisticated classification method; rather, the objective is to test the discrimination powers of the structural projection parameters. The test patterns are generated with random orientations, and more classification errors are due to an insufficient number of projections to handle the rotations. Most classification errors occur for pattern rotations midway between computed projections.

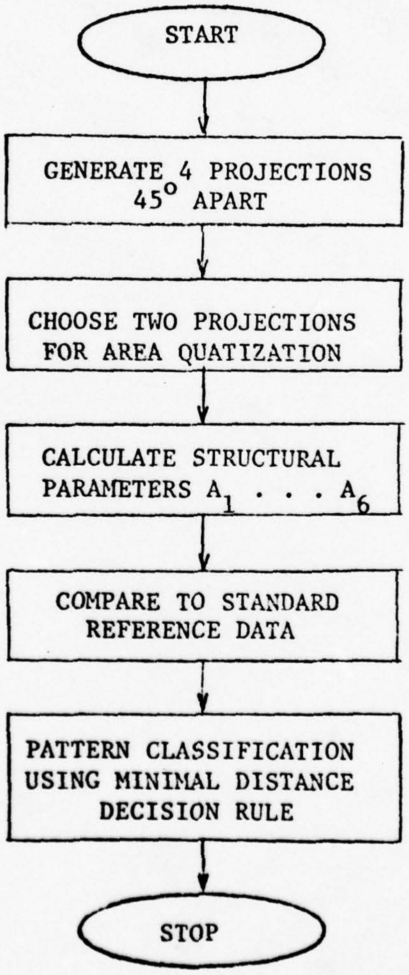

START
GENERATE 4 PROJECTIONS 45° APART
CHOOSE TWO PROJECTIONS FOR AREA QUATIZATION
CALCULATE STRUCTURAL PARAMETERS $A_1 \ldots A_6$
COMPARE TO STANDARD REFERENCE DATA
PATTERN CLASSIFICATION USING MINIMAL DISTANCE DECISION RULE
STOP

Figure 3.6. Classification algorithm

The results of the classification algorithm for a hand-generated test pattern set are given in Figure 3.7. The i, j entry of the matrix is the percentage of test pattern i classified as pattern j.

When noise is added to the same test pattern set, the experiment yields the results given in Figure 3.8. The added noise does not significantly reduce the effectiveness of the algorithm, demonstrating the noise immunity property of the global structural features.

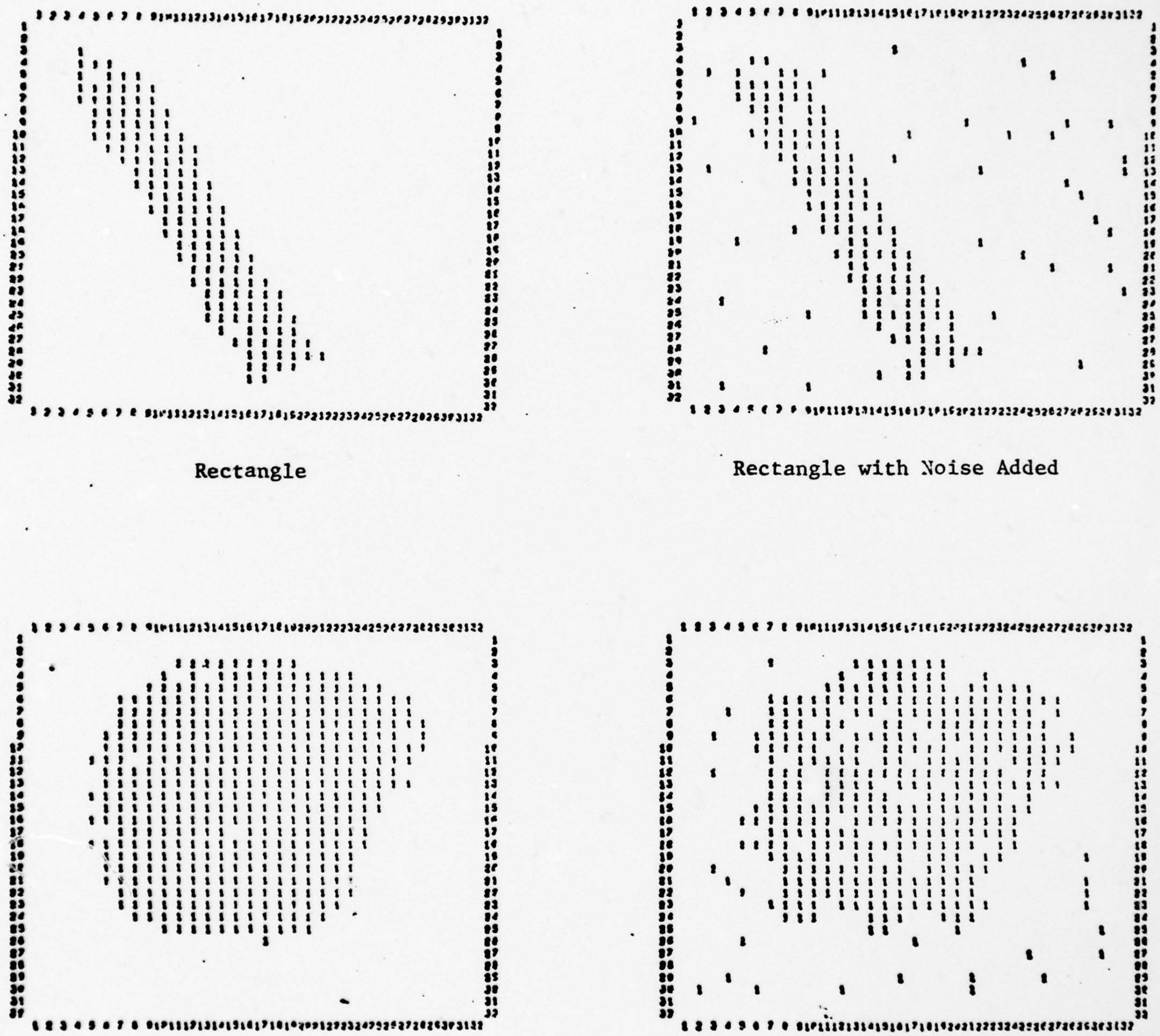
		□	○	△	✚
RECTANGLE	□	100	0	0	0
CIRCLE	○	0	100	0	0
TRIANGLE	△	0	0	90	10
CROSS	✚	10	10	10	70

Figure 3.7. Results for noise-free test patterns

		□	○	△	✚
RECTANGLE	□	90	10	0	0
CIRCLE	○	0	100	0	0
TRIANGLE	△	0	0	90	10
CROSS	✚	10	15	10	65

Figure 3.8. Results for test patterns with noise added

The results of this study demonstrate that the projection parameters have a remarkable discrimination ability. Some typical test patterns used in the study are shown below.



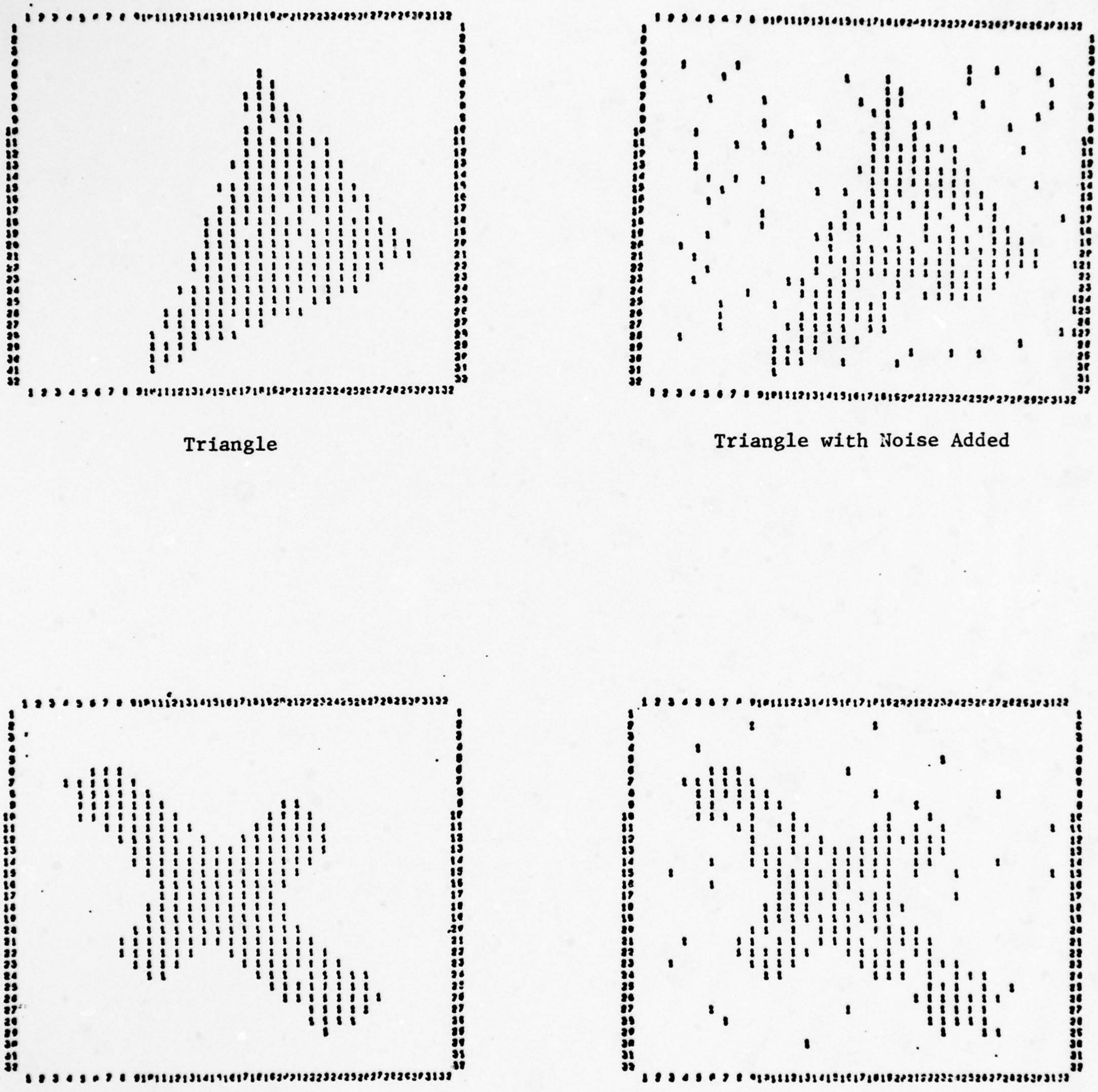
Rectangle

Rectangle with Noise Added

Circle

Circle with Noise Added

Figure 3.9 Sample test patterns



Triangle

Triangle with Noise Added

Cross

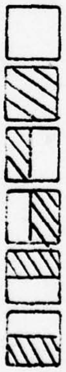
Cross with Noise Added

Figure 3.9 Continued

3.1 Pre-Prototype Method

In the pre-prototype system, the binary target data from the image decomposition module arrives serially for each horizontal scan and is shifted into an eight bit shift register. When the shift register is filled, the data is stored in a high speed buffer memory in such a manner that images over 4 x 8 windows are directly word addressable by the image processor. The image processor is implemented with a HP21MX minicomputer which accesses the 4 x 8 windows in the buffer memory and performs the projection computations and structural analysis required to recognize and locate the target image [2]. Each 4 x 8 window consists of two adjacent 4 x 4 windows called scan windows. The contents of each scan window are analyzed in parallel using scan window logic, producing discrete control signals that define the contents of the window and enable the computer's response. There are six different outputs from the scan window logic. These are:

- null window
- full window
- left window
- right window
- upper window
- lower window



The scan window logic is implemented with combinational logic. When the computer accesses a scan window in the buffer memory, the contents of the scan window are latched to the window logic which in turn enables the proper computer response. By sequentially processing the scan windows, the computer forms the projections and locates the target of interest.

A scanning algorithm sequentially scans a target image by accessing the scan windows in the buffer memory without stepping over to neighboring images within the parallelogram frame. The projections are computed while the target image is being scanned. When the target image is completely scanned, its structural parameters are established and its location and orientation are computed. A detailed description of the tracking algorithm is given in reference [1].

The main advantages of this approach are the speed of execution and the ability to locate multiple targets in the parallelogram frame. The major disadvantage is the loss of resolution caused by the scan window logic. For the target images to be tracked at WSMR, the loss of resolution is considered to be serious; a new approach was taken to obtain better resolution, using a separate projection computational module (PCM).

3.2 Projection Computational Module

The Projection Computational Module (PCM) is a specialized processor for accumulating the projections as the filtered video image arrives from the image decomposition module, and it computes the structural parameters which locate and characterize the target shape during the vertical retrace interval. The PMC retains the resolution of the TV camera by accumulating the projections for each pixel in the parallelogram frame. By processing the video image on a field-by-field basis, rather than a frame-by-frame basis, the incoming video data can be considered to form an N x M grid of pixels, with a new grid being generated at a rate of sixty frames per second. In anticipation of better CCD cameras, the PCM hardware is designed to process a 512 x 512 grid on a field-by-field basis.

The PCM accumulates the projections only within the target and plume parallelogram frames. However, the target and plume parallelogram frames can be made as large as the field-of-view of the camera without losing any resolution. The output of the image decomposition module can be considered to be an N x M binary matrix T whose entries are defined as

t_{ij} = 1, if i, j pixel is a target (or plume) point

t_{ij} = 0, if i, j pixel is a background point

As the TV camera scans the field, the elements of T are produced on a row-by-row basis. As the image decomposition module (IDM) produces the entries t_{ij}, a strobe is generated to the PCM only if the (i, j) pixel is within the parallelogram frame. Consequently, if the window size within the parallelogram frame is n x m, then the PCM serial input consists of n bursts of m values of t_{ij}. The bursts are separated from each other by the horizontal synchronization signal which is stripped from the raw video signal by the video acquisition module. Therefore, the PCM handles only a subset of the entire grid, namely, a smaller grid of n rows and m columns represented by the n x m binary matrix G defined by

g_{ij} = 1 if the i,j pixel is a target (or plume) point

g_{ij} = 0 if the i,j pixel is a background point.

PCM Computations

The main task of the PCM is to compute several parameters describing the projections of the target (or plume) points within the grid G_{ij}. These parameters are:

- the coordinates of the "center" of the top "half" of the target
- the coordinates of the "center" of the bottom "half" of the target
- a set of percentile points of the X- and Y-projections of target (plume) points within the entire grid

The division of the window into top and bottom halves furnishes the structure tracking algorithm with the information necessary to estimate the inclination of the target to the edges of the window. The division of the grid into top and bottom "halves" is done on the basis of equal numbers of target (plume) points in the top and bottom halves. The "center" of the top or bottom half is defined to be the median of the appropriate projection.

The distributions which must be computed are discrete integer-valued functions whose domain and range depend upon the size of the tracking window (i.e., the dimensions of the matrix G_{ij}, which are "n" rows and "m" columns). These are:

$D_x(i)$ distribution of target points in the entire window as a function of X

$D_y(j)$ distribution of target points in the entire window as a function of Y

$D_{xt}(i)$ distribution of target points in the top half of the window as a function of X

$D_{xb}(i)$ distribution of target points in the bottom half of the window as a function of X

Depending upon which tracking window is being processed, the word target may be replaced by the word plume. The domain and range of these distributions are:

	Range	Domain
D_x	[0,n]	[1,m]
D_y	[0,m]	[1,n]
D_{xt}	[0,n']	[1,m]
D_{xb}	[0,n-n']	[1,m]

where the n' specifies the "middle" of the target. Thus, n' is the value of the first index of G_{ij} at the center of the target and is the smallest positive integer which satisfies the relation

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} G_{ij} \leq \sum_{k=1}^{n'} \sum_{\ell=1}^{m} G_{k\ell}$$

Notice that the target points in the row $G_{n'j}$ are defined to be in the top half of the window.

The domains of the X-projections are the same, so that

$$D_x(i) = D_{xt}(i) + D_{xb}(i)$$

Also, the origin of the x-y coordinate system is defined to be the top left corner of the grid (the G_{11} position), and x increases to the right, while y increases downward.

The distributions are computed as follows:

$$D_x(i) = \sum_{k=1}^{n} G_{ki} \qquad i = 1, 2, \ldots m$$

$$D_{xt}(i) = \sum_{k=1}^{n'} G_{ki}$$

$$D_{xb}(i) = \sum_{k=n'+1}^{n} G_{ki}$$

$$D_y(j) = \sum_{k=1}^{m} G_{jk} \qquad j = 1, 2, \ldots n$$

Inspection of the above equations shows that if G_{ij} is generated serially by rows and if the value of n' is initially known, or can be estimated, then the distributions can be accumulated without requiring a memory for the matrix G_{ij}. A quantity which is inherently a more stable estimate is the left side of the implicit relation for n'. The technique used in this implementation is to replace the left side of that relation with the value computed from the previous field, that is

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} G_{ij}^{OLD} \leq \sum_{k=1}^{n'}\sum_{\ell=1}^{m} G_{k\ell}$$

This says than one has reached the center of the target in the field when the number of target points that have been counted (starting at the top of the field) is equal to half of the total number of target points in the previous field. This approximation seems reasonable, considering the high correlation between successive fields of video data. Of course the top and bottom distributions are meaningless for the first field that is processed when the system is initially started.

The target parameters which are described in the opening paragraph of this section are computed from continuous integer-valued functions corresponding to the above distributions as follows:

$$f_{xt}(x) = D_{xt}(i), \quad \text{if } i-1 \leq x < i$$

$$f_{xb}(x) = D_{xb}(i)$$

$$f_y(y) = D_y(j), \quad \text{if } j-1 \leq y < j$$

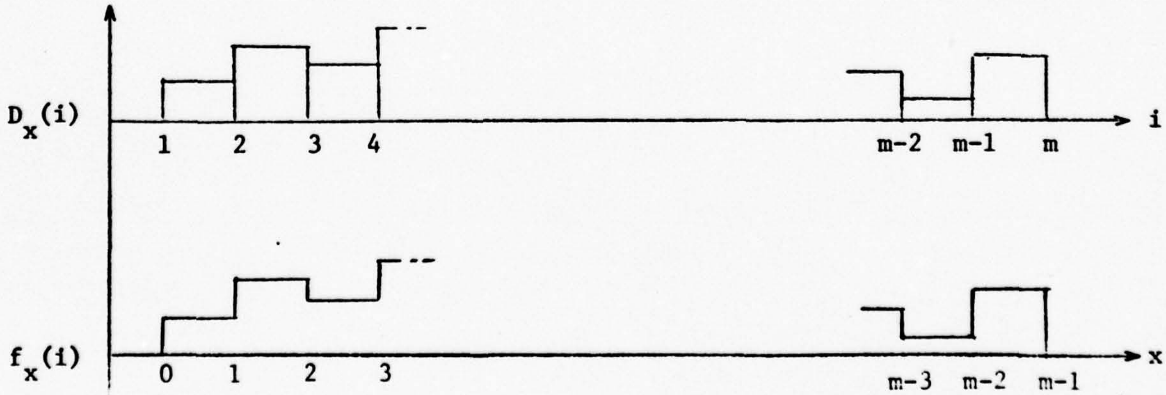


Figure 3.10. Function of a distribution

The range and domain of the continuous functions are:

	Range	Domain
$f_x(x)$	[0,n]	[0,m-1]
$f_{xt}(x)$	[0,n']	[0,m-1]
$f_{xb}(x)$	[0,n-n']	[0,m-1]
$f_y(y)$	[0,m]	[0,n-1]

The coordinates of the center of the top half of the target are denoted by $\left(X_y', Y_t'\right)$ and are computed as

$$\frac{N_1}{2} = \int_o^{X_t'} f_{xt}(x)dx$$

$$\frac{N_1}{2} = \int_o^{Y_t'} f_y(y)dy$$

where

$$N_1 = \int_o^{m-1} f_{xt}(x)dx$$

The coordinates of the center of the bottom half of the target are denoted by $\left(X_b', Y_b'\right)$ and are computed as

$$\frac{N_2}{2} = \int_o^{X_b'} f_{xb}(x)dy$$

$$\frac{N_2}{2} = \int_{n'}^{Y_b'} f_y(y)dy$$

where

$$N_2 = \int_o^{m-1} f_{xb}(x)dx$$

The following percentile points on the X and Y projections for the entire grid (Gij) are computed as follows:

$$\frac{i}{8} N_3 = \int_o^{P_{xi}} f_x (x)dx$$

$$\frac{i}{8} N_3 = \int_o^{P_{yi}} f_y(y)dy \; , \quad i = 1, 2, \ldots, 6, 7$$

where

$$N_3 = N_1 + N_2 = \int_o^{n-1} f_y(y)dy = \int_o^{m-1} f_x(x)dx$$

PCM Description

This section describes the architecture of the projection computational module (PCM). Programmed logic design is used to implement the operations described by the equations in the previous section. The projections are accumulated during the video horizontal scan, and the structural parameters are computed during the vertical retrace interval. The architecture is capable of performing all processing functions for two tracking windows (target and plume) each of up to 512 x 512 pixels in size. The processing time required to compute the structural parameters requires less than one millisecond of the vertical retrace interval.

The architecture of the PCM, as shown in Figure 3.11, consists of a microprogrammable control unit (MCU), central processing unit (CPU), interlaced target and plume projection memory (PM), a parameter communication memory (CM), and timing and control logic.

The MCU is implemented with a control store memory, microprogram counter (MPC), microinstruction register (MIR), and conditional branching logic, having the following characteristics:

- 64 x 36-bit word control store
- 70 nanosecond maximum cycle time
- conditional branching capability
- overlapped CP/MCU processing (optional)
- Schottky TTL implemented

The CPU consists of a 20 bit parallel processing unit implemented with ten Intel 3002 processing chips and a flexible bus structure, having the following characteristics:

- dynamically changeable bus structure
- 13 bit address bus width
- 20 bit data bus width
- 70 nanoseconds maximum cycle time
- Schottky TTI implemented

The CPU operates in conjunction with a main memory of 4K by 9 bit words with a 35 nanosecond read cycle and a 40 nanosecond write cycle, that is used to store the target and plume projections. The memory is interlaced to provide the required 96 nanosecond read-increment-write cycle time. When the structural parameters are computed, they are passed to the structural tracker by the 64 x 18 multiport RAM. This RAM can be accessed by the structural tracker.

The timing and control logic interfaces the target (T), plume (P), and strobe from the image decomposition module (IDM) to the PCM. Furthermore, the horizontal and vertical synchronization signals are required from the video acquisition module (VAM). Using these signals along with a 10 M Hz master clock, all timing and control signals are generated form the PCM.

It is recognized that a hardwired control unit would have certain advantages over microcoded control, but, for reasons detailed below, a microprogrammed configuration was chosen. Quoting Husson's [27] definition:

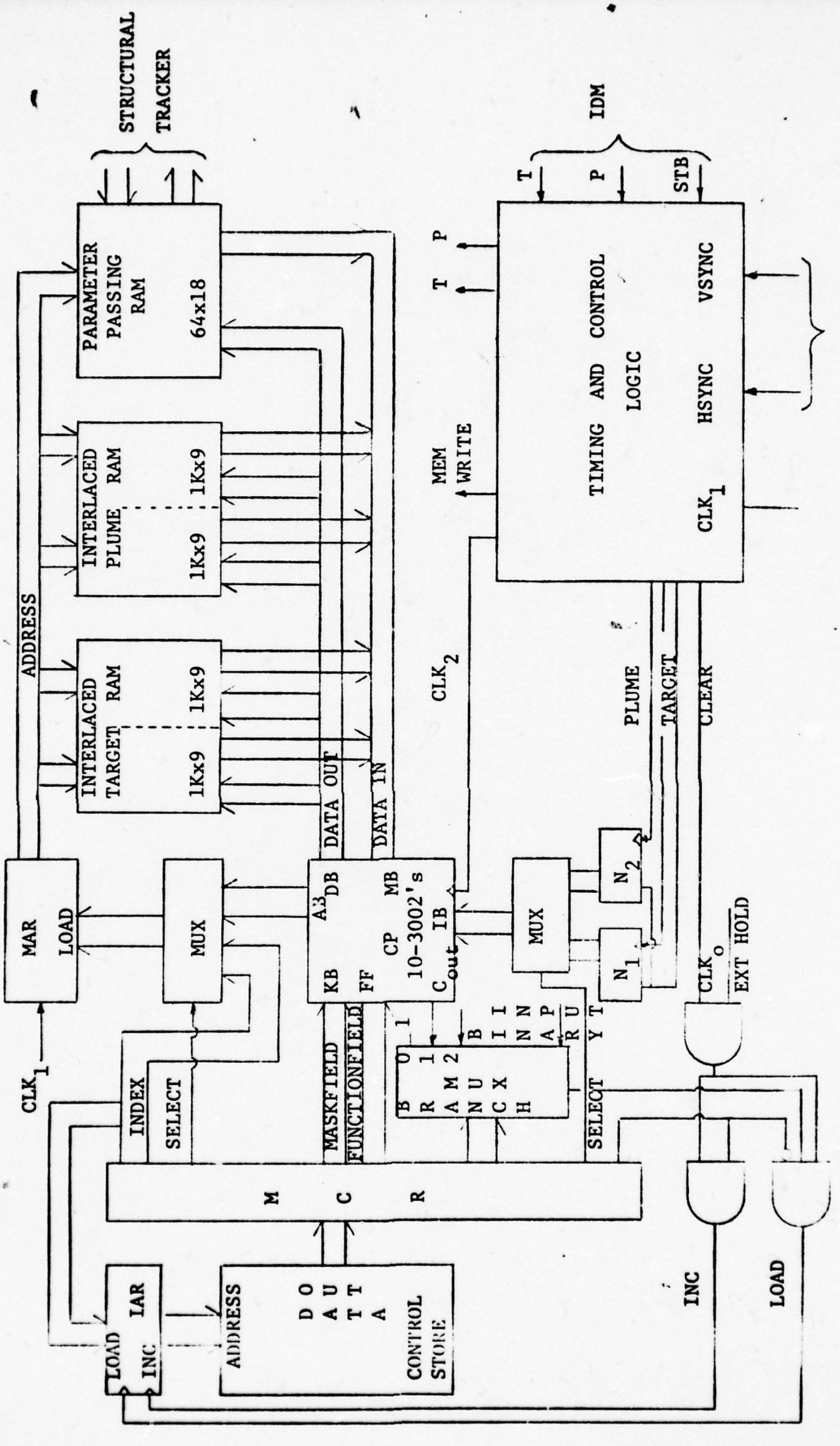


Figure 3.11. PCM architecture

> "Microprogramming is a technique for designing and implementing the control function of a data processing system as a sequence of control signals, to interpret fixed or dynamically changeable data processing functions. These control signals, organized on a word basis and stored in a fixed or dynamically changeable control memory, represent the states of the signals which control the flow of information between the executing functions and the orderly transitions between these signal states."

By inspection of the proposed CPU architecture, it is apparent that this task lends itself easily to microprogramming, since

1. the CPU (which is designed to optimize performance on this particular application) is synchronous, with precisely-defined timing points;
2. the CPU implementation has a minimum of control lines, which are encoded wherever possible;
3. the majority of the control lines form mutually-exclusive sets which can be encoded efficiently in the microcommand format, and which can be performed simultaneously during a single microcycle;
4. certain time-critical control sequences are implemented as hardwired control cycles which are initiated by the microcoded control unit, or an external strobe line;
5. in this application, the vast majority (over 99 percent) of the executed microcommands are single-microcommand loops, which allow the control store read cycle to be omitted. Thus, during these loops, the microcoded control unit is just as fast as hardwired controller would be.

In view of (5) above, pipelinning is not employed in the control store, since it would not increase performance significantly. Also, the main memory accessing is not overlapped with computations, since the decision to increment the MIR and fetch the next word of data from memory depends upon the results of the current microcycle's computations. An incorrect fetch requires a correction cycle to update the MIR to the correct value.

A microprogram-controlled processor has several advantages when compared to a hardwired-controller/processor configuration. One such advantage is flexibility. If the computational needs of the algorithm change, modification of the microprogram is often all that is required to perform any new arithmetic functions, whereas a hardwired controller may require extensive modifications. Of course, the degree of flexibility depends heavily upon the CPU architecture.

In addition, microprogramming gives uniformity and modularity to the control unit design, similar to that of the arithmetic and logic unit in the CPU. The CPU and the control unit are two complete and distinct entities which can be debugged separately in a systematic manner.

Finally, it is simple to obtain system diagnostics at any time by loading one or more diagnostic micro-routines which verify the correct operation of each data path and each portion of the CPU and memory by placing test patterns on the appropriate bus. Without this capability, debugging a parallel processing unit is very tedious.

Two possible disadvantages of microprogramming are cost and loss of performance. However, microcoded control units often cost less in terms of the sum of the actual hardware requirement, the design and debugging time, and their flexibility. As discussed below, the performance of the proposed microcoded controller is well within the required performance specifications of the control unit; thus, performance degradation due to microprogramming is not a pertinent consideration in evaluating the merits of microprogramming.

The CPU and the MCU constitute a unit into which the following signals are fed:

- the filtered video signal from the IDM, which indicates the presence of either a background point, or depending upon whether this is a target or a plume window, a target or plume point, respectively
- a sampling strobe for the video
- the horizontal and vertical synchronizing signals which are stripped from the raw video output of the camera by the VAM

The output of the unit is a set of 18 parameters for each tracking window that is defined. These parameters specify several characteristics of the distributions of target points in three regions of each tracking window:

- the top half of the window
- the bottom half of the window
- the entire window

These parameters are described more fully in the preceding section. They are guaranteed to be computed by the end of the vertical retrace interval following the field of interest and are stored as pseudo-floating point (integral portion and fraction, but no characteristic) binary data in a multi-port random-access memory (RAM). This RAM is accessible at any time by the structural tracking algorithm, although the data for a particular field is guaranteed valid only during the period of time between the end of the vertical retrace interval following that field and the start of the vertical retrace interval after the next field of video data (15.3 milliseconds later).

There are several trade-offs made in choosing the size and speed of the read-only memory (ROM) or RAM which holds the microcommands. It was found in the trial-and-error attempt to encode the necessary functions with various CPU architectures, that this structure is reasonably efficient for the operations that must be performed and requires a microprogram control store of approximately 35 words of about 30 bits each.

In order to provide a 64 x 35-bits control store, the memory chip best suited for the RAM implementation is the N82S09, which is a Schottky TTL RAM organized as 64 words of 9 bits. A total of four of these chips will provide 64 words of 36 bits, which are enough for both diagnostic and computational routines. The access time (address valid to data outputs valid) is 45 nanoseconds maximum.

Since the above mentioned chips are RAM, it is necessary to provide some means of loading the microprogram code into memory. This is accomplished by the following scheme:

- bus all "Data In" lines for the four chips in parallel
- tie all address lines for the four chips in parallel and drive them via a two-to-one line selector
- use two additional "write address" lines to multiplex the memory write pulse to one of the four memory chips via a one-to-four-line encoder
- leave the chips enabled at all times to reduce access times.

Thus, the data source for the microcode sees the control store as 256 words of nine bits, while the MCU logic sees the control store as 64 words of 36 bits.

Obviously, the data source could be one of many devices, such as a paper tape reader, magtape, or a computer. If a minicomputer is to be used elsewhere in the overall tracking system, it could be easily adapted to perform microcode loading too. After system checkout, the RAM can be replaced by a read-only memory, eliminating the microcode loading step but reducing the system flexibility by preventing loading of diagnostic routines.

The performance requirements of the CPU and MCU are fairly rigorous. It is anticipated that the operational version of the tracking system will require two tracking windows of different sizes. The design goal calls for a maximum of about 8 x 10^5 bits of video data per window. It is required that the CPU/MCU process this data on a field-by-field basis or at an effective rate of 4.8 x 10^7 bits per window per second. Since a single CPU/MCU can handle two windows concurrently, the effective data rate is 9.6 x 10^7 bits/second.

The video data is actually generated during a 15.3 millisecond interval at a maximum rate of about 96 nanoseconds per bit per window. As the data arrives, it is added to the current contents of the appropriate location in the main memory. At the end of the 15.3 millisecond interval, a dead time of about 1.3 millisecond occurs (the vertical retrace interval), during which the CPU/MCU must perform all the remaining computations and then clear the memory in preparation for the next field.

It is anticipated that the CPU and main memory are able to operate at a combined cycle time of 70 nanoseconds. This speed is obtained by restricting the number of levels of logic through which a signal must propagate in a single cycle and by using Schotty-clamped TTL logic wherever possible.

The 40 nanosecond cycle time of the RAM that is to be used for microcode storage, and other logic delays, contribute to yield a minimum MCU cycle time of about 70 nanoseconds. Thus, the effective MCU and CPU cycle time will be about 150 nanoseconds.

Instruction look-ahead during sequential instruction processing or single-instruction loops could be implemented, allowing the RAM access time to be effectively reduced to zero. In this case, the limiting factor for over-all speed would be the settling time of the CPU components. However, look-ahead increases the complexity of the branching hardware in the MCU.

The worst-case computational requirements call for about 3,500 microcontroller cycles per window. The computation time is about 525 microseconds per window, or 1,050 microseconds for two windows, which is within the 1.3 millisecond vertical retrace interval time slot.

The four distributions (D_x, D_y, D_{xt}, D_{xb}) that are defined previously are stored in a single, linearly-organized memory, with both n and m equal to 512. Thus:

Mem. Address	Contents	
0 - 511	$D_y(j)$,	$j = 1, . . . 512$
512 - 1023	$D_x(i)$,	$i = 1, . . . 512$
1024 - 1535	$D_{xt}(i)$,	$i = 1, . . . 512$
1536 - 2047	$D_{xb}(i)$,	$i = 1, . . . 512$

Note that the data $D_x(i)$ is redundant, since

$$D_x(i) = D_{xt}(i) + D_{xb}(i).$$

In fact, the hardware only stores D_y, D_{xt}, and D_{xb} when the data is being acquired in the window. Depending upon how the algorithm is microcoded, the memory reserved for D_x may be used for other purposes.

The reason for the particular memory address assignments for the distributions is the optimization of the computations. In fact, when computing, say X_t, the CPU accumulates the values in memory locations 1024, 1025, 1026, etc., until the accumulated value exceeds $N_1/2$, at which point the current value of the nine least-significant bits of the MAR is the APL ceiling of X_t.

Each memory word is 9 bits wide and contains a positive integer. Thus, the main memory's data bus is 9 bits wide, and the address bus is 11 bits wide. The cycle time is of the order of 50 nanoseconds.

In addition, there are two 18-bit counters whose control is implemented entirely in hardware and which contain N_1 and N_2, the total number of target points in the top and bottom halves of the window.

The microcontroller provides facilities for asynchronous and synchronous gating and timing signals, and conditional branching on the status of the ALU or the status of one of several external lines.

The microcommand words are organized as follows:

- asynchronous select fields which specify the operation of several data selectors in the CPU, allowing the bus configuration of the CPU to be dynamically altered during operation
- synchronous control fields which gate the MCU clock into several clocked elements in the CPU (such as counters and latches)
- an index field which contains either a branch address or a value loaded into the memory address register (MAR) for main memory accesses, which can also be preset to a particular value and then incremented or decremented to provide a loop counter

- a branch condition field which specifies the condition under which a conditional branch is to be executed
- enable fields which are used to selectively disable various external data inputs (e.g., from the Video Filter)

The above fields are independent and may be specified in any combination, with one exception: when an index value is specified as a preset input to the MAR, a branch cannot be specified.

The microcontrol unit consists of the following components:

- an instruction address register (IAR) of six bits containing the address in control store of the next microcommand to be executed
- a read-only memory (ROM) or random-access memory (RAM) of up to 64 words of 36 bits
- a microinstruction register (MIR) which latches the output of the ROM (or RAM)
- a 16-to-1-line selector which allows conditional branching, based upon the current CPU status or the status of any one of several external data lines

A single microinstruction is executed as follows:

- The microcommand is fetched from the control store at the address specified by the MPC and is loaded into the MIR.
- The asynchronous selects are made immediately available to the CPU to reconfigure the bus structure.
- If the select field S4 is one, the index field is made immediately available to the preset lines of the MAR.
- The branch condition field is used to select one of sixteen branch condition lines. Two of these lines are always zero and one and allow no-operation and unconditional branches to

be specified. The output of the selector is ANDed with the complement of S4, such that branching is impossible if the index field is being used as a preset value for the MAR.

- After all CPU buses have had time to settle, the next microcontroller clock occurs. At this time all the synchronous control fields are gated simultaneously onto their respective output lines for the duration of the clock.
- After the control store access time has elapsed, the next microcommand is loaded into the MIR.

When the branching condition is specified by an external line, branching occurs if the line is a logic "one" at the time fo the succeeding MCU clock. If the branching condition is an arithmetic/logic test in the CPU, the branch is taken/not-taken based upon the status of the CPU immediately prior to the next MCU clock; i.e., before any of the synchronous control fields have been used by the CPU.

The Signetics 8x02 Control Store Sequencer will neither help nor hinder the implementation. It merely replaces the MIR and performs the clear, increment, and branch functions required for incrementing the projections and for data processing. The stack operations (subroutine calls and looping facilities) can be useful at some later time, but are not necessary for the algorithms currently defined. If used, the control store word width would need to be increased by a minimum of three bits, and the MCU cycle time would increase by up to 20 nanoseconds. For this application, the IAR and the branching capability can be implemented equally well with a six-bit synchronous binary counter with clear, load, and increment. A single array of ten Signetics 3002's provides a 20-bit wide CPU which has the speed to process two windows, concurrently. This assumes a 45 nanosecond typical cycle time (Signetics).

3.3 Adaptive Tracking Algorithm

The objectives of the tracking algorithm are to maintain track and provide high resolution tracking data. These two objectives are often competitive, since higher resolution is obtained by increasing the zoom setting which augments the probability of losing the target due to the decreased camera FOV. When these objectives come in conflict, the tracking algorithm gives the highest priority to maintaining track.

The inputs to the tracking algorithm are the structural parameters for the target and plume images obtained from the PCM. The structural parameters are:

Target Parameters

- number of target points (NT)
- coordinates of center of top half of target (XTT, YTT)
- coordinates of center of bottom half of target (XTB, YTB)
- target projection percentiles in x direction $\{TPX_i | i = 1,2,...,7\}$
- target projection percentiles in y direction $\{TPY_i | i = 1,2,...,7\}$

Plume Parameters

- number of plume points (NP)
- coordinates of center of top half of plume (XPT, YPT)
- coordinates of center of bottom half of plume (XPB, YPB)
- plume projection percentiles in x direction $\{PPX_i | i = 1,2,...7\}$
- plume projection percentiles in y direction $\{PPY_i | i = 1,2,...7\}$

The number of target and plume points is used to define the existence and size of the target and plume images. The coordinates of the top and bottom halves are used to define the orientation and location of the target and plume images. The target projection percentiles are used to describe the shape of the target image and define a measure of confidence of the target location and orientation data. The plume percentiles are used to locate the tip of the plume and define a measure of confidence in the plume location and orientation data.

The inputs to the tracking algorithm characterize the size, position, and shape of the target and plume images and define a measure of confidence in the data as shown in Figure 3.12.

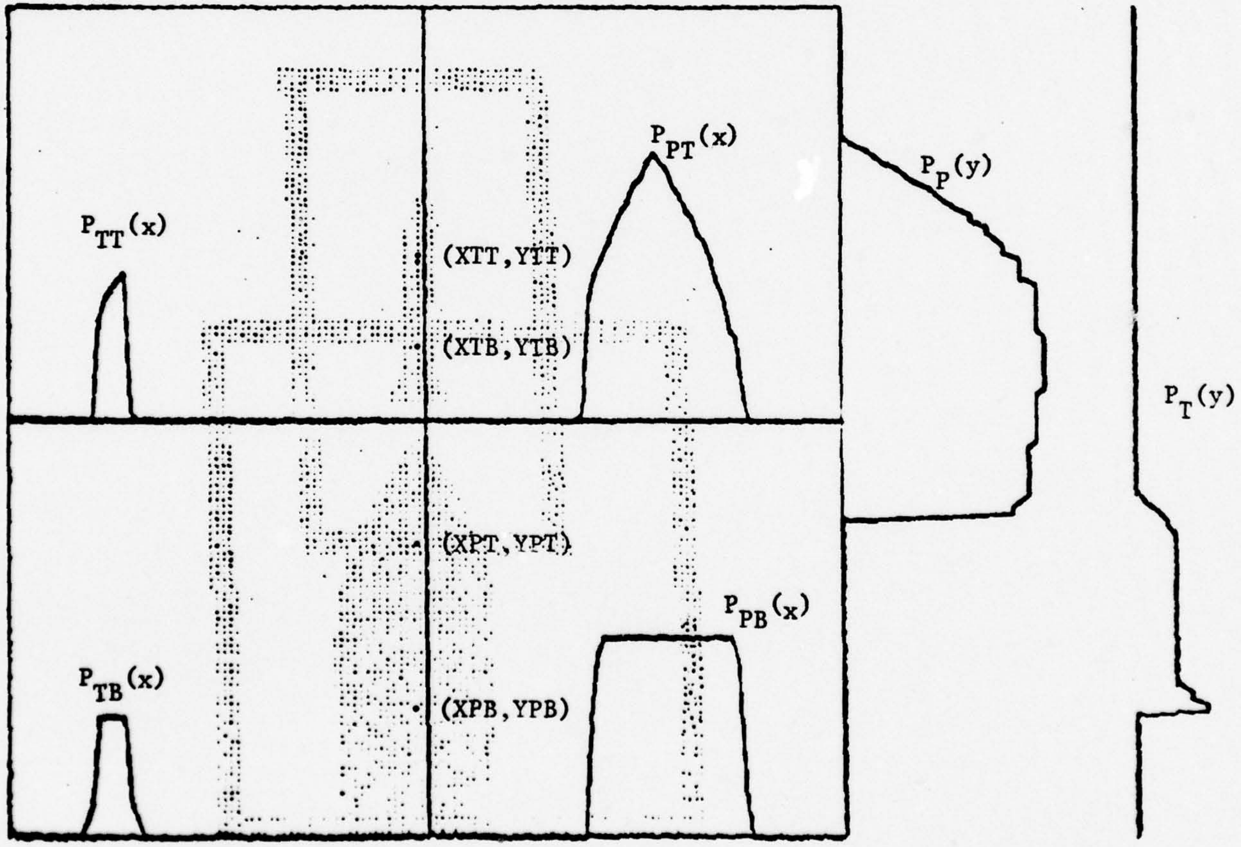


Figure 3.11. Typical tracking configuration

The tracking algorithm establishes the proper tracking strategy, outputs boresight, zoom setting, and orientation correction signals along with the confidence weight to the control processor, and controls the position and shape of the target and plume parallelogram frames for the image decomposition module. The outputs of the tracking algorithm are:

Outputs to Control Processor

- target X displacement from boresight (DX)
- target Y displacement from boresight (DY)
- target angle from vertical boresight (DY)
- desired change in zoom (DZ)
- confidence weight (W)

Outputs to Image Decomposition Module

- target parallelogram position
- target parallelogram shape
- target parallelogram orientation
- plume parallelogram position
- plume parallelogram shape
- plume parallelogram orientation

The outputs to the control processor are used to predict the target location, size, and orientation for the next frame. The boresight corrections signals are used to predict the azimuth and elevation pointing angle of the telescope. The rotation angle DR is used to rotate the image rotation element to keep the image vertical. The desired zoom change controls the zoom lens to keep the target visible within the FOV of the camera. The confidence weight is used by the control processor much like a Kalman weight to combine the measured and predicted values. When the confidence weight is low, the control processor relies more heavily on the recent trajectory to predict the location of the target on the next frame.

The outputs to the IDM define the size, shape, and position of the target and plume parallelogram frames. Two parallelograms are used so that the size, shape, and position of each parallelogram can be optimized for the image being tracked. Furthermore, the two parallelogram approach allows the target to be tracked when one of the images is lost. The parallelogram corresponding to the lost image can simply be expanded to the full FOV until the image reappears. There is no loss in resolution when the parallelograms are made larger; however, the parallelogram frames act like bandpass filters and reject unwanted noise outside the frame. The improved performance provided by the two parallelogram frames and the added flexibility of being able to track two different images in the FOV is considered as sufficient justification for the additional control logic necessary to implement two tracking parallelograms. When there is only one image in the FOV, one parallelogram frame can monitor the entire FOV while the other is tracking the target with a smaller view. If the target image escapes the smaller frame, the larger frame locates the target and directs the smaller frame back to the target. The size of the parallelogram frames is adjusted on the basis of the size of the target image and the amount of jitter in the target image location.

Some typical tracking situations are shown in the following figures.

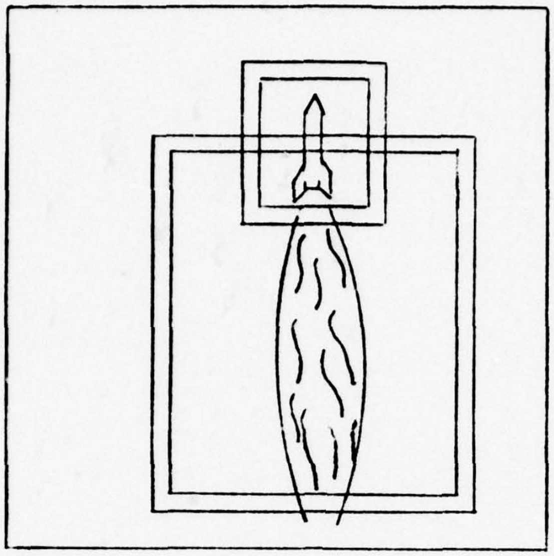


Figure 3.13. Missile tracking

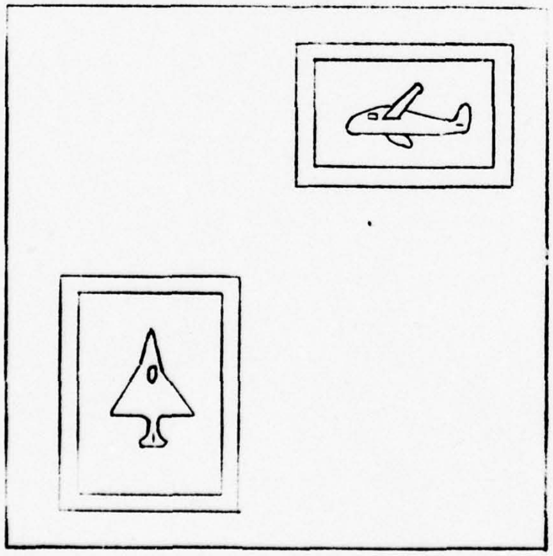


Figure 3.14. Two independent targets

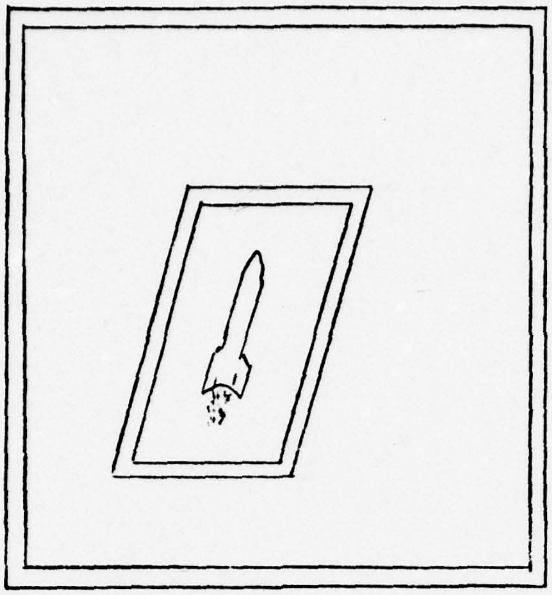


Figure 3.15. One target

Confidence Weight W

A confidence weight W is computed foı the target and plume images based upon how well they measure up to some desired features or properties. With W_T and W_P as the confidence weights characterizing the target and plume, respectively, we can define W as:

$$W = \alpha_T W_T + \alpha_P W_P$$

with contraint equations

$$\alpha_T + \alpha_P = 1$$

$$0 \leq W, W_T, W_P, \alpha_T, \alpha_P \leq 1$$

The values of α_T and α_P are dynamically adjusted by the tracking algorithm to give a realistic W under different tracking situations.

Target Confidence Weight W_T

The purpose of the target confidence weight W_T is to measure the probability that the located image is the desired target. An adaptive tracking algorithm updates the stored reference target image parameters dynamically so that the target can be tracked through different perspective views.

For each frame, the PCM computes the x and y projection percentile points $x_1, x_2, \ldots, x_7$ and $y_1, y_2, \ldots, y_7$, respectively. The target weight associated with a particular frame is given by

$$W_T = \begin{cases} 1 - \sum_{i=1}^{6} |\ell_i - \gamma_i| , & \text{if } \sum_{i=1} |\ell_i - \gamma_i| \leq 1 \\ 0 , & \text{else} \end{cases}$$

where

$$\ell_i = \frac{y_{i+1} - y_i}{\ell_T} , \qquad i = 1, 2, \ldots, 6$$

$$\ell_T = y_7 - y_1$$

The reference parameters γ_i, i = 1, . . ., 6 are computed in a manner similar to the computation of ℓ_i during initialization and continuously updated by the tracking algorithm using the formula

$$\gamma_i = \frac{N - 1}{N} \gamma_i + \frac{1}{N} \ell_i , \qquad i = 1, 2, \ldots, 6$$

Thus, W_T satisfies the constraint equation because for any frame, we have

$$\sum_{i=1}^{6} \gamma_i = \sum_{i=1}^{6} \ell_i = 1$$

Plume Confidence Weight W_P

The purpose of the plume confidence weight is to measure the probability that the located image is the plume image.

The plume in the FOV is often not an enclosed object. The shape of the plume within the plume window often varies significantly from frame to frame. Hence, two other features of the plume are used:

- plume density
- existence of a high density core

The plume weight is computed by

$$W_P = \alpha_D W_D + \alpha_C W_C$$

where

$$\alpha_D + \alpha_C = 1; \qquad 0 \leq W_D, W_C, \alpha_D, \alpha_C \leq 1$$

and W_D and W_C are measurements of the two above features.

To compute the two values W_D and W_C, the plume x and y percentiles points $(x_1, x_2, \ldots, x_7, y_1, y_2, \ldots, y_7)$ computed by the PCM are used.

$$W_D = \frac{NP}{2\,(x_7 - x_1)\,(y_7 - y_1)}$$

$$W_C = \begin{cases} 1, \text{ if } \dfrac{x_7 - x_1}{x_5 - x_3} \leq 3 \\ \\ 0, \text{ otherwise} \end{cases}$$

It is easy to show that W_D can only have values between 0 and 1.

The Tracking Algorithm as a Finite State Machine

The basis for an intelligent tracking algorithm is the ability to respond to a current input based upon the sequence of inputs that lead to the current state (or situation). A sequential machine possesses such a property because each state represents the collection of all input sequences that take the machine from the initial state to the present state (Nerode's tape equivalence). By defining an equivalence relation R on the tape set Σ^* as

$$xRy \text{ if } \delta(s_o, x) = \delta(s_o, y) \quad \forall\, x, y \in \Sigma^*$$

the tape set Σ^* can be partitioned into equivalent classes

$$[x] = s_i = \{y \mid xRy \;\forall\, y \in \Sigma^*\}$$

Consequently, a state represents all input sequences that produce a given tracking situation. This interpretation of input sequences transforms the development of the tracking algorithm into a problem of defining a finite state machine

$$TA = (S, I, Z, \delta, \omega)$$

The states of the machine $S = \{s_1, s_2, s_3, . . ., s_n\}$ define the different tracking situations that must be handled by the tracking algorithm. The input I to the finite state machine is derived from the image parameters and characterizes the size, shape, and location of the present target and plume images. The output set Z defines a finite set of responses that the tracking algorithm employs for maintaining track and retaining high resolution data. The next state mapping $\delta{:}S \times I \rightarrow S$ defines the next state $\delta(s_i, i_j) = s_k$ when an input i_j is applied to state s_i. The output mapping $\omega{:}S \rightarrow Z$ is a Moore output that defines the proper tracking strategy (response) for each state.

Input Set I

The inputs to the sequential machine discretely measure the value and rate of change of the size, location, and confidence weights of the target and plume images.

The important target and plume input variables are:

- image sizes (NT,NP)
- target and plume confidence weights (W_T, W_P)
- image displacement from boresight (ΔxT, ΔyT, ΔxP, ΔyP)
- rate of change in target size (ΔNT)
- rate of change in weight (ΔW_T, ΔW_P)
- rate of change in ΔxT, ΔyT, ΔxP, ΔyP

State Set S

The sequential machine TA has three major classes of states. The first class corresponds to a set of normal tracking states which require no modification in the tracking strategy. The second class corresponds to a set of states which indicate that the target image is moving out of the FOV. The third class is a set of states corresponding to situations when the image under track goes through abrupt changes in the shape and/or size within the FOV.

There are three major macro-states in the normal tracking state set:

- plume and target both under track (S_1)
- only one image (target or plume) under track (S_2)
- neither target nor plume under track (S_3)

The state S_1 corresponds to the most desirable tracking state for missile targets where both the target and plume images are being tracked. The second state S_2 corresponds to the situation where only one image is being tracked. For targets without plume, state S_2 is the normal tracking state. The third state S_3 handles the situations where the target and plume images are lost and tracking is guided by previous trajectory data.

There are two major macro-states in the FOV state set corresponding to one or both images leaving the FOV. Each macro-state can be further resolved into several states with different output strategies for driving the tracking algorithm back to the normal tracking state set. The first macro-state S_4 corresponds to the situation where one of the two images under track is leaving the FOV. The second macro-state S_5 corresponds to the situation where the only image under track is leaving the FOV.

There are two major macro-states in the abrupt change state set which correspond to the situation where the images are abruptly changing in the FOV. These macro-states can be further resolved into finer states with different outputs strategies to bring the tracking algorithm back to the normal tracking state. The first macro-state, S_6, handles the situation where one of the two images changes size or shape rapidly in the FOV. The other state, S_7, handles the situation where the only image under track is changing rapidly.

Output Set Z

The output set Z defines a finite set of responses that the structural tracker (i.e. the tracking algorithm) can make to maintain track while retaining high resolution data. Several of these responses are internal to the digital IDM and the structural tracker; hence, they are performed at electronics speed. However, the zoom lens operation is mechanical in nature and requires interaction with the control processor.

The output set Z contain the following responses:

- shape variation of the parallelogram windows
- location variation of parallelogram windows

- α_T and α_P value settings
- zoom setting variation
- structural parameters update procedure control

Next State Mapping δ

The structural tracker's view of the world is only a restricted version of what the control processor sees. However, it does have a higher resolution in that it can examine in detail what is inside each window; whereas the control processor receives only the information of Δx, Δy, W, etc. from the tracker. What the control processor does (to the optics train) is not fully known to the tracker - due mainly to the time limitation for the tracker to perform its tasks. Consequently, the tracker often has to, at best, guess at the situation it encounters and enter an appropriate state whose output will produce inputs that provide more information on the current situation.

The motivations of the next state mapping δ for the tracking algorithm TA are:

1. To enter a state in the normal tracking set with the best tracking condition attainable under current input conditions.
2. To enter a state whose output corresponds to strategies that will produce an input to achieve 1.

Output Mapping ω

The motivation for the output mapping is to associate with each state an intelligent strategy that will:

- maintain track and high image resolution
- achieve a desirable mapping to a next state that best describes the current situation
- provide accurate data for the Control Processor and the IDM

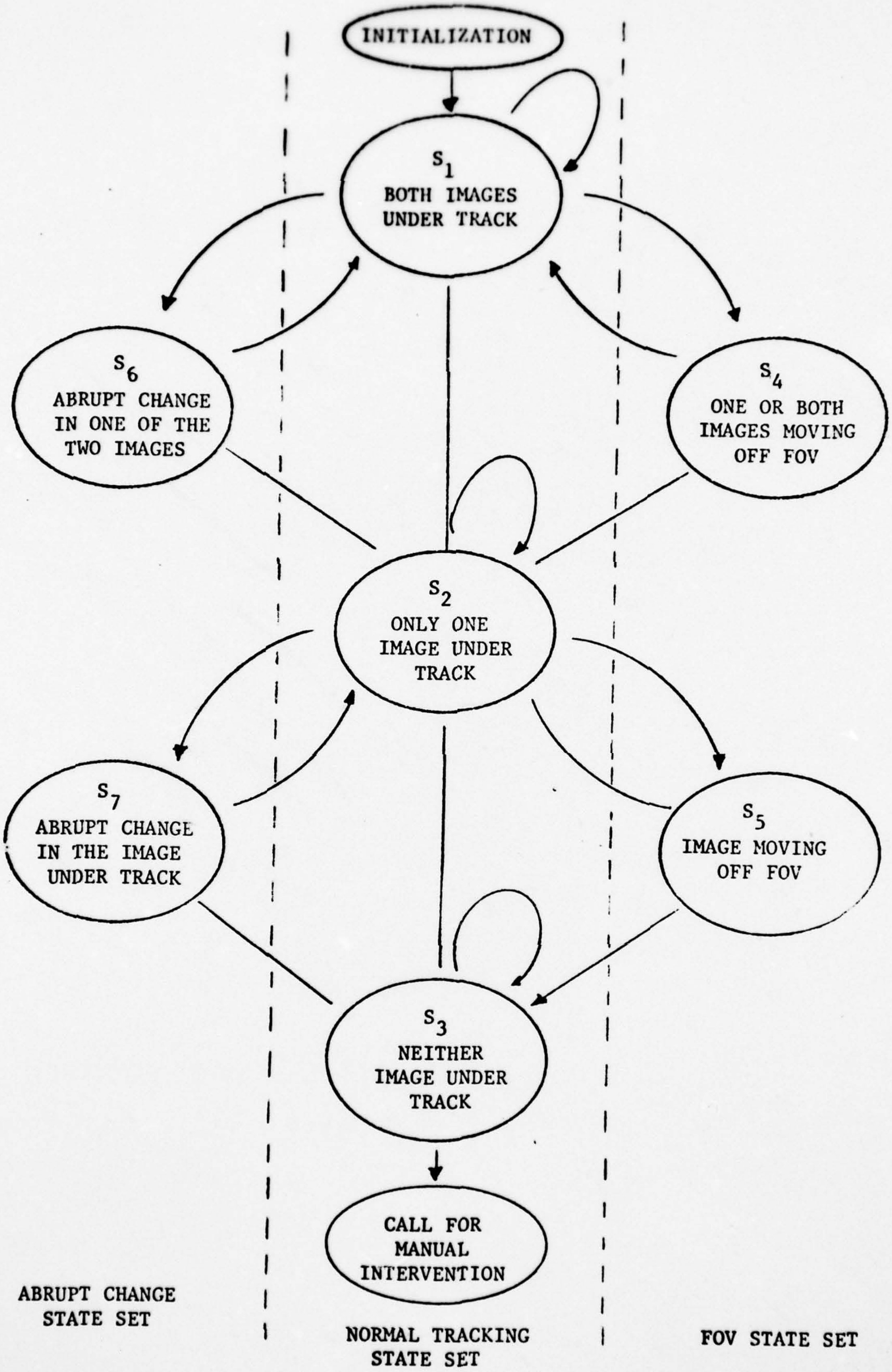

INITIALIZATION
S1
BOTH IMAGES UNDER TRACK
S6
ABRUPT CHANGE IN ONE OF THE TWO IMAGES
S4
ONE OR BOTH IMAGES MOVING OFF FOV
S2
ONLY ONE IMAGE UNDER TRACK
S7
ABRUPT CHANGE IN THE IMAGE UNDER TRACK
S5
IMAGE MOVING OFF FOV
S3
NEITHER IMAGE UNDER TRACK
CALL FOR MANUAL INTERVENTION
ABRUPT CHANGE STATE SET
NORMAL TRACKING STATE SET
FOV STATE SET

Figure 3.16. Tracking algorithm

For example, within the normal tracking state set, the different states require different values for α_T and α_P. In state S_1 where both target and plume images are under track, the values α_T and α_P would be set to 0.5 each. For state S_2, where only one image is being tracked, either α_T or α_P will be set to 1 and the other to 0, i.e., either $W = W_T$ or $W = W_P$. When neither image is under track in S_3, both α_T and α_P will be set to 0, resulting in $W = 0$.

Once the finite state machine TA has been defined, a standard ROM realization can be used to implement the tracking algorithm. The simplicity, modularity, and speed of the ROM realization make the finite state machine approach very attractive for the RTV tracking system.

IV. CONTROL PROCESSOR

The control processor closes the loop between the structural tracker and the optics and mount control system. The purpose of the control processor is to keep the tracking optics pointed at the target through launch and during flight. There are three sources of information available to the control processor: optical coordinates from the structural tracker, radar coordinates, and a manual override. The optical input to the control processor consists of target position boresight correction coordinates (Δx, Δy), zoom lens correction (Δz), orientation correction ($\Delta\phi$), and a structural weight (W) associated with these values. The optical data is available to the control processor every TV frame. The radar data comes from the range radar at 200 Hz rate (every 50 ms) and consists of azimuth (Θ_{AZ}), elevation (Θ_{EL}), range (R), and range rate ($\dot{R}$). There is, however, a 200 millisecond delay associated with the radar data that is caused by radar processing time.

There are several inherent differences between the optical and radar data. First, the optical system tracks the target in real-time with a two dimensional pointing vector; while the radar tracks the target in three dimensional space with a 200 millisecond delay. Secondly, when the target is visible, the optics data should be an order of magnitude more precise than the radar data. The radar data, however, is available when the optical data is lost due to clouds or poor visibility. The control processor must take these differences into consideration and generate the "best" estimate control signals to point the optics towards the target. The situation is similar to having two hunters, one with open sights and the other with a high power scope, track a moving target and direct the other to the target when the target is lost.

4.1 Control Structure

The control structure of the video tracking system is depicted in Figure 4.1. The tracking optics feeds the target image to the video processor which establishes the target coordinates with respect to the optics boresight. The control processor combines current target coordinates with previous target coordinates to point the optics toward the next expected target position forming a fully automatic tracking system. Furthermore, the control processor receives data from the tracking radar that can be utilized to continue track during poor visibility conditions. A manual override capability is provided to reacquire track if necessary.

The control processor configuration is shown in Figure 4.1. It consists of separate predictors for the optics and radar data. A decision module is also included which decides which predictor should be used or whether both should be neglected in the case of a manual override. Both the optical predictor and the radar predictor run simultaneously, solving the two and three dimensional tracking problems, respectively. Since the optical data is generally more precise, it is used as the primary tracking predictor and also to provide periodic updates for the radar predictor which is used during periods of poor optical visibility. This technique of using the more precise optical data to update the radar predictor is similar to the Cruise missile problem where the inertial navigation system is initialized in flight using radio position fixes. Using this technique, the radar tracking predictor error is periodically calibrated with the optical tracking system. If the optical data is lost, then the radar predictor can assume control with the benefit of a recent optical calibration.

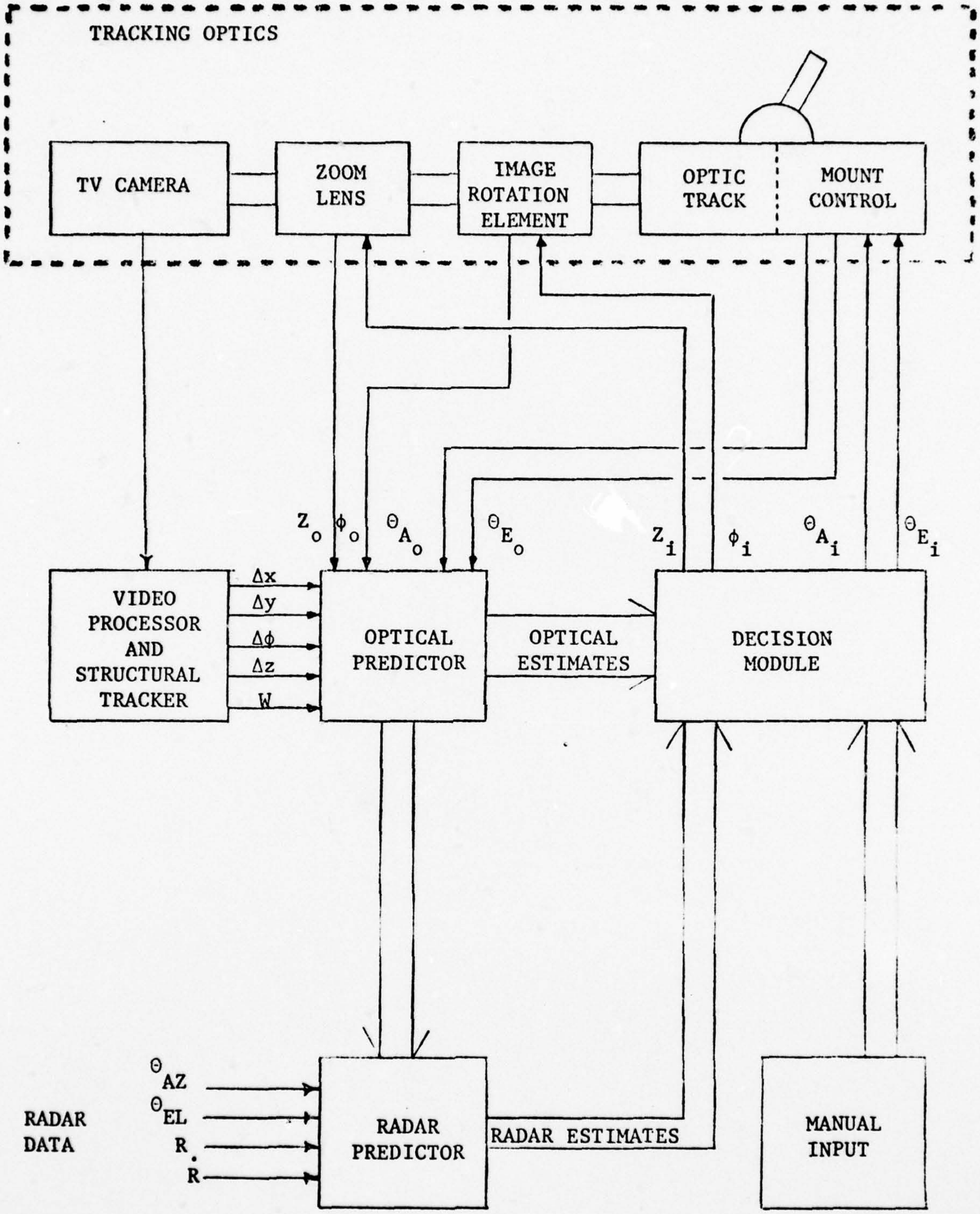


Figure 4.1. Control processor configuration

The purpose of the optical predictor is to generate the primary control signals: Θ_{E_i} and Θ_{A_i} for the tracking mount, Z_i for the zoom lens, and ϕ_i for the rotating element. The derivation of the optical predictor utilizes previous tracking data to predict the target location.

The optical data received from the structural tracker is illustrated in Figures 4.2 and 4.3. Figure 4.2 shows the measured values relative to the camera field-of-view, and Figure 4.3 illustrates the effect of the image rotation element by showing the camera FOV rotated relative to the vertical reference.

Let Θ_{A_o} and Θ_{E_o} denote the azimuth and elevation pointing angles of the camera boresight, ϕ_o denote the camera rotation angle, and $(\Delta x, \Delta y)$ denote the target displacement from boresight. The azimuth and elevation pointing angles of the target $(\Theta_{A_t}, \Theta_{E_t})$ can be established by rotating the axes and comparing the target displacements to the camera FOV, Θ_{FOV}/Z_o, which gives

$$\Theta_{A_t} = \Theta_{A_o} + \left(\frac{\Delta x}{X} \cos \phi_o + \frac{\Delta y}{Y} \sin \phi_o\right) \frac{\Theta_{FOV}}{Z_o \cos \Theta_{E_o}}$$

$$\Theta_{E_t} = \Theta_{E_o} + \left(\frac{\Delta y}{Y} \cos \phi_o - \frac{\Delta x}{X} \sin \phi_o\right) \frac{\Theta_{FOV}}{Z_o}$$

The cos Θ_{E_o} term in the azimuth equation is required, since the azimuth displacement is measured at Θ_{E_o} elevation and the azimuth gimbal is in the horizontal plane as shown in Figure 4.4. The azimuth angle $d\alpha$ measured at an elevation angle E_o is related to the azimuth angle in the horizontal plane by

$$R d\alpha = R \cos (\Theta_{E_o}) d\phi$$

and

$$d\phi = \frac{d\alpha}{\cos \Theta_{E_o}}$$

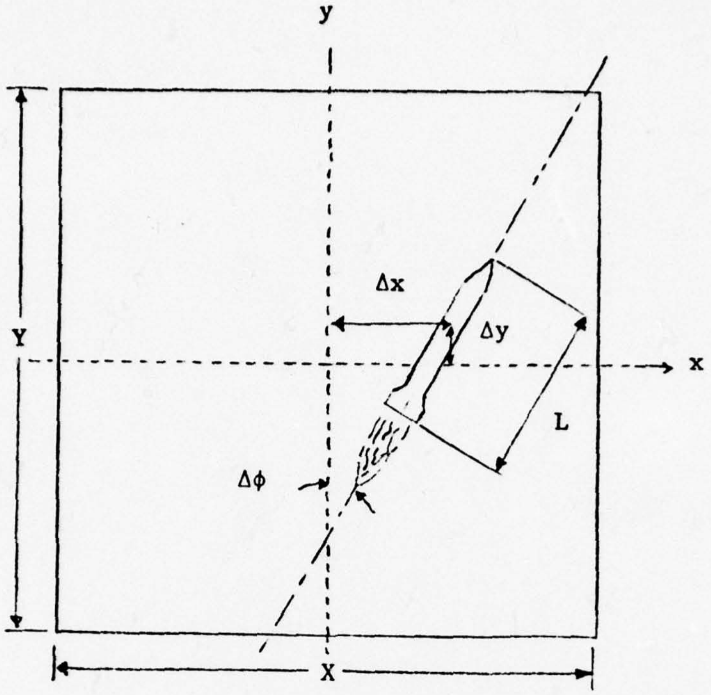


Figure 4.2 Target location parameters

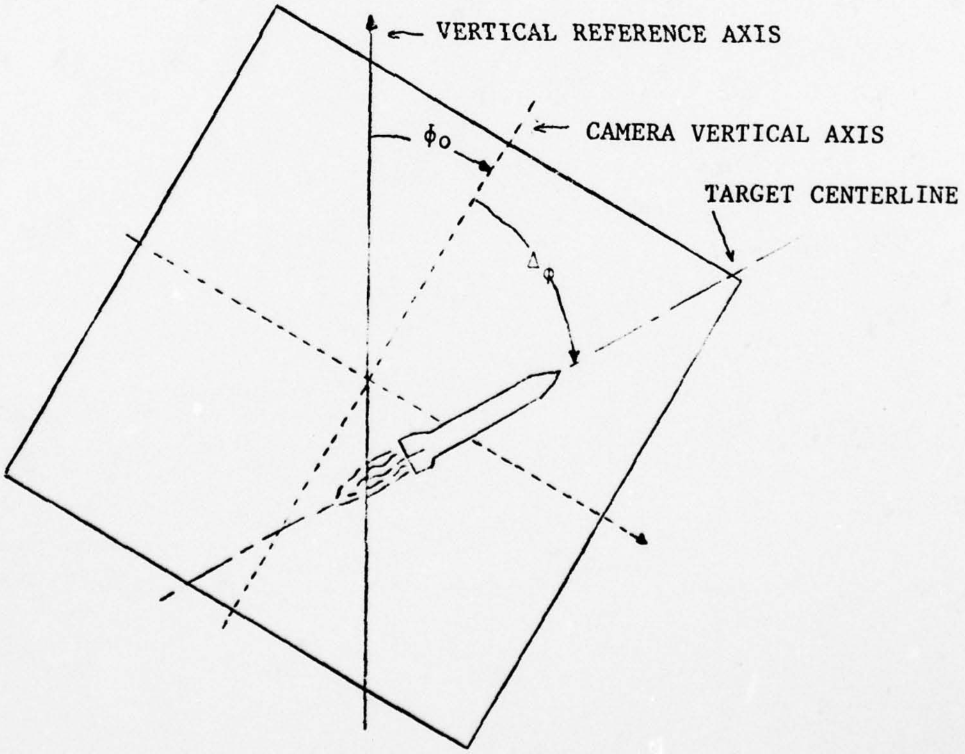


Figure 4.3 Camera rotation angle

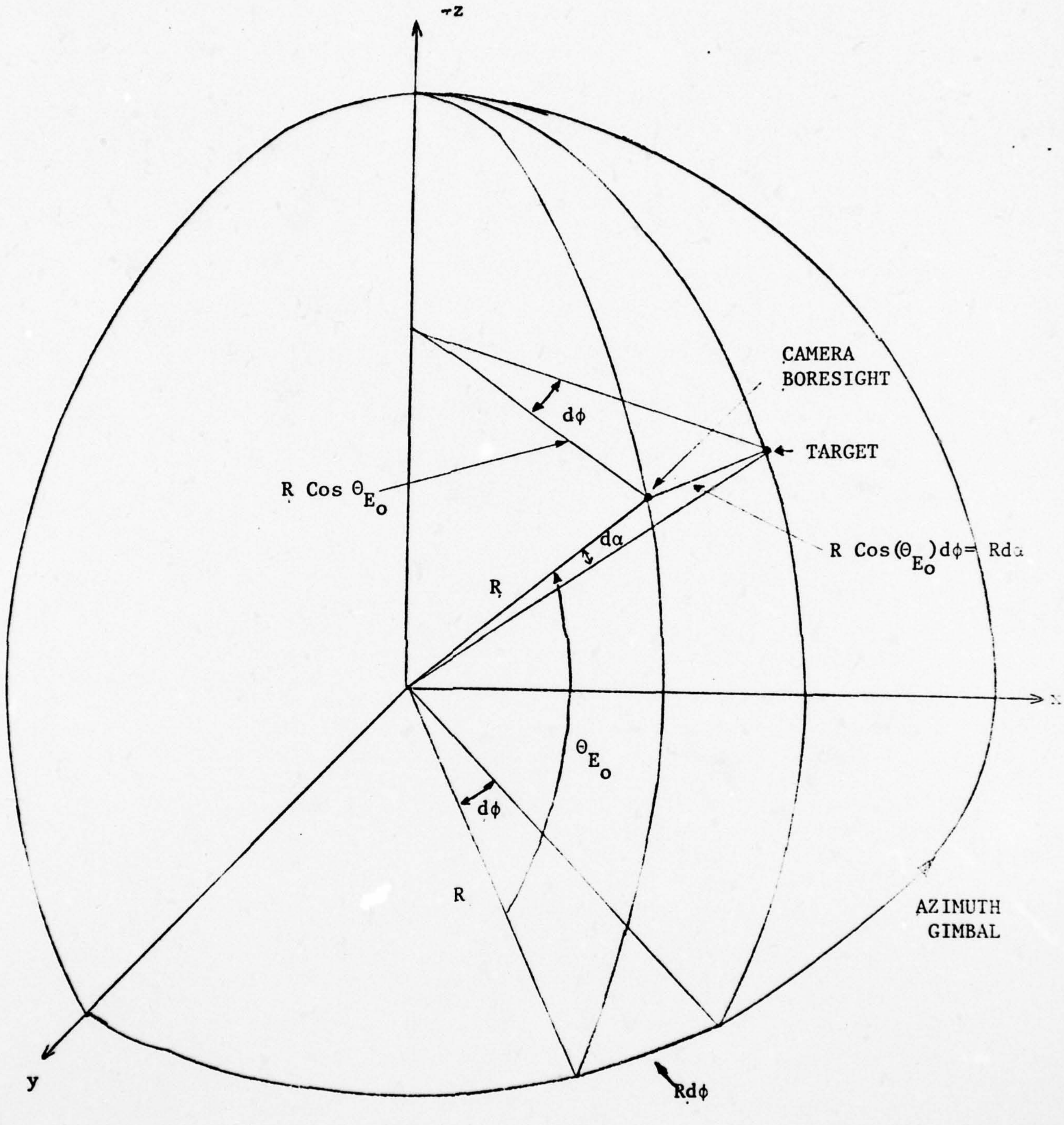


Figure 4.4. Azimuth computation

4.2 The RTV System Model

This section presents the analytic models of the dynamical systems to be controlled by the control processors. These models describe the dynamics of the TV camera mount azimuth and elevation gimbals, the zoom lens, and the image rotation element.

TV Camera Mount Dynamics

The camera mount control system provides a position tracking system capable of being directed by digital pointing data at an input rate of up to 200 samples per second. The azimuth and elevation gimbals are uncoupled and their transfer functions have the general forms

$$\frac{\Theta_{A_o}(s)}{\Theta_{A_i}(s)} = \frac{a_0 + a_1 s + a_2 s^2 + a_3 s^3 + a_4 s^4}{b_0 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4}$$

and

$$\frac{\Theta_{E_o}(s)}{\Theta_{E_i}(s)} = \frac{c_0 + c_1 2 + c_2 s^2 + c_3 s^3 + c_4 s^4}{d_0 + d_1 s + d_2 s^2 + d_3 s^3 + d_4 s^4}$$

Using actual engineering data from [25] and [26] the specific transfer functions are

$$\frac{\Theta_{A_o}(s)}{\Theta_{A_i}(s)} = \frac{28758 + \left(8558 + 834k_{3A}\right) s + 4321k_{3A} s^2}{28758 + 8867\ s + 1606\ s^2 + 32\ s^3 + s^4}$$

$$\frac{\Theta_{E_o}(s)}{\Theta_{E_i}(s)} = \frac{47168 + \left(10483 + 1060\ k_{3E}\right) s + 5467\ k_{3E} s^2}{47168 + 1087\ s + 2031\ s^2 + 36\ s^3 + s^4}$$

where k_{3A} and k_{3E} are the gain constants for the rate feed-forward loops in the azimuth and elevation channels. They have not been sepcified at this time, but will be determined through parametric analysis.

In terms of state-variables one has for azimuth

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -28758 & -3867 & -1606 & -32\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ 4321\ k_{3A}\\ 8558 - 137439\ k_{3A}\\ -245098 - 2541478\ k_{3A}\end{bmatrix}\Theta_{A_i}$$

$$\Theta_{A_o} = [1\ 0\ 0\ 0]\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}$$

and for elevation

$$\begin{bmatrix}\dot{x}_5\\ \dot{x}_6\\ \dot{x}_7\\ \dot{x}_8\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -47168 & -10876 & -2031 & -36\end{bmatrix}\begin{bmatrix}x_5\\ x_6\\ x_7\\ x_8\end{bmatrix} + \begin{bmatrix}0\\ 5467\ k_{3E}\\ 10483 - 195752\ k_{3E}\\ -330225 - 4056405\ k_{3E}\end{bmatrix}\Theta_{E_i}$$

$$\Theta_{E_o} = [1\ 0\ 0\ 0]\begin{bmatrix}x_5\\ x_6\\ x_7\\ x_8\end{bmatrix}$$

Zoom Lens Dynamics

The zoom lens control system has not been designed. However, for the present, assume the transfer function has the general form of a second order system

$$\frac{z_0(s)}{z_i(s)} = \frac{1}{q_0 + q_1 s + q_2 s^2}$$

AD-A059 052 NEW MEXICO STATE UNIV LAS CRUCES ENGINEERING EXPERIM--ETC F/G 17/8
A PRE-PROTOTYPE REAL-TIME VIDEO TRACKING SYSTEM.(U)
JAN 77 G M FLACHS, W E THOMPSON, R J BLACK DAAD07-76-C-0024

UNCLASSIFIED NL

2 OF 2
AD
A059052

END
DATE
FILMED
11-78
DDC

The corresponding state-variable form is

$$\begin{bmatrix} \dot{x}_9 \\ \dot{x}_{10} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -q_0 & q_1 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} z_i$$

$$z_o = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \end{bmatrix}$$

Since this project involves the specification of the zoom lens dynamics, the parameters will be established to meet the tracking requirements. At this time, assume a critically damped system and a natural frequency of around 20 Hertz (since the camera mount control system provides digital pointing data at up to 200 samples per second.)

The specific system equations are

$$\begin{bmatrix} \dot{x}_9 \\ \dot{x}_{10} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -15791 & -178 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} z_i$$

$$z_o = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_9 \\ x_{10} \end{bmatrix}$$

Image Rotation Element Dynamics

The control system for the image rotation element has not been designed. However, assume that the transfer function has the general form

$$\frac{\phi_o(s)}{\phi_i(s)} = \frac{1}{r_o + r_1 s + r_2 s^2}$$

The corresponding state-variable form is

$$\begin{bmatrix} \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -r_0 & -r_1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \phi_i$$

$$\phi_o = [\,1 \quad 0\,] \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix}$$

The parameters for the image rotation element dynamics will also be chosen to meet the tracking requirements. Assuming a critically damped system and a natural frequency of around 20 Hertz, the state equations are

$$\begin{bmatrix} \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -15791 & -178 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \phi_i$$

$$\phi_o = [\,1 \quad 0\,] \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix}$$

4.3 The Control Processor Development

The purpose of this section is to formulate the control processor problem and present some initial studies toward the solution of the problem. The control processor receives inputs from the structural tracker on the target location (Δx, Δy), orientation ($\Delta\phi$), and length (Δz), measured relative to boresight (Figure 4.2) and a confidence weight (W) that measures the reliability of the measured data. Using these inputs, the control processor predicts the desired tracker azimuth and elevation pointing angles ($\Theta_A(k+1)$, $\Theta_E(k+1)$) the zoom lens setting ($Z(k+1)$), and the image rotation setting ($\phi(k+1)$) for the next frame denoted by $(k+1)$.

Several considerations of the tracking problem make the development of the tracking estimator unique. First, the real-time constraint of providing the control signals between frames at frame rates up to sixty frames per second eliminates many lengthy computational methods. Secondly, a confidence weight derived from the structural shape of the target that measures the reliability of the measured data is available. Finally, the large class of targets and tracking environments require the estimator to be adaptive to the tracking environment. Based on these considerations, a predictor method has been developed that uses the confidence weight, much like a Kalman weight, to combine the measured and estimated values to predict the control signals for the next frame. Several simple predictors are defined and the "best" estimate of the predictors is used to define the control signals for the next frame. The "best" estimate is based upon how well the predictors predicted the last frame.

Since the form of the estimator equations is similar for the azimuth, elevation, zoom, and image rotation variables, the following notation will be used in the development.

k	frame index
j	time of predictor index
$\Theta(k)$	variable to be estimated or predicted (azimuth, elevation, rotation, or zoom)
$\Theta_m(k)$	measured value of $\Theta(k)$
$\hat{\theta}(k)$	estimated value of $\Theta(k)$ using measurements through k^{th} frame
$\hat{\theta}_j(k+1\|k)$	predicted value of $\Theta(k+1)$ using measurements through k^{th} frame and k index type predictor
$\hat{\theta}(k+1\|k)$	predicted value of $\Theta(k+1)$ using combination of set predictors
$W(k)$	confidence weight

Using the confidence weight in a manner similar to that of a Kalman gain [29 - 30], the estimated value is obtained from the measured and predicted values by

$$\hat{\theta}(k) = (1-W(k))\ \hat{\theta}(k|k-1) + W(k)\ \Theta_m(k)$$

where the confidence weight is normalized such that $0 \leq W(k) \leq 1$. Then the predictors having the function form

$$\hat{\theta}(k+1|k) = F(\hat{\theta}(k), \hat{\theta}(k-1), \ldots, \hat{\theta}(k-r))$$

rely more heavily on the estimated values when the weight is low. This is very important to continue track when the target passes through clouds or noisy backgrounds such as the mountain/sky interface.

The next step in the development is to establish a set of simple targets through the tracking environments of interest, yet simple enough to be performed in real-time. Due to the time constraints, only linear and quadratic predictors have been investigated; however, any predictor that can be performed within the real-time tracking constraints can be utilized.

Linear N Point Predictor

Assuming that the variable to be predicted is linear in time

$$\hat{\theta}(t) = [1 \quad t] \begin{bmatrix} q_o \\ q_1 \end{bmatrix}$$

and minimizing the mean square error

$$MSE = \sum_{k=1}^{N} (\hat{\theta}_j(k|k-1)-a_0-a_1t_k)^2$$

over the last N frames, we obtain the solution

$$\hat{\theta}_j(k+1|k) = [1 \quad t_{k+1}] \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

where

$$\begin{bmatrix} q_0 \\ q_1 \end{bmatrix} = \begin{bmatrix} N & \Sigma t_k \\ \Sigma t_k & \Sigma t^2_k \end{bmatrix}^{-1} \begin{bmatrix} \Sigma \hat{\theta}(k) \\ \Sigma \hat{\theta}(k)t_k \end{bmatrix}$$

and all summations are from k=1 to k=N. For N=2 and equally spaced points t_k=k we obtain the standard two point linear predictor

$$\hat{\theta}_j(k+1|k) = 2\hat{\theta}(k) - \hat{\theta}(k-1)$$

For N=3 and equally spaced points t_k=k we obtain a three point linear predictor

$$\hat{\theta}_j(k+1|k) = \frac{1}{3}[4\hat{\theta}(k) + \hat{\theta}(k-1) - 2\hat{\theta}(k-2)]$$

The linear predictor with N=2 has the advantage that it requires only two points (measured values) to initiate the prediction. This is importanc in initially acquiring track.

Quadratic N Point Predictor

By assuming that the variable to be predicted is quadratic in time

$$\hat{\Theta}\ (t) = [1 \quad t \quad t^2] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

and minimizing the mean square error over the last N points, we obtain the solution

$$\hat{\Theta}_j(k+1|k) = [1 \quad t_{k+1} \quad t^2_{k+1}] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

where

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} N & \Sigma t_k & \Sigma t^2{}_k \\ \Sigma t_k & \Sigma t^2{}_k & \Sigma t^3{}_k \\ \Sigma t^2_k & \Sigma t^3{}_k & \Sigma t^4{}_k \end{bmatrix}^{-1} \begin{bmatrix} \Sigma\ \hat{\Theta}(k) \\ \Sigma\ \hat{\Theta}(k)t_k \\ \Sigma\ \hat{\Theta}(k)t^2{}_k \end{bmatrix}$$

For N=5 and equally spaced points t_k=k, we obtain the five point quadratic predictor

$$\hat{\Theta}_j(k+1|k) = (9\hat{\Theta}(k) - 4\hat{\Theta}(k-2) + 3(\hat{\Theta}(k-5) - \hat{\Theta}(k-4)))/5$$

Predictor Combination

The final step in the derivation of the predictor is to combine the estimates from the different predictors into a combined estimate of the variable for the next frame. If there are n different predictors, then the combined estimate is given by

$$\hat{\theta}\ (k+1) = \sum_{j=1}^{n} \alpha_j\ \hat{\theta}_j(k+1) \quad \text{where} \quad \sum_{j=1}^{n} \alpha_j = 1$$

The weights α_j are determined on the basis of the errors

$$E_j(k) = |\ \hat{\theta}_j(k|k-1) - \hat{\theta}(k)\ |$$

made by the predictors on their estimate of the last frame. Initially, the linear two point and the quadratic five point predictors are utilized and the weights are chosen to be

$$\alpha_1 = 1-\alpha_2 = \frac{E_2(k) + 1}{E_1(k) + E_2(k) + 2}$$

The resulting combined predictor is given by

$$\hat{\theta}(k+1) = \alpha_1[2\hat{\theta}(k) - \hat{\theta}(k-1)] + \frac{\alpha_2}{5}\,[9\hat{\theta}(k) - 4\hat{\theta}(k-2) + 3(\hat{\theta}(k-5) - \hat{\theta}(k-4))]$$

Having the α_j weights established for each frame on the basis of how well the predictor estimated the last frame, provides a real-time adaptive method for predicting the desired azimuth, elevation, zoom, and image rotation response.

The N point linear and N point quadratic predictors are shown here for illustrative purposes. However, initial investigations show that the combination works quite well. The estimation scheme is set up so that it is not restricted to any certain prediction, and the use of other types will be investigated in future efforts as the need arises.

V. SIMULATION MODEL

The purpose of this simulation model is to provide a method for testing new design concepts and evaluating the performance of the RTV tracking system under realistic tracking conditions. The simulation model includes dynamic models for the target trajectory, the Contraves Model F Cinetheodolite tracking system, the RTV tracking system, and the control processor. The target is initially located at the launch site, $\vec{L} = (0, 0, 0)$, and its trajectory is computed in three space from acceleration profiles (A_x, A_y, A_z) in the x, y, z orthogonal directions. The tracker is located at $\vec{T} = (T_x, T_y, T_z)$ and it is initially positioned so that the target will pass through the FOV of the tracking optics.

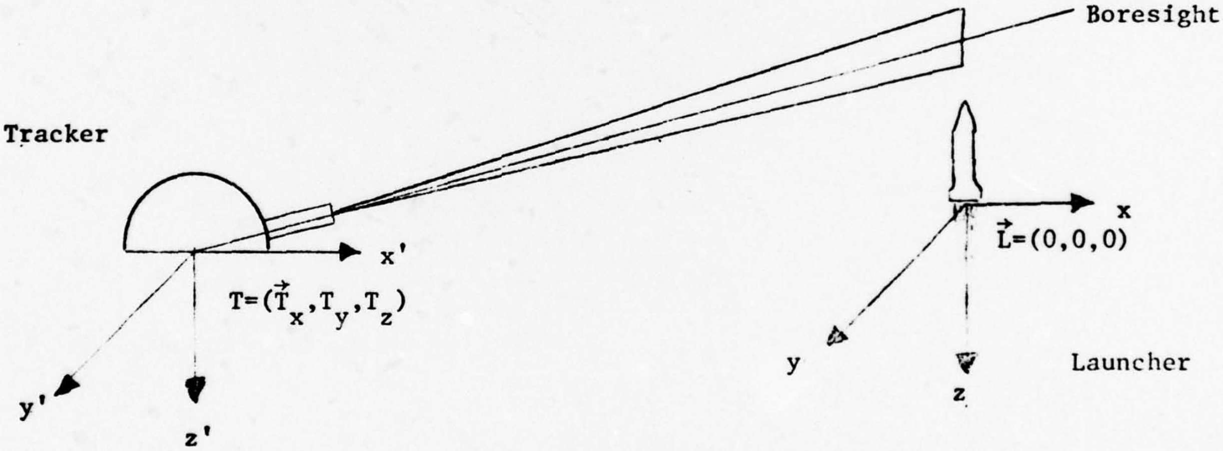


Figure 5.1. Tracking configuration

When the target passes through the FOV of the optics, the RTV tracking system locates the target image and provides the boresight corrections to the control processor, which provides the signals that direct the tracker into the automatic tracking model. The target image as seen by the tracker is established by placing a three dimensional target image onto the target trajectory and projecting the target image onto the plane normal to the camera FOV, thus simulating a realistic perspective of the target from the view of the tracking optics.

The optical image is statistically generated with intensity distribtuions for the target image F_T, the plume image F_P, the background image F_B, and the foreground images F_{FG}. The characteristics of these distributions were established through studies on the automatic programmable film-reader (APFR) at WSMR; however, experimental studies are planned to establish the characteristics of these distributions with a TV camera on the range. The intent of the simulation is to make the models as realistic as possible in order to evaluate the performance of the tracking system with the nasty realities of the tracking environment. Some of these are:

- Initial target acquisition problems.
- Rapid changes in the target intensity distribtuion caused by sun glare problems.
- Loss of target image due to foreground objects such as clouds.
- Structural changes in target image caused by target orientation and/or noisy background.
- Dynamics of the tracking optics which includes tracker, zoom lens, and image rotation control system.

The simulation model provides a method to evaluate the performance of the total tracking system for different targets and acceleration profiles. Tracking accuracy can be established by comparing the actual target location with the estimated target location determined by the RTV tracking system.

The simulation model (Figure 5.2) is written in Fortran IV language and utilizes an interactive Textronix graphics terminal to display the performance of the tracking system. The target and plume images are plotted with respect to the boresight of the camera FOV. When the target image is located by the structural tracker, the located target position and orientation are plotted relative to the actual location. The control processor estimates the location of the target on the next frame and provides the control signals to place the camera boresight at the target. The performance of the control processor can be evaluated by comparing the target position relative to the camera boresight.

The basic steps in the simulation model are:

Step 1: The program begins by initializing the target and tracker dynamic models. The target model is used to generate the target position and orientation from the specified acceleration profiles, and the tracker is initially positioned so that the target will pass through the FOV of the camera.

Step 2: When the target position and orientation are established, the target and/or plume image shapes are generated and projected onto the plane normal to the FOV of the camera, taking into account the zoom lens and image rotation systems.

Step 3: Having established the shape of the target, plume, and foreground images, the intensity distributions F_T, F_P, and F_{FG} are used to simulate the digitized output from the TV camera.

Step 4: The image decomposition module (IDM) places parallelogram frames about the target and plume images; analyzes the target, plume, and background intensity distributions; and classifies the points inside the target and plume parallelogram frames as either target, plume, or background. The resulting target and plume images are plotted beside the actual images to provide a visual display of the performance of the image decomposition algorithm.

Step 5: The binary target and plume outputs from the IDM are processed by the projection computation module (PCM) to establish the horizontal and vertical projections of the target and plume images. When the projections are completed, the structural parameters which locate and describe the structure of the images are computed.

Step 6: The structural tracking module utilizes the structural parameters to locate the target and plume images and computes the structural weight. An adaptive algorithm is used to update the stored structural characteristics of the target image.

Step 7: Using the location of the target and/or plume images relative to the camera boresight and the structural weight, the control processor computes the boresight correction signals to place the target image at boresight on the next frame. The performance of the control processor is established by comparing the position of the target image relative to boresight.

Step 8: The boresight control signals are used by the tracking optics model to establish the orientation of the tracking optics, the zoom lens setting, and the orientation of the image rotation element.

Step 9: The simulation continues by looping back to the target model to establish the location of the target image on the next frame and generate the image in the FOV of the camera.

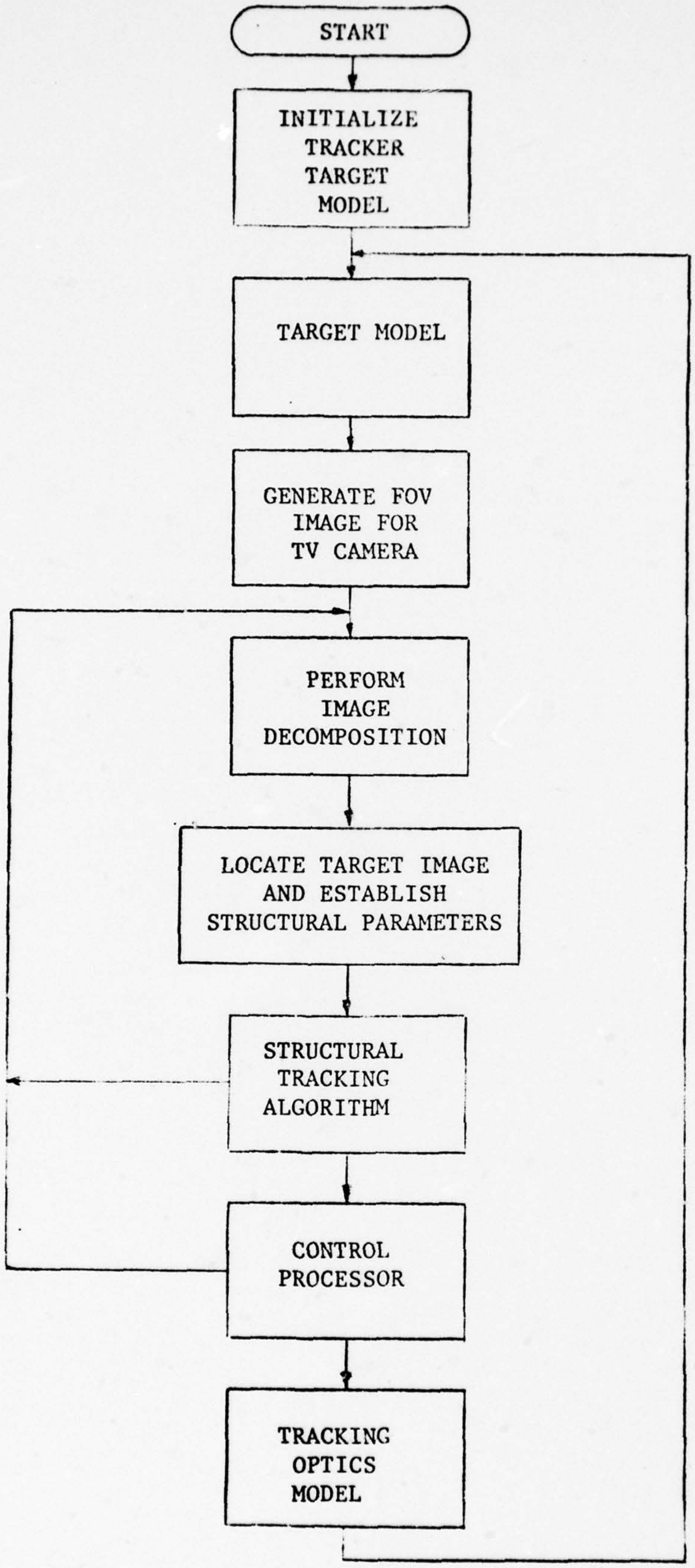

START
INITIALIZE TRACKER TARGET MODEL
TARGET MODEL
GENERATE FOV IMAGE FOR TV CAMERA
PERFORM IMAGE DECOMPOSITION
LOCATE TARGET IMAGE AND ESTABLISH STRUCTURAL PARAMETERS
STRUCTURAL TRACKING ALGORITHM
CONTROL PROCESSOR
TRACKING OPTICS MODEL

Figure 5.2 Simulation flowchart

The simulation model is implemented with a modular structure using the Fortran IV programming language. With the modular structure, the tracking algorithms can be easily changed and evaluated relative to the performance of the complete tracking system, which includes the tracking optics dynamics. The major modules of the simulation are:

- Main Driver (DRIVR)
- Target Trajectory Generator (TRJEC)
- Tracker Dynamics (TKRDY)
- Picture Generator (GENR)
- Image Decomposition Module (IDM)
- Projection Computation Module (PCM)
- Structural Tracking Module (STM)
- Control Processor Module (CONTR)

The first four of these modules are general enough to evaluate other state-of-the-art video tracking algorithms relative to the WSMR tracking system. The last four modules are required for the structural tracking algorithm; however, they represent the major problems that must be solved by an intelligent tracking algorithm.

Main Driver (DRIVR)

The main driver routine initializes the common area and calls the other modules to form the simulation structure. During the initialization phase, the operator defines the target and/or plume shapes; the target acceleration profiles; the tracker initial position and orientation; and the statistical distributions for the target, plume and background images. These data are used to simulate the target shape, trajectory, and image as seen by the camera mounted in the tracking optics.

Target Trajectory Generator (TRJEC)

The target trajectory generator integrates the acceleration profiles to define the target position in three space for each frame. By using the tracker location and orientation, the target position is established relative to the tracker boresight. Having the exact position of the target allows the simulation model to evaluate the tracking accuracy.

Tracker Dynamics Module

The tracker dynamics model simulates the dynamics of the Model F Cinetheodolite tracker, zoom lens, and image rotation element. The inputs to the model are:

- desired azimuth pointing angle
- desired elevation pointing angle
- desired image rotation
- desired zoom setting

The outputs of the tracker dynamic model for the given frame are:

- actual azimuth pointing angle
- actual elevation pointing angle
- actual image rotation
- actual zoom setting

Picture Generator Module

The picture generator utilizes the relative positions of the target and tracking optics to establish the target perspective. By taking into account the actual zoom and image rotation values and utilizing the target and plume shapes along with the intensity distributions, the picture generator simulates the digitized video output from the camera. The image in the FOV of the camera is simulated by a 512 x 512 grid of pixels, each pixel being defined by an intensity level L, $0 \leq L \leq$ LMAX(32).

Image Decomposition Module

The image decomposition module receives the discretized video image from the picture generator and suppresses the background intensities, forming binary pictures of the target and plume images. The IDM defines the target and plume parallelogram frames for characterizing the target, plume, and background intensities and decides for each pixel within the parallelogram frames whether the intensity belongs to the background, target or plume distributions.

Projection Computational Module

The projection computational module receives the filtered binary video image from the IDM on a point-by-point basis and computes the x and y projections for the target and plume images. When the frame is completed, the PCM computes the target and plume structural parameters which locate the target and plume images and define their structural characteristics.

Structural Tracking Module

The tracking module receives the structural parameters from the PCM on each frame and outputs the boresight correction signals to the control processor along with a confidence weight. The structural tracking module utilizes the sequential machine concept to characterize the different tracking situations and outputs an appropriate control strategy to the control processor. The tracking algorithm also controls the position and shape of the parallelogram frames for the IDM.

Control Processor Module

The control processor module receives the boresight correction signals and confidence weight from the structural tracker, predicts the location of the target on the next frame, and outputs the control signals to the tracker optics model. The confidence weight is used to combine the measured and predicted values, forming a history stack for the prediction processor. These simple prediction methods are currently used by the control processor. These are:

- two point linear predictor
- five point quadratic predictor

For each prediction method, the prediction errors on the last frame are used to form adaptive weights which are used to combine the linear and quadratic predictions into the combined estimate of the desired control signals. These control signals are then used by the tracker dynamics to compute the actual state of the tracking optics for the next frame, forming a complete automatic tracking system.

REFERENCES

[1] Flachs, G. M., et al., "A Real-Time Structural Tracking Algorithm", Proceedings of the 1976 National Aerospace and Electronics Conference (NAECON '76), Dayton, Ohio, May 1976.

[2] Black, R. J., Whitney, J. T. and Flachs, G. M., "A Pre-Prototype Real-Time Video Tracking System", Proceedings of NAECON '76, Dayton, Ohio, May 1976.

[3] Thompson, W. E. and Flachs, G. M., "A Structural and Dynamic Mathematical Model of a Real-Time Video Tracking System", Proceedings of the 1976 National Aerospace and Electronics Conference (NAECON '76), Dayton, Ohio, May 1976.

[4] Chang, S. K. and Chow, C. K., "Reconstruction of 3-D Objects from Two Orthogonal Projections and Its Application to Cardiac Cineangiography",

[5] U, Y. H. and Flachs, G. M., "Structural Feature Extraction by Projections", Proceedings of 1975 IEEE Region V Conference, Austin, Texas, April 1976.

[6] Gordon, R. and Herman, G., "3-D Reconstruction From Projections - A Review of Algorithms", Int. Review of Cytology, Vol. 38, 1974.

[7] Singer, R. A. and Behnke, K. W., "Real-Time Tracking Filter Evaluation and Selection for Tactical Applications", IEEE Transactions on Aerospace and Electronic Systems (TAES), January 1971.

[8] Spengarn K. and Weidemann, H. L., "Linear Regression Filtering and Prediction for Tracking Maneuvering Aircraft Targets", IEEE TAES, November 1972.

[9] Hampton, R. L. T. and Cook, J. R., "Unsupervised Tracking of Maneuvering Vehicles", IEEE TAES, May 1973.

[10] McAulay, R. J. and Denlinger, E., "A Decision-Directed Adaptive Tracker", IEEE TAES, March 1973.

[11] Fitts, J. M., "Aided Tracking as Applied to High Accuracy Pointing Systems", IEEE TAES, May 1973.

[12] Ludwig, D., and Lecours, M., "Synthesis and Analysis of Tracking Systems with Optimal Acquisition", IEEE TAES, May 1973.

[13] Thorp, J. S., "Optimal Tracking of Maneuvering Targets", IEES TAES, July 1973.

[14] Friedland, B., "Optimum Steady-State Position and Velocity Estimation Using Noisy Sampled Position Data", IEEE TAES, November 1973.

[15] Bershad, N. J., Merryman, P. and Sklansky, J., "Adaptive Trackers Based on Continuous Learning Theory", IEES TAES, March 1974.

[16] Pearson, J. B. and Stear, E. B., "Kalman Filter Applications in Airborne Radar Tracking", IEEE TAES, May 1974.

[17] Poirot, J. L. and McWilliams, G. V., "Applications of Linear Statistical Models to Radar Location Techniques", IEEE TAES, November 1974.

[18] Bhagavan, B. K. and Polge, R. J., "Performance of the g-h Filter for Tracking Maneuvering Targets", IEEE TAES, November 1974.

[19] Castella, F. R. and Dunnebacke, D. G., "Analytical Results for the X, Y Kalman Tracking Filter", IEEE TAES, November 1974.

[20] Moose, R. L., "An Adaptive State Estimation Solution to the Maneuvering Target Problem", IEEE Transactions on Automatic Control, June 1975.

[21] D'Appolito, J. A. and Roy, K. J., "Applications of Minimum Variance Reduced State Estimators", IEEE TAES, September 1975.

[22] Schooler, C. C., "An Optical α-β Tracking Filter for Systems with Model Inaccuracies", IEEE TAES, November 1975.

[23] Morgan, D. R., "A Target Trajectory Noise Model for Kalman Trackers", IEEE TAES, May 1976

[24] Alspach, D. L., "A Parallel Processing Solution to the Adaptive Kalman Filtering Problem with Vector Measurements", Computer and Electrical Engineering, Vol. 1, pp. 83-94, 1973.

[25] TR75 - 008, "Test Report of Instrument Data Converter and Contraves F Interface for G-22(sw-50)", Instrumentaion Directorate, Range Data Systems, White Sands Missile Range, New Mexico, August 1975.

[26] Engineering Memorandum #72-16, "Summing the Instrument Data Converter (IDC) Velocity Derived Signal to the Instrument Position Loop", Data Systems Branch, Range Modernization Division, White Sands Missile Range, New Mexico, 31 July 1972.

[27] Husson, S. S., "Microprogramming: Principles and Practices", Prentice-Hall, Inc., 1970, pp. 19-20.

[28] Arbib, M. A., "Theories of Abstract Automata", Prentice-Hall, Inc., 1969.

[29] Papoulis, A., "Probability, Random Variables and Stochastic Processes", McGraw-Hill, 1965.

[30] Sage, A. P. and Melsa, J. L., "Estimation Theory with Applications to Communication and Control", McGraw-Hill, 1971.

[31] Flachs, G. M., Thompson, W. E., Taylor, J. M. Black, R. J., Cannon, W., Rogers, R. and U, Y. H., "Mathematical Modeling and Simulation in a Programmed Design Method", submitted for presentation to the International Conference on Mathematical Modeling, Rolla, Missouri, 1977.

[32] Flach, G. M., Thompson, W. E., Taylor, J. M., Cannon, W., Rogers, R. and U, Y. H., "An Automatic Video Tracking System", submitted for presentation to the 1977 National Aerospace and Electronics Conference (NAECON '77).

[33] Arndt, R., "Automatic Missile Target Reading on the Automatic Film Reader", Optics Branch Technical Memorandum 72-2, White Sands Missile Range, September 1972.

[34] TR STEWS-ID-74-3, "A New Target Algorithm for the Automatic Programmable Film Reader", White Sands Missile Range, October 1974.

[35] Whitney, J., "A Pre-Prototype Real-Time Tracking Filter", New Mexico State University, Technical Report, April 29, 1976.

[36] Nilsson, N. J.,"Learning Systems", McGraw-Hill, New York, 1965.

[37] Tou, J. T. and Gonzalez, R. C., "Pattern Recognition Principles", Addison-Wesley, 1974.

APPENDIX A

A Software Design Tool for Microprogrammed Systems

TABLE OF CONTENTS

Abstract

This report describes the design philosophy, program capabilities, and program logic for a simulation program for clocked synchronous digital circuits.

This program is intended as a design aid for development of synchronous microprogrammed control units, although it is applicable to a much wider class of programs.

The input syntax is a vector-oriented non-algorithmic language which allows the use of some of the APL vector manipulation functions to describe actual hardware configurations on both the gate and functional levels without regard for chip pinning or fault detection. Several LSI functions are implemented as pre-defined components which may be used as "building blocks" to construct more complex circuits. Implemented functions include Boolean and two's complement vector operations and counters, latches, selectors, demultiplexors, and random-access memory LSI components. Facilities are provided for including subsystems from circuit libraries into the user's circuit.

Introduction

This program is capable of simulating a clocked (synchronous) digital circuit, and is intended to facilitate the design of microprogrammable control systems. It accepts as input a complete description of the digital circuit and produces as output the logic states of specified components and buses at intervals of one clock cycle. The simulation is logical rather than time-sliced and no attempt is made to detect races, logic "spikes", or fault location.

Since this is a completely self-contained simulation, the external stimuli for the circuit being simulated are generated by specifying a read-only memory (ROM) structure, the outputs of which are the stimuli, and whose address is incremented with each clock pulse.

The description of the circuit is written in a language that was designed for this application. Certain MSI and LSI logic functions, such as selectors, decoders, latches, counters, and randon-access memories, exist as predefined logic "building blocks" in the language. Combinational logic can be specified in a vector-oriented APL-like notation to construct other building blocks. This notation permits Boolean and arithmetic operations on mixed binary vectors and scalars and allows parenthesizing to any level.

The method of solution used in the program is a repetitive "relaxation" method in which the output of every combinational, asynchronous (i.e., non-clocked) element is computed based upon the current inputs to that element. This process is repeated until all outputs stabilize, that is, until no outputs change when the iteration is performed. When the system is stable, a clock pulse is generated for a part or for all of the synchronous (clocked) logic elements in the circuit. During the clock pulse, latches may be loaded, counters incremented, memory written into, etc., depending upon what the current outputs of the combinational logic are at that time.

Immediately prior to the clock, the program prints the current values of specified buses, or stops if a specified condition exists. After the clock, if no stop condition is met, the program again performs a successive relaxation upon the combinational logic.

In general, the relaxation method does not correctly model a non-clocked sequential circuit, since it does not take settling times and propagation delays into account. Since the language allows a restricted class of non-clocked sequential circuits to be specified, the experienced user can implement such circuits in this manner, although the practice is discouraged. Also, the program cannot detect whether the user's "combinational" logic equations are indeed purely combinational, so some care must always be taken to ensure that non-clocked sequential circuits are not accidentally created. It is better to transform all circuits to a Moore configuration, so that the predefined clocked logic components can be used whenever a sequential component is needed and the logic equations are purely combinational.

The simulation program consists of:

- a data base which contains the description of the circuit
- a translator which accepts card-image input files which contain the circuit description and store the information in the data base
- a routine which uses the current state of the circuit (contained in the data base) to compute the next state and contains the necessary code to implement each of the predefined

logic functions and an interpreter to "execute" the user's combinational logic equations

References to most predefined logic components are stored in table form, but logic equations are stored internally in text form and interpreted, rather than translated to machine code, to avoid the problems of machine-dependent code generation. Consequently, this is a semi-table-driven simulator/interpreter.

The first version of the simulator is written in IBM OS/360 FORTRAN G. It requires a byte-oriented facility and a word length of 32 bits. Minor modifications are required to allow the program to run on a 16-bit machine. Floating point arithmetic in not used.

This program is implemented in FORTRAN rather than APL for the following reasons:

- Program documentation is more easily incorporated into the program source file (via comment statements).
- FORTRAN is more widely used; this program may be more exportable.
- Better diagnostic messages are possible. For example, equations are parsed and interpreted by a FORTRAN routine rather than by the APL system's interpreter. The FORTRAN interpreter was coded to give explicit diagnostic error messages.
- The complexity of the circuit to be simulated is limited only by the available memory for the data base containing the circuit description, rather than to one APL workspace. The APL-PLUS file system is not suitable for this application.

- Output files from other programs (microprocessor assemblers, etc.) can be entered into this program via the OS/360 file concatenation or FORTRAN file sequencing facilities.
- Output files from other programs (microprocessor assemblers, etc.) and subsystems contained in circuit "libraries" can be entered into this program via the OS/360 file concatenation or FORTRAN file sequencing facilities.

The Input Card Deck

The user's deck may contain the following types of statements:

- statements which define buses
- statements which reference any one of a set of "predefined" logic components
- assignment-type statements (equations) used to specity the user's combinational logic and to connect buses in ways not provided by the predefined components
- DISPLAY statements which specify which buses are to be displayed (printed) immediately prior to the clock pulse
- STOP statements which specify the conditions for which the simulation is to stop
- END statement which signifies the end of the input stream.

The above statements may be in any order with the restriction that a bus must be defined before it is referenced by any statement. The program will accept as input any number of concatenated data-sets or datasets referenced by the OS/360 file sequencing facility. Program input is on FT05F001 and output is on FT06F001.

Only card columns one through 72 (inclusive) are processed. Columns 73 to 80 can contain anything. The text on a card must start in column one unless the card is a continuation of the preceeding card - in which case column one must be blank. Any of the above statement types can be continued on one or more continuation cards, although the total number of non-blank characters in columns one through 72 of the first card of a statement together with all of its continuation cards can not exceed 1600. There is o restriction on the actual number of continuation cards used.

An asterisk in column one indicates that the card is a comment card. Comment cards may be placed anywhere including immediately prior to a continuation card.

Spelling and punctuation must be adhered to <u>exactly</u>. The translation routines were designed to be very restrictive in this sense to simplify syntax analysis and error detection. However, all blanks are ignored, expcept as noted above to signify a continuation card, so blanks may be freely inserted to improve readability.

A typical deck setup might be as follows:

```
//                         (Job Control Language statements)
BUS,NAME= . . . .          (Bus definitions -- required)
.
.
.
SELECTOR,IN= . . . .       (Predefined components -- optional)
.
.
.
(logic equations)          (Combinational logic -- optimal)
.
.
.
```

```
DISPLAY,                     (DISPLAY statements -- optional)

.

.

STOP,                        (STOP statements -- optional)

.

.

.

END                          (END statement -- required)

/*                           ("end of data" marker)
```

Extensive error checking and detection code has been included in the program. All errors generate one or more descriptive messages which explain the exact cause of the error and, where applicable, a possible remedy for the condition.

Special Debug Features

The following two control cards are useful in debugging both the simulation program and the circuit to be simulated. Either or both of these cards may be inserted anywhere in the input stream prior to the END card.

The card

```
$B
```

produces a dump of the name and width of every bus that is defined up to this point in the input stream, as well as various indices and pointers associated with the bus.

The card

```
$C
```

produces a dump of every component that is defined up to this point in the input stream, and the name, subscript, and width of every bus associated with each component. Each of the user's logic equations is considered to be a component.

Columns three to 80 of the above two cards are ignored and may contain anything.

Defining a Bus

A bus is an asynchronous data or control-signal path of one or more bits. The number of bits in the bus is referred to as the width of the bus. A bus connects the output of a single component with the inputs of one or more components. Every bus in the system must have a bus definition card which contains the unique alphanumeric name of the bus, the width (in bits) of the bus, and, optionally, an octal value for the bus. The maximum bus width is 4096 bits.

Bus names may contain up to 16 characters. The first character must be a letter. The remaining characters may be letters or decimal digits.

A subscripted bus is indicated by a bus name followed by the value of the subscript (in decimal) enclosed in parentheses. A subscripted bus name is exactly equivalent to a single-bit bus and may be used wherever a single-bit bus is required. A bus subscript must be an unsigned decimal integer (without a decimal point) whose value is between zero and minus one, inclusive. No subscript computations are allowed.

If a value is specified, a bus is considered to be a hardwired constant. The state of such a bus never changes, the bus cannot be specified as an output bus of any component or be specified on the left-hand side of an assignment statement. This is the only way that literal values can be entered (except as the contents of a RAM or ROM). The user's logic equations (described later) cannot contain any explicit literals or constants except as bus subscripts or as part of a "take" or "drop" operator.

A bus is defined as

BUS,NAME=name,WIDTH=width

or

BUS, NAME=name,WIDTH=width,VALUE=value

for example, the card

BUS,NAME=JOE,WIDTH=6

defines a six-bit data or control bus named JOE. The card

BUS,NAME=WRITECONTROL,WIDTH=1

defines a single-bit data or control bus named WRITECONTROL.

The card

BUS,NAME=MINUSTWO,WIDTH=16,VALUE=177776

defines a 16-b9t constant bus named MINUSTWO whose value is the two's complement notation for a negative two.

A bus of width one is more conveniently referred to as a data or control line. A line may be defined as

LINE,NAME,=name

or

LINE,NAME=name,VALUE=value

For example, the card

LINE,NAME=WRITECONTROL

is equivalent to the previous definition of WRITECONTROL (above).

The card

LINE,NAME=ONE,VALUE=1

defines a single-bit constant bus named ONE whose value is one.

A bus may be referred to by its name or by its name and a subscript in parentheses following the name. The least-significant bit of an n-bit bus (say, B) is B(0); the most significant bit is B(n-1). Depending upon the usage of the bus, B(n-1) may be regarded as a sign bit for two's complement operations.

In general, when a bus is specified by name, all bits take place in the indicated operation. When a subscript is used, only the indicated bit takes place in the operation and the subscripted bus is exactly equivalent to a single-bit bus (a line). A subscripted bus name may be used wherever a line might be used.

When an initial value is specified for a bus, the octal digits are decoded starting at the right end of the octal number. Each bit is assigned to the corresponding bit of the bus starting with the least significant bit (B(0) above). Extra octal digits are ignored. If the octal number is not long enough to specify the value of the entire bus, the unspecified high-order bits default to zero.

Using Predefined Components

Several types of predefined components are available and may be included in the circuit. This section describes the operations performed by the predefined components and their specification syntax.

The order of parameters of the form "TYPE=" may be changed at will. Any parameter shown underlined below may be omitted entirely, along with the comma preceding it. Descriptive error messages are generated if the user violates any of the rules set forth below.

Note that all of the predefined components have data paths of arbitrary width. The actual width of the component is determined by the widths of the buses associated with that component.

The outputs of the synchronous predefined components are zero at the start of the simulation.

1. Selector (asynchronous)

SELECTOR,IN=A_1,A_2, . . . A_k,OUT=B, SEL=C,STB=D

defines an n-bit wide selector. If the value of the strobe bus D is one or the strobe bus in not specified, the output bus B is set to the value of the input bas $A_{1+(C)}$, where (C) denotes the value of the select bus C interpreted as an unsigned positive binary integer. If the strobe bus is specified and is zero, B is set to zero. The widths of the A_i and B must be the same. The width of the select bus C must be equal to $\log_2$ k, where k is the number of input buses specified. Note that k must be a power of two. Also, if the strobe is specified, its width must be one.

2. Demultiplexor (asynchronous)

DEMULTIPLEXOR,IN=A,OUT=B_1,B_2,. . . B_k,*SELECT=C,STB=D*

defines an n-bit wide demultiplexor (or "decoder"). If the value of the strobe bus D is one or the strobe bus is not specified, all of the output buses B_i are set to zero, except the bus $B_{1+(C)}$, which is set to the value of the input bus A. (C) denotes the value of the select bus C interpreted as an unsigned positive binary integer. If the strobe bus is specified and is zero, all of the B_i are set to zero. The widths of A and the B_i must be the same. The width of the select bus C must be equal to $\log_2$ k, where k is the number of output buses specified. Note that k must be a power of two. Also, if the strobe bus is specified, its width must be one.

3. Counter (synchronous)

COUNTER,IN=A,OUT=B,LOAD=C,CLR=D,INC=E,DEC=f,OVFL=G,UNFL=H

defines a counter which has synchronous load, clear increment, and decrement functions and overflow and underflow detection. If A is specified, the widths of A and B must be the same. The control buses C, D, E, F, G, and H must be of width one. All four of the lines C, D, E, and F are active high and determine the operation to be performed at the time that the next clock pulse occurs. If more than one control line is high, the first function in this list takes precedence:

1 -- CLR

2 -- LOAD

3 -- INC or DEC

If INC and DEC are both one, the value of the counter output bus B does not change.

If the clock pulse causes overflow or underflow to occur, the corresponding line (G, H above) is one until the next clear, load, increment, or decrement occurs (i.e., at least one clock cycle later).

Either INC or DEC must be specified. Both INC and DEC may be specified. If LOAD is specified, the IN must also be specified.

4. Latch (asynchronous)

LATCH,IN=A,OUT=B,LOAD=C,CLR=D

defines a latch which has synchronous load and clear functions. The widths of A and B must be the same. The buses C and D must be of width one. The control lines are active high and specify the action to be performed at the time that the next clock pulse occurs. If both the load and clear lines are one, the clear function takes precedence.

5. RAM (synchronous)

RAM,x WORDS OF y BITS,IN=A,OUT=B,ADR=C, WRT=D,VALUE=v_1,v_2,. . . v_k

where x and y are positive, unsigned decimal integers and A, B, C, and D are buses, defines a randon-access read/write memory with data input bus A, data output bus B, memory address bus C, and write-enable bus D. Data is read asynchronously from the memory address (specified by bus C) and placed on the output bus B. If the write-enable bus D is one at the time that the next clock pulse occurs, the data on bus A is stored at the memory address specified by C.

The widths of A and B must by y bits. The width of C must be the APL ceiling of $\log_2$ x. The bus D must be of width one.

The "VALUE=" parameter is optional. If used, it is followed by one or more octal numbers separated by commas which contain the initial contents of the RAM, starting at address zero. If more than x octal numbers are specified, only the first x numbers are used.

The octal numbers are decoded starting at the right end of the octal number. Each bit is assigned to the corresponding bit of the memory word, starting with the least significant bit. Extra octal digits are ignored. If an octal number is not long enough to specify the value of the entire memory word, the unspecified high-order bits of the memory word default to zero. If fewer than x octal numbers are present, the initial values of the unspecified RAM locations default to zero.

The number of words in the RAM need not be a power of two. If a nonexistent memory address is specified by the address bus C, the output bus B is set to zero, the write-enable line is ignored, and no error message is generated.

6. ROM (asynchronous)

ROM,x WORDS OF y BITS,OUT=A,ADR=B,VALUE=$v_1, v_2, \ldots v_k$

where x and y are positive, unsigned decimal integers and A and B are buses, defines a read-only memory with data output bus A and address bus B. Data is read asynchronously from the memory address (specified by bus B) and placed on the output bus A.

The width of A must be y bits. The width of B must be the APL ceiling of $\log_2 x$.

The "VALUE=" parameter is required. It must be followed by exactly x octal numbers separated by commas which contain the contents of the ROM starting at address zero.

The octal numbers are decoded starting at the right end of the octal number. Each bit is assigned to the corresponding bit of the memory word, starting with the least significant bit. Extra octal digits are ignored. If an octal number is not long enough to specify the value of the entire memory word, the unspecified high-order bits of the memory word default to zero.

The number of words in the ROM need not be a power of two. If a non-existent memory address is specified by the address bus B, the output bus A is set to zero and no error message is generated.

Logic Equations

The user's logic equations are used to implement functions that are not provided as predefined components and to set up data and control gating. The equations may contain any combination of the following operators:

=	assignment
&	Boolean AND
!	Boolean OR
!!	Boolean exclusive-OR
¬	Boolean complement
¬&	Boolean NAND
¬!	Boolean NOR
+	two's complement addition
-	two's complement subtraction or unary minus
nT	APL "take"
nD	APL "drop"
,	APL "catenate"

These operators are defined in terms of buses of arbitrary width. Any level of parenthesizing is allowed.

Statements are evaluated from right to left starting with the left-most parenthesized expression. The evaluation routine was designed after the APL right-to-left convention. The algorithm currently in use is equivalent to APL only if the statements are of the form

variable = expression

Consequently, multiple assignments in one statement are not allowed in the first version and the resulting evaluation is right-to-left.

Within the simulation program, each of the user's logic equations is treated as a "component" exactly like a latch, counter, etc. Thus, diagnostic messages referring to a component may be referring to an equation rather than a predefined component. In any case, diagnostic messages concerning any component will generate a dump of the component (similar to that produced by the $C card), so errors are easily pinpointed.

In general, two buses of different widths may be combined with a dyadic (two-operand) operator only if one of the operands is a bus of width one or a subscript bus, in which case that bit is extended (propagated) to form a bus of width equal to the width of the other bus*. If the bus widths differ and neither of the buses is either of width one or a subscripted bus, an error message is generated. These restrictions do not apply to the take, drop, or catenate operators.

Overflow and underflow indicators are not provided for the two's complement operators.

If the width of an evaluated expression differs from the width of the bus to which the value is to be assigned, an error message is generated.

The input translator ensures that every bus is the output bus of exactly one component (either a predefined component or an equation). Consequently, statements of the form

$A(0) = \text{expression}_1$

$A(1) = \text{expression}_2$

cause "multiple-difinition"-type errors and are therefore not allowed. This deficiency can be overcome by use of the catenate operator as

$A = \text{expression}_1, \text{expression}_2$

* Note that addition of a line of value one to a multi-bit bus has the effect of adding a <u>negative</u> one..

Description of Operators

In this section, A, B, and C denote bus names or subscripted bus names. Lower-case "n" denotes a decimal integer with an optional sign preceding it. If the sign if omitted, the number is assumed to be positive. The value of n must be between -4096 and 4096, inclusive. A decimal point cannot be used.

1)	=	(assingment)	A = B
2)	&	(AND)	A = B & C
3)	!	(OR)	A = B ! C
4)	!!	(exclusive-OR)	A = B !! C
5)	¬	(complement)	A = ¬ B
6)	¬&	(NAND)	A = B ¬ & C
7)	¬!	(NOR)	A = B ¬ ! C
8)	+	(addition)	A = B + C
9)	-	(subtraction)	A = B - C
10)	-	(unary minus)	A = -B
11)	nT	(take)	A = (n T) B
12)	nD	(drop)	A = (n D) B
13)	,	(catenate)	A = B , C

Note that the take and drop operations are denoted by "nT" or "nD" constructions enclosed in parentheses.

DISPLAY Statement

DISPLAY statements are optional statements which allow the user to print the values of buses immediately prior to the occurence of the clock pulse. The format of a DISPLAY statement is

DISPLAY,var_1,var_2,. . . var_k

where "var_i" is a bus name or a subscripted bus name. For example, a DISPLAY statement might look like

DISPLAY,A,B,C(12),D(0),E

The data is not necessarily printed in the order specified on the DISPLAY statement. The data is output as

- clock cycle number
- bus name
- bus subscript (if specified on the DISPLAY statement)
- bus value (in octal)

STOP Statement

STOP statements are optional statements which allow the user to stop the simulation when certain bits are one at the time immediately prior to a clock pulse. The format of a STOP statement is

STOP,var_1,var_2,. . . var_k

where "var_1" is the name of a bus or a subscripted bus name.

For example, a STOP statement might look like

STOP,A,B,(15),C(0),D

If a bus is specified, the stop condition is met wherever any bit of that bus in one at the time immediately prior to the clock pulse. If a subscripted bus name is specified, only that bit is tested. If any of the stop conditions on any STOP statement are met, the simulation halts after executing any DISPLAY statements that were specified. The values of the buses that are printed at that time are the same values that were tested by the STOP statements.

More complicated STOP conditions, for example, inequality of two lines, can be derived by defining an intermediate line, equating it to the "exclusive-OR" of the two lines, and then specifying the intermediate line in a STOP statement.

END Statements

The END statement is required and must be the last card in the user's input deck. The format of an END statement is

```
END
```

Any cards placed after an END statement are not processed.

If end-of-file is detected in the input stream prior to encountering an END statement, the program attempts to read from the OS/360 FORTRAN dataset having the next higher sequence number. Unpredictable results may occur if the next dataset in sequence is not specified by the user's Job Control Language (JCL) statements. Thus, the last concatenated dataset or the last FORTRAN dataset in the sequence must contain an END statement as its last card.

Program Table Summary

Bus Index Table (BIT) -- has exactly one entry for every bus in the circuit. The value of the entry for bus I, where I is greater than zero, is denoted by BIT(I) and is the DBT index of the BDB for bus I. Unused entries in the BIT are set to zero.

Component Index Table (CIT) -- has exactly one entry for every component in the circuit. The value of the entry for component J, where J is greater than zero, is denoted by CIT(J) and is the DBT index of the CDB for component J. Unused entries in the CIT are set to zero.

Stop Condition Table (SCTAB) -- contains the tokens of all buses that are specified as "stop conditions" on a STOP card. Unused entries in the SCTAB are set to zero.

Print Table (PRTAB) -- contains the tokens of all buses that are specified on a DISPLAY card. Unused entries in the PRTAB are set to zero.

Data Block TAble (DBT) -- large array which contains all DBDs and CDBs.

Data is entered into the above tables sequentially starting at index one. The first four tables above must have at least one unused (zero) entry to mark the current extent of the table.

A "token", as referred to above, is a fullword (32 bits) containing a BIT index and subscript value of a bus. The subscript value is -1 if no subscript was specified.

Job Control Statements

The Job Control Language (JCL) statements below are from a typical simulation run on OS/260 at NMSU. Note the use of the FORTRAN file sequencing facility to include a circuit subsystem (CLOCK) from the circuit library CKTLIB, followed by the user's card deck containing the description of the remainder of the circuit. The last card in the dataset referred to by the last DD statement (FT05F002 in this case) must be an END card.

```
//**   (charge card)
//**224K,2M,2000L,OC
//STEP1 EXEC PGM=DIGSYM
//STEPLIB DD DSNAME=DIGSYMTP,DISP=SHR
```

```
//FT06F001 DD SYSOUT=A
//FT45F001 DD UNIT=SYSDA,SPACE=(TRK,(0,1)),DCB=(LRECL=10)
//FT05F001 DD DSNAME=CKTLIB(CLOCK),DISP=SHR
//FT05F002 DD *
.
.    (user's deck)
.
.
END
/*
```

APPENDIX B

Technical Papers Presented

A PRE-PROTOTYPE REAL-TIME VIDEO TRACKING SYSTEM

R.J. Black, J.T. Whitney and G.M. Flachs
New Mexico State University
and
Alton L. Gilbert, U.S. Army, White Sands Missile Range

ABSTRACT

This paper describes a new approach to real-time optical tracking of high performance targets in noisy, multiple target environments. Previous attempts to develop a real-time optical tracking system have utilized contrast edge detectors and image correlators. Edge detectors are easily confused in noisy backgrounds and image correlators have not demonstrated the speed and accuracy required to evaluate multiple high performance targets.

A TV camera is used for the sensor and the video is digitized by a high speed analog to digital converter. A hardware picture decomposition module separates the targets from the background and stores a reduced binary picture into a high speed buffer memory. The contents of the buffer memory are processed by a structural image processor to identify and locate target images. Multiple targets within the target window are identified and located using structural projections. When the target is located control signals are generated to orient the camera towards the desired target. By utilizing an array of the trackers, each tracking one or more spatially close targets, the tracking system can handle a large class of multiple target tracking situations.

1. INTRODUCTION

There is much interest among tracking people in the development of an automatic real-time video (RTV) tracking system capable of tracking multiple targets in a noisy and low contrast background. Previous attempts at automatic optical tracking utilized contrast edge trackers or correlation detectors. Edge trackers are easily confused in a noisy background environment and correlation trackers have not demonstrated the accuracy and speed required for tracking the orientation and position of spatially separated multiple targets in a real-time environment. Optical tracking systems, consequently, still utilize high frame rate telescopic cameras that are manually aimed at the moving targets. The pointing angle of the camera's center of the frame is automatically recorded on each frame. The film is then processed and the target is manually located relative to the center of frame and the camera pointing angles are manually read off the film. Triangularization is then used to establish the spatial trajectory of the target using two or more cameras located at precisely known positions. Although much time and money is required to process and read the film, the data is of high quality and the system is flexible and it can satisfy a large class of multiple target tracking requirements.

Much work has been done at WSMR in developing target tracking algorithms for an Automatic Programmable Film Reader (APFR) to automatically locate targets and read the pointing angles off the film. The results of these efforts for automatically reading the film have led to the development of the RTV tracking system.

Initial efforts toward the development and evaluation of a pre-prototype RTV tracking system is currently being funded by White Sands Missile Range. The purpose of the pre-prototype system is to demonstrate that the picture decomposition and structural recognition algorithms developed on the APFR can be successfully implemented in hardware at frame rates in excess of 30 frames per second. The pre-prototype configuration (Figure 1) consists of a TV camera and video processor, picture decomposition module, and an image processor. The camera and video processor, acting as an eye in the RTV tracking system, provides a digitized picture image to the picture decomposition algorithm. The picture decomposition module separates the target densities from the background densities, reducing the picture to a binary picture which is stored in a high speed buffer memory. The image processor utilizes parallel processing hardware to analyze the picture in real-time to locate and structurally recognize targets of interest.

Although each RTV tracker is capable of tracking multiple targets within its view window, the real potential of tracking spatially separated multiple targets will be realized by an array of these trackers. One tracker can be utilized to search for targets over a large scene and assign local trackers to structurally identify and track targets of interests.

This work was supported by White Sands Missile Range under Contract No. DAAD07-76-C-0024.

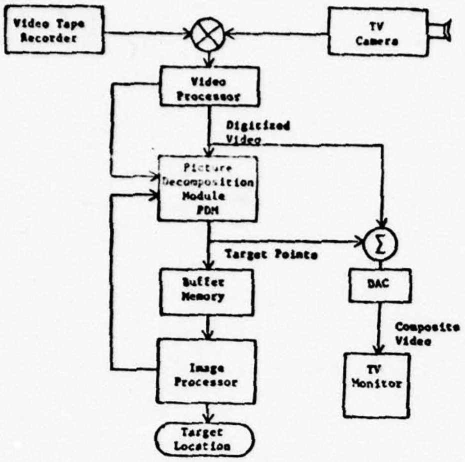


Figure 1. Pre-Prototype RTV Tracker

2. CAMERA AND VIDEO PROCESSOR

For the pre-prototype system a standard TV camera and a video tape recorder are being used to provide composite video signals of target images. This camera will eventually be replaced by a silicon diode array solid state camera that will be interfaced into the optics chain of the cinetheodolite telescope mounts used by the fixed cameras. The cinetheodolite mounts already have a position control system for pointing the camera which can be driven by the RTV tracking system. Radar data will also be used to point the camera in the vicinity of the target. This can be used to initially acquire track and to reestablish track after a target passes through a cloud.

The TV camera provides a composite video signal of the form shown in Figure 2.

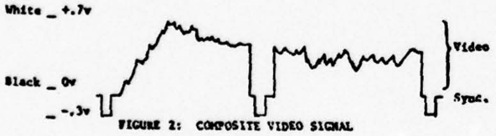


FIGURE 2: COMPOSITE VIDEO SIGNAL

The ADC provides a digitized representation of the composite video. Sixteen levels are initially being used to verify the potential of the tracking system. Additional levels and automatic gain control can be added to handle low contrast images. High speed voltage comparators (Signetics 521) and a ladder resistor network are used to obtain a high speed analog to digital convertor as shown in Figure 3. The input signal is compared to the voltage developed at each level of the resistor ladder network. The output of the i^{th} level will be a logical 1 for

$$\mathrm{Vin} \geq \frac{2i-1}{2N}\ \mathrm{VREF}$$

This will yield a N level conversion with the level detection occuring at the midpoint between the $(i-1)^{th}$ level and the i^{th} level. The sync signal is generated at output S. Resistor R_0 forms a termination impedance for the transmission line from the video source.

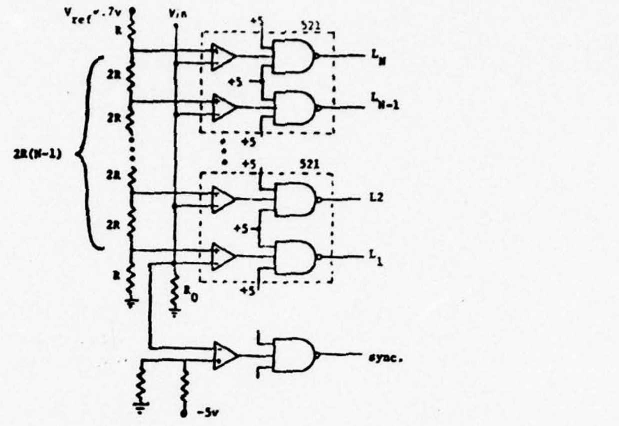


Figure 3. N Level Parallel ADC and Sync Detector

The composite video analog signal is reconstructed from the digitized representation for display on a TV monitor using a simple resistor summing circuit (Figure 4). The process of converting from analog to digital and back again to analog introduces less than fifty nanoseconds delay. For sixteen levels the digitization error is approximately 1/10 of the least significant bit.

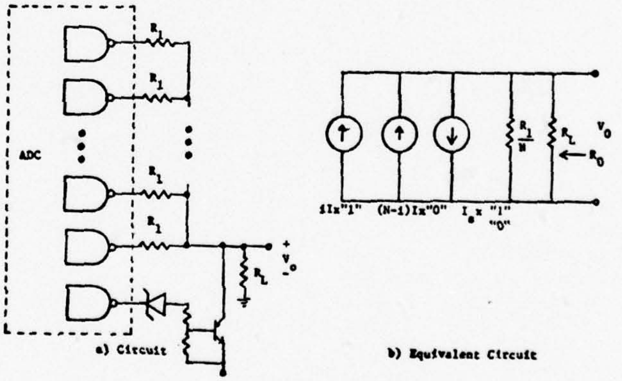


Figure 4. N Level DAC and Sync Insertion

Since the digitized signal is not converted to a binary number, the reconstruction is straight forward. The number of logical 1's is directly proportional to the original signal amplitude, allowing a resistor summing junction and a transistor to completely reconstruct the video for a monitor.

3. PICTURE DECOMPOSITION MODULE

The purpose of the picture decomposition module is to separate the target image from the background. Using the property that the target image has some intensities that are different from the immediate background, a parallelogram frame is placed about the predicted location of the target such that the target is included in the interior of the parallelogram frame as shown in Figure 5.

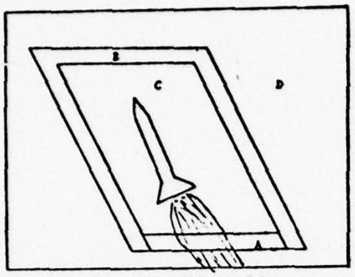
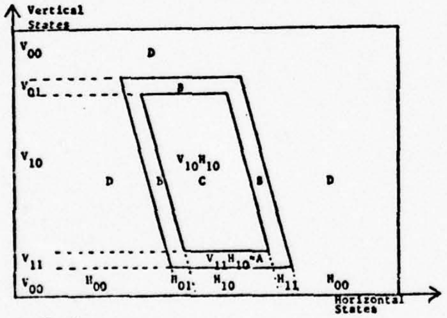


Figure 5. Parallelogram Frame

The parallelogram frame, consisting of the region between the two parallelograms, is partitioned into two regions A and B. Region B is utilized to provide a characterization of the immediate background densities, while region A is utilized to characterize the plume intensities when a plume is visible. The regions interior and exterior to the parallelogram frame are denoted by C and D respectively.

A digital state controller has been designed to separate the picture into the four regions A, B, C, and D. The controller has the ability to control the size, position, and angle of the parallelogram within the picture frame under program control. The controller, Figure 6, consists of four states for the horizontal scans and four states for the vertical scans that define and control the four regions in the picture frame.



$$A = V_{11}\cdot H_{10}$$

$$B = V_{01}\cdot\overline{H_{00}} + (H_{01} + H_{11})\cdot\overline{V_{00}}$$

$$C = V_{10}\,H_{10}$$

$$D = \overline{A + B + C}$$

Figure 6. State Regions

The vertical control is reset to state V_{00} with the vertical sync pulse which also loads a counter with the distance to the top of the parallelogram. The counter is decremented for each horizontal scan until it underflows and defines the next vertical state V_{01}. At this point another counter is activated to define the top border of the parallelogram. When the top border state V_{01} is finished, state V_{10} is activated which in turn activates V_{11}. When state V_{11} is finished, all counters are disabled until a vertical sync pulse occurs to start the process over. The horizontal control is essentially the same as the vertical control except that the initial value of state H_{01} can be incremented or decremented by a variable number of horizontal steps to define the slope of the parallelogram. The horizontal sync pulse resets the horizontal state controller. By utilizing counters to realize regions A, B, C, D in the picture frame, the position, size, and slope of the parallelogram frame can be accurately controlled by the image processor.

Using the state controller and two banks of counters, one for the background and the other for plume intensities, the intensities of the region A and the intensities of region B are established. The state controller enables the background counters in region B to establish a sampled statistical characterization $D_i(B)$ of the background intensities for the i^{th} frame. The plume counters are enabled by the state controller in region A to obtain a sampled statistical characterization $D_i(A)$ of the intensities in region A. Within the interior of the parallelogram frame, the picture decomposition module outputs a string of 0, 1, and 2 symbols that respectively represent background, target, and plume points. For each point in the interior parallelogram, a decision is made as to whether the intensity came from a distribution characterized by $D_i(B)+D_{i-1}(B)$, $D_i(A)\cdot\overline{D_i(B)+D_{i-1}(B)}$, or neither of these distributions. If it is decided that the sampled intensity came from $D_i(B)+D_{i-1}(B)$ the point is classified as a background point, if the intensity came from $D_i(A)\cdot\overline{D_i(B)+D_i(B)}$ the point is classified as a plume point, and if it is decided that the intensity came from neither of the above distribution, then the point is classified as a target point. Since the picture decomposition algorithm is implemented with an array of counters, the data rate can be easily increased from the standard 30 frames per second.

The counter array is composed of three banks of counters: one for the present frame background $D_i(B)$, one for the previous frame background $D_{i-1}(B)$, and one for the plume window $D_i(A)$. Each bank of counters is composed of a counter for each gray level. Each gray level counter is initially loaded with a noise rejection threshold T_{NR} and is decremented when its gray level occurs in the region defined by the state controller. The threshold T_{NR} is under program control and it is adjusted to provide immunity to random noise in high contrast regions of the picture. When the counter underflows, the intensity has occurred a sufficient number of times to decide that the given intensity is part of the region being characterized by the counter bank. Figure 7 shows the i^{th} gray level counter in each counter bank and it defines the logic that separates the picture into the target, plume, and background components within region C of the parallelogram. The outputs of all the counters in each bank are wired-ORed together onto three output buses. When the state controller is in region C, the three buses define whether the measured intensity occurred in the background region B of the current frame $(\overline{BC})$, the background region of the previous frame $(\overline{BP})$ or whether it occurred in the plume window A of the current frame $(\overline{BA})$. The logical expressions that defines the target, plume, or background points are:

Target Point = $\overline{BP}\cdot\overline{BC}\cdot\overline{BA}$

Plume Point = $\overline{BP}\cdot\overline{BC}\cdot\overline{BA}$

Background = $\overline{\text{Target V Plume}}$.

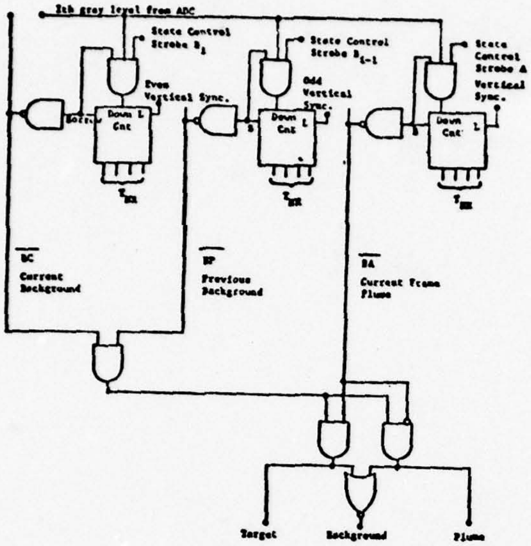


Figure 7. Intensity Distribution Counters

The output string from the picture decomposition module is stored in a high speed buffer memory (Figure 8). As the data arrives serially for each horizontal scan from the picture decomposition module, it is shifted into two eight bit shift registers; one for the target buffer and the others for the plume buffer. When the shift registers are loaded, the data is stored in the buffer memory so that the images over 4x8 windows are directly word addressable by the image processor.

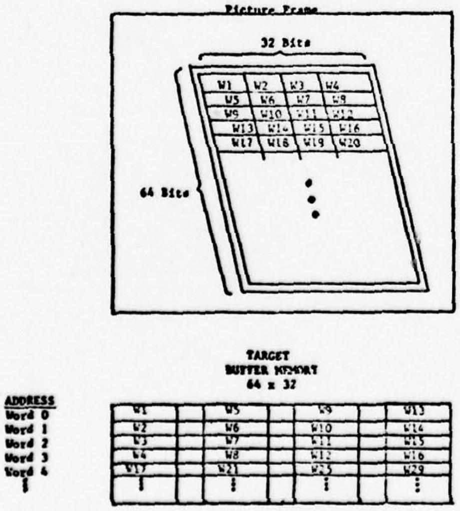


Figure 8. Buffer Memory Configuration

The target buffer memory consists of four 64x8 bipolar memories cacaded together to form a 64x32 high speed (50 n.s.) memory. When a word is accessed by the image processor, the image for two consecutive 4x4 windows becomes available. This look-ahead feature is utilized by the tracking algorithm to further reduce the bandwidth as shown in Table 1. The plume buffer memory is constructed in a similar manner and operates in parallel with the target buffer memory.

The following table shows the approximate bandwidths at various places in the system.

TV Video	5 Mhz.
Output of the ADC	195 ns/pt.
Into the Buffer Memory	1.56 μs/Storage
Out of the Buffer Memory	256 μs/Scan Window

Table 1. Bandwidth Considerations

4. IMAGE PROCESSOR

The image processor analyzes the contents of the 4x4 scan windows stored in buffer memory as they are filled by the picture decomposition module. A HP2116 minicomputer is currently being used as a programmable controller to access the scan windows in buffer memory to structurally locate and recognize targets. A bipolar microcomputer system is being designed to replace the HP2116, reducing the size and cost of the tracker. The contents of the scan windows are analyzed in parallel by scan window logic, producing discrete control signals that define the contents of the scan window and enable the computer's proper response. This eliminates the time consuming bit processing of the image. There are six distinct outputs from the scan window logic. These are:

Null window	□
Full window	▨
Left window	◧
Right window	◨
Upper window	⬒
Lower window.	⬓

The scan window logic is implemented with combinational logic and threshold gates.

When the computer accesses a scan window in buffer memory, the contents of both the target and plume buffer memories are latched for the scan window logic. The outputs of the scan window logic then define the computers response. Consequently, little time is spent processing the scan window image. By sequentially processing the view windows, the computer utilizes a fast structural location and recognition algorithm to locate and track targets in the view of the camera. A detailed description of the tracking algorithm is given in another paper "A Real-Time Tracking Algorithm" presented at this conference.

A simplified block diagram of the pre-prototype RTV tracking system is given in Figure 9. The minicomputer is interfaced to the picture decomposition module and buffer memories by two microcircuit interface cards. Each microcircuit interface card provides bi-directional transfer of sixteen bits of data along with status and control signals. The output bus of the first interface defines the address of the scan window to the buffer memory and its input bus receives the results from the scan window logic. The second interface sends signals to the picture decomposition module to control the position, size, and orientation of the parallelogram. It receives data corresponding to the statistical properties of the background, target, and plume intensities.

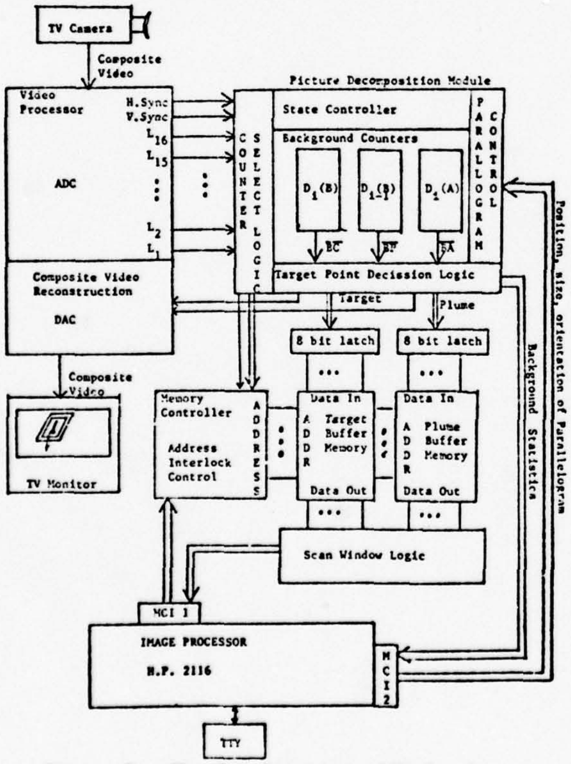


Figure 9. The Pre-Prototype RTV Tracker

REFERENCES

[1] Flachs, G. M. and Gilbert, A. L. "Automatic Boresight Correction in Optical Tracking". Proceedings of the IEEE 1975 National Aerospace and Electronics Conference.

[2] Flachs, G. M. "Real-Time Tracking Algorithm", Final Report for grant DAHCOA-15-G-0038, ARO, Oct., 1975.

[3] Flachs, G. M. "A New Target Tracking Algorithm for the Automatic Programmable Film Reader", WSMR Technical Report, STEWS-ID-74-3. Oct., 1974.

[4] Optical Sensor and Tracker (OSAT) Technical Report submitted to WSMR by DBA Systems, Inc., June, 1971.

[5] Bradley Lester "Use of two Electro-Optical Sensors Driven By Computer Generated Scan Patterns To Form Real-Time Two Station Tracking System", Proceedings of the IEEE 1975 National Aerospace and Electronics Conference.

A REAL-TIME STRUCTURAL TRACKING ALGORITHM

Gerald M. Flachs, Wiley E. Thompson, Yee Hsun U
Dept. of Electrical & Computer Engineering
New Mexico State University
Las Cruces, New Mexico 88003

and

Alton L. Gilbert
Instrumentation Directorate
US Army White Sands Missile Range
White Sands Missile Range, New Mexico 88002

ABSTRACT

This paper presents a real-time tracking algorithm for automatic optical tracking systems. A view window is positioned about the target from the predicted target data. Using a statistical target filter, picture decomposition of this micro-environment is performed to separate the target (and possibly other patterns in the neighborhood) from the background. Projections of these objects are generated and the target of interest is structurally identified and its location and orientation precisely determined. The algorithm can be extended to track a large class of objects in a multiple target environment.

INTRODUCTION

A new target tracking algorithm is presented for real-time optical tracking systems. Optical tracking systems still utilize high frame rate telescoptic fixed cameras that are manually aimed at the target. Optical encoders record the pointing angle of the camera directly on the film. The film is processed and the target position and orientation is manually read off the film along with the camera's pointing angle. The system is versatile and it yields high quality data with the advantage that it can be used in multiple target tracking situations. The film processing and reading effort, however, is time consuming and expensive. Consequently, there is much interest in developing a real-time optical tracking system that retains the accuracy and versatility of optical tracking systems. The hardware technology for such a system is currently feasible; however, present tracking algorithms have not demonstrated the required tracking reliability and accuracy. Previous attempts to implement an automatic optical tracking system have utilized contrast edge trackers or correlation trackers. Edge trackers are easily confused by noisy backgrounds such as the interface between the mountain and sky backgrounds. Correlation trackers have demonstrated the ability to track targets in a noisy background; however, they generally utilize a small processing window about the target and track the target using a coarse frame-to-frame correlation algorithm. Consequently, only coarse structural properties of the target are utilized and the accuracy of position and orientation measurements do not compare to the standard optical tracking system.

Major progress in the development of a real-time structural tracking algorithm has been achieved by using a statistical picture decomposition method to separate the target image from the background and convert the image to a binary picture, and by implementing mathematical projections to structurally identify and locate targets of interest. By combining image processing and pattern recognition techniques with fast parallel processing hardware, the algorithm performs statistical background analysis, picture decomposition, and target location and structural recognition in real-time.

The algorithm has been implemented into a pre-prototype tracking system at the Image Processing Lab of the Department of Electrical and Computer Engineering, NMSU, utilizing equipment and facilities available from the department and WSMR. The hardware implementation is discussed in another paper, "A Pre-prototype Real-Time Video Tracking System," presented at this conference.

THE TARGET TRACKING ALGORITHM

The target tracking algorithm is responsible for locating the target image, finding its position and orientation, and predicting the location of the same target on subsequent frames, all within a fixed time limit as determined by the frame rate of the TV camera. The problem of locating the target in the TV window is complicated by the fact that the background is not uniform. Typically, it changes initially from desert, then to a mountain, and finally to a sky scene during the flight of the target. It is the interfaces between these background scenes that have traditionally caused many problems for tracking systems. The diversity of target shapes such as missiles, helicopters, airplanes, and balloons, as well as the ability of one target to exhibit various shapes at different view angles, further complicate the problem and requires the algorithm to adapt to the changing target structure and distinguish the target structure from other patterns in the background.

The Real-Time Video (RTV) Target Tracking Algorithm can be divided into the following main modules:

.Picture Decomposition
.Micro Image Processor
.Structural Recognition and Location

The picture decomposition algorithm separates the target images from the background and generates a binary image of the picture. It is assumed that the TV window has been digitized into an NxN picture array with L gray levels. The picture function F; $Z^2 \rightarrow Z^1$, Z^i denotes an i-dimensional integer space, maps a given point denoted by the tuple (i,j) into the gray level values assigned by

$$F(i,j) = \ell;\ 1 \leq \ell \leq L \text{ and } 1 \leq i,\ j \leq N.$$

The micro image processor solves the multiple target problem by using a buffer memory to scan, identify, and locate each distinct target. Targets are structurally identified and located by using the theory of projections [3]. A projection in the x-y plane of a function f(x,y) along a certain direction w onto a straight line z perpendicular to w is defined to be

$$P_w(z) = \int f(x,y)\ dw$$

as shown in Figure 1. In general, a projection integrates the gray levels of a picture along parallel lines through the pattern generating a function called the projection. For binary digitized patterns, the projection gives the number of object points along parallel lines; hence, it is a distribution of the target points for a given view angle.

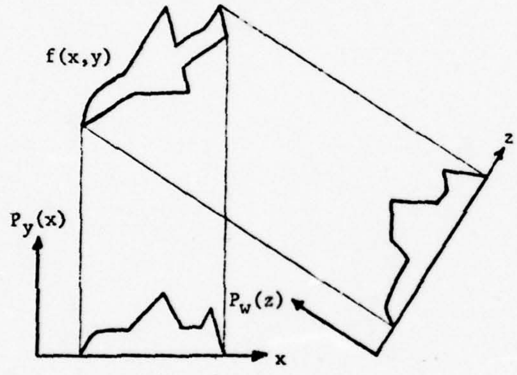


Figure 1: Projections

PICTURE DECOMPOSITION

For any pattern recognition system, some form of raw data reduction is usually necessary. This is essential for real-time application so that target detection and identification can be performed within the specified time limit.

The picture decomposition module separates the picture into three parts:

- . Background
- . Target
- . Plume

Each of the three parts defines one or more regions within the target window and hence is a set of grid points. Denote the background, target, and plume sets by B, T, and P, respectively. No convexity and connectivity constraints are assumed on these sets. If T or P are partitioned into non-connected subsets, then they can be written as

$$T = \{T_{a_1},\ T_{a_2}, \ldots, T_{a_m}\}$$

$$P = \{P_{b_1},\ P_{b_2}, \ldots, P_{b_n}\}$$

where

$$T_{a_i} \cap T_{a_j} = 0 \text{ for } i \neq j$$

$$P_{b_i} \cap P_{b_j} = 0 \text{ for } i \neq j$$

This partition occurs when there are several targets and plumes within the target window. Since a target may not have a plume and the existence of a plume does not guarantee the existence of a visible target, no relation is defined between subscripts a_i and b_i.

Two parallelograms are centered around and oriented in the direction of the target as shown in Figure 2. The interior parallelogram is defined so as to contain the target of interest. The region between the two parallelograms forms the target filter and it is partitioned into two subregions, R and S. Region R is utilized to provide a characterization of the background intensities in the target neighborhood, while region S is utilized to provide a characterization of the plume intensities. The background, target, and plume sets within the target window are defined by

$$S = \{(i,j) \mid F(i,j) \varepsilon V_R\}$$

$$T = \{(i,j) \mid F(i,j) \varepsilon \overline{V}_R \cap \overline{V}_S\}$$

$$P = \{(i,j) \mid F(i,j) \varepsilon \overline{V}_R \cap \overline{V}_S\}$$

where

$$V_R = \{\ell \mid \text{card}\{(p,q) \mid F(p,q) = \ell\ \forall (p,q) \varepsilon R\} > T_{NR}\}$$

$$V_S = \{\ell \mid \text{card}\{(p,q) \mid F(p,q) = \ell\ \forall (p,q) \varepsilon S\} > T_{NR}\}$$

$$T_{NR} = \text{noise reject threshold}$$

and provides useful information to locate targets within the target window.

Picture decomposition using the target filter is of vital importance to the RTV Tracking System. It significantly reduces the amount of raw data by extracting the target of interest from a noisy background scene, transforming the picture from a multi-gray level pattern into a binary one. This allows the other modules of the RTV Tracking System to process the binary image using fast, simple algebraic techniques that can often be implemented in hardware and/or firmware at a reasonable cost and complexity.

MICRO IMAGE PROCESSOR (MIP)

The Micro Image Processor processes the data in the target and plume buffer memories to locate objects within the target window. For each object found, three projections (the two 45° diagonal projections and the x-projection) are generated during the MIP scanning process. Instead of accessing the binary image point-by-point serially, the MIP considers the target or plume pattern as an array of 4x4 scan windows as shown in Figure 3. By addressing a scan window, the MIP accesses simul-

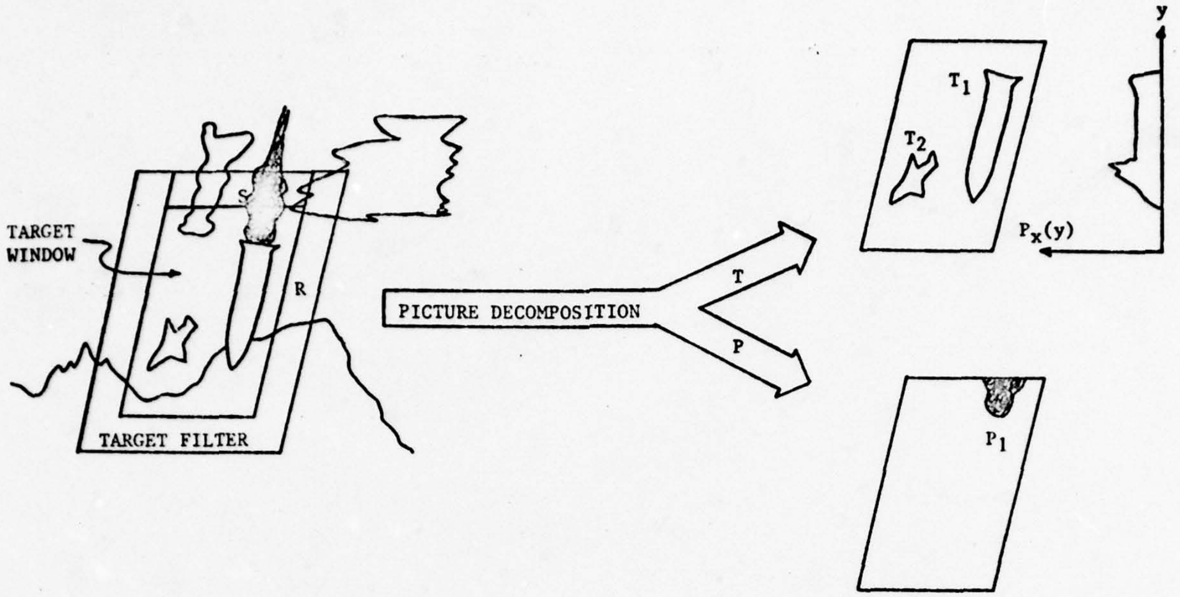


Figure 2: Picture Decomposition

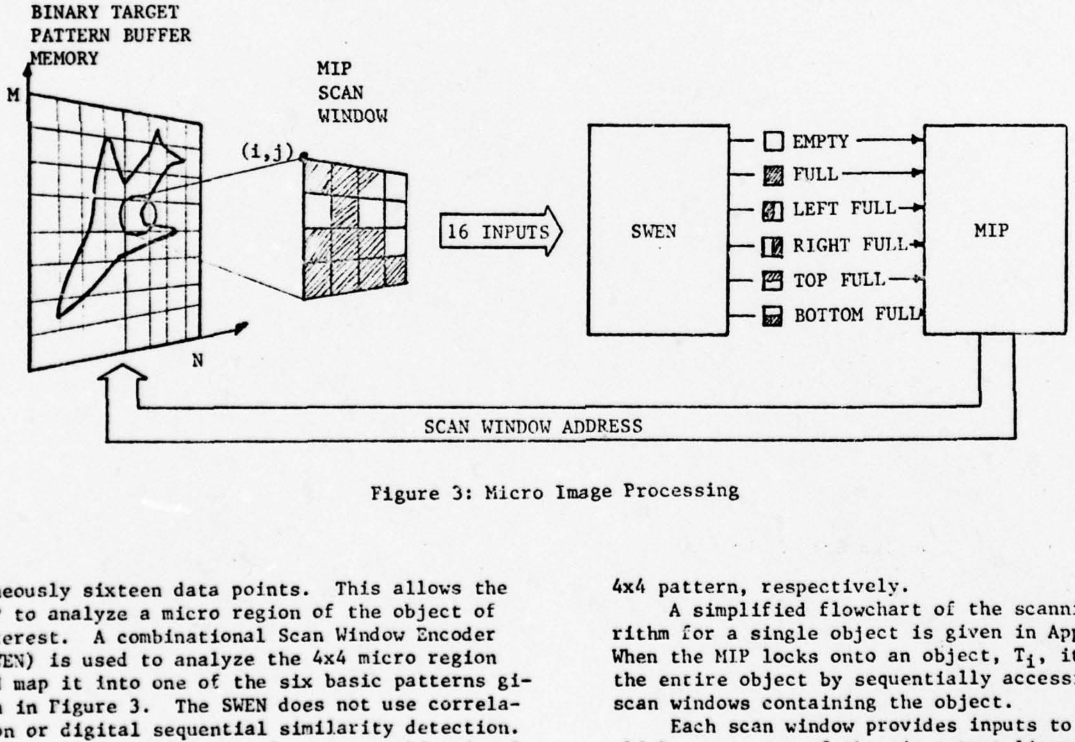
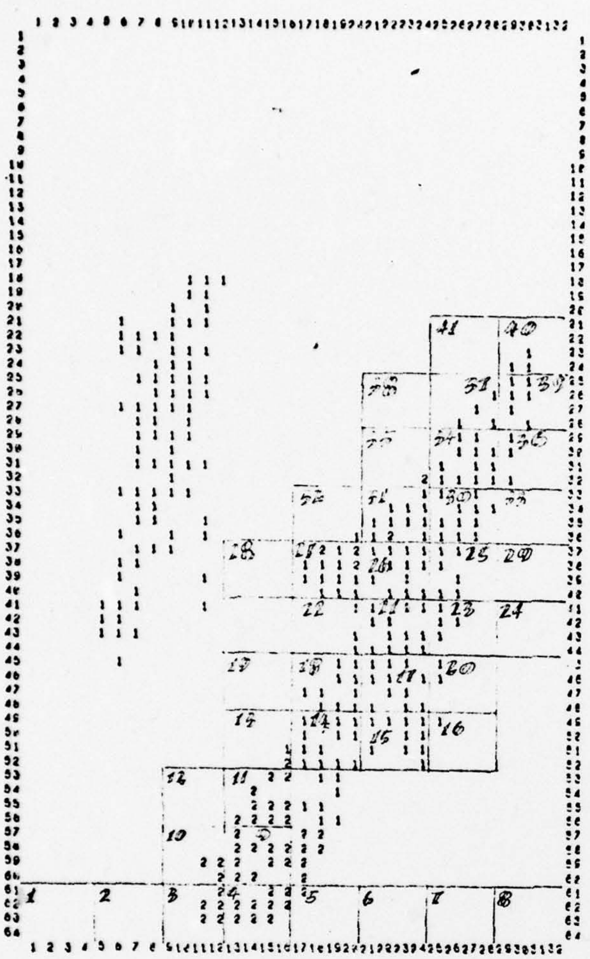


Figure 3: Micro Image Processing

taneously sixteen data points. This allows the MIP to analyze a micro region of the object of interest. A combinational Scan Window Encoder (SWEN) is used to analyze the 4x4 micro region and map it into one of the six basic patterns given in Figure 3. The SWEN does not use correlation or digital sequential similarity detection. Rather, it is a 16-input, 6-output combinational pattern classifier over a 4-dimensional pattern space with feature vector $X = (x_1, x_2, x_3, x_4)$ where $x_1, \ldots, x_4$ are the number of object points within the right, left, top, and bottom halves of the 4x4 pattern, respectively.

A simplified flowchart of the scanning algorithm for a single object is given in Appendix A. When the MIP locks onto an object, T_i, it scans the entire object by sequentially accessing the scan windows containing the object.

Each scan window provides inputs to the SWEN which causes one of the six output lines to turn on. This information allows the MIP to increment the projections and to determine either the address of the next scan window to be accessed or establish if the object T_i has been completely



N=32 M=64

1-target point 2-plume point

Sequence of scanning is given by handprinted numbers at top left corner of each scan window

Figure 4: The Scanning Process

scanned. One important feature of the scanning algorithm is that, given the address of a scan window within T_i, it will automatically scan over the entire object and generate its three projections without stepping over to a neighboring object set T_j within the target window. A sufficient condition for the scanning algorithm to be valid for a nonconvex set T_i is

$$T_j \not\subset \text{convh}(T_i) \quad \text{for } i \neq j$$

where convh(T_i) is the convex hull of T_i[4]. For most target tracking applications, this limitation is acceptable. The target window can be considered as an array of 4x4 scan windows as shown in Figure 3.

The MIP algorithm (Figure 5) begins by locating all plumes that intersect the S region of the target filter. By locating the tip of each plume, targets above the plume tip are sequentially scanned and their respective projections generated and the targets are structurally identified and located. The total number of target points C_T provides information as to whether all objects within the target window have been detected. If there exists an object of interest still undetected, the difference between the target points distribution and the sum of all x-projections of the targets located

$$P_x^T(y) - \sum_{i=1}^{I} P_x^{T_i}(y)$$

provides valuable information as to where new targets can be found. For tracking targets without plumes, the MIP algorithm starts at entry point B and uses the $P_x^T(y)$ information right away to locate a target. Even for targets having a plume, this approach is also valid; however, plumes usually have simple convex shape and are much easier to locate than targets.

STRUCTURAL RECOGNITION AND LOCATION

Tracking algorithms that do not utilize structural information to identify and distinguish the desired target from the background objects are easily confused in a noisy background or multiple target situations such as that caused by missile staging. To be able to identify a variety of target structures such as airplanes, missiles, helicopters, and balloons, and distinguish them from background patterns, it is necessary to utilize a structural tracking algorithm.

It has been shown that for sufficient large numbers of projections, a multi-gray level digitized pattern can be uniquely reconstructed [5]. This means that structural features of a pattern are contained in the projection. The picture decomposition algorithm described earlier reduces a multi-gray level picture to a binary pattern where 0 is used for background and a 1 is used for a target point. This simplifies the construction of projections and eliminates interference of structural information by intensity variation within the target pattern, and consequently, fewer projections are required to extract the structural information. The MIP algorithm further isolates each target within the target window and generates three projections for structural analysis. Although any convex, symmetric pattern can be reconstructed by only two orthogonal projections [7], three projections are used to obtain better structural characterization of nonconvex targets and they provide a more accurate measurement of target extreme points.

It is convenient to transform the projection functions into a parametric model for structural analysis. Area quantization offers the advantage of easy implementation on a computer. This process transforms a projection into k rectangles of equal area (Figure 6), where

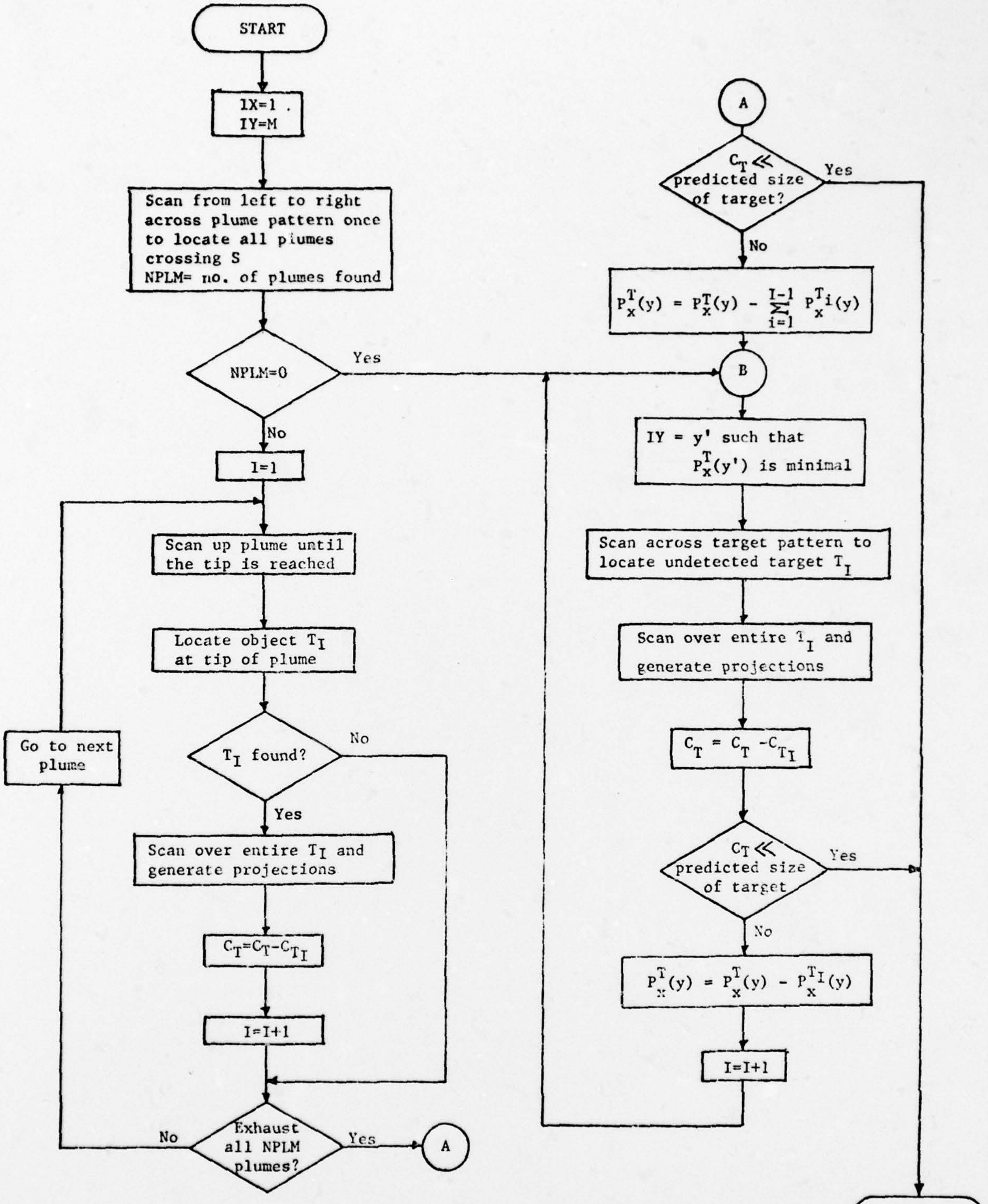

START
IX=1 .
IY=M
Scan from left to right across plume pattern once to locate all plumes crossing S
NPLM= no. of plumes found
NPLM=0
Yes
No
I=1
Scan up plume until the tip is reached
Locate object T_I at tip of plume
T_I found?
No
Yes
Scan over entire T_I and generate projections
$C_T=C_T-C_{T_I}$
I=I+1
Go to next plume
Exhaust all NPLM plumes?
No
Yes
A
A
$C_T \ll$ predicted size of target?
Yes
No
$P_x^T(y) = P_x^T(y) - \sum_{i=1}^{I-1} P_x^{T_i}(y)$
B
IY = y' such that $P_x^T(y')$ is minimal
Scan across target pattern to locate undetected target T_I
Scan over entire T_I and generate projections
$C_T = C_T - C_{T_I}$
$C_T \ll$ predicted size of target
Yes
No
$P_x^T(y) = P_x^T(y) - P_x^{T_I}(y)$
I=I+1
END

Figure 5: MIP Algorithm

$$\sum_{i=1}^{h}\int_{z_i}^{z_{i+1}} P(z)dz = \frac{1}{h}\int_{z_1}^{z_{h+1}} P(z)dz \quad (i=1,2,\ldots,h)$$

Studies [6] have indicated that for k = 8 and three projections, it is possible to structurally identify triangles, rectangles, squares, crosses, and circles. The decision regions for k = 4 and two projections P(a) and P(b) oriented 45° apart, are given in Figure 7. These results suggest that, given a prior knowledge of the type(s) of patterns a target can assume during its flight, it is possible to use its projections to distinguish the target from other object patterns possessing different structures. Having the structure encoded into parameters allows the algorithm to adapt the structural parameters frame by frame to the target's structure. The extreme points of an object (nose and tail in the case of a missile), can be established by solving simple linear equations using extreme points of the projections.

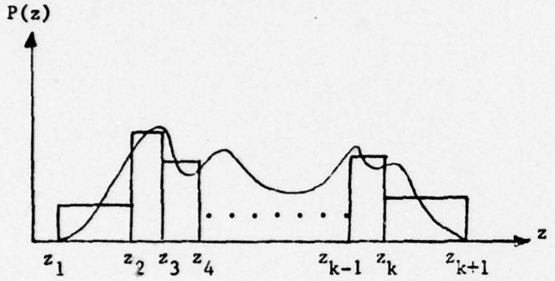


Figure 6: Area Quantization

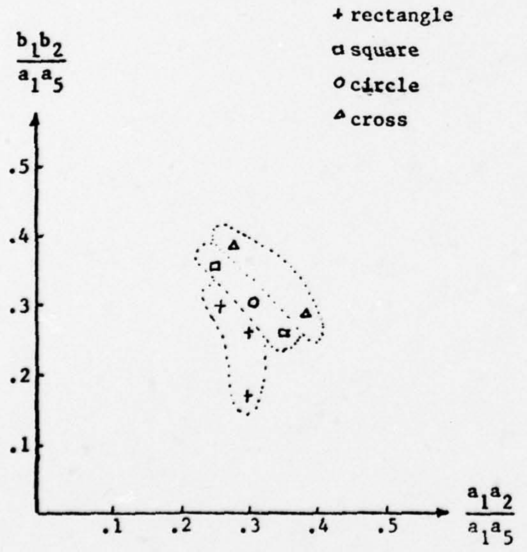


Figure 7: Decision Regions for k = 4

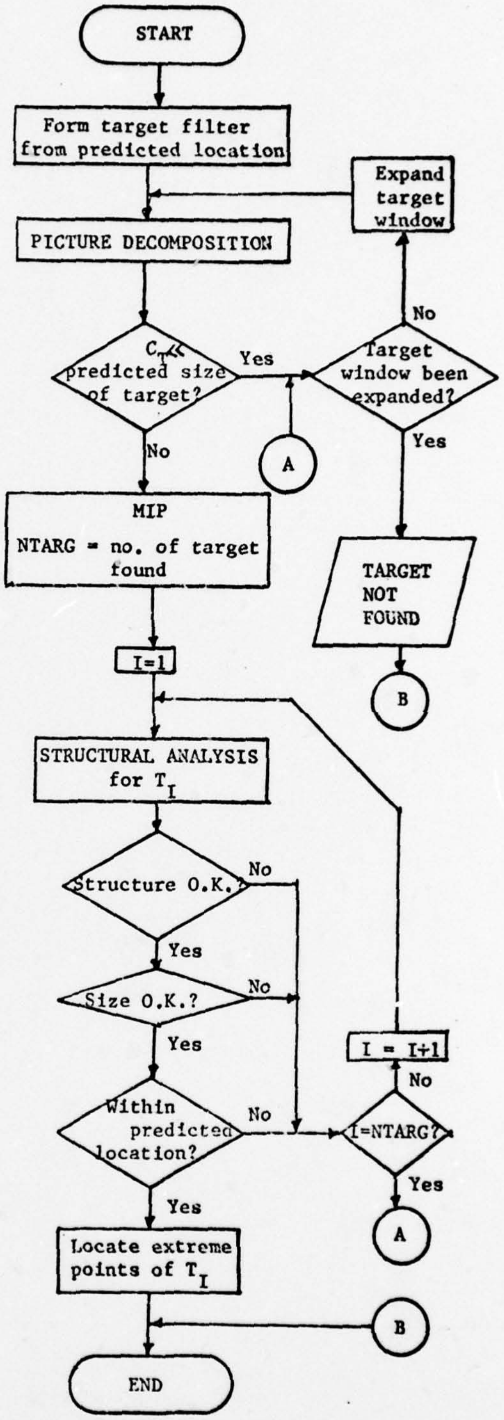

START
Form target filter from predicted location
Expand target window
PICTURE DECOMPOSITION
C_T<< predicted size of target?
Yes
No
Target window been expanded?
No
Yes
A
MIP
NTARG = no. of target found
TARGET NOT FOUND
B
I=1
STRUCTURAL ANALYSIS for T_I
Structure O.K.?
No
Yes
Size O.K.?
No
Yes
I = I+1
No
Within predicted location?
No
I=NTARG?
Yes
Yes
A
Locate extreme points of T_I
B
END

Figure 8: RTV Tracking Algorithm

THE RTV TRACKING ALGORITHM

The conceptual flow chart of a Real-Time Video Target Tracking Algorithm for a single target is given in Figure 8. For a multiple target environment, there are two solutions. When all targets of interest are spatially close, the size of the target window can be enlarged to enclose all predicted target locations. The MIP can then keep track of all targets using a target file. However, this approach is costly in terms of processing time and data resolution. An alternate approach would be to have an array of MIP's controlled by a master MIP controller. The master MIP controller would search a large scene for possible targets and direct local MIP trackers to identify and track targets of interest. Each local MIP would be responsible for tracking one target. Having an array of these trackers, many targets specially separated can be simultaneously tracked, providing a solution to a large class of multiple target tracking situations.

APPENDIX A. SINGLE OBJECT SCANNING ALGORITHM

INITX,INITY - Coordinates of initial window within the object.

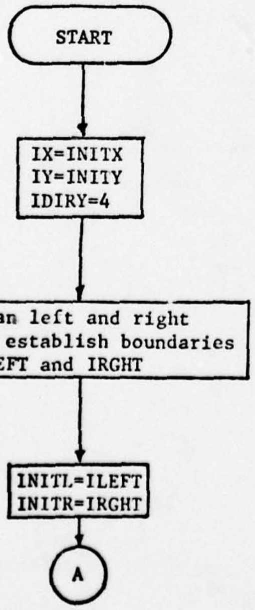
INITL,INITY - Left and right boundary of first row of windows scanned.

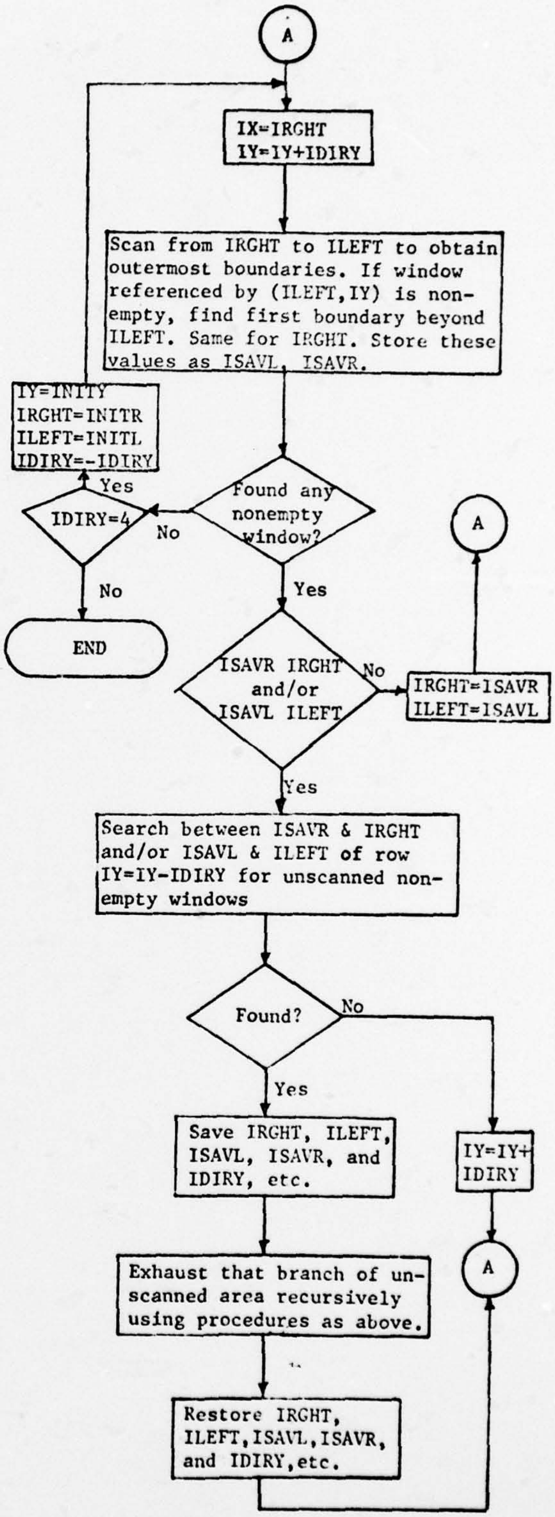
IX,IY - Scan window reference address for accessing object buffer memory.

IDIRY - y directional increment for scan window, equal to ±4.

ILEFT,IRGHT - Left and right boundary of previous row of windows scanned.

ISAVL,ISAVR - Left and right boundary of current row of windows scanned.




A
IX=IRGHT
IY=IY+IDIRY
Scan from IRGHT to ILEFT to obtain outermost boundaries. If window referenced by (ILEFT,IY) is nonempty, find first boundary beyond ILEFT. Same for IRGHT. Store these values as ISAVL, ISAVR.
IY=INITY
IRGHT=INITR
ILEFT=INITL
IDIRY=-IDIRY
Yes
IDIRY=4
No
Found any nonempty window?
No
END
Yes
A
ISAVR IRGHT and/or ISAVL ILEFT
No
IRGHT=ISAVR
ILEFT=ISAVL
Yes
Search between ISAVR & IRGHT and/or ISAVL & ILEFT of row IY=IY-IDIRY for unscanned nonempty windows
Found?
No
Yes
Save IRGHT, ILEFT, ISAVL, ISAVR, and IDIRY, etc.
IY=IY+IDIRY
Exhaust that branch of unscanned area recursively using procedures as above.
A
Restore IRGHT, ILEFT,ISAVL,ISAVR, and IDIRY,etc.

ACKNOWLEDGEMENT

This research was supported by White Sands Missile Range under Contract No. DAAD07-76-C-0024. The authors would like to thank Mrs. B. Vigil who cheerfully type the manuscript.

REFERENCES

1) FLACHS,G.M. and GILBERT,A.L. "Automatic Boresight Correction in Optical Tracking"; Proceedings of the IEEE 1975 National Aerospace and Electronics Conference.

2) FLACHS, G.M. "A New Target Tracking Algorithm for the Automatic Programmable Film Reader"; White Sands Missile Range Technical Report, STEWS-ID-74-3; Oct. 1974.

3) VAINSHTEIN, B.K. "Synthesis of Projection Functions"; Soviet Physics-Doklady, Vol. 10, No. 2; August 1971.

4) YAGLOM, I.M. and BOLTYANSKII, V.G. (Translated by KELLY, P.J. and WALTON, L.F.) Convex Figures; Holt, Rinehart and Winston; 1961.

5) GORDON, R. and HERMAN, G. "Three Dimensional Reconstruction from Projections - A Review of Algorithms"; Int. Review of Cytology; Vol. 38, 1974.

6) FLACHS, G.M. and U, Y.H. "Structural Feature Extraction by Projections"; Proceedings of IEEE 1976 Region V Conference.

7) CHANG, S.K. and CHOW, C.K. "Reconstruction of 3-Dimensional Objects from Two Orthogonal Projections and its Application to Cardiac Cineangiography"; IEEE Trans. Computers, Vol. C-22 No. 1; Jan. 1973.

A STRUCTURE AND DYNAMIC MATHEMATICAL MODEL OF A REAL-TIME VIDEO TRACKING SYSTEM

Wiley E. Thompson and Gerald M. Flachs
Department of Electrical and Computer Engineering
New Mexico State University
Las Cruces, New Mexico 88003

ABSTRACT

A control structure for a Real-Time Video (RTV) Tracking System is formulated and a dynamic state model of the system is developed for evaluating the accuracy and tracking performance of the total system configuration. The input driver for the model consists of a target spatial trajectory along with the target and background image, as presented to the TV camera. The outputs of the model consist of the azimuth and elevation pointing angles of significant points of the target. By utilizing a software simulation of the Video Image Processor and Tracking Algorithm, a closed loop model is formed which provides a method for evaluating the RTV Tracking System performance for different target structures, backgrounds, and target maneuvers.

SYSTEM DESCRIPTION

The overall structure for the Real-Time Video (RTV) Tracking System, for a single station, is depicted in Figure 1. Basically, the TV camera sends a video picture to the Video Image Processor and Tracking Algorithm (VIP) which (as discussed in detail in [1], [2]) locates the target relative to camera boresight. This information is transferred to the Data Processor. As an example, if the target were a missile, the data from the (VIP) might be nose (x_n, y_n) and tail (x_t, y_t) locations of the missile relative to boresight, as shown in Figure 2. In addition, camera roll angle ϕ_o (Figure 3) and zoom setting z_o are transferred from the camera to the Data Processor while camera azimuth angle θ_{Ao} and elevation angle θ_{Eo} are transferred from the camera mount system to the

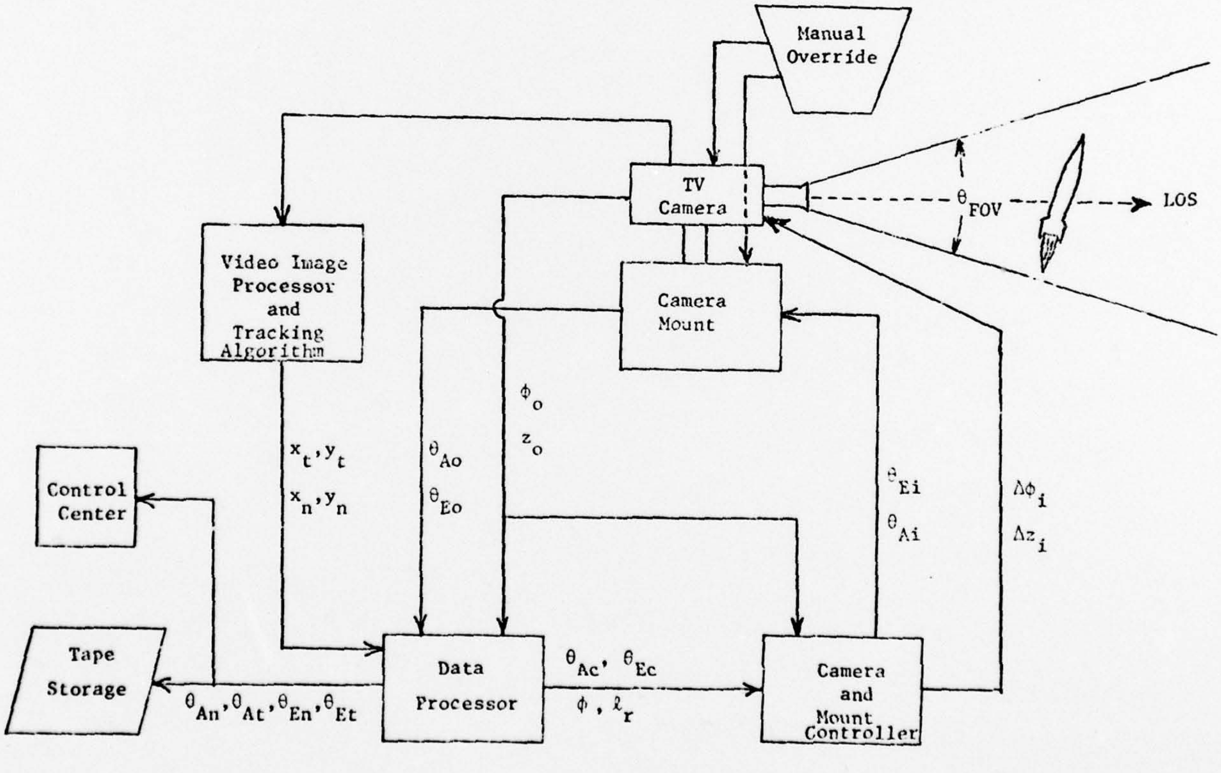


Figure 1. Overall Structure for the Real-Time Video Tracking System

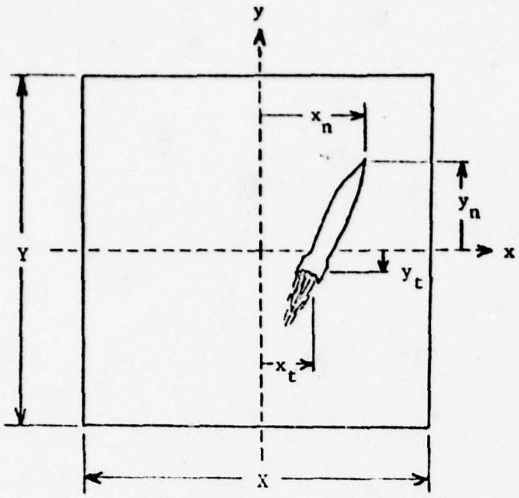


Figure 2. Target nose and tail locations relative to camera boresight.

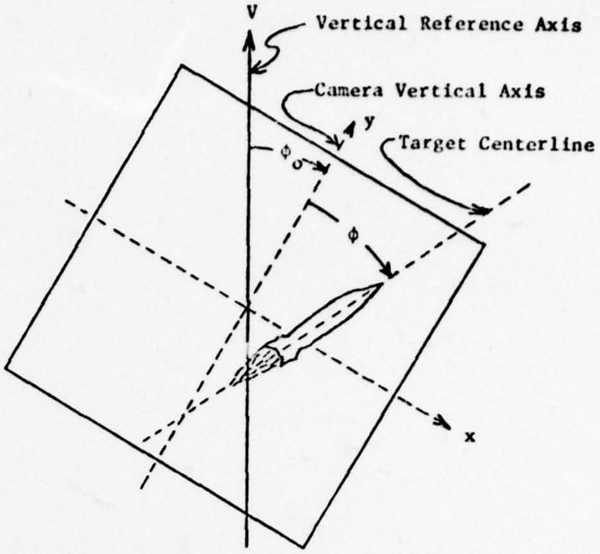


Figure 3. Camera roll angle ϕ_o relative to vertical reference axis V and target centerline angle ϕ relative to the camera vertical axis y.

Data Processor. The Data Processor then computes azimuth and elevation pointing angles corresponding to, say the nose and tail of the missile, i.e., θ_{An}, θ_{En}, θ_{At}, θ_{Et}, and stores them on magnetic tape along with the corresponding value of time. In addition, the Data Processor computes target centroid azimuth angle θ_{Ac}, elevation angle θ_{Ec}, target centerline angle with the camera vertical axis ϕ (Figure 3), and a size ratio parameter ℓ_r. This data is passed on to the Camera and Mount Controller (CMC). The controller also receives ϕ_o and z_o from the camera. This information is used by the CMC to predict target position and compute azimuth θ_{Ai} and elevation θ_{Ei} pointing angle input commands to the mount, along with zoom and roll angle change to the camera. The manual override control is to aid in initialization and re-establishing track in case of loss of track.

SYSTEM MODEL

In this section, each of the components, with the exception of the VIP, are considered in detail and the mathematical models are presented, along with the system interrelationships involving the component variables.

Data Processor

The field of view angle θ_{FOV} of the camera is a function of the camera zoom setting z_o. Let

$$\theta_{FOV} = f(z_o) \qquad (1)$$

Then the nose and tail azimuth and elevation angles as computed from the camera azimuth and elevation pointing angles, θ_{FOV}, ϕ_o, and the VIP output by the relations

$$\left.\begin{aligned}
\theta_{An} &= \theta_{Ao}+(\frac{x_n}{X}\cos\phi_o+\frac{y_n}{Y}\sin\phi_o)\theta_{FOV}\\
\theta_{At} &= \theta_{Ao}+(\frac{x_t}{X}\cos\phi_o+\frac{y_t}{Y}\sin\phi_o)\theta_{FOV}\\
\theta_{En} &= \theta_{Eo}+(\frac{y_n}{Y}\cos\phi_o-\frac{x_n}{X}\sin\phi_o)\theta_{FOV}\\
\theta_{Et} &= \theta_{Eo}+(\frac{y_t}{Y}\cos\phi_o-\frac{x_t}{X}\sin\phi_o)\theta_{FOV}
\end{aligned}\right\} \qquad (2)$$

These expressions are based on the assumption that the angles added to θ_{Ao} and θ_{Eo} are given by their tangents. This is a very good assumption since the quantities in parentheses are less than unity and the maximum possible value of θ_{FOV} is on the order of 1^o.

When time arguments are omitted for simplicity, as in (1) and (2), it is assumed that the variables correspond to the same sampling instant. Uniform sampling and control intervals are assumed, and "k" denotes the instant kT, where T is the sample interval. In this sense, all the variables in (1) and (2) implicitly have the argument "k." The target centroid azimuth and elevation angles are computed using the relations, respectively

$$\begin{aligned}
\theta_{Ac} &= \frac{\theta_{An}+\theta_{At}}{2}\\
\theta_{Ec} &= \frac{\theta_{En}+\theta_{Et}}{2}
\end{aligned} \qquad (3)$$

The angle that the target centerline makes with the vertical axis of the camera is computed as

$$\phi = \tan^{-1}\left[\frac{x_n - x_t}{y_n - y_t}\right] \qquad (4)$$

and the size ratio parameter, ℓ_r, is computed as

$$\ell_r = \max\left[\left|\frac{x_n - x_t}{X}\right|, \left|\frac{y_n - y_t}{Y}\right|\right] \qquad (5)$$

Camera and Mount Controller

One purpose of the controller is to compute, and issue to the camera mount, azimuth and elevation pointing angle commands so that the distance between the camera line of sight (LOS) and the target centroid is minimized. Since the target may be moving, the input commands are based on a predicted location of the target centroid. For simplicity in the preliminary investigations, a first order prediction is used. In particular, the azimuth and elevation angle input commands are given, respectively, by

$$\begin{aligned} \theta_{Ai}(k) &= 2\theta_{Ac}(k) - \theta_{Ac}(k-1) \\ \theta_{Ei}(k) &= 2\theta_{Ec}(k) - \theta_{Ec}(k-1) \end{aligned} \qquad (6)$$

This control predicts that the azimuth and elevation angles of the target will change the same amount during the next sample interval, from their present value, as they changed over the past sample interval.

Another function of the controller is to change the roll angle of the camera, ϕ_o, so that the angle, , between the camera vertical axis and the target centerline is minimized. Again, the control is based on a predicted angle for the target centerline, so that the input command $\Delta\phi_i$, representing the predicted desired change in ϕ_o, is given by

$$\Delta\phi_i(k) = 2\phi(k) + \phi_o(k) - \phi(k-1) - \phi_o(k-1) \qquad (7)$$

A third duty of the controller is to compute and issue commands for control of the zoom lens of the camera so as to maintain a certain specified value for the size ratio parameter, ℓ_r. Let ℓ_d be the desired value for ℓ_r. Then the error in ℓ_r is

$$\Delta\ell_r = \ell_d - \ell_r \qquad (8)$$

so that the needed change in zoom setting is

$$\Delta z_i = g(\Delta\ell_r) \qquad (9)$$

Camera Mount

The camera mount control system provides a position tracking system capable of being directed by digital pointing data at an input rate of up to 200 samples per second.

The transfer functions for the camera mount servosystems can be shown to have the following forms [3], [4]

$$\frac{\theta_{Ao}(s)}{\theta_{Ai}(s)} = \frac{a_1+a_2s+a_3s^2+a_4s^3+a_5s^4}{b_1+b_2s+b_3s^2+b_4s^3+s^4} \qquad (10)$$

and

$$\frac{\theta_{Eo}(s)}{\theta_{Ei}(s)} = \frac{c_1+c_2s+c_3s^2+c_4s^3+c_5s^4}{d_1+d_2s+d_3s^2+d_4s^3+s^4} \qquad (11)$$

for the azimuth and elevation servosystem, respectively. The parameter values in these equations, along with their physical significance and means of changing them, are given and discussed in [3], [4]. A phase variable state model representation for these systems is

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -b_1 & -b_2 & -b_3 & -b_4\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}\theta_{Ai}$$

$$\theta_{Ao} = [(a_1-a_5b_1)(a_2-a_5b_2)(a_3-a_5b_3)(a_4-a_5b_4)]\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} + a_5\theta_{Ai} \qquad (12)$$

for the azimuth position servosystem and

$$\begin{bmatrix}\dot{y}_1\\ \dot{y}_2\\ \dot{y}_3\\ \dot{y}_4\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -d_1 & -d_2 & -d_3 & -d_4\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}\theta_{Ei}$$

$$\theta_{Eo} = [(c_1-c_5d_1)(c_2-c_5d_1)(c_3-c_5d_3)(c_4-c_5d_4)]\begin{bmatrix}y_1\\ y_2\\ y_3\\ y_4\end{bmatrix} + c_s\theta_{Ei} \qquad (13)$$

for the elevation position servosystem.

TV Camera

In the actual RTV Tracking System, the TV camera will be mounted on and interfaced to the telescopic optics train of the Contraves Theodolite Camera Mount System. For purposes of this model, the TV camera includes the scanning process, image rotation, and zoom lens system. The scanning process is included as part of the simulated driver for the model. The image rotation and zoom lens system are characterized, respectively, by

$$\phi_o(k+1) = \phi_o(k) + \Delta\phi_i(k) \qquad (14)$$

and

$$z_o(k+1) = z_o(k) + \Delta z_i(k) \qquad (15)$$

where $\Delta\phi_i(k)$ and $\Delta z_i(k)$ are given by (7) and (9), respectively.

The direct sum of component models and interrelation equations (1)-(9), (12)-(15), along with a software simulation of the Video Image Processor and Tracking Algorithm, form a closed loop model for simulating and analyzing the RTV Tracking System.

FUTURE STUDIES AND SIMULATIONS

The control structure and dynamic model, as presented, form a basis for the initial design and evaluation of the RTV Tracking Systems. Simulation studies involving the model will provide valuable information regarding potential tracking accuracy and reliability relative to various target structures, target maneuvers, and background scenarios. Dynamic simulations providing access to all important system variables will be valuable in design improvement and modification in the total system. For example, changes in the target tracking algorithm can be quickly evaluated. Also, simulations will indicate the effectiveness of the first order predictions by the Camera and Mount Controller as well as possible need for higher order predictions. A sensitivity analysis will also be performed to indicate the relative importance of various system variables and parameters to the tracking accuracy of the system.

ACKNOWLEDGEMENT

This work was supported by White Sands Missile Range under Contract No. DAAD07-76-C-0024.

REFERENCES

[1] Flachs, G.M., Y.H. U, W. E. Thompson, and A.L. Gilbert, "A Real-Time Video Tracking Algorithm," Proc. of IEEE 1976 NAECON, May 1976.

[2] Black, R.J., G.M. Flachs, and J. Whitney, "A Pre-Prototype Real-Time Video Tracking System," Proc. of IEEE 1976 NAECON, May 1976.

[3] Stephenson, H.B., "Summing the Instrument Data Converter (IDC) Velocity Derived Signal to the Instrument Position Loop," Engineering Memorandum No. 72-16, Data System Branch, Range Modernization Division, White Sands Missile Range, N.M., July 1972.

[4] "Test Report of the Instrument Data Converter and Contraves F Interface for G-22 (SW-5D)," Lockheed Electronics Co., Inc. Technical Report TR75-008 to WSMR, August 1975.

[5] Grommes, R.J., and C.J. Yi, "Analysis and Simulation of a Video Tracking System," Proc. 1974 IEEE Systems, Man, Cybernetics Conference, pp. 93-98.

STRUCTURAL FEATURE EXTRACTION BY PROJECTIONS

Yee Hsun U and G.M. Flachs
Department of Electrical and Computer Engineering
New Mexico State University
Las Cruces, New Mexico 88003

ABSTRACT

The theory of projections can be applied to extract structural information from an object pattern. Area quantization is used to transform a projection function into a parametric model for structural analysis. Utilizing the parametric structural model, a pattern classification experiment is performed to recognize a simple class of binary patterns. The results of the experiment demonstrate the feasibility of using projections to extract global structural features for pattern recognition.

INTRODUCTION

The subject of representation and feature extraction of pictorial structures has received much intense study in the last decade. Most efforts concentrate on defining some basic primitives at the local level and developing a global model for an object of interest from an interconnected network of the primitives [1,2,3].

Define the global structural feature of an object as the geometric shape it assumes when the multi-gray level representation of the object has been reduced to a binary one. We will assume, unless otherwise stated, such an interpretation for the word "structure" throughout this paper, as differs from the generally accepted meaning of "structural" in pattern recognition which connotes an approach that characterizes a pattern as being composed of primitive elements that are related to each other in well defined ways [6]. Some work performed on pattern classification using global structural features is given by [4,5]. These approaches suppress certain micro details at the local level; however, inherent in these techniques is a high degree of immunity to noise.

The problem under consideration in this paper is the structural recognition of object patterns under real-time processing constraints. Little attention has been devoted to the development of real-time structural recognition algorithms, although there are many applications in weapon systems, tracking systems, and surveillance systems. Projections are used to structurally characterize binary patterns. Area quantization of the projections provides a parametric model for real-time structural recognition.

PROJECTION

The projection in the x-y plane of a picture function f(x,y) along a certain direction w onto a straight line z perpendicular to w is defined to be [7]

$$P_w(z) = \int f(x,y)dw \qquad (1)$$

In general, a projection integrates the gray levels of a picture along parallel lines through the pattern generating a function called the projections (Figure 1). For digitized binary patterns, the projection function gives the summation of object points along parallel lines [8]; hence, it is a distribution of the target points for a given view angle. The theory of projection is used mainly as a tool for reconstructing multi-gray level complex pictures [9]. For a binary picture, any convex symmetric pattern can be reconstructed from two orthogonal projections [8].

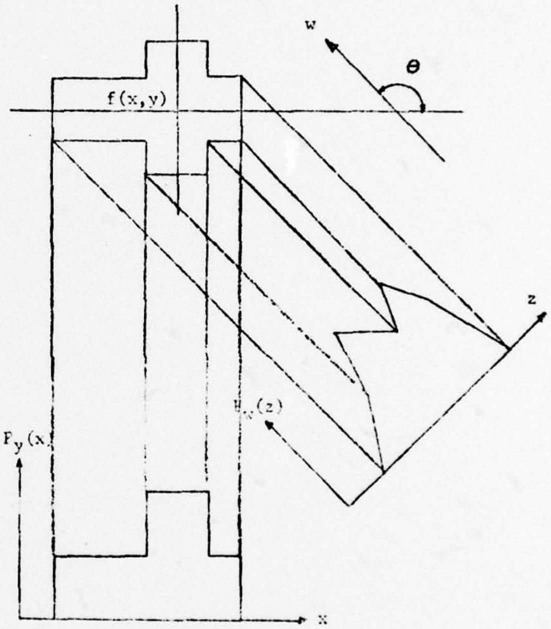


Figure 1: Projections

The pattern recognition experiment presented in this paper utilizes the following structural properties of the projection function $P_w(z)$ for a binary pattern of an object A.

1. A convex $\Rightarrow P_w(\alpha z_1+(1-\alpha)z_2) \geq \alpha P_w(z_1)+(1-\alpha)P_w(z_2)$
   $\forall$ w and $z_1,z_2 \in z$; $0 \leq \alpha \leq 1$; $z_1 \leq z_2$
2. A is symmetric with respect to two orthogonal axis
   $\Rightarrow P_w(z)$ is symmetric $\forall w$
3. $\int_z P_w(z) = \int_y P_x(y) \quad \forall w$ and $\forall x$
4. If a triangle has two symmetric projection functions 45° apart, then the mid point of each projection must project through a node of the triangle.

AREA QUANTIZATION

It is necessary to transform a projection function into a parametric model for structural analysis. Area quantization offers the advantage of easy implementa-

This research is supported by White Sands Missile Range, New Mexico under Contract No. DAAD07-76-C-0024 and by the U.S. Army Research Office, Durham, North Carolina, under Grant DAH C04-75-G-0038.

tion on a computer. This process transforms a projection function $P_w(z)$ into k rectangles of equal areas (Figure 2), where

$$\int_{z_i}^{z_{i+1}} P_w(z)dz = \frac{1}{k}\int_{z_1}^{z_{k+1}} P_w(z)dz \qquad (2)$$

where

$i = 1,2,\ldots,k$

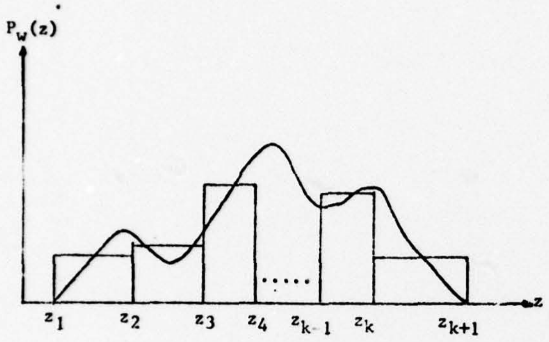


Figure 2: Area Quantization

Another important feature of the area quantization model for a projection function of an object is that the ratio of line segments

$$\frac{z_i z_j}{z_k z_l} \quad i \neq j,\ k \neq l$$

is object size invariant. Consequently, these parameters measure the structure of the object independent of size.

A PATTERN CLASSIFICATION EXPERIMENT

A basic class of simple patterns is defined in Figure 3. It includes triangle, cross, circle, and rectangle. These patterns were chosen because a large class of more complex shapes can be approximated by them. With the exception of the triangle, all the figures have two orthogonal axis of symmetry. Properties 2 and 4 in the section on projection make it reasonable to differentiate a triangle from the rest by simply checking if the projection is nonsymmetric. To classify the rest of the simple patterns, the first three patterns in Figure 3, with the inequality constraint set to be equal, were digitized into 32x32 binary picture arrays. These arrays were presented to a computer which generated two projection functions 45° apart and area quantized them for k = 8. Six parameters were calculated for each pattern and their values are given in Figure 4. The value $z_i z_j$ and the angle θ are as defined in equation (2) and Figure 1, respectively. These three sets of six parameters each are used as a standard reference in the classification experiment, whose simplified flowchart is given in Figure 5.

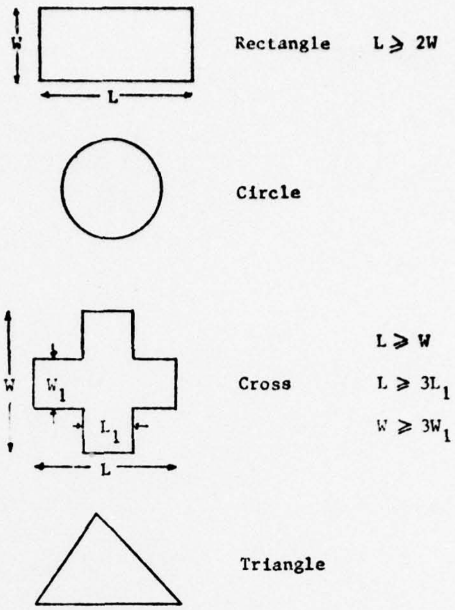


Figure 3: A Basic Class of Simple Patterns

Four projection functions are initially generated for each input pattern; however, only two projections are area quantized and used in the classification algorithm. The widest projection and its 45° counterclockwise neighbor are selected to reduce the rotational effect. Six parameters $A_1,\ldots,A_6$, as defined in Figure 4, along with a symmetry measure of the projections, are utilized to distinguish between rectangles, circles, crosses, and triangles. For a given pattern, these structural parameters are computed and compared to the standard reference (Figure 4) values for each basic pattern. The object is classified to the category with the best match utilizing a minimal distance decision rule.

		Rectangle	Circle	Cross
$\theta = 135°$	A_1	0.25	0.368	0.417
	A_2	0.5	0.599	0.722
	A_3	0.75	0.805	0.861
$\theta = 90°$	A_4	0.434	0.368	0.380
	A_5	0.643	0.599	0.553
	A_6	0.821	0.805	0.744

$$A_1 = \frac{z_1 z_2}{z_1 z_5} \qquad A_2 = \frac{z_1 z_3}{z_1 z_5} \qquad A_3 = \frac{z_1 z_4}{z_1 z_5}$$

A_4, A_5, and A_6 are similarly defined

Figure 4: Area Quantization Data for k = 8

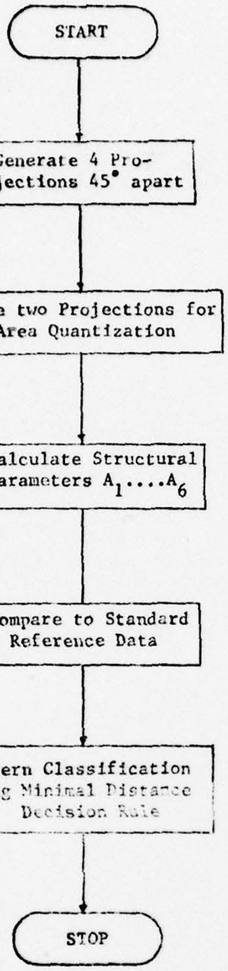

START
Generate 4 Projections 45° apart
Choose two Projections for Area Quantization
Calculate Structural Parameters $A_1 \ldots A_6$
Compare to Standard Reference Data
Pattern Classification using Minimal Distance Decision Rule
STOP

Figure 5: Classification Algorithm

		▭	○	△	✚
Rectangle	▭	100	0	0	0
Circle	○	0	100	0	0
Triangle	△	0	0	90	10
Cross	✚	10	10	10	70

Figure 6: Results for noise-free
Test Patterns

		▭	○	△	✚
Rectangle	▭	90	10	0	0
Circle	○	0	100	0	0
Triangle	△	0	0	90	10
Cross	✚	10	15	10	65

Figure 7: Results for Test Patterns
with Noise Added

The classification algorithm has been used to identify test patterns for each of the four basic patterns. The results of the classification algorithm for hand-generated test pattern set are given in Figure 6, when the i,j entry of the matrix is the percentage of test pattern i classified as pattern j.

Noise was added to the same test pattern set and the experiment was repeated with results given in Figure 7. The added noise did not significantly reduce the effectiveness of the algorithm, demonstrating the noise immunity property of the global structural features. Typical test patterns utilized in this experiment are given in Appendix A.

DISCUSSION

Projections provide a method for extracting global structural parameters for pattern recognition algorithms. The projection can be implemented in hardware and the area quantization implemented with parallel processing hardware, providing a feasible method for real-time structural recognition. Although the decision rule utilized to classify the patterns in this paper is extremely simple, the global structural features demonstrated remarkable structural discriminating ability and deserve further research.

REFERENCES

[1] Fischler, M.A. and Elschlager, R.A., "The Representation and Matching of Pictorial Structures," IEEE Trans. Comp., Vol. C-22, No. 1, January 1973.

[2] Underwood, W.E. and Kanal, L.N., "Structural Descriptions, Transformational Rules, and Pattern Analysis," Proc. 1st Int. Joint Conf. on Pattern Recognition, Washington, D.C., 1973.

[3] Feng, H.Y.F. and Pavlidis, T., "Decomposition of Polygons into Simpler Components: Feature Generation for Syntactic Pattern Recognition," IEEE Trans. Comp., Vol. C-24, No. 6, June 1975.

[4] Wong, E. and Steppe, J.A., "Invariant Recognition of Geometric Shapes," Methodologies of Pattern Recognition (S. Watanabe-ed), Academic Press, 1969.

[5] Nakano, Y. et.al., "Improvement of Chinese Character Recognition Using Projection Profiles," Proc. 1st Int. Joint Conf. on Pattern Recognition, Washington, D.C., 1973.

[6] Blackwell, F.W., "Combining Mathematical and Structural Pattern Recognition," Proc. 2nd Int. Joint

Conf. on Pattern Recognition, Copenhagen, Denmark, 1974.

[7] Vainshtein, B.K., "Synthesis of Projecting Functions," Soviet Physics - Doklady, Vol. 10, No. 2, August 1971.

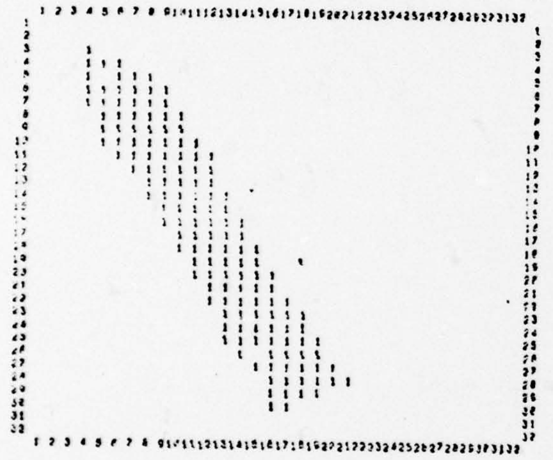
[8] Chang, S.K. and Chow, C.K., "Reconstruction of 3-D Objects from Two Orthogonal Projections and its Application to Cardiac Cineangiography," IEEE Trans. Comp., Vol. C-22, No. 1, January, 1973.

[9] Gordon, R. and Herman, G., "Three Dimensional Reconstruction from Projections - A Review of Algorithms," Int. Review of Cytology, Vol. 38, 1974.

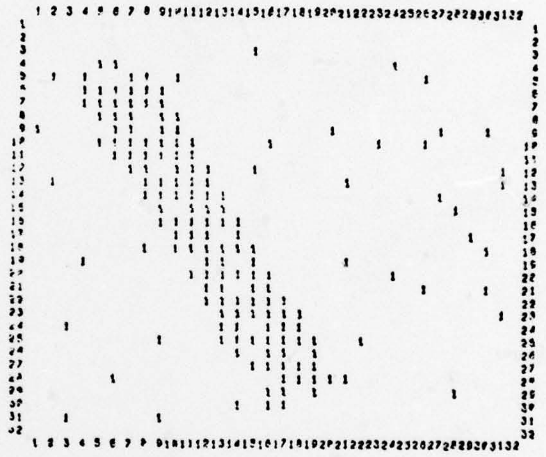
[10] Kashyap, R.L. and Mittal, M.C., "Picture Reconstruction from Projections," IEEE Trans. Comp., Vol. C-24, No. 9, September 1975.

APPENDIX A

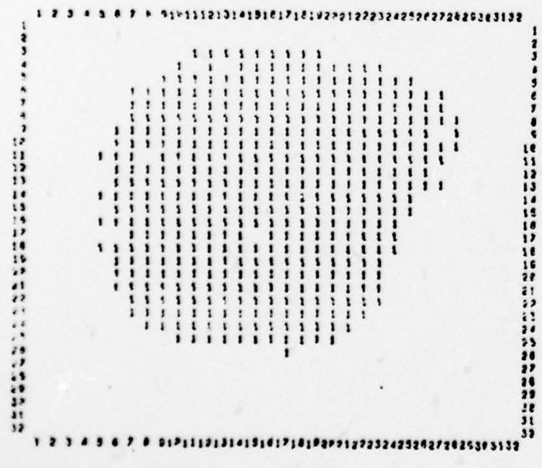
Sample Test Patterns



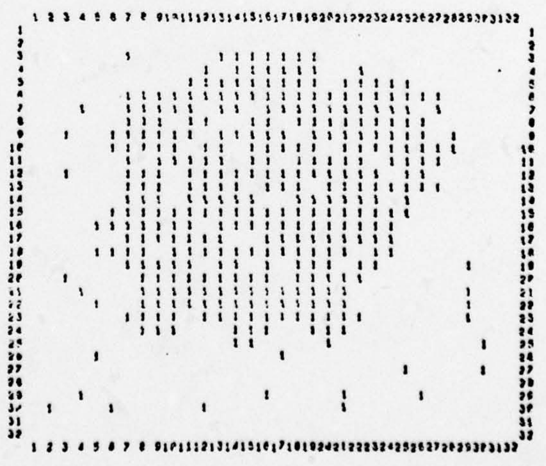
Rectangle



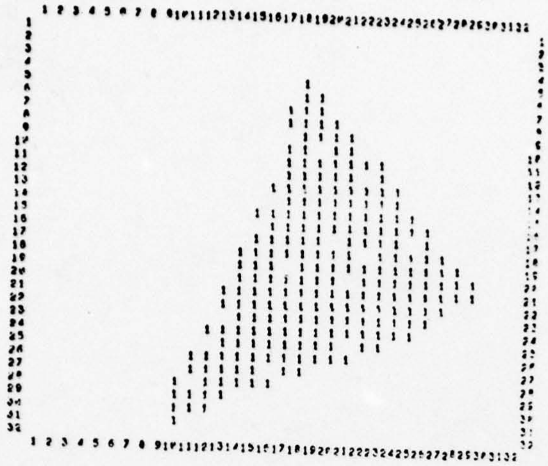
Rectangle with Noise Added



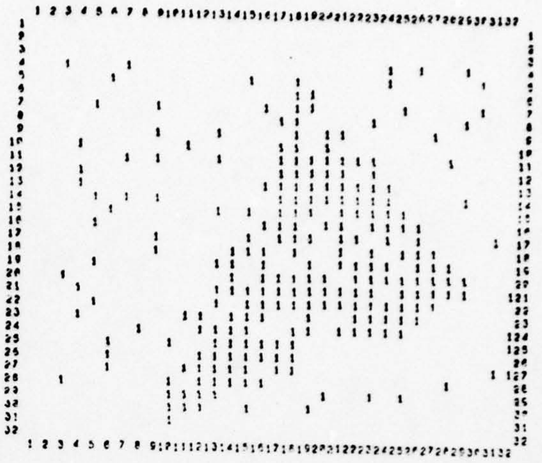
Circle



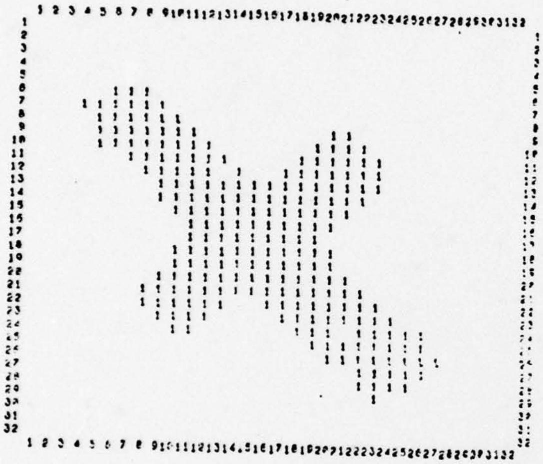
Circle with Noise Added



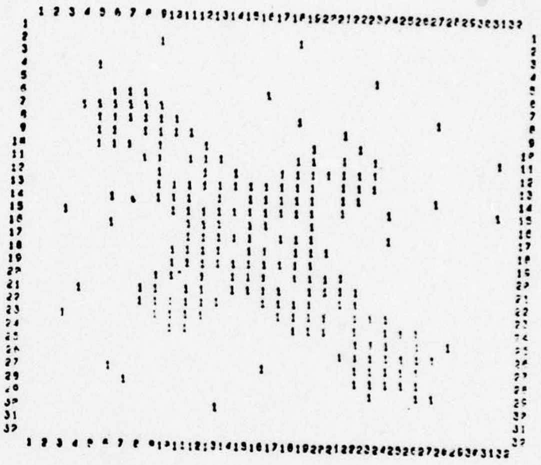
Triangle



Triangle with Noise Added



Cross



Cross with Noise Added