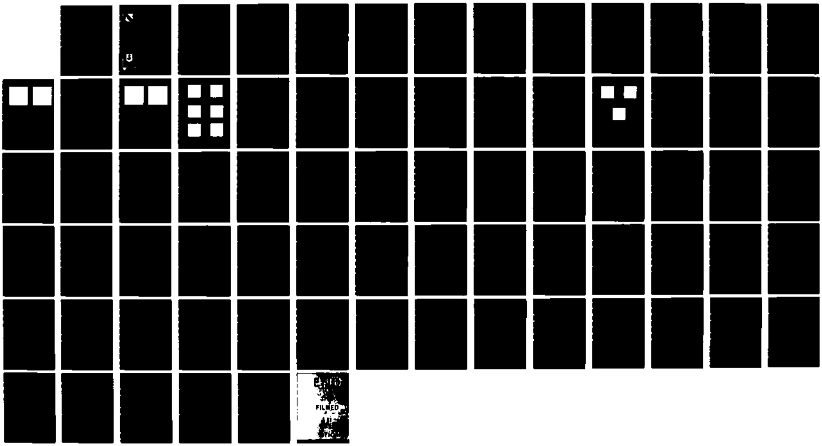
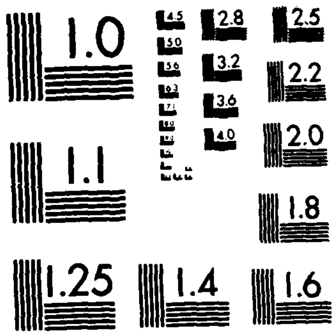
AD-A128 622 OPTICAL TRACKING INVESTIGATIONS INTO IMAGE INFORMATION AND ADAPTIVE GUIDANCE(U) FRANK J SEILER RESEARCH LAB UNITED STATES AIR FORCE ACADEMY CO J D LEDBETTER APR 83 FJSRL-TR-83-0004 1/1

UNCLASSIFIED F/G 17/8 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FRANK J. SEILER RESEARCH LABORATORY

FJSRL-TR-83-0004 APRIL 1983

OPTICAL TRACKING INVESTIGATIONS INTO IMAGE INFORMATION AND ADAPTIVE GUIDANCE

FINAL REPORT

CAPTAIN JAMES D. LEDBETTER

PROJECT 2305-F2-65

AIR FORCE SYSTEMS COMMAND

UNITED STATES AIR FORCE

This document was prepared by the Guidance and Control Division, Directorate of Aerospace-Mechanics, Frank J. Seiler Research Laboratory, United States Air Force Academy, Colorado Springs, CO. The research was conducted under Project Work Unit Number 2305-F2-65, Adaptive Concepts in Electro-Optical Tracking Applications. Captain James D. Ledbetter was the Project Scientist in charge of the work.

Inquiries concerning the technical content of this document should be addressed to the Frank J. Seiler Research Laboratory (AFSC), FJSRL/NH, USAF Academy, Colorado Springs, CO 80840. Phone AC 303-472-3122.

This report has been reviewed by the Commander and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

James D. Ledbetter

JAMES D. LEDBETTER, Captain, USAF
Project Scientist

T. Saito

THEODORE T. SAITO, Lt Col, USAF
Director, Aerospace-Mechanics Sciences

WILLIAM D. SIURU, JR., Col, USAF
Commander

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER FJSRL-TR-83-0004	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Optical Tracking Investigations Into Image Information and Adaptive Guidance		5. TYPE OF REPORT & PERIOD COVERED Final Report Sep 79 - Aug 82
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Captain James D. Ledbetter		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Frank J. Seiler Research Laboratory USAF Academy Colorado Springs, CO 80840		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2305-F2-65
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE April 1983
		13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

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

18. SUPPLEMENTARY NOTES

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

Image Processing
Texture Analysis
Segmentation
Optical Tracking
Adaptive Guidance
Reachable-set Theory

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

A two part study into critical aspects of optical tracking systems is presented. The basic question of what comprises information in a textural image is addressed in a study to investigate the application of an erosion process to a binary image. The objective of the erosion is to enhance the image properties that match the properties of the structural element, or mask, used in the erosion process. The resulting image can be used for further processing, but the enhanced properties would now dominate other qualities or features in the image.

(CONTINUED ON REVERSE)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 Continued:

A statistical study of various textural edges is presented. The use of the variance and mean to indicate textural transitions is investigated on two different types of textural edges.

An adaptive guidance technique based on reachable set theory is compared to pursuit and proportional navigation approaches presently employed in missile systems. It is shown that this approach adapts to maneuvering targets far better than the traditional approaches. The computational requirements for the algorithm are evaluated in terms of present digital hardware capability. The results indicate that using current technology, substantial improvements in guidance system performance can be realized.

Accession For

NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification ______

By ______
Distribution/

Availability Codes

Dist	Avail and/or Special
A	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure No. Page No.

PREFACE

The work summarized in this report falls into two separate, though related, areas. Image processing techniques to extract target data from an optical sensor and the subsequent guidance algorithm to use that data certainly form a closed loop system with regards to purpose. However, the intent here is to look at basic concepts in both problem areas without regard to "closing the loop" and forming a complete tracking system. The work reported on texture analysis of image data was done by Captain James Ledbetter, Frank J. Seiler Research Laboratory. The work on the reachable-set guidance evaluation was performed and written by Dr. Michael Larimore and Dr. Claude Wiatrowski, University of Colorado at Colorado Springs, with minor revisions by Captain Ledbetter for inclusion in this report.

SECTION I

1.0 Introduction and Background

The continued advancement of solid state imaging technology has resulted in an increased emphasis on the development of low-cost, light weight sensor arrays for Air Force surveillance, reconnaissance, and weapons control systems. The use of these sensors as imaging devices is attractive due to elimination of high voltage vacuum tube technology of conventional vidicon tubes[1,2]. The sensor's small size and low power requirements insure both linear and array devices will have a strong impact in tracking applications where limited space and power availability are factors. Additional technology verification has been provided by an Air Force Avionics Laboratory program which conducted a parametric analysis of an array tracker system to determine its performance characteristics and found that the charge coupled device (CCD) imaging array was not the limiting factor in most tracking applications[3]. More recently, the Jet Propulsion Laboratory has reported the design of a CCD tracker for a space mission[4].

This report documents efforts undertaken to investigate two different problem areas in optical tracking. The first area is that of target segmentation in the optical field of view. The segmentation process must be performed early in the processing of the image data and is a combination of boundary detection and texture analysis techniques[5]. Of particular interest in this study is the information gained from the image texture. This interest is a result of natural terrain where many military targets would be located. Such terrain is not predominately characterized by tonal edges but by textural changes.

The second area of interest is adaptive guidance and control. An "intelligent" weapons system utilizing an optical sensor might ultimately incorporate software that would allow it to adapt to the changing scenarios it "sees" by anticipating or predicting what a target might do to optimize its position in an encounter. The present guidance techniques of pursuit or proportional navigation can be deceived by an intelligent target capable of radical manuevers in the final seconds of the encounter. This report examines a promising guidance technique that can adapt to maneuvers. The technique is evaluated in terms of its implementation with present day technology.

SECTION II. TEXTURE ANALYSIS INVESTIGATION

2.0 Introduction

Image segmentation is a major component of any machine image analysis requirement. Based on the quality of the segmentation, other important image descriptors, or features, can be defined to further represent the image data. However, while humans find it very easy to "see" a collection of objects when they view a scene, machines "see" only an array of equally weighted pixel values which vary in intensity based on the amount of light incident upon them. Objects, therefore, are not "seen", only pixel intensity from which the objects comprising the image must be determined. Pixel intensities, and the way they are arranged, comprise the basic elements of information available to segment collections of pixels into objects of interest or regions which have more or less homogeneous properties.

Pixel intensities and their arrangement comprise the inextricable relationship of tone and texture properties in the image. Both properties are always present in an image, one usually predominating over the other. When an area has very little variation in pixel intensity, the predominant property is tone. When an area has wide variation in pixel intensity, the predominant property is texture. The size of the area in this distinction is critical. Its crucial nature arises when describing a given texture in terms of its tonal primitives and a given spatial organization. The term "spatial organization" requires the declaration of the size of the area concerned. For that local area, the texture is then the combination of one or more tonal primitives, regions with tonal properties, and a spatial rule specifying their arrangment. The segmentation of images where the information varies between the tonal properties and the textural properties is very difficult unless a priori knowledge is available on the statistics or properties of the texture. Without this knowledge, local area operations must be used to discern the statistics of the texture. However, the computation of these properties is a function of the size of the local area. This interrelationship usually results in the boundary between two textural areas being blurred or broadened due to the averaging of the properties of the regions when the local operator is straddling the boundary.

When regions of uniform texture are defined there are many different ways to approach deriving the properties of the textures. A very useful survey of differing approaches to texture definition is given in Reference 6. One method which has proven very useful in classifying textures is that of determining concurrence matrices and defining the texture based on features of those matrices[7]. The power of this approach is that it characterizes the spatial interrelationships of the grey tones in a textural pattern. Its weakness is that it does not capture the shapes of the tonal primitives.

In this section of the report, information on two investigations into texture segmentation techniques is presented. The image used was almost entirely textural in content. The primary interest was in determining a fast, simple "information indicator" that could be used to enable an adaptive segmenter to switch between tonal and textural operations in finding edges for further boundary definition. Section 2.1 describes the image used. Sections 2.2 and 2.3 describe an erosion study and a statistics study, respectively, of different textural edges in the image.

2.1 Image Information

The images used in these investigations were 512x512 subpartitions from a 4000x3000 pixel image from the Seasat-A synthetic aperature radar (SAR) sensor. This sensor operated from July through early October 1978 and generated a large amount of land and sea data. Since it is a radar imager, the information in the images is almost entirely textural in content. Figures 2.1 and 2.2 are the images used in the study. The broad, dark shaded bands on the images are a result of the technique used to photograph the screen of the video monitor in the International Imaging Systems Model 70 image processing system used in this study. Various textural edges in these images are used in the studies detailed in the following sections.

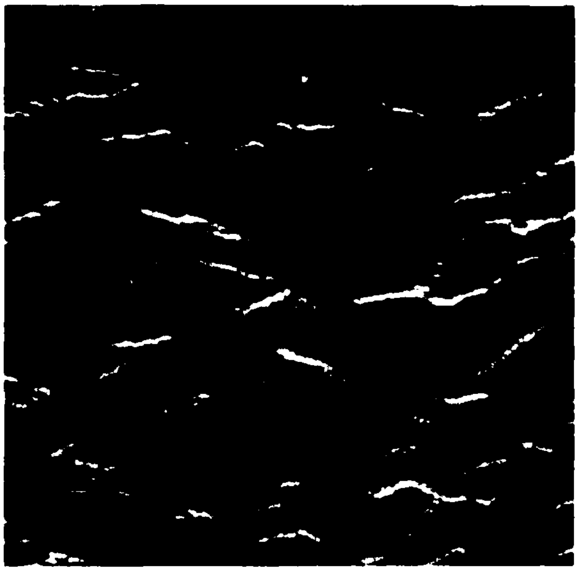


Figure 2.1 Seasat Image 1

Figure 2.2 Seasat Image 2

2.2 Erosion Study

Erosion is a filtering approach applied to binary images. The process is summarized in Reference 6 as follows. The basic idea is to define a structural element as a set of resolution cells constituting a specific shape such as a line or square and to generate a new binary image by translating the structural element through the image and retaining only those pixels as 1's where there is a match between the structural element 1's and the image 1's. The process, in effect, erodes the binary image as successive translations of the structural element are made through the image. This process is shown very simply in Figure 2.3.

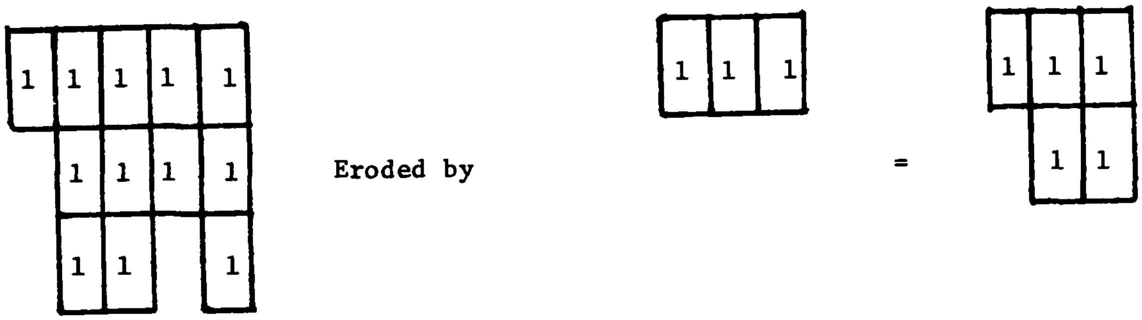


Figure 2.3 Erosion Process

The textural feature obtained for each translation of the structural element through the image is the number of 1's left after each cycle. For binary images this is the same as the area. The area versus the number of erosion cycles is plotted and yields what is called the covariance function.

The power of this approach is that it emphasizes the shape aspects of the tonal primitives of the texture. It has found wide application in the analysis of microstructures. Since the texture in Figures 2.1 and 2.2 are very fine, i.e., the tonal primitives are very small, it seemed that this approach could be used to determine shape characteristics, i.e., orientation, width, and density of the different textural edges. The edges considered in this study are shown in Figure 2.4 and Figure 2.5. The coordinates of the crosshairs define the names of the two edges, i.e., edge (430, 312) and edge (127, 122). The edges were partitioned into a 32x32 image for the analysis.

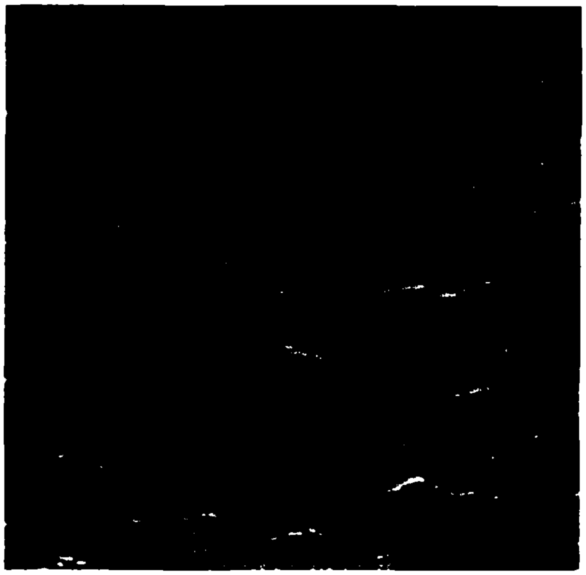


Figure 2.4
Location of Edge (430,312)

Figure 2.5
Location of Edge (127,122)

The important parameters in the erosion procedure are the binary image and the shape and size of the structural element, or mask, used to erode the image. The binary image is important from the standpoint of the threshold level used to generate it from the original grey level image. Figures 2.6, 2.7, 2.8 are the results of thresholding Figure 2.1 at levels where 10%, 20%, and 30%, respectively, of the grey levels in the original are above the threshold level. All pixels above the threshold value are set to 1 in the binary image while all pixels below the threshold are set to 0.

The structural elements used in this study were primarily chosen to determine how well the orientation and density of the edges could be determined. Figures 2.9, 2.10, 2.11 show the effect of one, two, and three erosion cycles, respectively, on Figure 2.6. The structural element used was a horizontal line three elements long, i.e., [1,1,1]. The covariance plots resulting from the complete erosion process on edge (127, 122) and edge (430, 312) in Figures 2.6, 2.7, and 2.8 are shown in Figures 2.12 and 2.13.

Figure 2.6
Seasat Image 1 - 10% Threshold

Figure 2.7
Seasat Image 1 - 20% Threshold

Figure 2.8
Seasat Image 1 - 30% Threshold

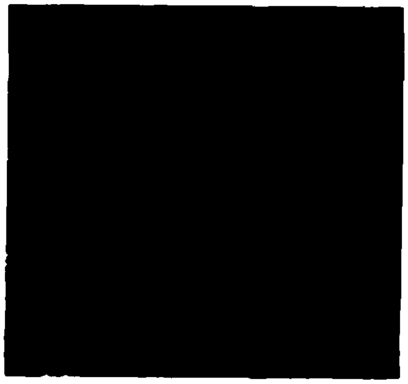


Figure 2.9
One Erosion - 10% Threshold

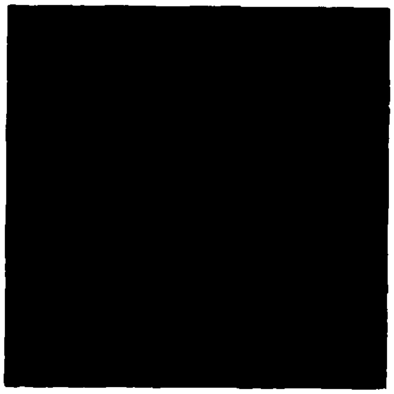


Figure 2.10
Two Erosions - 10% Threshold

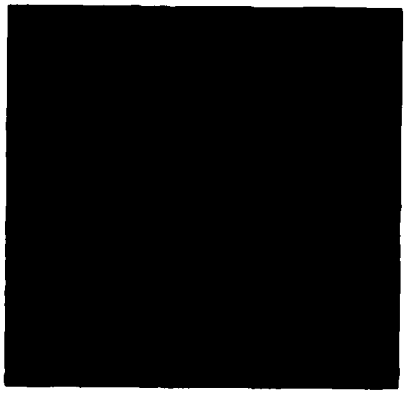


Figure 2.11
Three Erosions - 10% Threshold

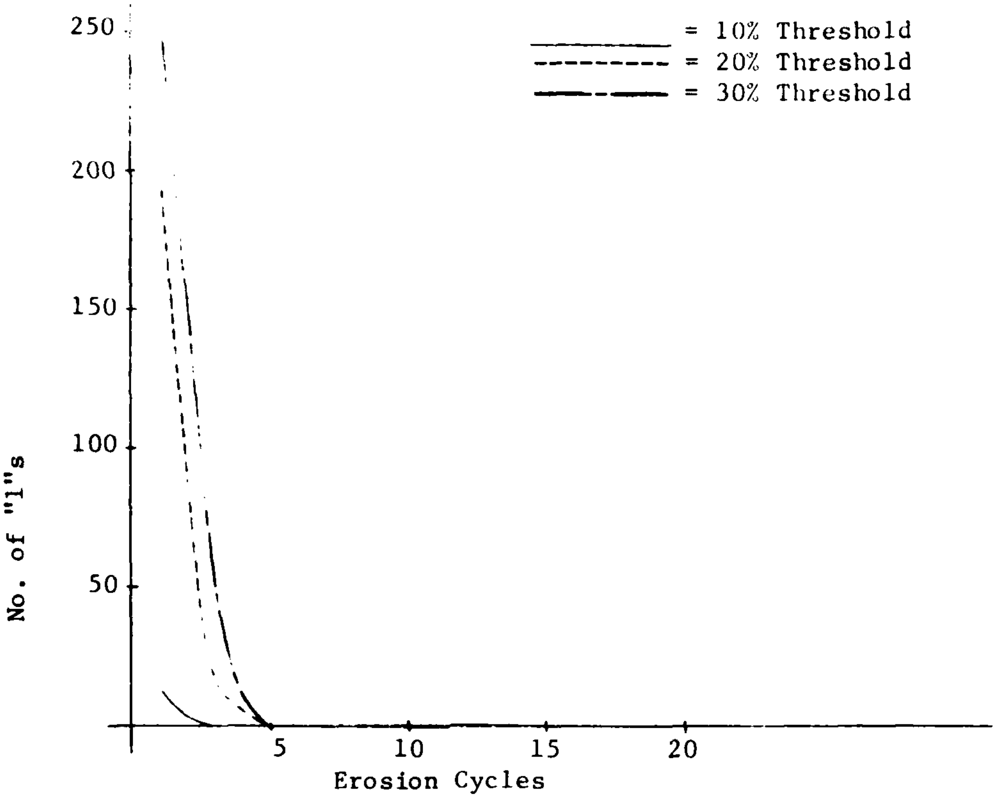


Figure 2.12 Covariance for Edge (430,312), --- Mask

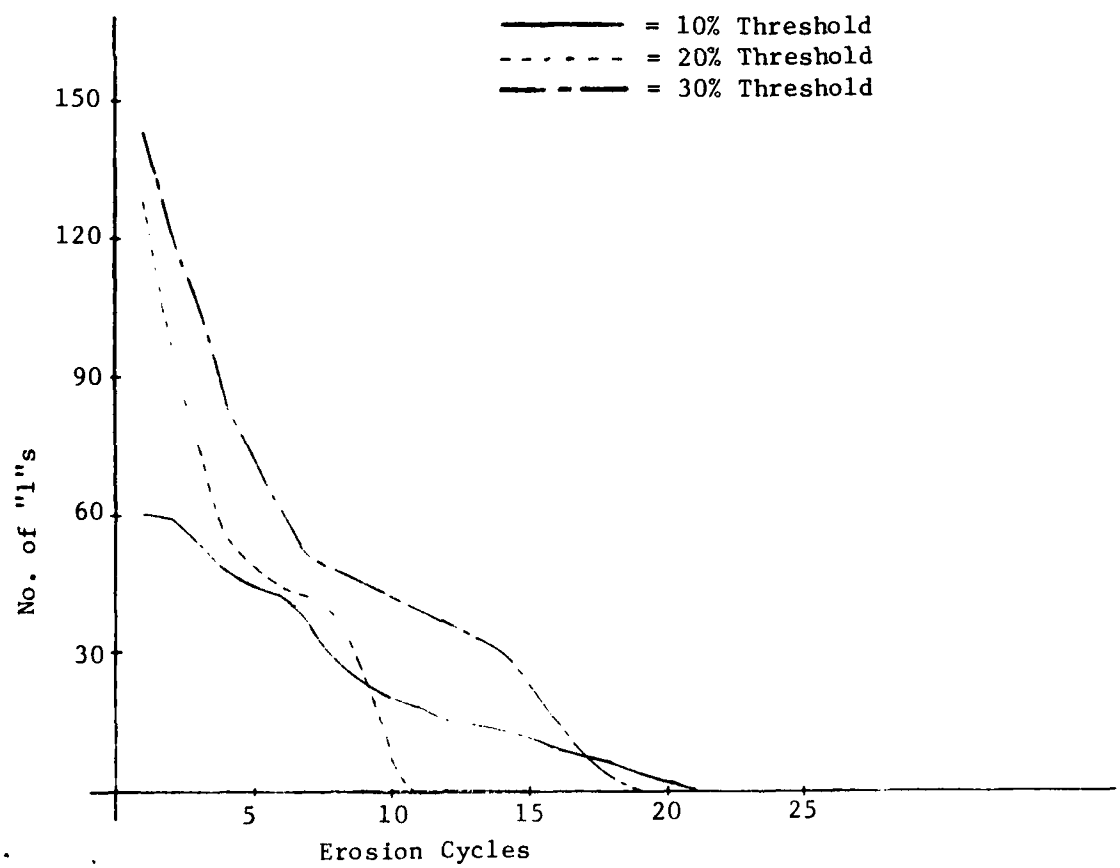


Figure 2.13 Covariance for Edge (127,122), --- Mask

Edge (127, 122) and edge (430, 312) were chosen because of their decidedly different characteristics, as evident in Figures 2.4 and 2.5. An examination of the covariance plots in Figures 2.12 and 2.13 shows that erosion with the three-wide, horizontal structural element results in plots that are markedly different also. Edge (430, 312) is oriented at approximately 45 degrees and is relatively broad but not very dense. Visually it is not perceived as having any horizontal qualities due, primarily, to its density and orientation. This is also evident on the covariance plot of Figure 2.13. The very rapid erosion to zero in five cycles for every threshold level indicates very little in the way of horizontal qualities. The large area, or number of 1's, at the 30% threshold level is indicative of the size of the edge but, due to its orientation, the horizontal mask quickly shows that the structural primitives comprising the texture of this edge are not horizontal in nature.

Edge (127, 122) is visually perceived as a dense, relatively thin horizontal edge. Its covariance plot in Figure 2.12 immediately reflects the horizontal orientation through the large number of cycles necessary to erode. Even more important is the piecewise linear nature of the plots. Constant slope is an indication that the same number of pixels are being eroded for each cycle. This implies that a very uniform structure matching the orientation of the structural element is present. Figures 2.9, 2.10, and 2.11 also show that during the erosion process the horizontal qualities of the edge were amplified by the horizontal mask.

Two other structural elements masks were applied to edge (430, 312). These elements were oriented at 135 and 45 degrees. The covariance plots for them are shown in Figure 2.14 and Figure 2.15. The same properties are evident as discussed previously for the horizontal mask. The 45 degree mask immediately shows the orientation of the edge. The piecewise linear structure emphasizes the orientation match. During the erosion process the 45 degree property was emphasized by the mask.

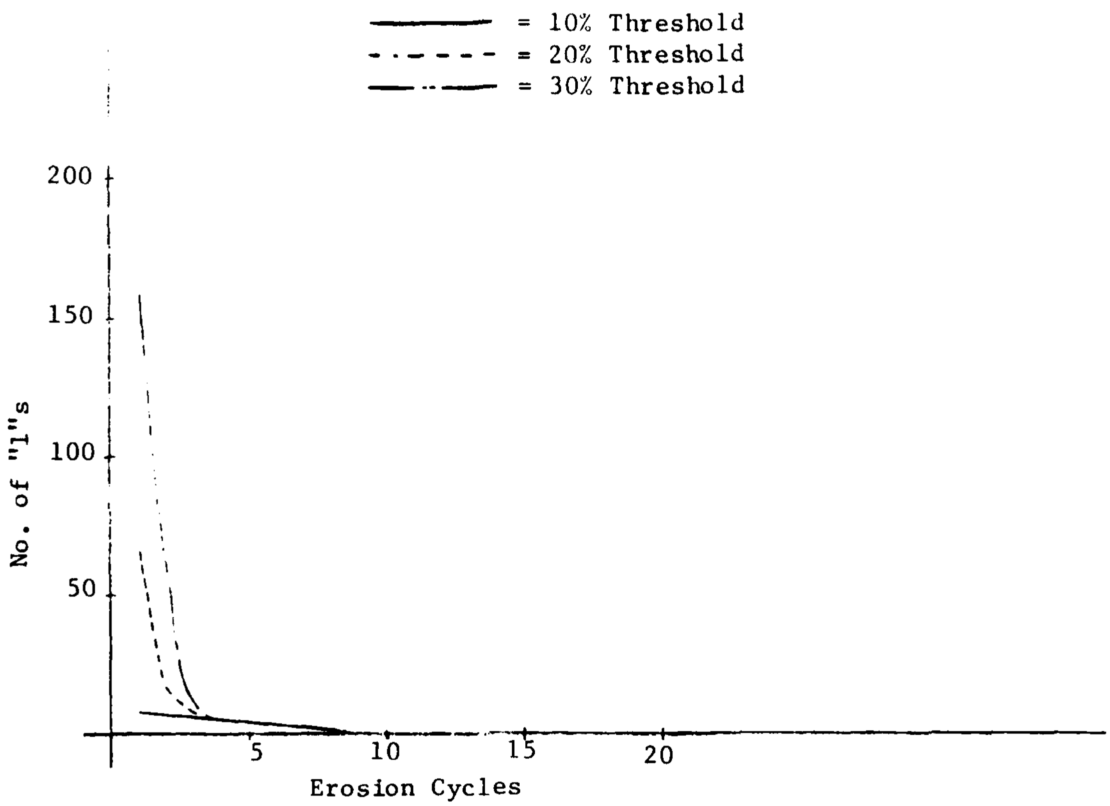


Figure 2.14 Erosion Cycles Covariance for Edge (430, 312), -- Mask

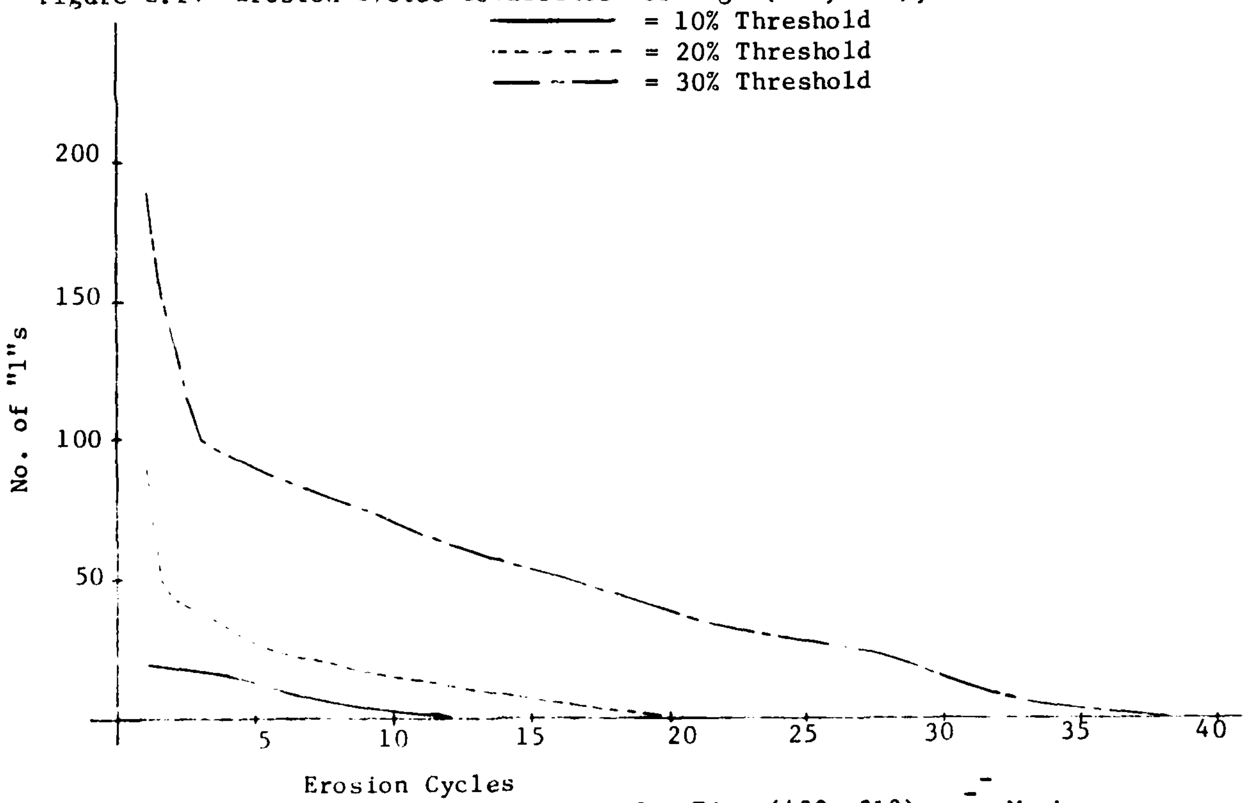


Figure 2.15 Erosion Cycles Covariance for Edge (430, 312), -- Mask

2.2.1 Conclusions

The erosion process was very good for emphasizing the basic properties of the edges considered in this study. Emphasis was placed on orientation and density properties in this study. There is no reason that periodicity of texture structures cannot be identified by using structural elements designed to accent that property. Such masks would consist of 1's separated by 0's with the blank space determining the period.

The texture images resulting from this SAR sensor seem to lend themselves very well to the texture analysis approach. One problem area is that it is an iterative process requiring sometimes many passes. One possible very good use for this process would be to use it as a preprocessing step. A few erosion cycles could be performed to accent any areas of an image that match the properties the structural element was designed to reveal. Then the resulting image could be used for further processes but the properties of interest would now dominate other qualities or features in the image. This could aid the feature selection process for representing image data.

2.3 Statistics Study

K.I. Laws[8] developed and reported on "texture energy" transforms which performed better than co-occurance statistical approaches. For a zero mean field, the texture energy measure is the standard deviation since the variance would be the average of squared signal values, an energy measure in the formal sense of the word. If the image had been previously filtered, the texture energy measures the local energy within the pass band.

Since either the variance or the standard deviation alone has been shown to be sufficient to extract texture information, a statistical study of two types of textural edges was performed. The study consisted of determining the mean and standard deviation of a local area as that area was moved across the textural edges. The edges used in this study are shown in Figure 2.2 and Figure 2.16.

Since the mean and standard deviation are local operations, the size of the local area, or window, is an important parameter. Two sizes of windows were used, a 32x32 and a 16x16 area. There was no particular reason for choosing these sizes other than they fit within the texturally distinct areas that comprised the region around the edges. For faster iterative operation, smaller sizes would be more appropriate.

The location of the window is defined by the upper left hand corner pixel co-ordinates. When the window was moved through the edges it was placed to the left of the edges and moved completely through them. This is normally done in a raster-type scan in most hardware. For an interesting excursion, a path perpendicular to the orientation of the edge was used to determine if being able to depart from the normal raster-scan method of convolving a local operator through an image could be beneficial.

Statistics were also determined for the edges after they had been filtered with a Sobel operator. The Sobel gradient is an edge detection operation. It has been used in texture discrimination studies with good results. It is a nonlinear 3x3 operator which is defined by the following masks:

$$x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

For each pixel the Sobel magnitude is determined by

$$SBL = \sqrt{x^2 + y^2}$$

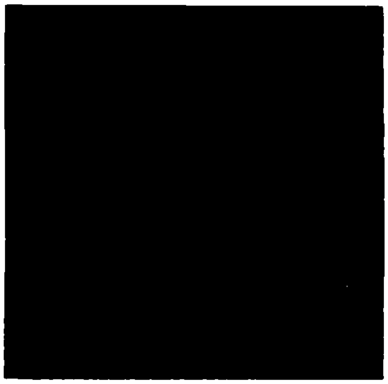
In this study, the square of the Sobel magnitude was used and the operation was labeled SOBELSQ. Figures 2.17 and 2.18 show the results of applying this mask to the edges of interest.

Figures 2.19 - 2.24 show the results of the statistical measurements. As is evident in Figure 2.19, the 32x32 mask size does not descriminate the narrow edge (430,312). This is a result of the edge comprising too small a percentage of the mask area as the mask is moved through it. Figure 2.20 shows that the smaller mask size greatly improves the edge descrimination capability. It also shows that the SOBELSQ operation provides little improvement. Figure 2.21 indicates that an appreciable improvement in locating the edge boundary can be obtained by moving the mask on a path perpendicular to the edge. This results from the increased percentage the edge has in the mask as it first enters the mask.

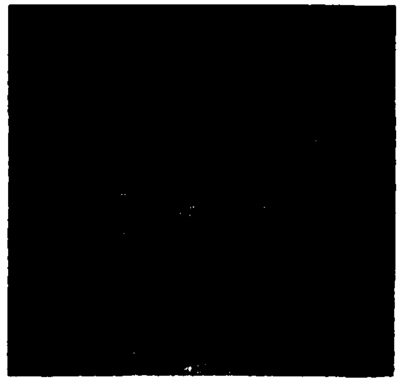
Figures 2.22 - 2.24 show the results on edge (72,372). This edge is different from the narrow edge (430,312). It has more width and, therefore, should show a double mode characteristic as the mask is passed through it. In fact, both the 32x32 and 16x16 masks do not meet this expectation, as shown in Figures 2.22 and 2.23. The standard deviation measure is enhanced by the Sobel operation in the 16x16 case. Figure 2.24 shows the double mode characteristic is obtained when the mask is moved on a perpendicular path to the edge orientation.

2.3.1 Conclusions

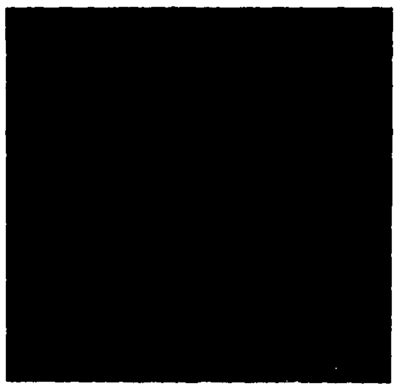
As expected, mask size and path direction are important in determining the effectiveness of a filtering operation. The usefulness of the variance and standard deviation as texture measures for these images is not completely evident from the limited results. Certainly under the right conditions of mask size and filter preprocessing, such as a Sobel magnitude operation, the usefulness of the variance as a texture measure could be enhanced. Certainly the present results do not conclusively point to the variance aa the sought for texture information indicator for these texture images. Without a priori knowledge of the texture characteristics, however, it remains an effective texture measure.



Location of Edge (72,372)
Figure 2.16



SOBELSQ Operation on Fig. 2.1
Figure 2.17



SOBELSQ Operation of Fig. 2.2
Figure 2.18

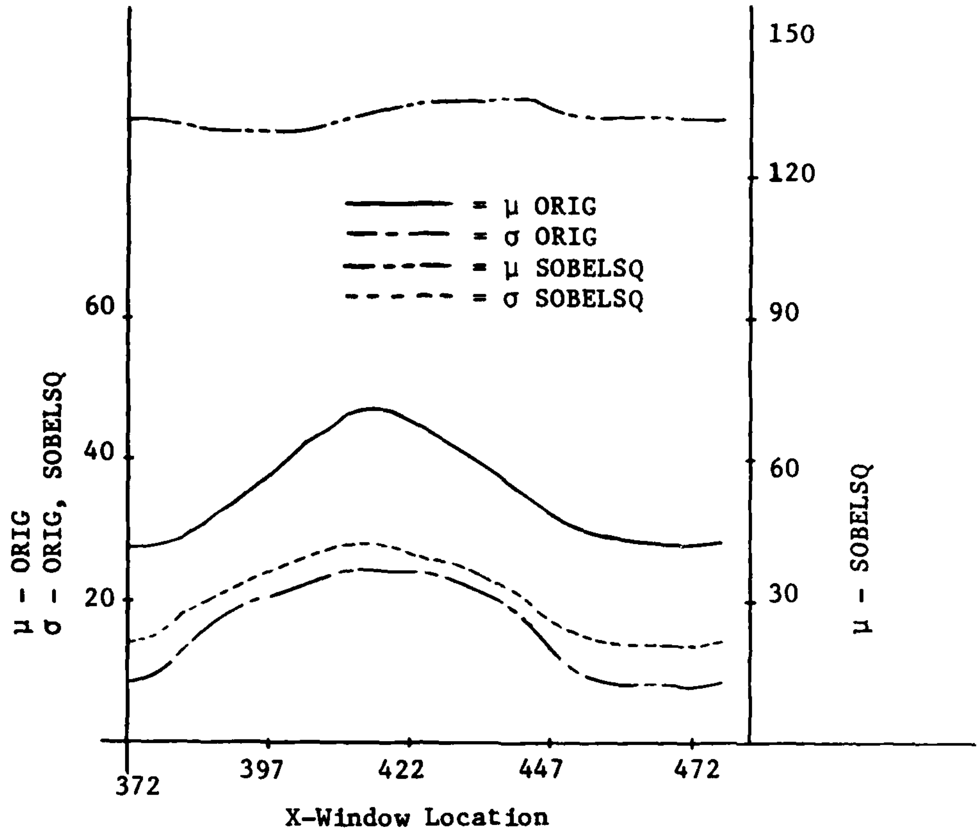


Figure 2.19 32x32 Statistics for Edge (430, 312)

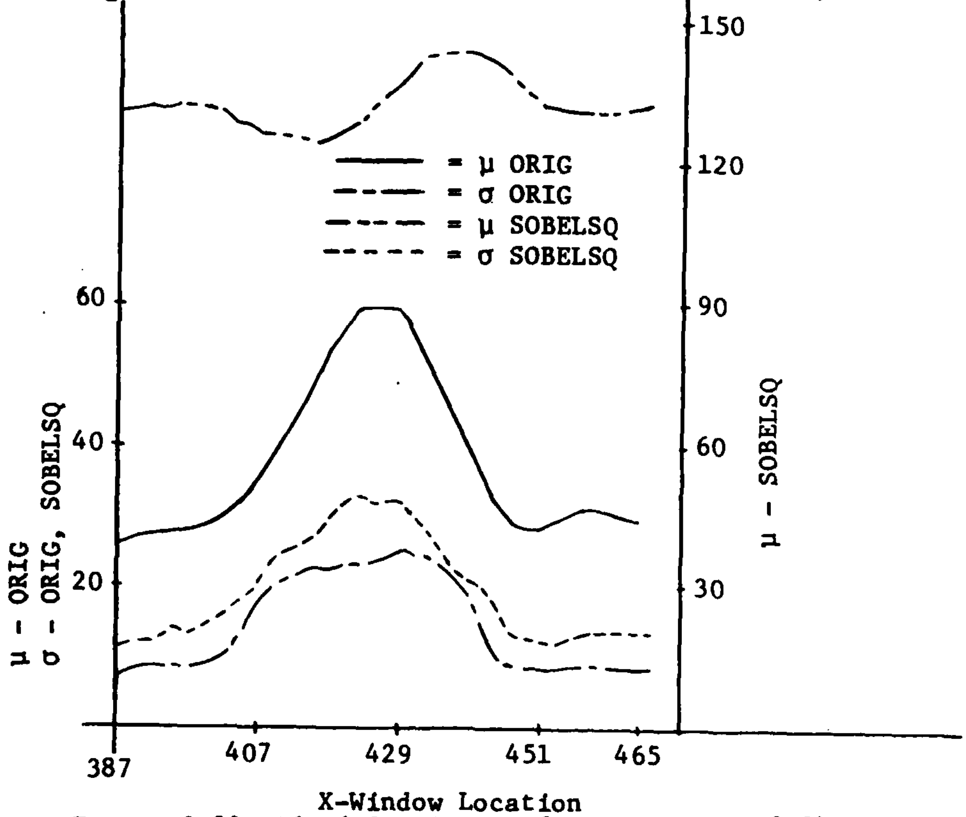


Figure 2.20 16x16 Statistics for Edge (430, 312)

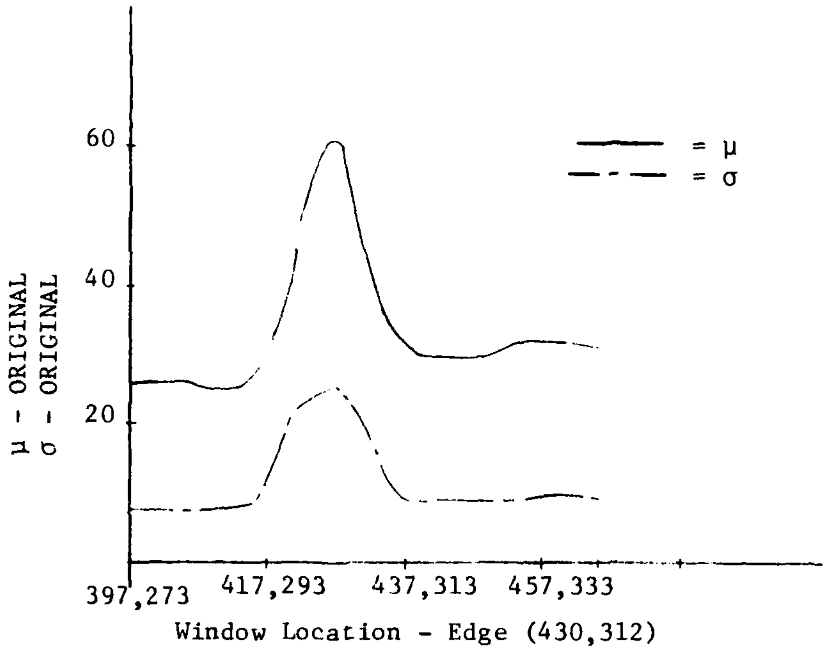


Figure 2.21 16x16 Statistics for Perpendicular Path

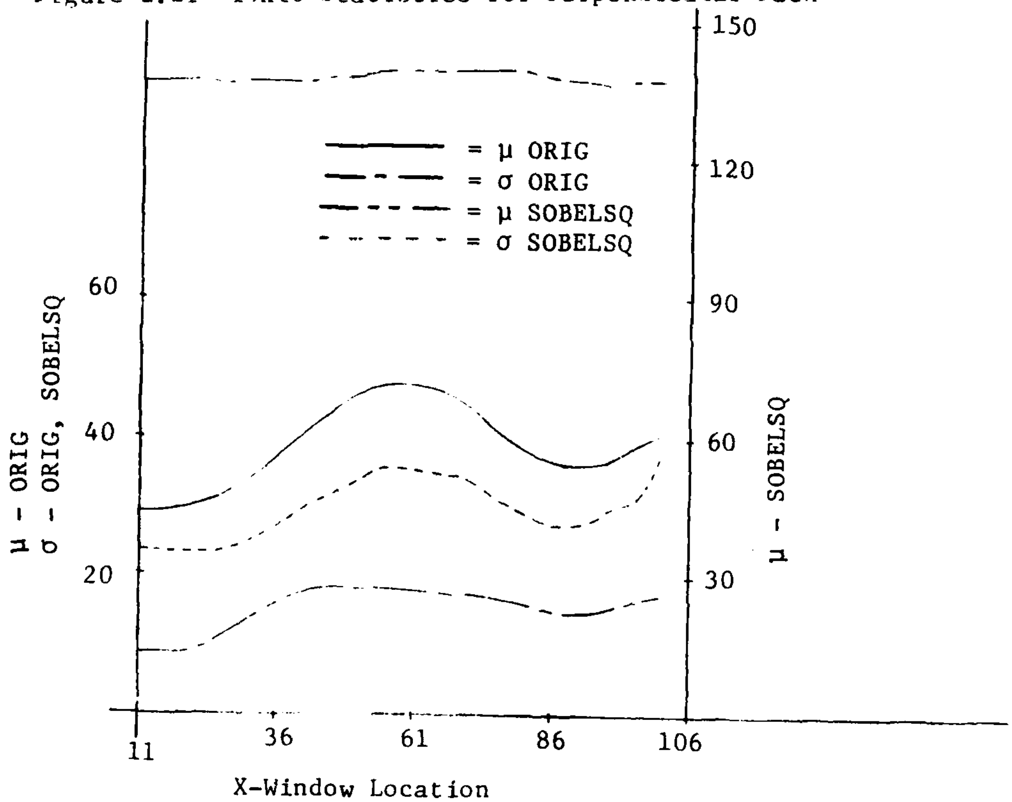


Figure 2.22 32x32 Statistics for Edge (72, 372)

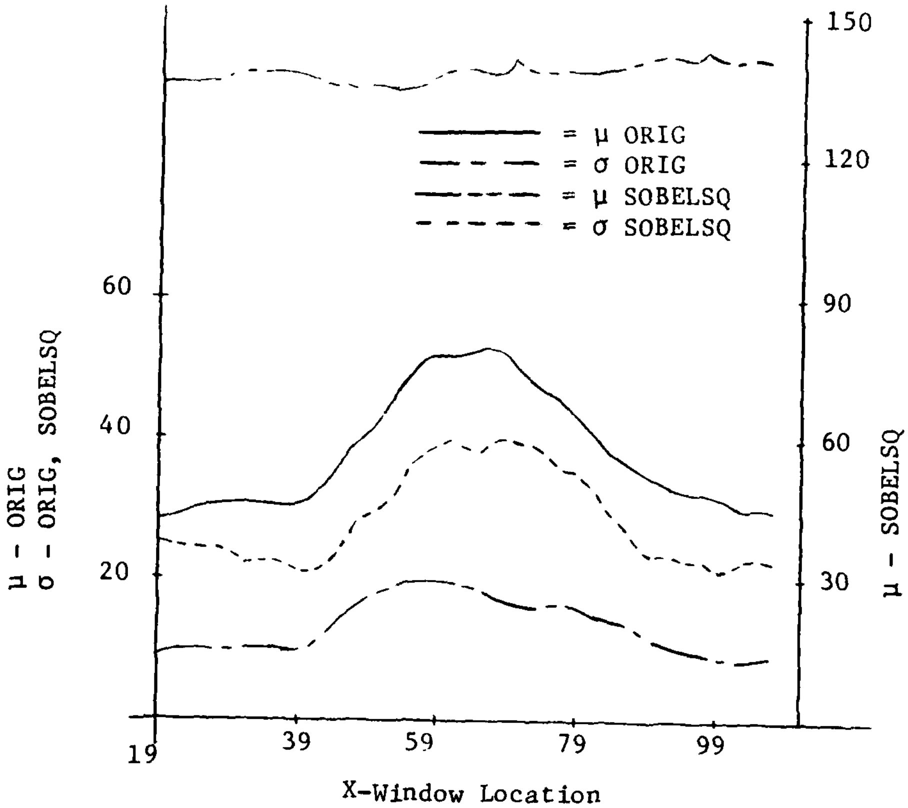


Figure 2.23 16x16 Statistics for Edge (72, 372)

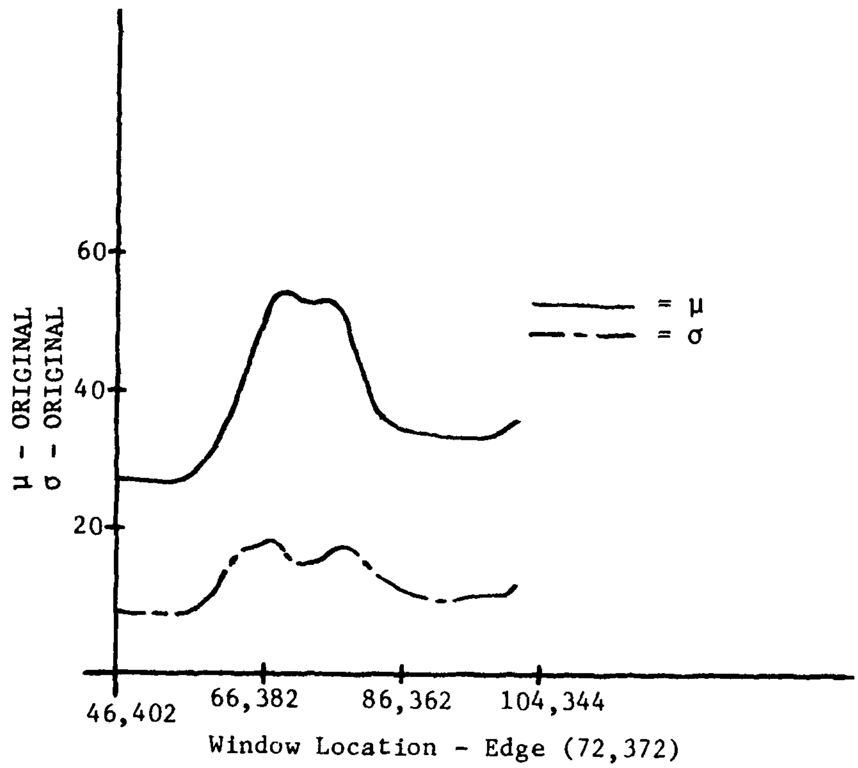


Figure 2.24 16x16 Statistics for Perpendicular Path

SECTION III. REACHABLE-SET GUIDANCE TECHNIQUE

3.0 Introduction

Guidance of short range air-launched missiles has been based largely on a static technology for the last decade. Conventional guidance of "fire and forget" weapons has been dominated by variations of proportional and pursuit navigation, founded in optimal control for non-maneuvering interception[9-13]. They require simple angle measurements, and are easily implemented in analog hardware. Within terminal saturation constraints such techniques provide adequate accuracy for scenarios involving non-maneuvering vehicles. Yet, an intelligent target capable of radical maneuvers can deceive a conventional guidance strategy by purposely inducing terminal saturation in the final seconds of the encounter.

In the presence of maneuvers, formulation of an effective guidance strategy becomes far more complex[14-18]. Control effort must in some sense anticipate target trajectories by modeling maneuver capabilities and the target response, both deterministic and random, to the closing missile. Stated in this framework, the problem reduces to a differential games formulation, defining optimal evasion/pursuit strategies[19]. Yet, the solution, even for very contrived scenarios, leads to a significant numerical burden, unsuitable for practical usage.

While such an elaborate approach will be of doubtful value in this context for some time to come, it does serve to indicate that onboard intelligence can be used to greatly improve a missile's advantage over its adversary. Indeed, the capabilities of digital hardware have matured to the point where serious consideration must be given to advances in guidance strategy over the conventional techniques. Of particular value would be a study of the tradeoffs possible between level of intelligence (i.e., hardware capabilities) and overall missile performance (i.e., miss distance and aspect angle). An extensive evaluation of this nature would illuminate key factors necessary for the development of future weapons delivery systems. By reducing computational requirements to a common denominator, i.e., currently available hardware, practicality with respect to physical size limitations can be assessed; this also allows extrapolation of practicality into the future with projected advances in microelectronics. Of course, it is the performance of any given control strategy that ultimately determines if the necessary hardware is warranted.

The work effort described by this report does not attempt such a survey, but rather, a small fraction of such a study was conducted; a single promising guidance technique was examined and compared with benchmark simulation tests of conventional guidance in several encounter scenarios. Then, using currently available technology, the architecture of the required hardware was examined. The results indicate that using current technology, substantial improvements in missile performance can be realized. Of course, this is simply a single point of the overall study, and does not pretend to proclaim the best currently available technique.

The subject of this evaluation is a guidance law developed in Reference 20, based on the concept of reachable set theory for dynamic systems[21]. It was chosen because it is representative of a class of guidance strategies that could be of immediate value, i.e., it (1) has an intuitive structure, (2) calls for moderate computation in the form of a systematic search, and (3) is suited for a sampled data context. Variations can be appended to the basic law to adapt it to other types of encounters by using a cost function based on physical limitations[22-23].

The following sections present details regarding: Encounter Model (Section 3.1), Conventional Guidance Implementation (Section 3.2), Advanced Guidance Implementation (Sections 3.3 through 3.5), and Observations of Performance From Simulations (Section 3.6).

3.1 Encounter Model

Before discussing the guidance techniques in detail, it is necessary to describe the model of the target and missile behavior. For the most part, the assumptions and numerical values used in reference 20 were drawn upon. The target was allowed a constant forward speed of 1000 ft/sec; maneuver-induced drag was assumed compensated by forward thrust. The maximum turn was governed by a 6 G constraint on normal acceleration. Due to the short duration of the close-range encounter, the target was constrained to maneuver in a single plane having a "tilt" angle δ with respect to its initial velocity vector, as shown in Figure 3.1.

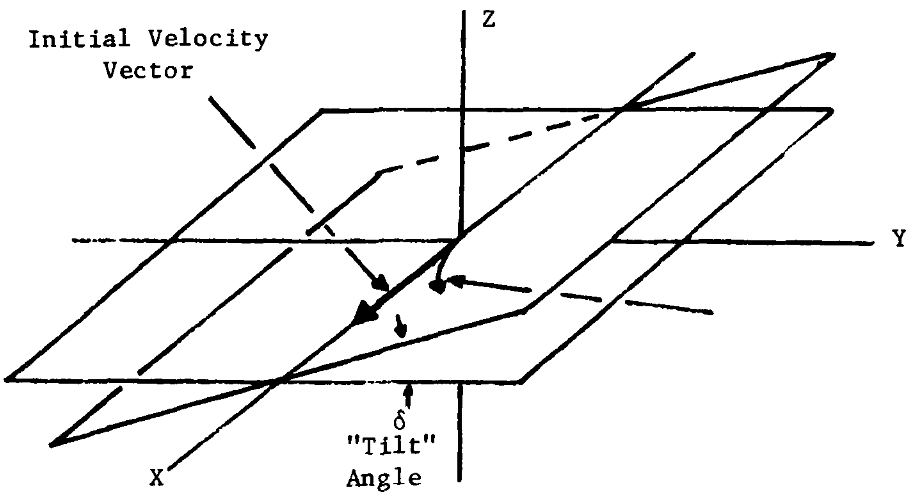


Figure 3.1 Maneuver Plane Definition

The missile model was somewhat more complex. It was assumed to have an initial speed of 1000 ft/sec, launched from its host aircraft. The thrust was given as 4700 lb, with a burn tiem of 2.6 sec. The fuel load was 50 lb of the initial 165 lb. Guidance was by means of a normal lift vector, magnitude $a_{\perp}$, at an angle σ with respect to the missile body as shown in Figure 3.2.

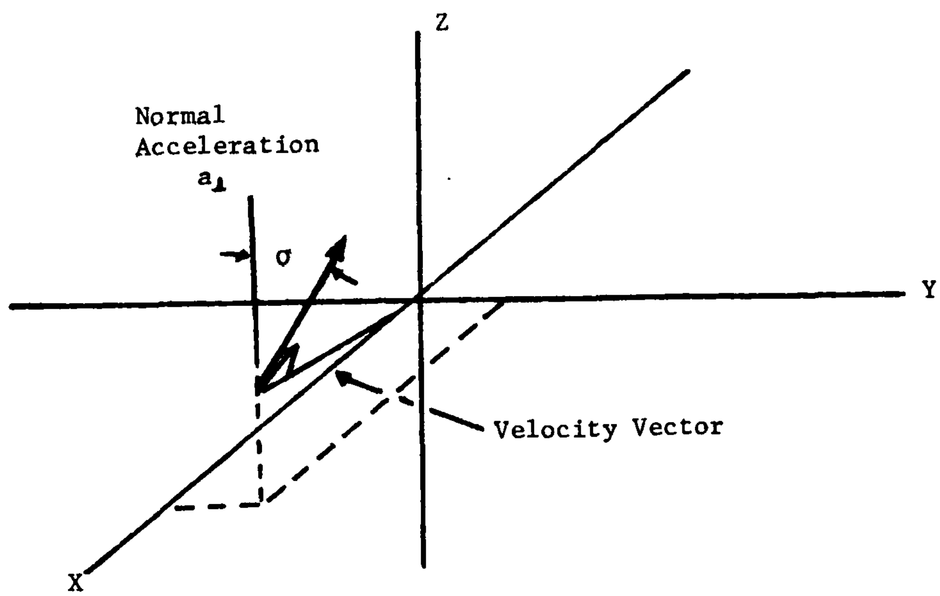


Figure 3.2 Missile Acceleration Vector

In the interest of practicality, a parabolic drag force law was used[20]

$$D = K_1 \, v^2 C_{DO} + K_2 \, \frac{(a_\perp \, w/g)}{v^2} \, C_L$$

where

C_{DO} = 2.3 - zero lift drag coefficient

C_L = .0025 = induced drag coefficient

K_1 = proportionality factor = .001

K_2 = proportionality factor = 1000

w = missile weight, function of time

g = gravitational acceleration

v = missile speed

Scenarios were defined by the relative positions of the target and missile and their respective velocity vectors. The target was then allowed a maneuver strategy bounded by 6 G's in turn rate, at a fixed angle of tilt. Missile guidance was done by choice of lift vector as a function of closure.

3.2 Implementation of Conventional Guidance

For the purposes of simulation and benchmark evaluation, conventional guidance was represented by two techniques; proportional navigation and pursuit guidance. Linear combinations of their respective components can be used where the weighting constants may be functions of range, i.e., favoring pursuit initially and proportional on final approach[24]. Here, each was used in its purest form.

3.2.1 Proportional

The magnitude of the normal acceleration for proportional guidance is given as

$$a_{\perp}(t) = C\,|\dot{\theta}_{LOS}(t)V_c|$$

where

$\dot{\theta}_{LOS}(t)$ = rotational rate, with respect to inertial space, of the line of sight angle to the target

V_c = closing speed

C = navigation constant (normally between 3 and 6)

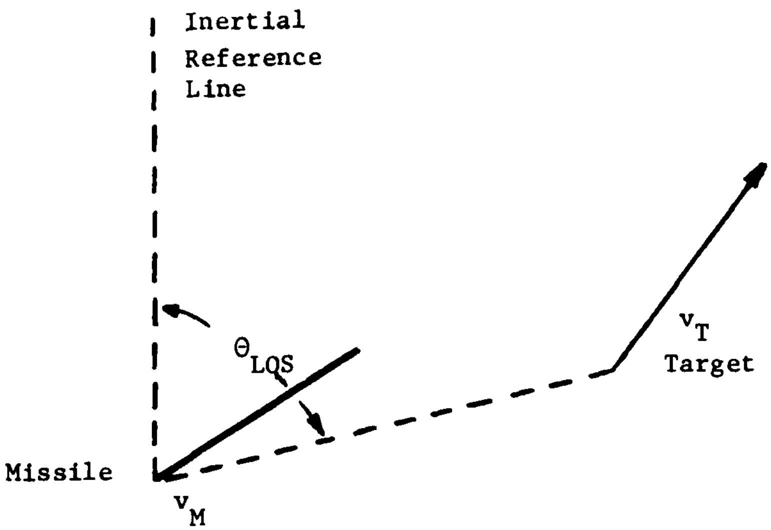


Figure 3.3 Proportional Guidance

In three dimensions, the angle of the missile acceleration orientation is chosen to rotate the missile velocity vector toward the target's relative displacement vector.

While in practice this calculation is done in analog hardware, for simulation purposes the necessary derivative was approximated by a sampled-data finite difference.

$$a_{\perp}(kT) = C \left| \frac{\theta_{LOS}(kT)-\theta_{LOS}((k-1)T)}{T} \; V_c \right|$$

where T was a small sampling interval. At each such iteration, a new control effort was computed and used to drive the system's dynamic equations.

3.2.2 Pursuit

For this second case the magnitude of control is given by

$$a_{\perp}(t) = C|\theta_{LOOK}(t)|V_c$$

with orientation σ chosen as in the previous case. The angle θ_{LOOK} is measured from the missile body axis to the target location.

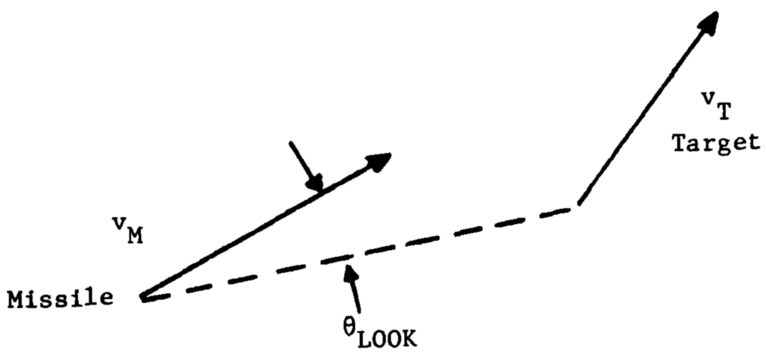


Figure 3.4 Pursuit Guidance

Again, this angle was sampled at uniform intervals for the purposes of simulation.

As mentioned earlier, such techniques have long been used with success. They prove adequate for launchings where terminal saturation is avoided due to sluggish target evasion or close range. The principal advantages lie in its means of implementation; the measurements required are simple, i.e., only a reasonable guess at closing speed and an accurate estimate of the displacement angle. The simple measurements and their associated sensors, coupled with the analog control hardware, make such navigation schemes attractive from the point of view of cost and physical size.

Yet, because these schemes are based on non-evasive targets, they tend to de-emphasize any initial launch advantage, and prefer to postpone offensive counter-maneuvering until late in the scenario. That is, since the controller does not expect changes in the target trajectory, it responds only after a maneuver becomes evident at the angle sensor. This is often too late for adequate course correction, and invariably results in terminal saturation and miss distance dependent on the agility of the target. One means of dealing with this terminal miss effect is to increase the warhead size and the kill radius.

3.3 Advanced Guidance

The terminal effects associated with conventional guidance are largely responsible for the interest in more sophisticated navigation techniques capable of anticipating and responding to target maneuvers. A missile launched with a high kill probability means that interception is highly likely for all valid target maneuvers. That is, the target's position in space and time will be contained in the set of all points reachable by the missile; as time-to-go decreases, this reachable set shrinks. The guidance controller must maintain the initial advantage by anticipating target attempts to exit the missile's shrinking reachable set. For this, the controller should examine all valid target trajectories and respond as if the worst one (from the missile's viewpoint) were to be used.

Clearly, this is an ill-posed problem which can be rendered tractable by the quantitative observation in reference 20. Using a differential games analysis of this framework of assumptions, it can be shown that the target maximizes time-to-go when its maneuver is a maximal acceleration turn. For our purposes, the tilt angle δ of this maneuver is unimportant; a complicated function of relative position and attitude. Using this observation, a tractable guidance scheme can then be implemented. In brief, the controller determines the target's trajectory, assuming the target makes its maximum turn at some selected tilt angle δ. Upon examination of a set of such angles, the missile responds as if the target were to choose the "worst case", i.e., anticipating the optimal maneuver for the target. After a brief interval, e.g., determined by computational requirements, the process is repeated using updated position and attitude information, yielding a "closed-loop" implementation.

The detailed operation, including mathematical particulars, is as follows:

Given observations of relative displacement, missile and target heading and speed, assuming a relative coordinate system to be described later, then

a. the controller assumes that the target chooses a maneuver plane defined by δ and uses its optimal maneuvers to maximize time-to-go.

b. Next, the necessary missile heading coordinates (ϕ , θ) to effect an intercept are computed. This is done using the following time function expressions for relative Cartesian displacement:

$$\Delta x(t) = \Delta x(0) + V_m t \ \cos(\phi) \cos(\theta) - \frac{1}{a_t} \sin(V_t t / a_t)$$

$$\Delta y(t) = \Delta y(0) + V_m t \ \sin(\phi) \cos(\theta) + \frac{1}{a_t} \cos(\delta) [\cos(v_t t / a_t) - 1]$$

$$\Delta z(t) = \Delta z(0) + V_m t \ \sin(\phi) + \frac{1}{a_t} \sin(\delta) [\cos(v_t t / a_t) - 1]$$

where

V_t = target speed

a_t = target turn rate (maximum)

V_m = missile speed

The closed form integration results by assuming constants for speed, maneuver angle, and heading angle. At the interception time t_f

$$\Delta x = \Delta y = \Delta z = 0$$

By defining $\ell = V_m t_f$ as the distance traveled by the missile, both ϕ and θ can be eliminated to yield a single scalar equation in the time-to-go:

$$\begin{aligned} f(t_f) = \ell^2 &- [\Delta x(0) - \frac{1}{a_t} \sin (V_t t_f / a_t)]^2 \\ &- [\Delta y(0) - \frac{1}{a_t} \cos(\delta) (\cos(V_t t_f / a_t) - 1)]^2 \\ &- [\Delta z(0) - \frac{1}{a_t} \sin(\delta) (\cos(V_t t_f / a_t) - 1)]^2 \end{aligned}$$

This can be solved for the positive root t_f using a Newton-Raphson search; then the actual missile heading can be found using closed-form evaluations:

$$\phi = f_\phi(t_f) = \sin^{-1} \left[\frac{[\Delta z(0) + \frac{1}{a_t} \sin (\delta) [\cos(V_t t_f / a_t) - 1)]}{V_m t_f} \right]$$

$$\theta = f_\theta(t_f) = \cos^{-1} \left[\frac{[\Delta x(0) + \frac{1}{a_t} \sin (V_t t_f / a_t)]}{V_m t_f \cos (\phi)} \right]$$

At this point, the velocity vector that the missile should have for interception is known, (V_m, ϕ, θ), given the specific target maneuver angle δ.

c. Given the actual missile heading, the acceleration necessary to yield an average velocity vector of (V_m, ϕ, θ) over the interval $(0, t_f)$ is approximated by

$$a_{\perp}(\delta) = \frac{\Delta V}{t_f} = \frac{2V_m \S}{t_f}$$

where § is the angle separating initial and desired velocity vectors. This relationship is depicted in Figure 3.5 and represents the control effort necessary to respond to a specific maneuver.

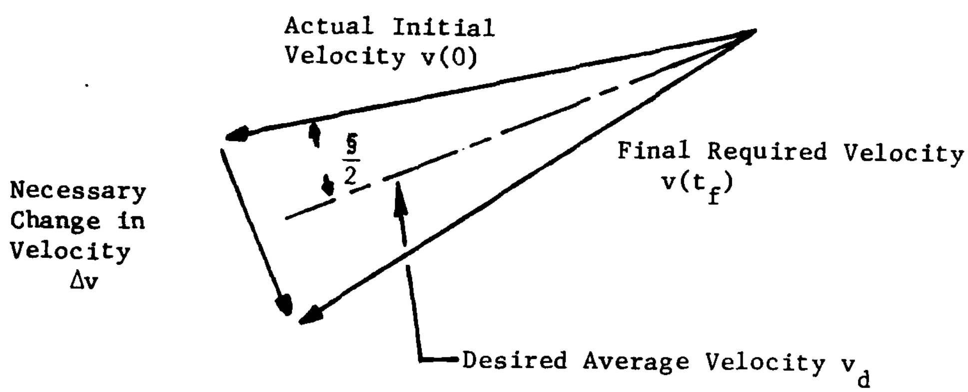


Figure 3.5 Required Missile Velocity Change

d. Conceptually, $a_{\perp}$function a (δ) exists, giving the necessary missile control effort as a function of target maneuver angle δ. The maximum of this function defines the worst case evasion that the target can choose. Hence, the missile controller anticipates this as the maneuver, and responds with normal acceleration $a_\perp^*$, oriented at the angle of vector ΔV defined above. The computation involved will be detailed in the next section.

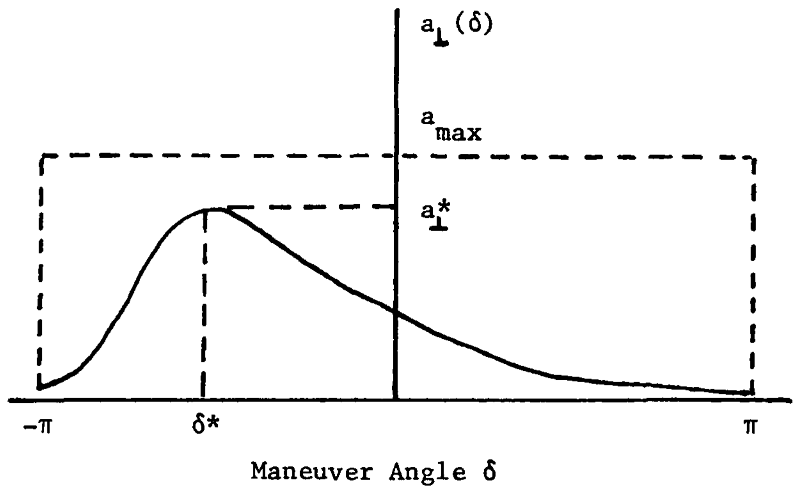


Figure 3.6 Missile Acceleration Function

Note that an interesting feature emerges for such an analysis. At any given instant, the function $a_\perp(\delta)$ summarizes the advantage that the missile has over its target. That is, if a means of successful evasion exists, then for some angle δ, $a_\perp(\delta)$ exceeds the maximum allowable normal acceleration of the missile. Such information could be particularly valuable to a pilot if presented as a "probability of hit" measure.

While computation and architecture will be discussed in subsequent sections, one consideration must be mentioned here. This guidance scheme in its closed-loop form must actually be updated in a continuous fashion.

Yet due to computational operations, it becomes sampled data control, with the update or sampling interval determined by the necessary computation.

3.4 Algorithm Requirements

In this section, the requirements of the advanced reachable-set based guidance scheme are discussed. First, an overall description of the software is presented, addressing the structure of the Fortran listing of GUIDE found in Appendix A. Then, using findings from tests using this software, currently available hardware is evaluated.

A simplified flowchart in Figure 3.7 is included here to aid in the description of the Fortran listing in the Appendix. Upon first entry, initialization steps are encountered, allowing the user to set interactively a number of options and parameters. At run time, this section (lines 34-74) is skipped, and actual control computation proceeds as follows:

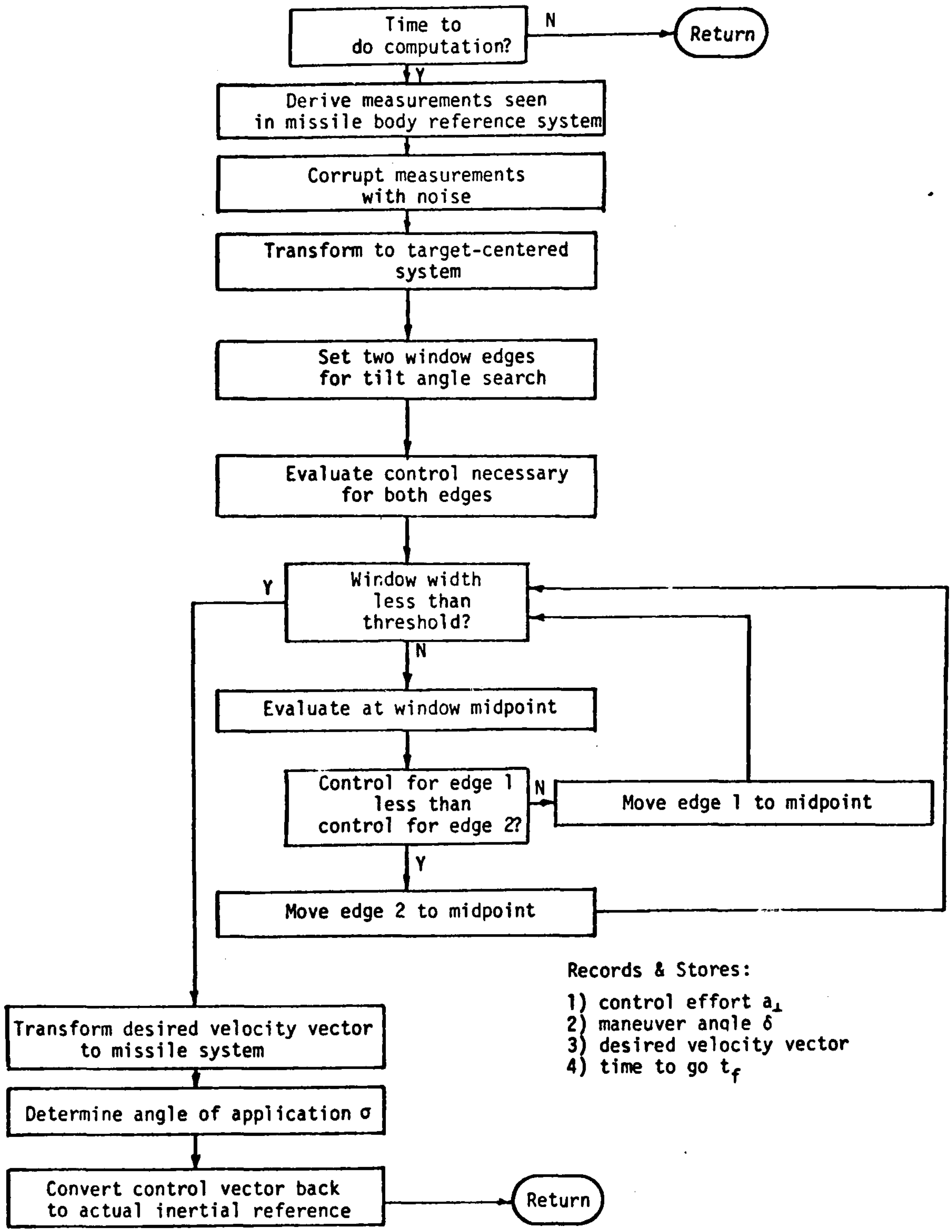


Figure 3.7 Flowchart

1) First, the elapsed time from the previous control calculation is checked against the specified update interval. If insufficient time has passed, there is an immediate return back to the calling program. The control effort previously determined is maintained.

2) When the update interval has elapsed, a new control effort is determined from current measurements. Passed to the routine are the actual states of missile and target with respect to an inertial reference. The pairs of six-dimensional vectors specifically contain (X, Y, Z) position values, followed by spherical velocity information (V, ϕ, θ), i.e., speed, azimuth and elevation of the velocity vector. These are transformed by linear algebraic rotation to a missile-centered coordinate system. Use of this coordinate system as a basis for necessary measurements eliminates the need for a strapped-down inertial reference in the missile. Assuming the turn rates are slow with respect to the update interval, only moderate degradation in the overall performance is experienced.

The new coordinate system is defined by an X-axis in the direction of the missile axis, and its positive Y-axis lying in the plane of the target point.

3) The five measurements in this system (range, target azimuth, target speed, velocity vector azimuth and elevation) are appropriately corrupted by random measurement noise.

At this point the target measurements have been conditioned as if the missile sensors had gathered them; i.e., they represent the information available from sensors having only the missile and target as directional reference. Thus, the code in the subroutine to this point (line 100) is simply overhead computation. Actual control computation made by the missile begins at line 130. Certain sections of the code associated with the missile are also overhead, performing certain initialization computations. As such, they are executed only one time per pass so efficient coding was not felt necessary. The computation bound loops, however, must be studied for improvement in efficiency before actual implementation.

4) To begin, the missile-centered measurements were transformed to a target-centered system, defined analogously to the missile-centered system. This step is necessary due to the problem formulation described in Section 3.3. This is done in lines 130 through 165.

5) At this point, the search for the worst case control effort begins. A window is defined straddling the angle δ^* determined as the target's worst maneuver at the previous update time. In practice, assuming a sufficiently fast update rate, the tilt angle of worst maneuver changes slowly, so will remain within such an interval. (For the simulations conducted, an interval of 30^o was used.) For the two extreme maneuver angles, $\delta_1=\delta^* - \frac{\Delta}{2}, \delta_2=\delta^* + \frac{\Delta}{2}$ the solution is found for the necessary missile velocity vector direction to effect an intercept. As described earlier, this is done using a Newton-Raphson search for the time-to-go, t_f, and then solving a closed-form expression for ϕ and θ. Assuming good starting values (the previous value found for t_f is adequate) convergence will require only two or three iterations. For each window edge, the necessary control effort $a_\perp$ is found, again described in Section 3.3.

6) The process of search and evaluation is repeated using the window's midpoint.

7) The smaller of the control efforts computed for the window edges is determined and the corresponding tilt angle abandoned in favor of the midpoint angle. That is, the window is halved by rejecting the maneuver angle that represents the lesser threat. The control effort as well as the desired velocity vector parameters are stored for the new window edge.

8) The new window width is checked against a prespecified threshold. If it exceeds the threshold, the binary search continues by repeating step (6) above. If it is indeed less, the maximization of control effort is complete.

9) At this point, all pertinent information about the worst possible maneuver is available. Specifically, this includes the time to intercept, t_f, the required control effort, $a_\perp$, and the desired missile velocity vector, (V, ϕ, θ). The vector is transformed back to missile-centered system.

10) Finally, the angle of application for the control vector is determined. At this point the missile has sufficient information to determine the necessary control surface deflections to generate its acceleration vector.

11) The last section of the subroutine (lines 297 through 309) simply map the acceleration vector back to the inertial system, for compatibility with the calling program.

3.5 Computational Requirements

The proposed algorithm is heavily arithmetic bound. As a result, computational requirements are easily estimated since the time required for floating-point arithmetic will be predominant and will be a good estimate of total computation time required. Additionally, software emulated floating-point arithmetic would obviously not be acceptable.

The algorithm was divided into its major parts as shown in the block diagram of Figure 3.8 and computational times estimated for each part. An 8086 microcomputer operating at 5 MHz with an auxiliary 8087 floating-point processor was used for all time estimates. Estimates were made by counting floating-point operations (including load and store) in the original Fortran program and multiplying by the appropriate 8086/87 instruction time. Pessimistic estimates were made at all times.

In Figure 3.8, the start-up search was ignored since it is only made once and contributes little to the computational problem's dynamics. Times for each of the subprocesses are shown in Figure 3.8. The total computational time required by the 8086/87 processor is given by:

$$37{,}877 + 14{,}586\ N + 9{,}304\ M + 7{,}293\ M\ N$$

where N is the number of Newton iterations needed to calculate the direction of the velocity vector for intercept and M is the number of binary search passes required to search for the maneuver angle. Typical values are three Newton iterations and M=4 corresponding to dividing the maneuver angle search window into 16 segments. For these values, the total computational time is approximately 206.4 milliseconds, about an order of magnitude slower than desired.

3.5.1 Trigonometric Look-Up Table

The algorithm was searched for significant opportunities for improvement. Although the 8087 floating-point processor was used for calculating a variety of functions, the sine and cosine functions were especially prevalent and time consuming. The 8087 calculates these two functions from the tangent function via trigonometric identities. Another time estimate was calculated using the 8087 for all calculations except for sine and cosine evaluations. These latter functions were assumed to be stored in a look-up table in read-only-memory. The results of this estimate are also shown in Figure 3.8. The total computation time required using the 8086/87 and a look-up table for sine and cosine is given by:

$$17{,}893 + 5{,}766\ N + 4{,}024\ M + 2{,}883\ M\ N$$

For the same conditions, N=3 and M=4, as in the previous example, total computation time was approximately 85.9 milliseconds. Although a significant improvement over the original estimate of 206.4 milliseconds without the look-up table, further reduction of computational time was desirable to improve the performance of the algorithm.

3.5.2 Multiple Processors

The next logical step was to attempt configurations of multiple 8086/87 processors. Simulation showed that end-to-end computational delay was critical and not sampling rate. Thus, pipelined processor configurations were ruled out as not reducing end-to-end delay but only increasing sampling rate. Parallel computation was clearly necessary. Examining Figure 3.8, most computation is clearly required in setting up the search window and searching for the maneuver angle. Fortunately, each of these two tasks could be configured to allow parallel processing. Each edge of the search window could be found independently. Each half of the search window could be searched independently. Figure 3.9 is the block diagram for the implementation of the algorithm. The time estimates in Figure 3.9 assume two completely independent 8086/87 systems, each with its own sine and cosine look-up table. These systems are loosely coupled. The total computational time for the dual processor system is given by:

$$12{,}554 + 2{,}883\ N + 2{,}012\ M + 1{,}442\ M\ N$$

Again, for N=3 and M=4, the total computation time is approximately 46.6 milliseconds, a reasonable performance for this algorithm. As an added bonus, the second processor could be used for sensor and actuator conditioning as shown in Figure 3.10. The times when the second processor would not be needed by the control algorithm are exactly these times when data are input and output.

3.5.3 Future Refinements

Clearly, all three performance estimates are encouraging. Even the single processor estimate of 206.4 milliseconds is sufficiently fast that a newer-generation processor will be able to reduce this time to an acceptable value. Cost and performance figures, normalized to a single processor system without look-up table, are shown in Table 3.1. Notice that although the dual processor system's performance is greater, its cost performance ratio is actually greater than that of the single processor with look-up table. A very desirable investigation would be to actually code and test this algorithm on an 8086 with 8087 floating-point processor and look-up table for sine and cosine. It is likely that an improvement of a factor of 2 over the pessimistic estimates could be found. If the performance of the single processor system could be so improved, the difficulties and costs of the dual processor could be avoided. In any case, the actual implementation of the single processor system would provide more accurate data on which to base performance estimates of other systems.

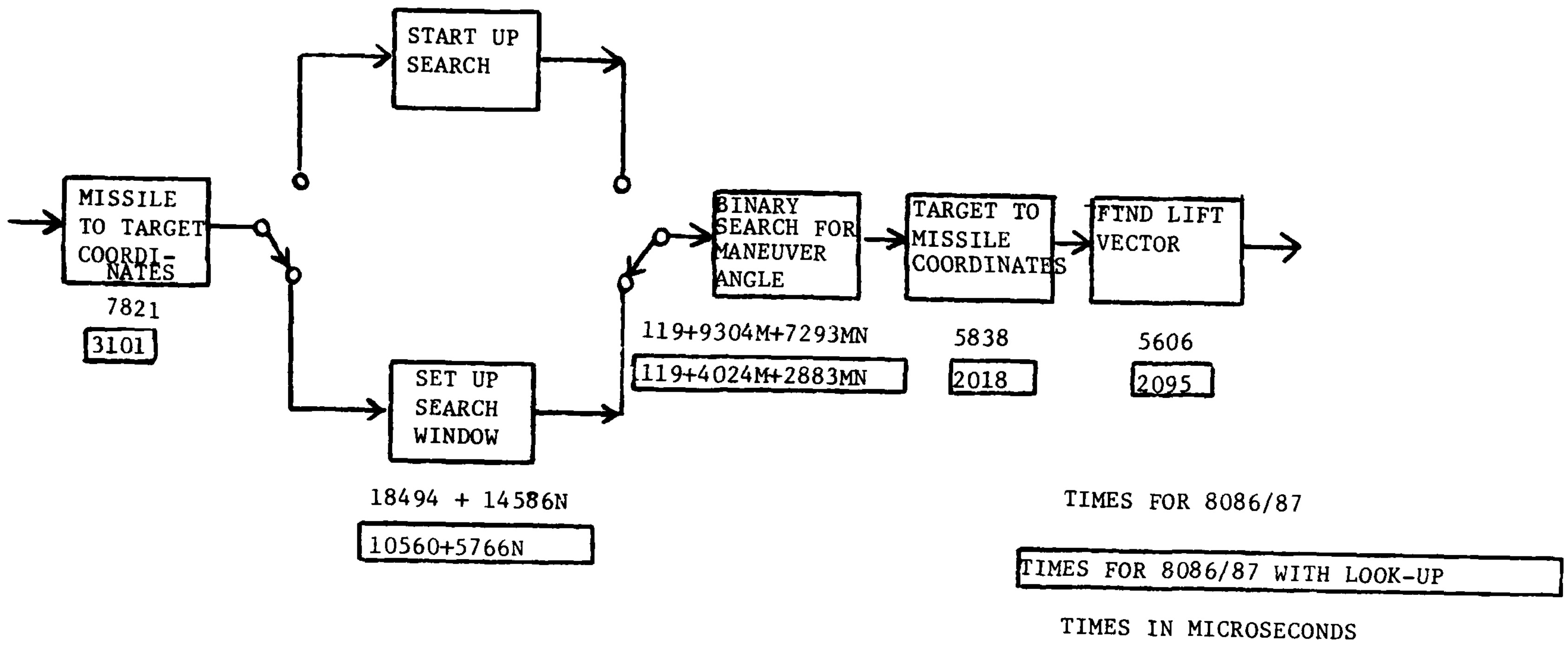


Figure 3.8 Single Processor Solutions

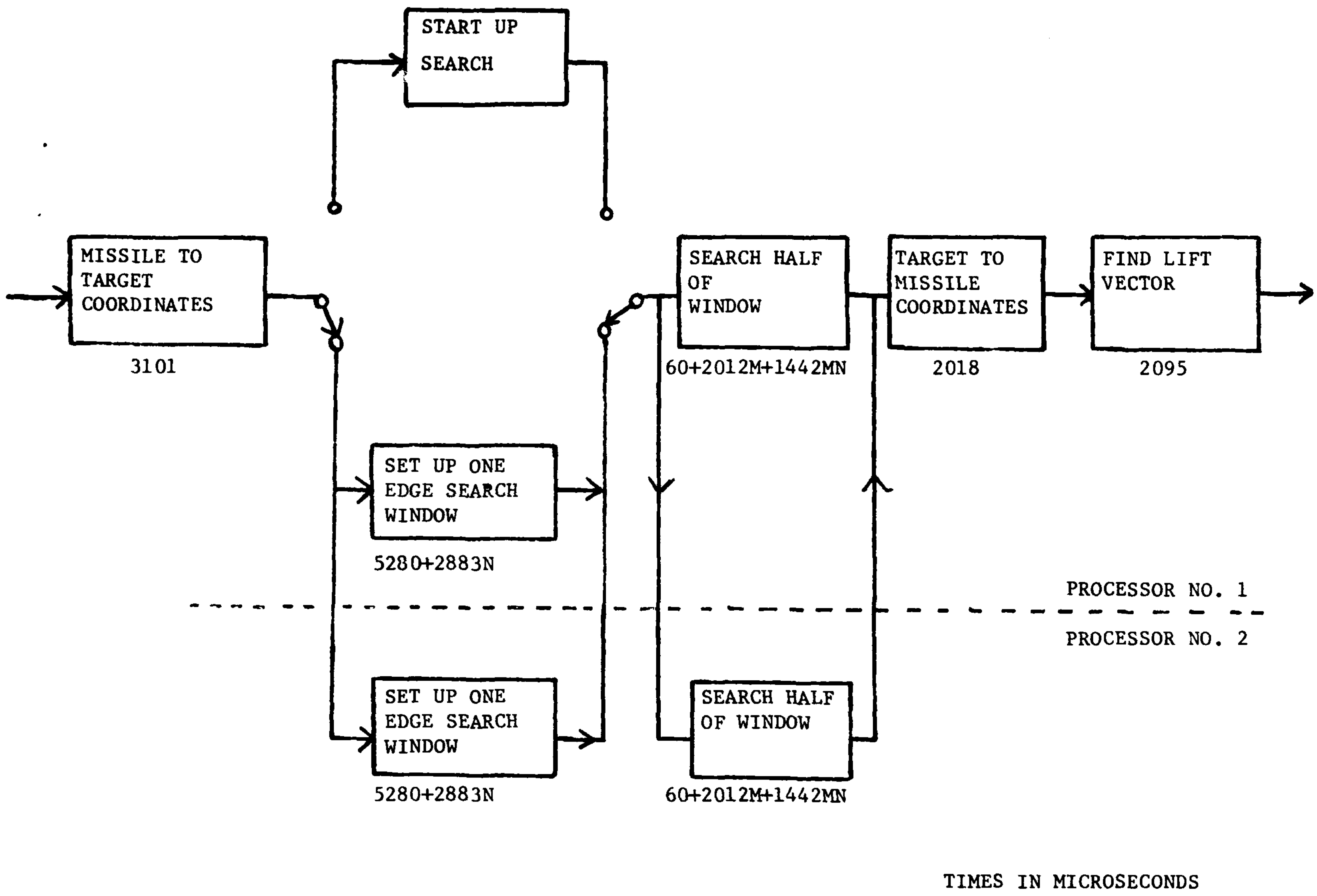


Figure 3.9 Dual Processor Solution

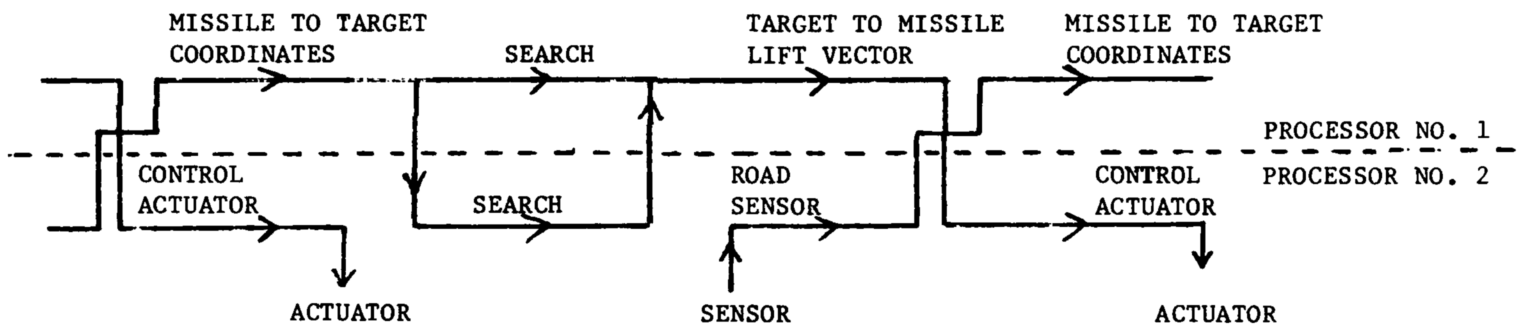


Figure 3.10 Additional Uses for Second Processor

Table 3.1. Cost and Performance of Architectural Alternatives

SYSTEM	PERFORMANCE	COST	COST/PERFORMANCE
8086/87 Processor	1.0	1.0	1.0
8086/87 Processor with Look-up Table	2.4	1.1	0.46
DUAL 8086/87 Processors with Look-up Table	4.4	2.2	0.5

3.6 Performance

The achievable performance of an advanced guidance technique determines if the cost of additional sensors and computational hardware is warranted. For the sake of comparison with conventional guidance, several classes of tests were simulated using the target and missile models described earlier. In this way, the robustness of the reachable set approach could be assessed by systematically degrading the assumptions and measurements entering into its formulation. Two representative encounters were used, shown in Figure 3.11: Scenario two involved a rear attack with the target maneuvering by pulling up at two o'clock, i.e., a low crossing rate. Scenario five dealt with a side attack on a target maneuvering as before, involving a high crossing rate.

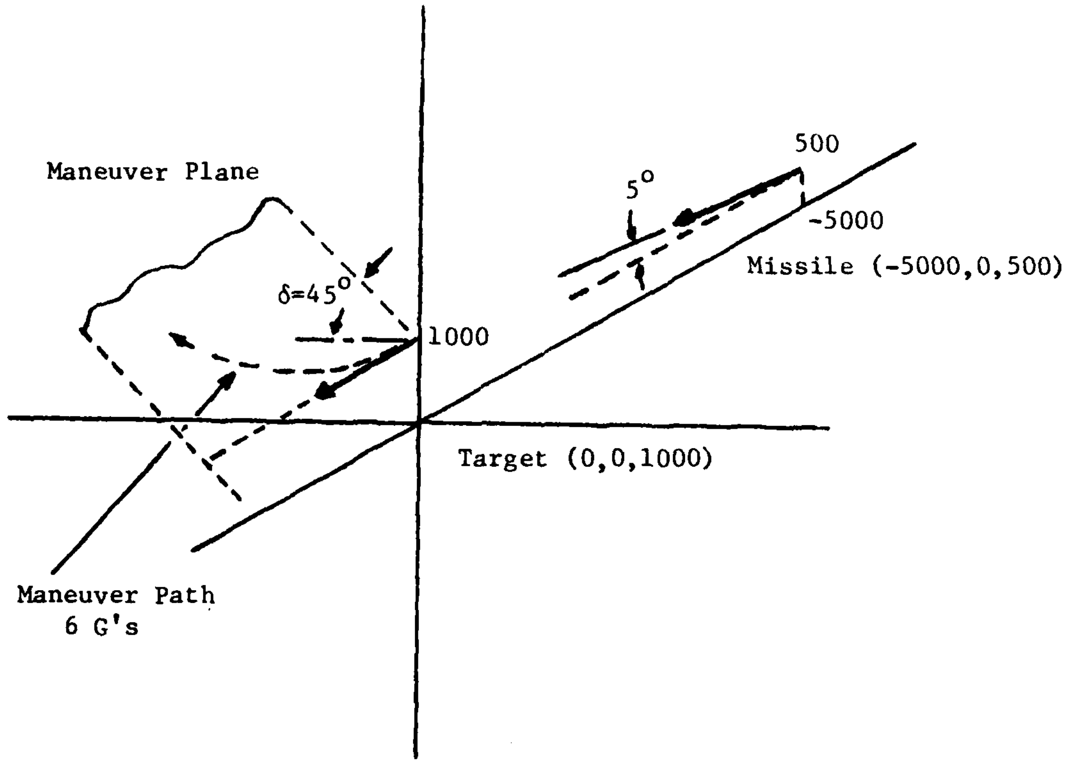


Figure 3.11a Scenario 2: Rear Attack

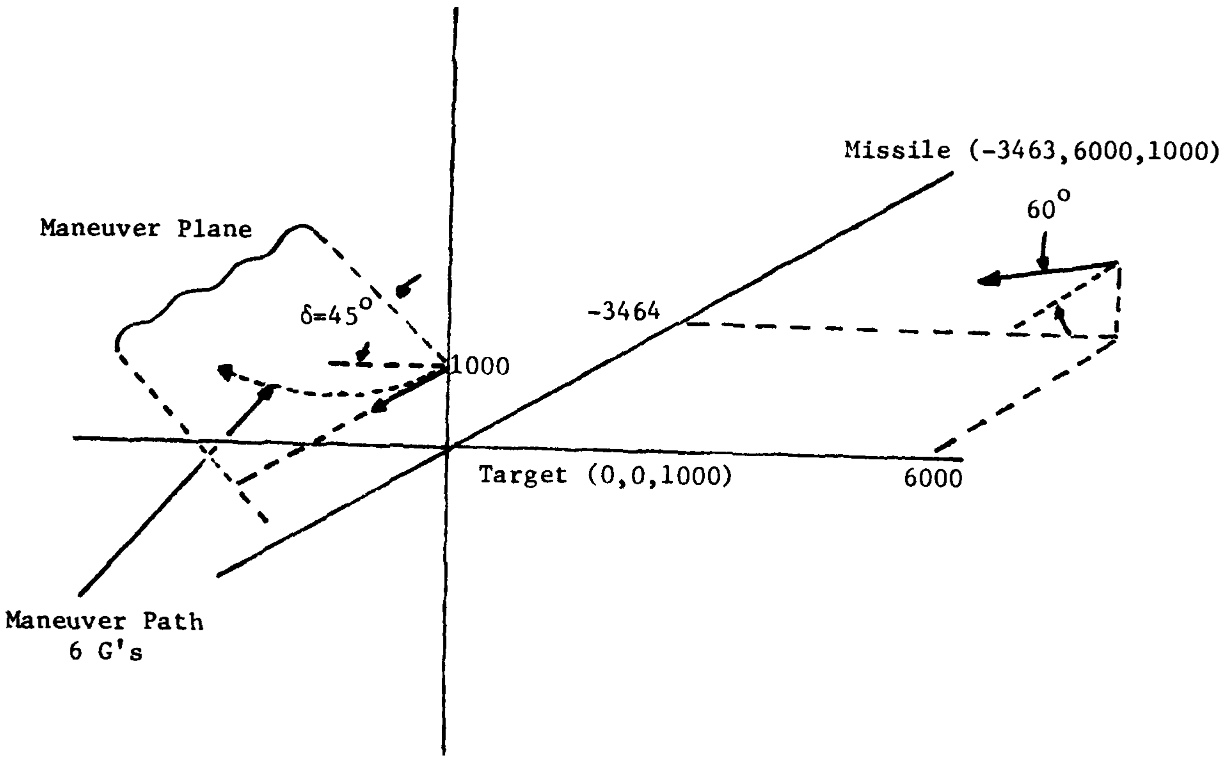


Figure 3.11b Scenario 5: Side Attack

3.6.1 Perfect Measurements

To evaluate the potential for success, simulations were first run using perfect measurement information. The resulting miss distances are given in Table 3.2. While insignificant computational delay was assumed, a sampling or update rate of 100 msec was used for the reachable set approach. It can be seen, as would be expected, that performance of conventional techniques is severely degraded by high crossing rate. In both cases, the terminal efforts required by conventional guidance saturated the allowable limits of the missile. On the other hand, the advanced approach gave a control effort well distributed over the encounter's duration, indicating a high degree of anticipation, as noted in reference 20. Ideally, the effort should be monotonically non-increasing, whereas in practice a small degree of "upturning" of control effort is witnessed on final approach. In both scenarios the improvement in miss distance over that of conventional guidance exceeds two orders of magnitude, and gives virtual target contact.

Table 3.2. Miss Distances

	Scenario 2	Scenario 5
Pro-nav	16'	150'
Pursuit	19'	170'
Reachable Set	.11'	.28'

$T_s = .1$ s

3.6.2 Computational Delay

The scope of computation described in Sections 3.4 and 3.5 is clearly a significant factor in implementation. In practice the length of time from measurement availability to completion of control calculation ultimately governs the rate at which course refinements can be made. Clearly, this implementation delay serves to degrade performance since actual application of the control effort comes when the measurements have lost some validity. To study the implications of this effect for the reachable-set based approach, a series of simulations were formulated allowing variations of computation time lag from 0 to 50 msec. The update interval as before was fixed at 100 msec. Figure 3.12 shows these results graphically for the two scenarios. The performance behaves in a roughly parabolic fashion as time lag is increased. For Scenario two, the miss distance went from .1 to 3 feet; for Scenario five, from .2 to 9 feet. This is expected from an intuitive viewpoint, since high crossing rate would imply a faster obsolescence of measurement data. Nevertheless, accuracy remained considerably better than that of conventional guidance under ideal conditions.

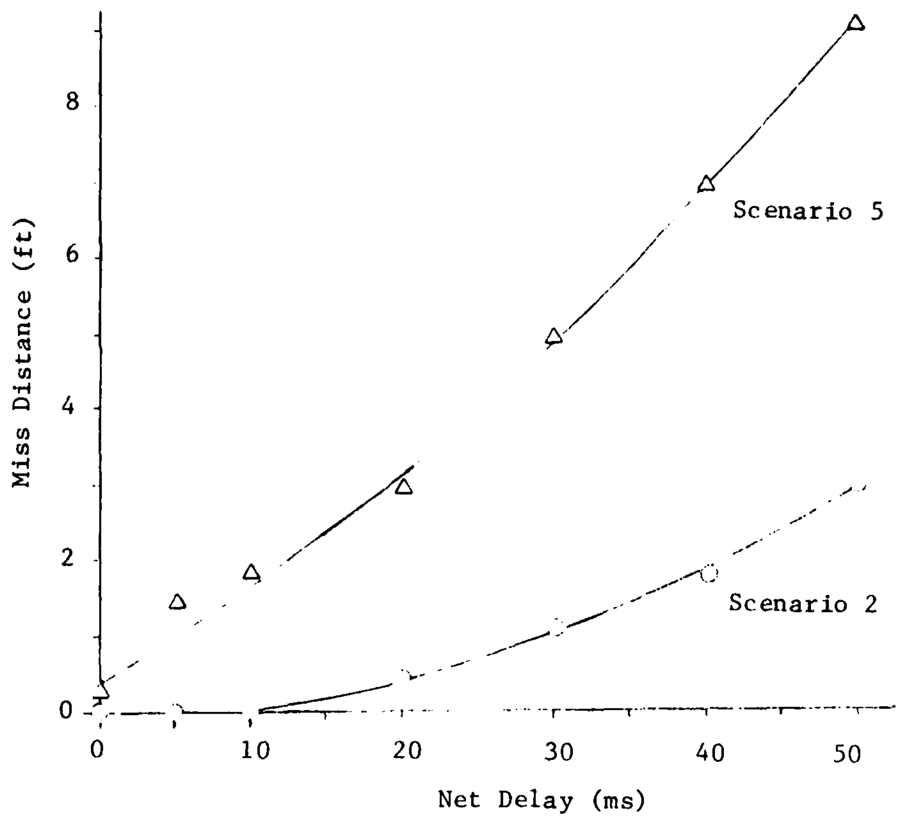


Figure 3.12.
Performance versus Computation Delay
Reachable Set Guidance, Missile Body Reference

3.6.3 Measurement Error Effects

The most significant requirement of the reachable-set approach, aside from computational hardware, is a set of extensive measurements, and their associated sensors. The necessary information includes relative displacement (target range and displacement angle) and relative velocity (target speed and heading). The required accuracy of these measurements dictates the sensor cost and complexity. To evaluate the performance sensitivity with respect to measurement errors, a third set of simulations was conducted.

1) Range Error. To each measurement of target range a white Gaussian error sample was added. The standard deviation of the error was specified as a percentage of the actual instantaneous range; the distribution was truncated to give only positive range measurements. Figure 3.13 shows an ensemble mean over 25 samples of performance versus range error standard deviation. It can be seen that moderate to severe penalties result from random range inaccuracies. For a standard deviation of less than 75%, the performance is still superior to conventional guidance.

2) Speed Error. Controlled inaccuracies were incorporated into speed measurements in much the same manner. Performance degradation was more pronounced than for range errors, but the basic shape remains unchanged, as seen in Figure 3.14.

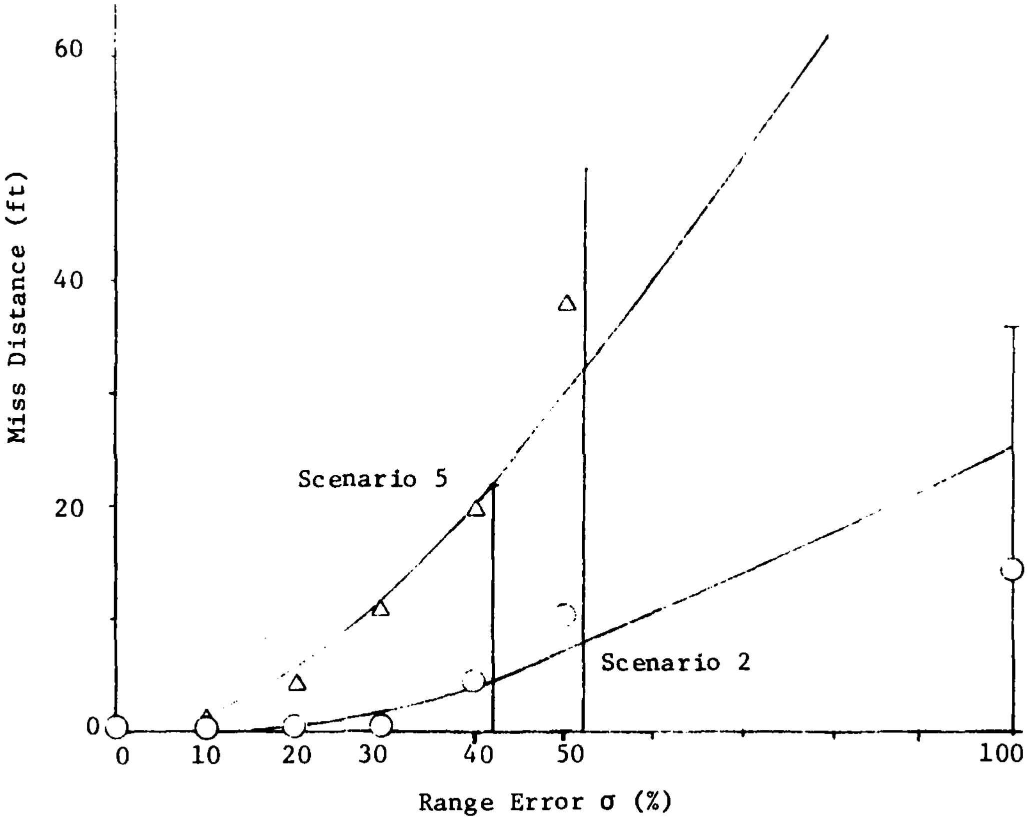


Figure 3.13.
Performance versus Range Error
Reachable Set Guidance, Missile Body Reference

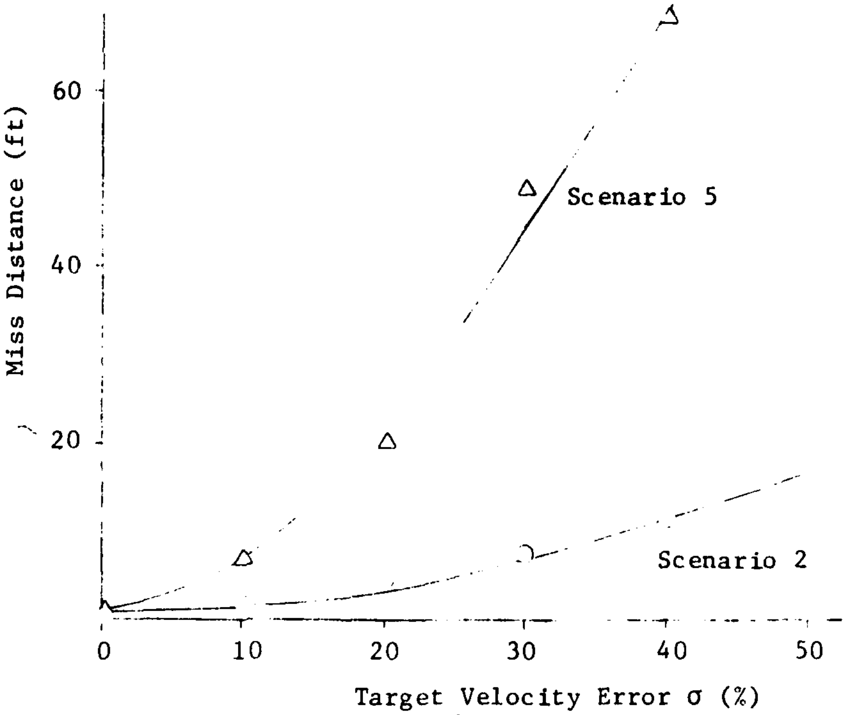


Figure 3.14.
Performance versus Velocity Error
Reachable Set Guidance, Missile Body Reference

3) Target Heading. The angular measurements necessary were degraded by a white Gaussian disturbance added to each sample. The standard deviation of the error was given in absolute degrees. As seen from Figure 3.15, both scenarios were essentially equally affected by errors in the target heading angle. Five degree (90 mrad) standard deviation resulted in an expected miss of 25 feet, a substantial degradation.

4) Target Position. For the implementation used in the simulations, relative target position was generated from a measurement of azimuth angle from a reference axis. This second angle is apparently far more critical than the other. As indicated by Figure 3.16, a smaller standard deviation of three degrees (55 mrad) results in the same mean miss distance of 25 feet.

Summarizing the observations from the simulations:

1) Given equivalent noiseless measurements, the reachable set based guidance has potential for tremendous performance improvements over conventional guidance. This is due to the reduction of terminal saturation effects, and the even distribution of control effort.

2) Actuator lag due to computation time degrades performance to some degree, although the effect may not be serious. Higher crossing rates increase the severity of the degradation.

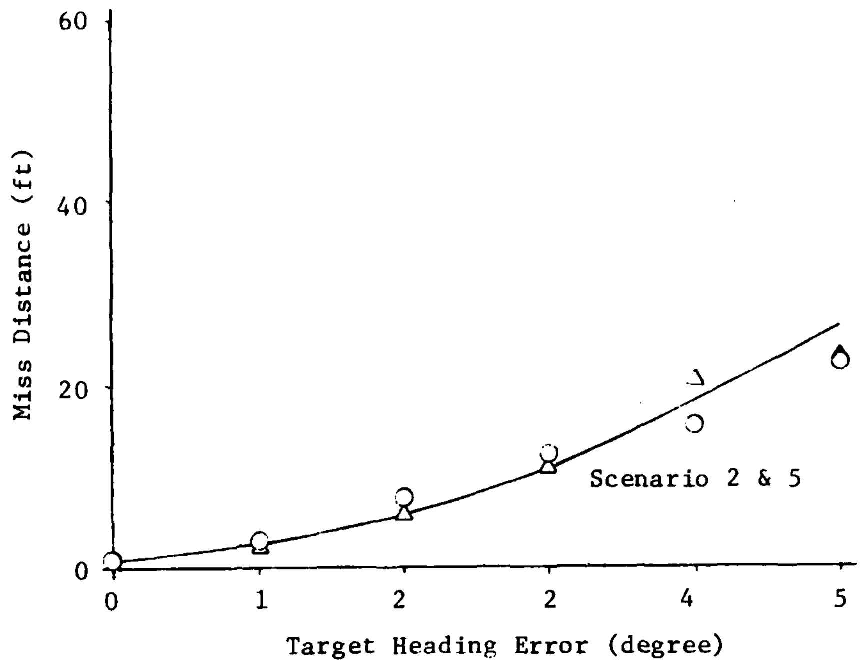


Figure 3.15
Performance versus Target Heading Error
Reachable Set Guidance, Missile Body Reference

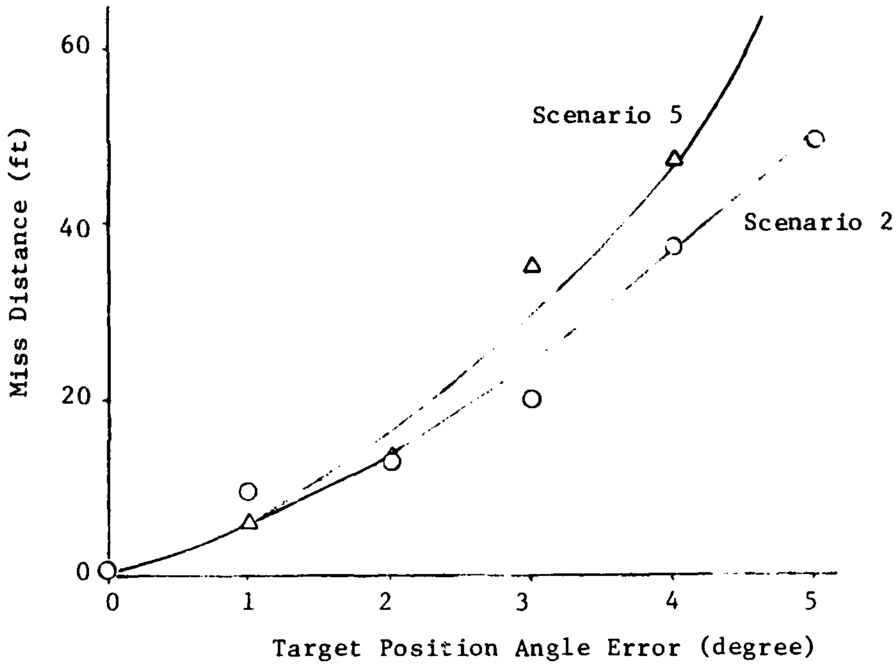


Figure 3.16
Performance versus Target Position Angle Error
Reachable Set Guidance, Missile Body Reference

3) Range accuracy is apparently not critical. Distance measured by conventional means should be adequate. Image ranging might very well be possible.

4) Likewise, relative speed accuracy is not critical.

5) On the other hand, target heading requires measurement accuracy on the order of a few degrees for suitable performance.

6) Target position azimuth angle apparently required even a tighter tolerance, but well within present sensor accuracies.

The preliminary results described here are merely intended to indicate the sensitivity of a typical advanced guidance technique to <u>independent</u> errors. The apparent sensor accuracy required is stringent. Of course, an actual system would be able to incorporate a tracking or smoothing algorithm in software, reducing a sensor's raw error substantially, and consequently enhance missile performance.

BIBLIOGRAPHY

1. N. Shannon and K. Stich, "Imaging Sensors for RPUs," Proceedings of the Society of Photo-Optical Instrumentation Engineers - Airborne Reconnaissance, Vol 101, pp. 26-38, 1977.

2. D.F. Barbe, "Imaging with Charge-Coupled Devices," Journal of Spacecraft, Vol 14, pp. 300-305, 1977.

3. J. Rodgers and W.E. Taylor, "Solid State Imaging Device Parameter Study for Use in Electro-Optic Tracking Systems," Final Report for AFOSR Grant 76-3022, Oct 1977.

4. R.H. Stanton and R.W. Armstrong, "A CCD Tracker for Closed-Loop Instrument Pointing at Halley's Comet," AIAA Paper 81-0088, AIAA 19th Aerospace Sciences Meeting, St Louis, Missouri, January 1981.

5. J. Sklansky, "Image Segmentation and Feature Extraction," IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-8, No. 4, April 1978.

6. R.M. Haralick, "Statistical and Structural Approaches to Texture," Proceedings of the IEEE, Vol 67, No. 5, May 1979.

7. R.M. Haralick, K. Shannugam, I. Dinstein, "Textural Features for Image Classification," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-3, No. 6, November 1973.

8. K. Laws, "Textured Image Segmentation," University of Southern California Image Processing Institute Report 940, January 1980.

9. Arthur E. Bryson and Yu-Chi Ho, Applied Optimal Control, Blaisdell Publishing Company, 1969.

10. Arthur E. Bryson, "Linear Feedback Solutions for Minimum Effort Interception, Rendezvous, and Soft Landing," AIAA Journal, Vol 13, No. 8, pp 1542-1544, August 1965.

11. Mauicio Guelman, "A Qualitative Study of Proportional Navigation," IEEE Transactions on Aerospace and Electronic Systems, Vol AES-7, No. 4, pp 637-643, July 1971.

12. M. Guelman, "Proportional Navigation with a Maneuvering Target," IEEE Transactions on Aerospace and Electronic Systems, Vol AES-8, No. 3, pp 364-371, May 1972.

13. Stephen A. Murtaugh and Harry E. Criel, "Fundamentals of Proportional Navigation," IEEE Spectrum, Vol 3, No. 12, pp 75-85, December 1966.

14. Tom L. Riggs, Jr., "An Overview -- Optimal Control and Estimation for Tactical Missiles Research Program," IEEE 1979 National Aerospace and Electronics Conference, Dayton, Ohio, pp 752-756, May 1979.

15. D.D. Sworder, "Guidance Laws for Accelerating Targets," Proceedings Thirteenth Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove, CA, pp 88-92, November 1979.

16. K.C. Wei, and A.E. Pearson, "Control Law for an Intercept System," Journal of Guidance and Control, Vol 1, No. 5, pp 298-304, September 1978.

17. Paul Zarchan "Representation of Realistic Evasive Maneuvers by the Use of Shaping Filters," Journal of Guidance and Control, Vol 2, No. 4, pp 290-295, July 1979.

18. John J. Deyst, Jr., and Charles F. Price, "Optimal Stochastic Guidance Laws for Tactical Missiles," Journal of Spacecraft, Vol 10, No. 5, pp 301-308, May 1973.

19. Shaul Gutman, "On Optimal Guidance for Homing Missiles," Journal of Guidance and Control, Vol 2, No. 4, pp 296-300, July 1979.

20. N.K. Gupta and B. Sridhar, "Reachable Sets for Missile Guidance Against Smart Targets," IEEE National Aerospace and Electronics Conference, pp 780-787, May 1979.

21. David M. Salmon and Walter Heinz, "Reachable Sets Analysis -- An Efficient Technique for Performing Missile/Sensor Tradeoff Studies," AIAA Journal, Vol 11, No. 7, pp 927-931, July 1973.

22. B. Sridhar and N.K. Gupta, "Accurate Real-Time SRAAM Guidance Using Singular Perturbation Optimal Control," IEEE National Aerospace and Electronics Conference, pp 772-779, May 1979.

23. B. Sridhar and N.K. Gupta, "Missile Guidance Laws Based on Singular Perturbation Methodology," Journal of Guidance and Control, Vol 3, No. 2, pp 158-165, March 1980.

24. Richard James Casler, Jr., "Dual Control Guidance Strategy for Homing Interceptors Taking Angle-Only Measurements," Journal of Guidance and Control, Vol 1, No. 1, pp 63-70, January 1978.

APPENDIX

REACHABLE SET GUIDANCE ROUTINE

```
FTN4,L
      SUBROUTINE GUIDE(IFLAG,XT,XM,U,TSAMP)
C
C
C     800613  ROUTINE TO COMPUTE REACHABLE-SET GUIDANCE
C     800708  MOD TO SEARCH ONLY LIMITED ANGLE RANGE
C     800711  MOD TO USE MISSILE CENTERED REFERENCE SYSTEM
C     800801  MOD TO RESTART WITHOUT PROMPTING
C
C     IFLAG = INITIALIZATION FLAG =0 SETUP
C                                 =2 GO
C
C     XT = TARGET STATE VECTOR, (X,Y,Z) & (V,PHI,THETA)
C
C     XM = MISSILE STATE VECTOR, SAME AS XT
C
C     U = 3-D CONTROL COMMAND VECTOR (THRUST,ANORMAL,SIGMA)
C
C     TSAMP = SAMPLING INTERVAL
C
C
      DIMENSION XT(1),XM(1),U(1)
      DIMENSION HIT(3),DEL(3),TEM(3),TET(3),A(3),B(3)
      DIMENSION HOLD(4)
      DIMENSION DELX(3),AD(3),MD(3)
      DIMENSION HAM(5,5),MAR(2,5)
      DIMENSION IX(3)
      DATA THRUST/4700./
      DATA PI/3.1415927/
      DATA IHUH/0/
C
C     ENTER HERE
C
      IF(IFLAG) 50,5,50
    5 IF(IHUH) 45,10,45
   10 WRITE(1,11)
   11 FORMAT("REACHABLE SET GUIDANCE, ENTER TURN RATE FOR TARGET #+")
      READ(1,*) RK
      WRITE(1,12)
   12 FORMAT("INPUT UPDATE INTERVAL (SEC), ANGLE TOL, ANGLE WINDOW #+")
      READ(1,*) UPDAT,DEPS,DSTEP
      WRITE(1,13)
   13 FORMAT("INPUT HIT TOLERANCE & NUMBER OF SEARCH ITERATIONS #+")
      READ(1,*) EPS,MITER
      ITER=0.
      JFLAG=0
      WRITE(1,15)
   15 FORMAT("INPUT FRACTIONAL ERROR ON RANGE MEASUREMENTS #+")
      READ(1,*) RPER
      WRITE(1,16)
   16 FORMAT("VELOCITY MEASUREMENTS @+")
      READ(1,*) VPER
      WRITE(1,17)
   17 FORMAT("BEARING ANGLE MEASUREMENT #+")
      READ(1,*) BSIG
      BSIG=BSIG*PI/180.
```

```
      WRITE(1,19)
   19 FORMAT("VELOCITY VECTOR MEASUREMENT ?+")
      READ(1,*) ASIG
      ASIG=ASIG*PI/180.
      WRITE(1,20)
   20 FORMAT("RANDOM SEEDS ?+")
      READ(1,*) IX(1),IX(2)
      DO 35 K=1,100
   35 XNOIS=RGN(IX)
      INUM=1
      RETURN
C
C     RESTART
C
   45 ITER=0
      JFLAG=0
      TIN=0.
      RETURN
C
C     RUN TIME
C
   50 TIME=TSAMP*ITER
      REMAIN=AMOD(TIME,UPDAT)
      ITER=ITER+1
      IF(AMOD(TIME,UPDAT).GE.TSAMP*.99) RETURN
C
C
C     GIVE US THE MEASUREMENTS OBSERVED IN MISSILE SCALE
C
      VM=XM(4)
      CALL MISRF(0,XT,XM,RANGE,BEAR,VT,PHIM,THETM)
      SCALE=180./PI
C
C     CORRUPT MEASUREMENTS
C
   51 RTEMP=RANGE*(1.+RPER*RGN(IX))
      IF(RTEMP.LT.0.) GO TO 51
      RANGE=RTEMP
      BEAR=BEAR+BSIG*RGN(IX)
   52 RTEMP=VT*(1.+VPER*RGN(IX))
      IF(RTEMP.LT.0.) GO TO 52
      VT=RTEMP
      PHIM=PHIM+ASIG*RGN(IX)
      THETM=THETM+ASIG*RGN(IX)
C
C*********************************************************************
C
C     NOW, GIVEN THE OBSERVATIONS: RANGE, BEARING , TARGET SPEED (ABS),
C     AND TARGET VELOCITY DIRECTION, I.E.
C
C     RANGE
C     BEAR
C     VT
C     PHIM
C     THETM
C
```

```
C         1)  TRANSFORM MISSILE POSITION AND VELOCITY VECTORS INTO TARGET
C             CENTERED COORDINATE SYSTEM, WHERE X AXIS IS ALIGNED WITH
C             TARGET VELOCITY VECTOR.
C
C         2)  SEARCH FOR WORST MANUEVER PLANE BY BINARY SEARCH.  GIVES
C             NECESSARY VELOCITY VECTOR FOR INTERCEPT.
C
C         3)  TRANSFORM THIS DESIRED VELOCITY VECTOR BACK INTO MISSILE
C             CENTERED COORDINATES.
C
C         4)  DETERMINE DIRECTION OF ACCELERATION VECTOR, GIVEN BY SIGMA.
C
C
C     COMPUTE VALUES NECESSARY FOR COMPUTATION
C
C     DISPLACEMENT VECTOR (TARGET CENTERED)
C
      DELX(1)=-RANGE*COS(BEAR)
      DELX(2)=-RANGE*SIN(BEAR)
      DELX(3)=0.
C
C     TRANSFORM POSITION VECTOR INTO TARGET CENTERED WITH X AXIS PROPE
C     ALIGNED WITH TARGET VELOCITY
C
      CALL MTRAN(PHIM,DELX,HOLD)
      TEMP=HOLD(2)
      HOLD(2)=HOLD(3)
      HOLD(3)=TEMP
      CALL MTRAN(THETM,HOLD,DELX)
      TEMP=DELX(2)
      DELX(2)=DELX(3)
      DELX(3)=TEMP
C
C     TRANSFORM VELOCITY VECTOR IN SAME WAY
C
      HOLD(1)=1.
      HOLD(2)=0.
      HOLD(3)=0.
      CALL MTRAN(PHIM,HOLD,TEM)
      TEMP=TEM(2)
      TEM(2)=TEM(3)
      TEM(3)=TEMP
      CALL MTRAN(THETM,TEM,HOLD)
      TEM(1)=HOLD(1)
      TEM(2)=HOLD(3)
      TEM(3)=HOLD(2)
C
C     'DELX' IS POSITION VECTOR, 'TEM' IS VELOCITY VECTOR
C
C     CONVERT BACK TO SPHERICAL COORDINATES IN NEW SYSTEM
C
      PHIST=ATAN2(TEM(2),TEM(1))
      THETST=ATAN2(TEM(3),SQRT(DOT(TEM(1),TEM(1),2)))
C
C     DO THE ACTUAL CONTROL COMPUTATION
C
```

```
C      AT START-UP, SEARCH MUST BE OVER WHOLE CIRCLE FOR WORST MANEUVER
C      (MAY BE DONE PRIOR TO LAUNCH BY HOST VEHICLE COMPUTER AND DOWN-LO
C
       IF(JFLAG) 70,60,70
C
C      START-UP SEARCH DONE INITIALLY
C      LOOK AT 0,90,180,270 ANGLES FOR WORST PAIR. . . BASE BINARY SEARC
C      ON THOSE STARTING POINTS
    60 L=0
       JFLAG=1
       AM=-1.E37
       DO 62 LZ=1,4
       DELTA=(LZ-1)*PI/2.
       CALL INTER(DELX,RK,DELTA,VT,VM,TIN,HIT,EPS,MITER,ICOUN,IER)
       IF(IER.EQ.0) TIN=HIT(1)
       CALL VECAN(PHIST,THETST,HIT(2),HIT(3),ZETA)
       ATEST=2.*VM*SIN(ZETA)/HIT(1)
       IF(ATEST.LT.AM) GO TO 61
       AM=ATEST
       L=LZ
    61 HAM(LZ,1)=AM
       HAM(LZ,2)=DELTA
       HAM(LZ,3)=HIT(2)
       HAM(LZ,4)=HIT(3)
       HAM(LZ,5)=HIT(1)
    62 CONTINUE
       DO 63 K=1,5
    63 HAM(5,K)=HAM(1,K)
       HAM(5,2)=2.*PI
       IF(L.NE.1) GO TO 65
       M=3
       IF(HAM(4,1).LT.HAM(2,1)) M=1
       GO TO 66
    65 M=-1
       IF(HAM(L-1,1).LT.HAM(L+1,1)) M=1
    66 DO 67 K=1,5
       HAM(1,K)=HAM(L+M,K)
    67 HAM(2,K)=HAM(L,K)
C
C
C      COME HERE FOR ACTUAL BINARY SEARCH
C      COMPUTE ANGLE DIFFERENCE, EXIT WHEN SMALL ENOUGH
C      OTHERWISE LOOK AT MIDPOINT, TOSS OUT SMALLER OF END POINTS
C
    68 DIFDEL=ABS(HAM(1,2)-HAM(2,2))
       IF(DIFDEL.LT.DEPS*PI/180.) GO TO 80
       DELTA=(HAM(1,2)+HAM(2,2))/2.
       CALL INTER
       CALL VECAN
       ATEST=2.*VM*SIN(ZETA)/HIT(1)
       L=1
       IF(HAM(1,1).GT.HAM(2,1)) L=2
       HAM(L,1)=ATEST
       HAM(L,2)=DELTA
       HAM(L,3)=HIT(2)
       HAM(L,4)=HIT(3)
```

```
      HAR(L,5)=HIT(1)
      GO TO 68
C
C
C     AFTER BEING INITIALIZED, COME HERE FOR ALL TIME # 0 TO
C     SET UP CURRENT SEARCH.
C     TAKES WINDOW AROUND OLD MANUEVER ANGLE AND SETS UP END POINTS
C     FOR SEARCH.
C     THEN GOES BACK UP TO BINARY SEARCH PART.
C
   70 M=-1
   71 DELTA=DELHOL+DSTEP*PI/180.*M
      CALL INTER
      CALL VECAN
      ATEST=2.*VM*SIN(ZETA)/HIT(1)
      HAR((M+3)/2,1)=ATEST
      HAR((M+3)/2,2)=DELTA
      HAR((M+3)/2,3)=HIT(2)
      HAR((M+3)/2,4)=HIT(3)
      HAR((M+3)/2,5)=HIT(1)
      IF(M.EQ.1) GO TO 68
      M=1
      GO TO 71
C
C
C     FINISH UP FINDING WORST MANUEVER ANGLE BY LOOKING AT END
C     POINTS OF LAST INTERVAL.  OUTPUTS SPHERICAL ANGLES OF DESIRED
C     VELOCITY VECTOR.
C
   80 M=1
      IF(HAR(1,1).LT.HAR(2,1)) M=2
      PHIBAR=HAR(M,3)
      THETBR=HAR(M,4)
      TIN=HAR(M,5)
      AM=HAR(M,1)
      DELHOL=HAR(M,2)
C
C     MAP DESIRED VELOCITY VECTOR BACK INTO MISSILE CENTERED COORDINATE
C     SYSTEM
C
C     SPHERICAL TO CARTESIAN
C
      TEM(1)=COS(PHIBAR)*COS(THETBR)
      TEM(2)=SIN(PHIBAR)*COS(THETBR)
      TEM(3)=SIN(THETBR)
C
C     ROTATION BACK
C
C
      TEMP=TEM(2)
      TEM(2)=TEM(3)
      TEM(3)=TEMP
      CALL MTRNI(THETM,TEM,HOLD)
      TEMP=HOLD(2)
      HOLD(2)=HOLD(3)
      HOLD(3)=TEMP
```

```
            CALL STRAI(PHI ,HOLD,TEM)
      C
      C
      C     BACK TO SPHERICAL IN MISSILE SYSTEM
      C
            PHIL=ATAN2(TEM(2),TEM(1))
            THETD=ATAN2(TEM(3),SQRT(DOT(TEM(1),TEM(1),2)))
      C
      C     FIND LIFT VECTOR ANGLE SIGMA FROM THIS ACCELERATION ANGLE
      C
            CALL TWIST(V,,V,,PHID,THETD,SIG)
      C
      CCCCCC FINISHED AT THIS POINT*****************************************
      C
      C     CONVERT THIS LIFT VECTOR BACK TO INERTIAL REFERENCE
      C
            CALL MISRF(1,XT,XM,RANGE,BEAR,VT,PHT,THETA,HOLD,SIG)
            TEM(1)=COS(XM(5))*COS(XM(6))
            TEM(2)=SIN(XM(5))*COS(XM(6))
            TEM(3)=SIN(XM(6))
            DEL(1)=-TEM(2)
            DEL(2)=TEM(1)
            DEL(3)=0.
            CALL VMPL(DEL,1./SQRT(DOT(DEL,DEL,3)),DEL,3)
            SIG=ATAN2(DOT(HOLD,DEL,3),HOLD(3))
            IF(AM.EQ.-1.E37) GO TO 140
            U(1)=THRUST
            U(2)=AM
            U(3)=SIG
        140 RETURN
            END
```

** NO ERRORS*

```
FTN4,L
      SUBROUTINE MISRF(IFLAG,XT,XM,RANGE,BEAR,VT,PHI,THETA,AVEC,SIG)
C
C     800711   ROUTINE TO COMPUTE OBSERVATIONS IN MISSILE CENTERED
C              COORDINATE SYSTEM, GIVEN INERTIAL REFERENCE COORDINATES,
C     800714   MOD FOR RIGHT-HANDED COORDINATE SYSTEM
C
C
C     IFLAG = 0, GIVE ME THE MEASUREMENT, # 0 RETURN RETRANSFORMED A VE
C
C     XT = TARGET STATE VECTOR, (X,Y,Z) X (V,PHI,THETA)
C
C     XM = MISSILE STATE VECTOR ETC
C
C     RANGE = SEPARATION DISTANCE
C
C     BEAR = AZIMUTH ANGLE TO TARGET
C
C     VT = TARGET SPEED
C
C     PHI = BEARING ANGLE OF TARGET VELOCITY
C
C     THETA = ELEVATION ANGLE OF TARGET VELOCITY
C
C     AVEC = VECTOR FOR CONROL EFFORT
C
C     SIG = ANGLE FOR LIFT
C
      DIMENSION XT(1),XM(1)
      DIMENSION DEL(3),VMVEC1(3),VMVEC2(3),VMVEC3(3),VTVEC(3)
      DIMENSION RVEC(3),HOLD1(3),HOLD2(3)
      DIMENSION AVEC(3)
C
C     GOING OR COMING
C
      IF(IFLAG) 100,9,100
C
C     COMPUTE RELATIVE DISPLACEMENT IN INTERTIAL COORDINATES
C     AND RANGE
C
    9 DO 10 K=1,3
   10 DEL(K)=XT(K)-XM(K)
      RANGE=SQRT(DOT(DEL,DEL,3))
C
C     COMPUTE VELOCITY UNIT VECTORS IN INERTIAL COORDINATES
C
      VMVEC1(1)=COS(XM(5))*COS(XM(6))
      VMVEC1(2)=SIN(XM(5))*COS(XM(6))
      VMVEC1(3)=SIN(XM(6))
      VTVEC(1)=COS(XT(5))*COS(XT(6))
      VTVEC(2)=SIN(XT(5))*COS(XT(6))
      VTVEC(3)=SIN(XT(6))
C
C     COMPUTE RANGE UNIT VECTOR
C
      CALL VMUL(RVEC,1./RANGE,DEL,3)
```

```
C
C      CONSTRUCT "Y" BASIS VECTOR
C
       CALL VMUL(HOLD1,DOT(RVEC,VMVEC1,3),VMVEC1,3)
       CALL VSUB(VMVEC2,RVEC,HOLD1,3)
       CALL VMUL(VMVEC2,1./SQRT(DOT(VMVEC2,VMVEC2,3)),VMVEC2,3)
C
C      CONSTRUCT "Z" BASIS VECTOR FROM X CROSS Y
       VMVEC3(1)=VMVEC1(2)*VMVEC2(3)-VMVEC1(3)*VMVEC2(2)
       VMVEC3(2)=VMVEC1(3)*VMVEC2(1)-VMVEC1(1)*VMVEC2(3)
       VMVEC3(3)=VMVEC1(1)*VMVEC2(2)-VMVEC1(2)*VMVEC2(1)
C
C      COMPUTE POSITION AZIMUTH
C
       TEMP=DOT(RVEC,VMVEC1,3)
       IF(ABS(TEMP).GT.1.) TEMP=SIGN(1.,TEMP)
       BEAR=ACOS(TEMP)
C
C      COMPUTE VELOCITY AZIMUTH
C
       X=DOT(VTVEC,VMVEC1,3)
       Y=DOT(VTVEC,VMVEC2,3)
       Z=DOT(VTVEC,VMVEC3,3)
       PHI=ATAN2(Y,X)
       IF(ABS(Z).GT.1.) Z=SIGN(1.,Z)
       THETA=ASIN(Z)
       VT=XT(4)
       RETURN
C
C      RETURN LIFT VECTOR
C
   100 CALL VMUL(HOLD1,SIN(SIG),VMVEC2,3)
       CALL VMUL(HOLD2,COS(SIG),VMVEC3,3)
       CALL VADD(AVEC,HOLD1,HOLD2,3)
       RETURN
       END

**  NO ERRORS*
```

```
FTN4,L
      SUBROUTINE INTER(XIN,K,DELTA,VT,V,TIN,HIT,EPS,MITER,ITER,IER)
C
C     800530  ROUTINE TO SOLVE FOR VELOCITY ORIENTATION FOR INTERCEPT
C              GIVEN TARGET INFORMATION AND NOMINAL MISSILE SPEED
C
C     801625  MOD TO USE ALTERNATE SOLUTION, ALGEBRAIC ELIMINATION OF
C           .  ANGLE VARIABLES FIRST
C
C     XIN = INITIAL MISSILE POSITION WRT TO TARGET (X0,Y0,Z0)
C           (TARGET INITIAL VELOCITY COLINEAR TO X AXIS)
C
C     K = TARGET MAX TURN RATE
C
C     DELTA = TARGET TURN PLANE INCLINE
C
C     VT = TARGET NOMINAL SPEED
C
C     V = MISSILE NOMINAL SPEED
C
C     TIN = INITIAL VALUE FOR GUESS OF INTERCEPT TIME (IF 0, COMPUTES
C
C     HIT = VECTOR OF INTERCEPT INFORMATION (TIME,PHI,THETA)
C
C     EPS = MAXIMUM RANGE ERROR SQUARED TO TOLERATE (INPUTTED)
C
C     MITER = MAXIMUM NUMBER OF NEWTON STEPS TO TRY (INPUTTED)
C
C     ITER = NUMBER OF NEWTON STEPS COMPLETED (OUTPUTTED)
C
C     IER = ERROR RETURN NUMBER   =0 OK
C                                 =1 SINGULARITY CONDITION
C                                 =-1 FAILURE TO CONVERGED IN MITER STEP
C
C
      DIMENSION XIN(3),HIT(3)
      DIMENSION SCR1(3),SCR2(3)
      REAL K
      DATA PI/3.1415927/
      DATA VNOM/2500./
C
C     MAKE INTELLIGENT INITIAL GUESS (TAILCHASE)
C
      VM=V
      IF(V.LT.VNOM) VM=VNOM
      ITER=0
      TF=TIN
      IF(TIN.GT.0.) GO TO 1
      RANGE=SQRT(DOT(XIN,XIN,3))
      TF=RANGE/VM
C
C     NEWTON STEP LOOP
C
    1 Y=(VM*TF)**2-(XIN(1)-VT/K*SIN(K*TF))**2
     &   -(XIN(2)+VT/K*COS(DELTA)*(COS(K*TF)-1.))**2
     &   -(XIN(3)+VT/K*SIN(DELTA)*(COS(K*TF)-1.))**2
```

```
      YD=2.*VM*VM*TF-2.*(XIN(1)-VT/K*SIN(K*TF))*(-VT*COS(K*TF))
     &   -2.*(XIN(2)+VT/K*COS(DELTA)*(COS(K*TF)-1.))
     &   *(-VT*COS(DELTA)*SIN(K*TF))
     &   -2.*(XIN(3)+VT/K*SIN(DELTA)*(COS(K*TF)-1.))*(-VT*SIN(DELTA)
     &   *SIN(K*TF))
      IF(ABS(YD).LT.1.E-5) GO TO 99
      IF(SQRT(ABS(Y)).LE.EPS*VM*TF) GO TO 89
      TF=TF-Y/YD
      IF(TF.LT.0.) TF=.001
      ITER=ITER+1
      IF(ITER.GE.MITER) GO TO 109
      GO TO 1
C
C     OK RETURN
C
   89 IER=0
      ARG=(-XIN(3)-VT/K*SIN(DELTA)*(COS(K*TF)-1.))/(VM*TF)
      IF(ABS(ARG).GT.1.) ARG=SIGN(1.,ARG)
      THETA=ASIN(ARG)
      RNUM=(-XIN(2)-VT/K*COS(DELTA)*(COS(K*TF)-1.))
      RDEN=(-XIN(1)+VT/K*SIN(K*TF))
      PHI=ATAN2(RNUM,RDEN)
   95 HIT(1)=TF
      HIT(2)=PHI
      HIT(3)=THETA
      RETURN
C
C     SINGULARITY RETURN
C
   99 IER=1
      RETURN
C
C     FAILURE TO CONVERGE RETURN
C
  109 IER=-1
      RETURN
      END
```

** NO ERRORS*

```
FTN4,L
      SUBROUTINE MTRAN(ANG,XIN,XOUT)
C
C     820711  ROUTINE TO TRANSFORM IN ROTATIONAL SENSE A 3-SPACE
C             VECTOR
C
C     ANG = ROTATIONAL ANGLE
C
C     XIN = 3-SPACE VECTOR
C
C     XOUT = ROTATED VECTOR
C
      DIMENSION XIN(3),XOUT(3)
      XOUT(3)=XIN(3)
      XOUT(1)=COS(ANG)*XIN(1)+SIN(ANG)*XIN(2)
      XOUT(2)=-SIN(ANG)*XIN(1)+COS(ANG)*XIN(2)
      RETURN
      END
```

** NO ERRORS*

```
      SUBROUTINE [illegible](ANG,XIN,XOUT)
C
C     BRD714  INVERSE TRANSFORMATION
C
      DIMENSION XIN(1),XOUT(1)
      XOUT(3)=XIN(3)
      XOUT(1)=COS(ANG)*XIN(1)-SIN(ANG)*XIN(2)
      XOUT(2)=SIN(ANG)*XIN(1)+COS(ANG)*XIN(2)
      RETURN
      END
```

** NO ERRORS*

FILM