

NO-A184 973

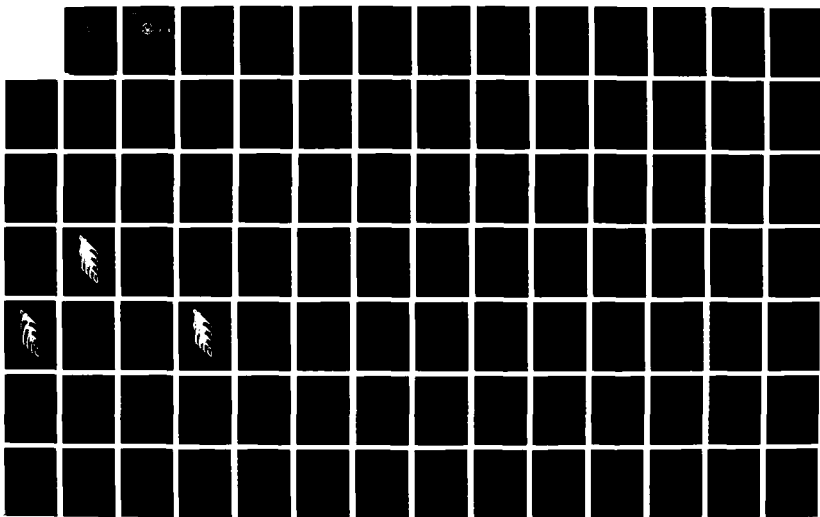
CORRECTION OF INERTIAL NAVIGATION SYSTEM DRIFT ERRORS
FOR AN AUTONOMOUS L. (U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA H D RICKENBACH SEP 87 NP552-87-839

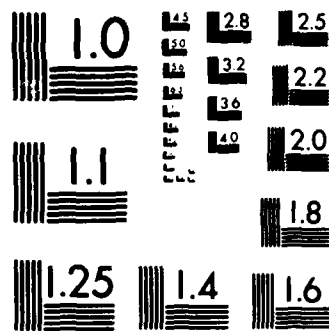
1/1

UNCLASSIFIED

F/G 12/9

ML





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD-A184 973

NPS52-87-039

DTIC FILE COPY

(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
OCT 06 1987
S D

CORRECTION OF INERTIAL NAVIGATION SYSTEM
DRIFT ERRORS FOR AN AUTONOMOUS LAND
VEHICLE USING OPTICAL RADAR TERRAIN DATA

Mark Douglas Rickenbach

September 1987

Thesis Advisor:

R. B. McGhee

Approved for public release; distribution is unlimited.

Prepared for:
Ohio State University Research Foundation
Columbus, OH. 43212

DEPARTMENT OF THE NAVY

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Rear Admiral R. C. Austin
Superintendent

D. A. Schradly
Provost

This thesis prepared in conjunction with research sponsored
in part by Ohio State Univ. Research Foundation under RF
Project No. 716520.

Reproduction of all or part of this report is authorized.

Released By:



GORDON E. SCHACHER
Dean of Science and Engineering

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			5 MONITORING ORGANIZATION REPORT NUMBER(S)		
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-87-039			7a NAME OF MONITORING ORGANIZATION Prof. Kenneth J. Waldron Dept. of Mech. Eng., Ohio State Univ.		
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b OFFICE SYMBOL (if applicable) 52	7b ADDRESS (City, State, and ZIP Code) 2075 Robinson Laboratory 206 W. 18th Ave. Columbus, Ohio 43210		
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER RF Project No. 716520 RF Purchase Order No 496549			
8a NAME OF FUNDING/SPONSORING ORGANIZATION Ohio State Univ. Research Foundation		8b OFFICE SYMBOL (if applicable)	10 SOURCE OF FUNDING NUMBERS		
8c ADDRESS (City, State, and ZIP Code) 1314 Kinnear Road Columbus, Ohio 43212		PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) Correction of Inertial Navigation System Drift Errors for an Autonomous Land Vehicle Using Optical Radar Terrain Data					
12 PERSONAL AUTHOR(S) Rickenbach, Mark D.					
13a TYPE OF REPORT MS Thesis		13b TIME COVERED FROM TO		14 DATE OF REPORT (Year, Month, Day) 1987 September	
15 PAGE COUNT 96		16 SUPPLEMENTARY NOTATION			
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Robotics, Walking Machines, Adaptive Suspension Vehicle, Computer vision, Inertial Navigation		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) It is generally agreed that some type of vision system capable of providing a preview of terrain is an important attribute of a driverless vehicle. One approach to providing this capability is to use active images such as those obtained by sonar or radar. This thesis is concerned with a computer simulation study of an approach to data processing for range images obtained from an optical radar system using a scanning laser beam. The system studied is modeled after the ERIM scanner mounted on the Adaptive Suspension Vehicle walking machine developed at Ohio State University. Both the problem of registering successive images in the presence of vehicle motion and of optimally averaging such images to obtain more accurate terrain elevation data are investigated in the thesis.					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ OTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL Robert B. McGhee			22b TELEPHONE (Include Area Code) 408-646-2095		22c OFFICE SYMBOL 52Mz

Approved for public release, distribution is unlimited

**Correction of Inertial Navigation System
Drift Errors for an Autonomous Land
Vehicle Using Optical Radar Terrain Data**

by

Mark Douglas Rickenbach
Lieutenant Commander, United States Navy
B.S. Applied Math, University of Idaho, 1974

Submitted in partial fulfillment of the
requirements for the degree of

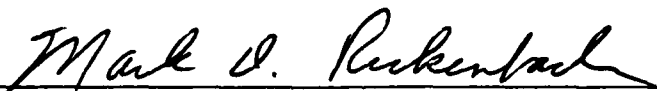
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

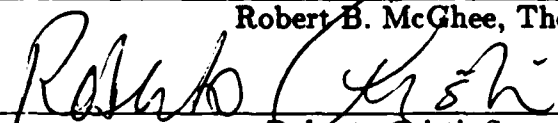
September 1987

Author:

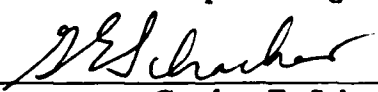

Mark D. Rickenbach

Approved by:


Robert B. McGhee, Thesis Advisor


Roberto Cristi, Second Reader


John P. Powers, Chairman,
Department of Electrical and
Computer Engineering


Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

It is generally agreed that some type of vision system capable of providing a preview of terrain is an important attribute of a driverless vehicle. One approach to providing this capability is to use active images such as those obtained by sonar or radar. This thesis is concerned with a computer simulation study of an approach to data processing for range images obtained from an optical radar system using a scanning laser beam. The system studied is modeled after the ERIM scanner mounted on the Adaptive Suspension Vehicle walking machine developed at Ohio State University. Both the problem of registering successive images in the presence of vehicle motion and of optimally averaging such images to obtain more accurate terrain elevation data are investigated in the thesis.



Accession For	
NTIS CRAGI	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution	
Avail. and/or	
Dist	
A-1	

TABLE OF CONTENTS

I.	INTRODUCTION	7
A.	GOALS	7
B.	ORGANIZATION	8
II.	SURVEY OF PREVIOUS WORK	10
A.	INTRODUCTION	10
B.	MOBILE ROBOT SYSTEMS	10
1.	Early Robot Systems	10
2.	Autonomous Land Vehicles	14
3.	DARPA Adaptive Suspension Vehicle	15
C.	DESCRIPTION OF OPTICAL RADAR	16
1.	Physical Description	16
2.	Performance Data	16
D.	ESTIMATION TECHNIQUES	17
1.	Regression Analysis	17
2.	Iterative Methods	18
E.	SUMMARY	21

III. PROBLEM STATEMENT AND PROPOSED SOLUTION

METHOD	22
A. INTRODUCTION	22
B. IMMEDIATE-AREA TERRAIN SENSING AND PATH SELECTION	22
C. TERRAIN SENSING USING THE OPTICAL RADAR	24
D. PROPOSED METHOD OF DATA FILTERING	30
1. Stationary Walker Case	30
2. Compensating for Walker Movement	31
a. Gradient Descent Method	32
b. Grid Search Method	35
E. SUMMARY	36

IV. EXPLANATION AND JUSTIFICATION OF SIMULATION

MODEL	37
A. INTRODUCTION	37
B. TERRAIN AND OPTICAL SCANNER MODELS	37
C. DATA HANDLING DURING SIMULATION	45
E. SUMMARY	50

V. SIMULATION PERFORMANCE	51
A. INTRODUCTION	51
B. SIMULATION RESULTS	51
1. Stationary Walker Case	51
2. Moving Walker Case	59
C. CONCLUSIONS	71
VI. SUMMARY AND CONCLUSIONS	74
A. RESEARCH CONTRIBUTIONS	74
B. RESEARCH EXTENSIONS	75
LIST OF REFERENCES	77
APPENDIX: PROGRAM LISTING	79
INITIAL DISTRIBUTION LIST	93

I. INTRODUCTION

Robots have long been in man's thinking. From the early science fiction writers to today's cartoons, the idea of robots that operate on their own has fascinated us. There have been, and still are, many research programs concerned with realization of an autonomous robot. These programs include wheeled machines, tracked vehicles, and legged vehicles.

There are many uses and advantages to each type of robotic machine. Legged vehicles or walking machines have an advantage over wheeled or tracked vehicles in that they can traverse a greater variety of terrains. A walking machine can traverse terrain that is heavily wooded, can walk over muddy terrain, and can negotiate obstacles that a wheeled or tracked vehicle may not be able to overcome. However, before any robotic machine can travel anywhere by itself, it has to be able to see where it is going and decide where it wants to go based on what it sees. Once all of these research areas have been mastered, widespread application of autonomous robots may follow.

A. GOALS

The goal of this thesis is to explore methods to obtain accurate terrain altitude information that may be used by some type of route planning program for an autonomous vehicle. The process of obtaining these altitude values will

include some type of filtering of noisy information and correction of errors in the vehicle's location information. The sensors involved in this process are assumed to be some type of inertial navigation system (INS) and an optical radar scanner.

The vehicle used as a reference for setting up the model used in this thesis is the Ohio State University Adaptive Suspension Vehicle (ASV), which is a six-legged walking machine. The ASV is equipped with an optical terrain scanner manufactured by the Environmental Research Institute of Michigan (ERIM). [Ref. 1]

A secondary purpose of this thesis is to develop a simulation model for the optical scanner and terrain that will be used in the testing of the techniques presented in the thesis. These models could also be used in future studies that involve a scanner and terrain.

B. ORGANIZATION

Chapter II presents a discussion of work dealing with early robotic systems as well as current projects under development. The optical terrain scanner used as a reference in this thesis is discussed in some detail. Finally, a general discussion on regression analysis and Kalman filtering are presented.

The reason for obtaining an accurate terrain altitude map is discussed in Chapter III. A detailed discussion is presented on how the inertial navigation and optical scanner data are mathematically manipulated to obtain the x and y coordinates and the terrain altitude of the terrain being scanned. Finally, a more

detailed discussion of Kalman filtering and regression analysis as they apply to this thesis are presented.

Chapter IV presents the simulation models used for the terrain and the optical scanner. Also, flow charts showing how data is handled during the simulation runs are included.

Finally, Chapter V discusses the simulation experiments conducted and their results. The last chapter, Chapter VI, puts forward some conclusions about the work presented in this thesis followed by some recommendations for future work.

II. REVIEW OF PREVIOUS WORK

A. INTRODUCTION

For many years, man has been working toward the goal of making an autonomous robot. However, before a robot or machine can be truly autonomous, it has to have some way of seeing the world in which it is to operate. In this chapter, past robotic systems and their vision systems are reviewed along with ways of handling the data from these vision systems.

B. MOBILE ROBOT SYSTEMS

1. Early Robot Systems

Inventors have been fascinated for centuries with the idea of creating machines that act like animals or human beings. The earliest of these were "clockwork" machines like the "artificial duck" built by Jacques de Vaucanson in the 1730's, and Baron Wolfgang von Kempelen's chess-playing automaton of the late eighteenth century (later proved a hoax). Clockwork machines, as the name implies, were based on complex mechanical gears and linkages which provided the timing and required movements for the machine. Until the advent of digital computers and component miniaturization technology, robots remained little more than mechanical novelties. [Ref. 2: pp. 254-264]

In the 1960's, researchers at Johns Hopkins University built what could possibly be considered the first truly autonomous mobile robot. The sole activity of this robot was to navigate the hallways of the laboratory "searching" for electrical power outlets to keep its batteries charged. Special purpose circuitry (rather than a programmable computer) was used to direct the robot's actions, and information from an elementary sonar system kept the robot centered between the walls. Although simplistic, the machine demonstrated an ability to interact with its environment and to sustain itself through its own actions (i.e. searching for outlets and recharging its batteries). [Ref. 2: pp. 254-264]

The next major step in autonomous robot development was accomplished in 1969 at the Stanford Research Institute (SRI). A mobile robot dubbed "Shakey" was linked to a digital computer via radio giving it the much needed additional "brainpower" to more fully interact with its environment and solve relatively complex problems. Using a television camera as a sensor and hierarchical software control, Shakey was able to negotiate obstacles and accomplish simple path planning when moving from one location to another. This first attempt at machine interaction with an uncertain environment encountered many problems. Rather complex image processing algorithms were required to discern potential obstacles and complicated computer programs were needed to map the obstacles onto a spatial grid coordinate system that the robot could "understand". Shakey kept track of its own position through a crude "dead reckoning" system in which sensors counted the number of rotations of the robot's

wheels to determine how far it had moved from its initial starting position. It was soon discovered that, while multiple sensors can provide a more comprehensive representation of the environment, coordinating the incoming data from those sensors represents a major hurdle. For instance, on the Shakey robot, wheel slippage caused position errors which in turn induced grid map drift and resulted in multiple representations of individual obstacles cluttering the map. [Ref. 3: pp. 10-15]

For all its problems, the Shakey robot remained state-of-the-art until the mid-1970's. During this decade, research involving incorporation of new technologies into autonomous robots continued. Advances in integrated electronics manufacturing made possible relatively powerful single-board computers and smaller, more compact sensor systems. Still, many of the robots built during this time frame were tethered to an off-board computer either via cables or radio link. Among the robots of this period were the Jet Propulsion Laboratory's Mars Rover, the Stanford Cart, and France's Laboratoire d'Automatique et d'Analyse des Systemes' HILARE robot. These robots all shared the common goal of integrating multiple sensor systems and developing algorithms to allow autonomous operation in uncertain environments. [Ref. 3: pp. 20-25]

The JPL Rover used a laser range finder, stereo TV cameras, tactile, and proximity sensors to observe its environment, and a gyrocompass and optical encoders on the wheels to keep track of its own position. Like the Shakey robot,

sensor integration and error propagation problems plagued the system and, due to the heavy computing requirements, the robot was dependent on its mainframe computer link. [Ref. 3: pp. 20-30]

The Stanford Cart employed a complex image processing scheme to improve obstacle recognition. A TV link to an off-board computer and a moveable, slide-mounted camera system were used to reduce imaging errors, but this system proved slow and very sensitive to light and shadow effects. Like previous robots, the Stanford Cart's dead reckoning system induced errors into the system which it could not overcome. [Ref. 3: pp. 20-25]

The French Hilare robot incorporated a number of techniques most often associated with artificial intelligence. "Expert system" modules worked together, sharing information about navigation, obstacle identification, etc., under a computing hierarchy of onboard microcomputers linked to an off-board minicomputer and mainframe. The Hilare used a laser range finder and a prediction algorithm to anticipate what the obstacle map would "look" like after the robot moved. It could then correct its perception and/or position information to merge the predicted and actual image information and thereby alleviate many of the errors experienced in the earlier systems. [Ref. 3: pp.20-25]

A departure from wheeled vehicles is the Ohio State University (OSU) Hexapod. The Hexapod is a six-legged walking machine which first operated in 1977. Although the OSU Hexapod is strictly a laboratory robot, a truly autonomous walking machine, like its animal counterpart in nature, would be

ideally suited to traversing rough, unpredictable terrain. The OSU machine's legs are controlled to provide static stability; that is, at least three legs are on the ground providing a stable supporting tripod for the robot at all times. The robot's sensors include stereo TV cameras and motion sensors to maintain stable body position while moving or climbing. While sophisticated computer routines automate many of the Hexapod's balance, leg movement, and coordination functions, the Hexapod remains tethered to its external power supply and minicomputer and relies on human interaction for navigation and for some foot-placement decisions. [Ref. 4: pp. 3-17]

2. Autonomous Land Vehicles

The Autonomous Land Vehicle (ALV) is the newest, and thus far the most successful, of the various autonomous wheeled robots which have been tested to date. Designed for the Defense Department's Advanced Research Projects Agency (DARPA), the vehicle is a hydraulically powered, eight-wheeled platform equipped with the computers and sensors required to negotiate unfamiliar territory without human intervention. The ALV's mission is conceptually more difficult than the robots previously described since it is designed to operate outdoors in a much more unpredictable environment. The ALV's sensors include a TV camera and optical radar for sensing the environment, and an inertial navigation system for position information. During initial testing, the sensor data was successfully integrated and processed, providing the vehicle with "road following" information able to keep it between ditches. [Ref 5]

Another ALV was built in 1985 by FMC Corporation. This vehicle is an armored personnel carrier which is a tracked, instead of wheeled vehicle. This ALV has an inertial navigation system, a vehicle control computer, and a sonic imaging sensor. The architecture of the FMC vehicle consists of 5 subsystems called a Planner, Observer, Mapmaker, Pilot, and Vehicle Control. In general, the Planner plans the route of the ALV using preloaded digitized maps of the local terrain. The Observer uses the sensors input to create an obstacle map, then the Mapmaker generates a pilot map using the information from the Planner and the obstacle map. The Pilot uses the pilot map and guides the vehicle along an optimum path by passing instructions to the Vehicle Control subsystem. The first test of the FMC ALV was conducted in 1985 in which it successfully avoided obstacles and performed path execution at a speed of 5 mph. [Ref. 6: pp. 14-23]

3. DARPA Adaptive Suspension Vehicle

A more advanced six-legged walking machine, called the *Adaptive Suspension Vehicle* (ASV), is currently undergoing test and evaluation at Ohio State University. The ASV differs from the OSU Hexapod in that it is completely self-sufficient (no external power or computing required). The ASV uses an internal combustion engine to provide power to its hydraulic systems and on-board computers. The ASV carries its own computers that control leg movement and stability, and provide vision information. The vision system consists of an optical radar mounted on top of the control cab. At present, the ASV requires an on-board human operator to plan and control the motion of its body using a

three-axis control stick. With continuing research, the ASV could become the first autonomous legged walking machine. [Ref. 1: pp. 1-60]

C. DESCRIPTION OF OPTICAL RADAR

1. Physical Description

The ASV is equipped with an optical terrain scanner. The optical scanner is manufactured by the Environmental Research Institute of Michigan (ERIM). The scanner is 26 inches wide, 12.5 inches high and weighs 75 pounds [Ref.1: pg. 35]. The optical scanner consists of a scanning mechanism, transmitter optical train, and receiver optical train. The scanning mechanism consists of a nodding mirror and a four-sided polygon mirror which combine to scan up and down and left and right. The transmitter optical train consists of a GaAlAs laser diode, collimating lens, anamorphic prism pair and a beam expansion telescope. [Ref. 7: pp. 3-4].

2. Performance Data

The optical scanner has a scan rate of 2 Hertz. The horizontal field of scan is +40 to -40 degrees in 128 increments while the vertical field of scan is from -15 to -75 degrees elevation in 128 increments. The instantaneous field of view is 0.5 degrees and the range resolution is 0.125 feet. The maximum horizontal range is 32 feet. [Ref. 7: pp. 2-3]

Due to various factors such as weather, obstacles and equipment errors, the range value returned by the ERIM scanner is not exact [Ref. 7: pg. 5]. The

ASV has or will have gyro or magnetic compasses for azimuth, elevation and roll inputs along with an altimeter for altitude information and geoposition satellite (GPS) inputs for horizontal position. Even though the ASV will be able to receive position inputs from GPS and the altimeter, they are inaccurate. That is, although the gyro's give angles within fractions of a degree, the errors associated with GPS and the altimeter are of the order of tens of feet; therefore, x, y, and z need to be corrected. Detailed information regarding these two sources of terrain mapping error was not available at the time of writing of this thesis.

D. ESTIMATION TECHNIQUES

1. Regression Analysis

In regression analysis, a number of noisy measurements of a variable are used to approximate a functional relationship. Using the notation of Ref. 8, if the measured variable of the system is

$$y_0(t) = y(t, \vec{z}_0) + \epsilon(t) \quad (2.1)$$

then the objective of regression analysis is to find a vector, $\vec{z}$, which approximates the true parameter vector, $\vec{z}_0$, in the presence of the measurement error, ϵ . Typically, this is accomplished by minimization of some type of sum squared error function.

If the response of the system under consideration can be represented linearly as

$$\bar{y}_0 = A\bar{z}_0 + \bar{z} \quad (2.2)$$

where $\bar{y}_0$ is a vector of samples of y_0 , A is the coefficient matrix, $\bar{z}_0$ is the true parameter vector, and $\bar{z}$ is a random error vector, then the *sum squared error function*, Φ , can be defined as

$$\Phi = (\bar{y}_0 - A\bar{z})^T (\bar{y}_0 - A\bar{z}) \quad (2.3)$$

where $\bar{z}$ is the trial parameter vector. To minimize Φ , the partial derivatives of Φ with respect to each component of $\bar{z}$ are equated to zero. Applying this to Eq. (2.3), the result is

$$\frac{\partial \Phi}{\partial \bar{z}} = \nabla \Phi = -2A^T \bar{y}_0 + 2A^T A\bar{z} = 0 \quad (2.4)$$

or

$$A^T A\bar{z} = A^T \bar{y}_0 \quad (2.5)$$

Solving Eq. (2.5) for the least squares estimate of the true parameter vector, the result is

$$\bar{z} = [A^T A]^{-1} A^T \bar{y}_0 \quad (2.6)$$

This is the *least squares parameter estimate* of the true parameter, $\bar{z}$. [Ref. 8: pp. 65-68]

2. Iterative Methods

The Kalman filter is an iterative method used to obtain an optimal estimate of a variable in an environment that has noise present. Unlike regression

analysis, in using a Kalman filter, it is not necessary to store past measurements for present or future computations. For the purposes of this discussion, the notation in Ref. 9 is used. It is assumed the reader has some knowledge of Kalman filtering.

Assume a system is described by

$$\bar{x}_k = \Phi_{k-1} \bar{x}_{k-1} + \bar{w}_{k-1} \quad (2.7)$$

and

$$\bar{z}_k = H_k \bar{x}_k + \bar{v}_k \quad (2.8)$$

where $\bar{x}_k$ is the state variable at time t_k , Φ_{k-1} is the transition matrix at time t_{k-1} , $\bar{w}_{k-1}$ and $\bar{v}_k$ are the random noise vectors with zero mean and covariances Q_k and R_k , respectively, $\bar{z}_k$ are the measurements, and H_k is the observation matrix. An updated estimate of the state, $\hat{\bar{x}}_k(+)$, based on the measurement $\bar{z}_k$ and the past estimate, $\hat{\bar{x}}_k(-)$, can be obtained from the recursive form

$$\hat{\bar{x}}_k(+) = K'_k \hat{\bar{x}}_k(-) + K_k \bar{z}_k \quad (2.9)$$

where K'_k and K_k are the time-varying weighting matrices (Kalman gain matrices).

[Ref. 9: pp. 60-110]

If $\tilde{\bar{x}}_k$ denotes the estimation error, such that

$$\tilde{\bar{x}}_k(+) = \bar{x}_k + \tilde{\bar{x}}_k(+) \quad (2.10)$$

and

$$\tilde{\bar{x}}_k(-) = \bar{x}_k + \tilde{\bar{x}}_k(-) \quad (2.11)$$

then Eqs. (2.8-2.11) yield

$$\tilde{\mathbf{x}}_k(+) = [K_k' + K_k H_k - I] \tilde{\mathbf{x}}_k + K_k' \tilde{\mathbf{z}}_k(-) + K_k \bar{\mathbf{v}}_k . \quad (2.12)$$

By definition, $E[\bar{\mathbf{v}}_k] = 0$ and if $E[\tilde{\mathbf{z}}_k(-)] = 0$, this estimator will be unbiased for any $\mathbf{x}_k$ if

$$K_k' + K_k H_k - I = 0 . \quad (2.13)$$

Thus, Eqs. (2.10) and (2.12) become

$$\hat{\mathbf{x}}_k(+) = \hat{\mathbf{x}}_k(-) + K_k [\mathbf{z}_k - H_k \hat{\mathbf{x}}_k(-)] \quad (2.14)$$

and

$$\tilde{\mathbf{x}}_k(+) = (I - K_k H_k) \tilde{\mathbf{x}}_k(-) + K_k \bar{\mathbf{v}}_k . \quad (2.15)$$

By definition, the error covariance matrix, P_k , is given by

$$P_k(+) = E[\tilde{\mathbf{x}}_k(+) \tilde{\mathbf{x}}_k(+)^T] \quad (2.16)$$

and

$$P_k(-) = E[\tilde{\mathbf{x}}_k(-) \tilde{\mathbf{x}}_k(-)^T] . \quad (2.17)$$

If it is assumed that the measurement errors are uncorrelated, then

$$E[\tilde{\mathbf{x}}_k(-) \bar{\mathbf{v}}_k^T] = E[\bar{\mathbf{v}}_k \tilde{\mathbf{x}}_k(-)^T] = 0 . \quad (2.18)$$

Applying Eq. (2.12) to Eqs. (2.16) and (2.17)

$$P_k(+) = (I - K_k H_k) P_k(-) (I - K_k H_k)^T + K_k R_k K_k^T . \quad (2.19)$$

The Kalman gain matrix, K_k , is defined as

$$K_k = P_k(-) H_k^T [H_k P_k(-) H_k^T + R_k]^{-1} . \quad (2.20)$$

Substituting Eq. (2.20) into Eq. (2.19)

$$P_k(+) = [I - K_k H_k] P_k(-) . \quad (2.21)$$

From Eqs. (2.14), (2.20) and (2.21), a recursive filter can be set up to obtain the optimal estimates of $\hat{z}_k(+)$ [Ref. 9: pp. 60-110].

E. SUMMARY

As has been discussed in this chapter, there have been in the past and are now in progress, many projects to develop autonomous vehicles. To become autonomous, such a vehicle ought to have a vision system and one such system was described. In the next chapter, methods for handling the vision information are applied to the problem addressed by this thesis.

III. PROBLEM STATEMENT AND PROPOSED SOLUTION METHOD

A. INTRODUCTION

Before a vehicle can become autonomous, it needs to know what the terrain it is operating in looks like then be able to pick a path it wants to travel along. In this chapter, some path selection techniques are discussed in general terms along with some detailed discussion on filtering of terrain data.

For the remainder of this thesis, when reference is made to an optical radar scanner, the ERIM discussed in Section C of Chapter II is assumed to be the one in use. This assumption will show up in performance data assumed for the data conversions and simulation.

B. IMMEDIATE-AREA TERRAIN SENSING AND PATH SELECTION

The need for information about the terrain in the immediate area of a walking machine is crucial. As long as the machine has a human driver, he can navigate the terrain manually, picking where to walk and, if necessary, placing the feet in appropriate positions. However, before a walking machine can become truly autonomous, it needs to be able to navigate on its own and pick its own path to walk along. Moreover, even in the case of a manned vehicle, it is highly desirable that the driver be concerned only with control of the body of the vehicle with foothold selection being accomplished automatically from vision data.

There have been many approaches and schemes for solving the problem of selecting adequate footholds and avoiding obstacles. The Autonomous Land Vehicle (ALV) uses a vision system that is a combination of a color video TV camera and the ERIM optical radar scanner [Ref. 5: pp. 19-23]. The color camera detects differences in color between the road and off-road areas, while the optical scanner is used strictly to detect differences in smoothness between the two areas. A major disadvantage of this system is that the color camera is greatly affected by changes in lighting and weather conditions.

Another obstacle avoidance system is used in the FMC ALV. In that system, the Observer uses information from a sonic image sensor to detect obstacles not already known on the stored digitized map and then creates an obstacle map. The sonic image sensor is a phased array sensor that has a range of 32 meters and an arc of 120 degrees. The sensor utilizes sonic waves and produces a map every 0.2 seconds. [Ref. 6: pp. 14-23]

With a vision system consisting of just an optical radar terrain scanner, the problem of autonomous navigation becomes even more difficult. Before an "optimal" path can be picked to walk along, accurate information about the terrain altitude in the immediate area has to be known. Once this information is available, a scheme can be used to decide what areas can be walked over and what areas cannot.

One possible terrain classification scheme was developed by Poulos in Ref. 10. In his work, a least squares quadratic surface is fitted to a surface pixel and its

eight neighbors. From the x, y, z information of each pixel, he uses the gradient vector and Hessian matrix and solves for the eigenvalues of the Hessian matrix. From these solutions, he classifies each pixel as a saddle, depression, ridge, plane, valley, hill or pass [Ref. 10: pp. 50-69]. To further refine the terrain picture, he also takes into account the slope associated with a pixel. Depending on what criteria is set for what is a safe slope to traverse, each pixel is also classified as being level or having a safe or unsafe slope. The property of being level, safe slope, or unsafe slope is called the *primary* terrain cell classification, while the above mentioned categories derived from the Hessian matrix are called the *secondary* classification [Ref. 10:pp.70-78].

With each terrain cell classified, it becomes possible to use some type of artificial intelligence technique to select the "optimal" path to traverse [Ref.11]. The only problem is to make sure that the terrain information provided to the classifying scheme is the best and most accurate possible.

C. TERRAIN SENSING USING THE OPTICAL RADAR

To be able to use the information obtained from the optical scanner, the data must first be converted into Cartesian coordinates. This conversion of the data will allow manipulation of any prior or future information that is not in the same data format as the incoming data [Ref. 7: pp. 2-3]. In order to convert the optical scanner information into Cartesian coordinates, it is necessary to use not only the

scanner information, but also the information from the Inertial Navigation System (INS).

To do the scanner data conversion, it was decided to model the INS and optical scanner systems as a nine link manipulator and to use the Denavit-Hartenberg (D-H) transformation for this purpose [Ref 12:pp. 36-41]. The inputs from the INS system consist of an x, y, z translation from the INS origin, and the body azimuth, elevation and roll angles. The inputs provided by the optical scanner are scanner elevation and azimuth angles and range to the terrain. The first three links and the last link of the model are translational transformations, while the others are rotational transformations. Figure 1 is the representation of the resulting nine link manipulator. Table 1 explains what each value in Figure 1 represents.

In drawing Figure 1 and setting up the D-H transformation matrices, the following assumptions were made: [Ref. 12:pp.36-41]

1. The reference coordinate system for the ASV body is positive z down, positive x out of the front of the vehicle, and positive y out the right side.
2. The z_{i-1} axis lies along the axis of motion of the ith joint and the z_i axis is normal to the z_{i-1} axis.
3. When possible, the direction of the ith twist axis, z_i , associated with a particular link in the manipulator model is picked to make the twist angle (α_i) positive.

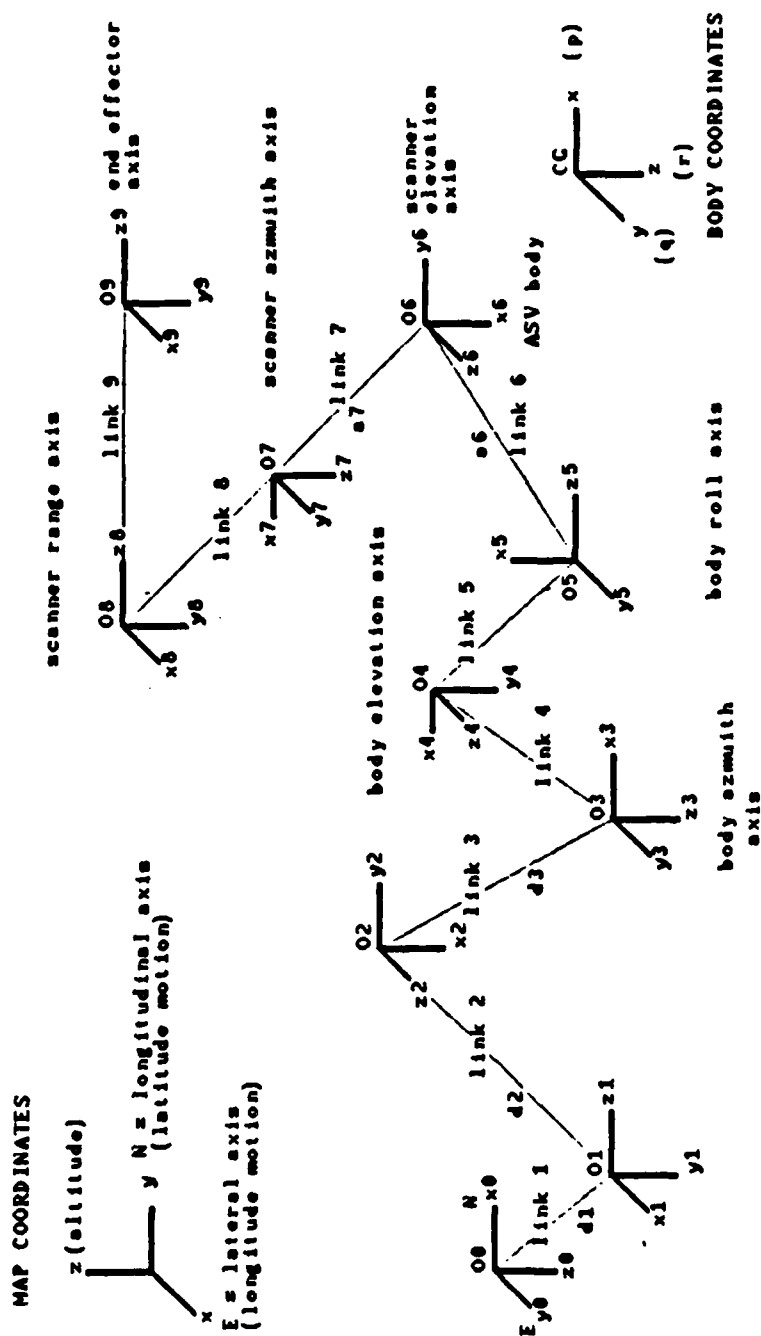


Figure 1
Nine Link Manipulator Representation

TABLE 1
D-H TRANSFORMATION SYMBOL MEANINGS

Symbol	Represents
d_1	displacement from the INS origin in the z direction
d_2	displacement from the INS origin in the x direction
d_3	displacement from the INS origin in the y direction
θ_4	ASV body azimuth angle
θ_5	ASV body elevation angle
θ_6	ASV body roll angle
θ_7	scanner elevation angle
θ_8	scanner azimuth angle
d_9	range to terrain
a_6	distance between INS origin and scanner origin
a_7	distance between scanner mirrors

The following definitions are needed for the discussion of the D-H transformation matrices for link i : [Ref. 12:pp.36-41]

1. Link length (a_i) is the linear displacement from the inboard motion axis to the outboard motion axis along the twist axis, (z_i).
2. Twist angle (α_i) is the angular displacement of the outboard motion axis, (z_i), from the inboard motion axis, (z_{i-1}), about the twist axis, (z_i).
3. Joint displacement (d_i) is the linear displacement of the outboard twist axis, (z_i), from the inboard twist axis, (z_{i-1}), measured along the inboard motion axis, (z_{i-1}).
4. Rotation angle (θ_i) is angular displacement of outboard twist axis, (z_i), from the inboard twist axis, (z_{i-1}), measured about the inboard motion axis, (z_{i-1}).

Using these assumptions and definitions, Table 2 shows the joint and link parameter values for the ASV scanner system. The values in Table 2 in parentheses are for the *reference configuration* shown in Figure 1 while the others are fixed values.

Using the information in Table 2, the general D-H transformation matrix, Eq. (3.1), can be used to form each link as follows:

$${}^{i-1}A_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & ac\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & as\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

In Eq. (3.1), ${}^{i-1}A_i$ is the D-H transformation matrix for link i going from origin ($i-1$) to origin i . Also, $c\theta_i$ represents $\cos\theta_i$, and $s\theta_i$ represents $\sin\theta_i$. To obtain the

TABLE 2
LINK AND JOINT PARAMETER VALUES

Link i	θ_i	d_i	α_i	a_i
1	90	d_1	90	0
2	90	d_2	90	0
3	90	d_3	90	0
4	θ_4 (180)	0	90	0
5	θ_5 (-90)	0	90	0
6	θ_6 (180)	0	90	a_6
7	θ_7 (-90)	0	-90	a_7
8	θ_8 (-90)	0	90	0
9	0	d_9	0	0

Cartesian coordinates of the terrain at the end of the scanner beam, each transformation matrix is multiplied in order as shown in Eq. (3.2).

$${}^0A_9 = {}^0A_1 z {}^1A_2 z {}^2A_3 z {}^3A_4 z {}^4A_5 z {}^5A_6 z {}^6A_7 z {}^7A_8 z {}^8A_9 \quad (3.2)$$

In order to simulate the scanning of terrain on the graphics computer, the matrix 0A_6 which specifies the position of the ASV body is computed and then multiplied by the matrix 6A_9 to get the coordinate transformation for the beam tip. Using Eq. (3.1) and Table 2, the following results are obtained

$${}^0A_6 = \begin{bmatrix} c4c5c6+s4s6 & c4s5 & c4c5s6-s4c6 & d_2 \\ s4c5c6-c4s6 & s4s5 & s4c5s6+c4c6 & d_3 \\ s5c6 & -c5 & s5s6 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

and

$${}^6A_9 = \begin{bmatrix} c7c8 & -s7 & c7s8 & d_9c7s8+a7c7 \\ s7c8 & c7 & s7s8 & d_9s7s8+a7s7 \\ -s8 & 0 & c8 & d_9c8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

In Eqs. (3.1), (3.3), and (3.4), the notation has again been abbreviated so that, for example, c7 stands for $\cos\theta_7$ and s7 stands for $\sin\theta_7$. Also, the origins of the scanner and the INS system are assumed to be in the same place, thus making $a_6 = 0$. While this is in fact not the case for the ASV, this assumption simplifies the computations needed to evaluate Eq. (3.2) and at the same time has no effect on the validity of the simulation studies carried out in the work of this thesis.

D. PROPOSED METHOD OF DATA FILTERING

Under the present scheme used by Ohio State University, the ASV scans the terrain using the optical radar scanner, and using whatever information the INS provides, a terrain map is calculated and stored. On all successive scans, the terrain map is recalculated with the given data, and the resulting new terrain elevation values replace the old ones. Each map is stored using the x, y, and z values calculated using the noisy range values provided by the optical radar scanner, which makes these x, y, and z values incorrect. The following discussion presents an approach to correcting for the z value errors separately from the x and y value errors. It is proposed to use Kalman filtering for the z values and regression analysis for the x and y values.

1. Stationary Walker Case

In the case where the ASV is standing still, the only significant errors introduced into the system are the noisy range values from the optical radar which in turn affect the calculated values of the altitude, z. In this case, to obtain the optimal values of the altitude for each terrain cell, Eqs. (2.14), (2.20) and (2.21) can be used. If the assumption is made that H_k in Eq. (2.14) is equal to 1, then Eq. (2.14) becomes

$$\hat{z}_k(+) = \hat{z}_k(-) + K_k[\bar{z}_k - \hat{z}_k(-)] \quad (3.5)$$

where $\hat{z}_k(+)$ is the estimate of the terrain altitude after the range measurement. Also, $\hat{z}_k(-)$ is the estimate before the measurement and K_k is the Kalman gain

matrix. Finally, $\bar{z}_k'$ is the value of the altitude calculated using the noisy range measurement.

From Eq. (2.20), $P_k(-)$ is defined as the variance of the error before the measurement and R_k is the variance of the measurement. For the purposes of this problem, R_k is the same as the variance of the altitude with respect to the range and $P_k(-)$ is the total variance of the altitude. Thus, R_k can be written as

$$R_k = \sigma_r^2 = \left(\frac{\partial z}{\partial \text{range}} \right)^2 \sigma_{scan}^2 \quad (3.6)$$

where σ_{scan}^2 is the variance associated with each scan of the optical radar scanner. In the case of $P_k(-)$, there is not only a variance in the z direction but there is also a variance in the x and y calculations due to the noisy range values. Thus, $P_k(-)$ can be written as

$$P_k(-) = \sigma_{total}^2 = \sigma_z^2 + \left(\frac{\partial z}{\partial x} \right)^2 \sigma_x^2 + \left(\frac{\partial z}{\partial y} \right)^2 \sigma_y^2 \quad (3.7)$$

Using Eqs. (3.6) and (3.7), Eqs. (2.20) and (2.21) can be written as

$$K_k = \sigma_{total}^2(-) [\sigma_{total}^2 + \sigma_r^2]^{-1} \quad (3.8)$$

and

$$P_k(+) = [\sigma_z^2 \sigma_{total}^2(-) | \sigma_z^2 + \sigma_{total}^2(-) |^{-1} \quad (3.9)$$

2. Compensating for Walker Motion

When the ASV is walking, or when there is an INS initialization error, the problem becomes one of computing the optimal value for the altitude and ensuring it is stored in the correct terrain cell. Due to the inaccuracies of the

altimeter and GPS inputs discussed in Section C of Chapter II, it is necessary to estimate the difference in the horizontal coordinates, Δx and Δy . For this part of the problem, regression analysis will be used to select the correct terrain cell. Then the optimal altitude value calculated from Kalman filtering will be stored. Two methods of regression analysis will be discussed for use in this problem: the gradient descent method and a grid search method.

a. Gradient Descent Method

In this approach, there is a criterion function, Φ , for a given set of parameters. The criterion function will be the sum squared error between the range returned by the optical radar scanner, $\hat{R}$, and the range computed from the internal terrain map pre-stored in the ASV computers, R . From this, Eq. (2.3) can be written as

$$\Phi = \sum (\hat{R} - R)^T (\hat{R} - R) \quad (3.10)$$

To correct for the x and y coordinate drift, the gradient of Φ , $\nabla\Phi$, and the Hessian matrix, H , will be estimated using a 3 x 3 grid around the INS values of the x and y coordinates. Using the procedure described by Poulus in Ref. 10, the 3 x 3 terrain cell mask is set up with the cell of interest at point where $x=0$ and $y=0$ with its associated value of Φ_0 . Table 3 shows the relative distances of the cells and the criterion functions associated with them.

A quadratic function can be fitted to the cells of interest as a function of x and y which has the form

TABLE 3
RELATIVE DISTANCES OF TERRAIN
CELL MASK AND CRITERION FUNCTIONS

x=-1 y=1 Φ_1	x=0 y=1 Φ_2	x=1 y=1 Φ_3
x=-1 y=0 Φ_4	x=0 y=0 Φ_0	x=1 y=0 Φ_5
x=-1 y=-1 Φ_6	x=0 y=-1 Φ_7	x=1 y=-1 Φ_8

$$\hat{R} = k_1 + k_2x + k_3y + k_4x^2 + k_5xy + k_6y^2 \quad (3.11)$$

Eq. (3.11) can be written as

$$\hat{R} = A_{9 \times 6} k_{6 \times 1} \quad (3.12)$$

where the subscripts are the dimensions of the matrices and

$$k = (k_1 \ k_2 \ k_3 \ k_4 \ k_5 \ k_6)^T \quad (3.13)$$

and

$$A_i = (1 \ x_i \ y_i \ x_i^2 \ x_i y_i \ y_i^2) \quad (3.14)$$

By substituting Eqs. (3.12-3.14) into Eq. (3.10) and taking the partial derivatives with respect to the k_i 's, Eq. (2.4) becomes

$$\frac{\partial \Phi}{\partial k_i} = \nabla \Phi = -2A^T \Phi^T + 2A^T A k = 0 \quad (3.15)$$

or

$$A^T A k = A^T \Phi^T \quad (3.16)$$

Rearranging terms, Eq. (2.6) can be written as

$$k = [A^T A]^{-1} A^T \Phi^T \quad (3.17)$$

The gradient of the terrain cell of interest is [Ref. 10: pg. 59]

$$\nabla \Phi = k_2 \hat{i} + k_3 \hat{j} \quad (3.18)$$

and the Hessian matrix is [Ref. 10: pg.60]

$$H = \begin{bmatrix} k_4 & 2k_5 \\ 2k_5 & k_6 \end{bmatrix} \quad (3.19)$$

Applying Eqs. (3.18) and (3.19), the optimum amount of Δx and Δy can be computed as

$$(\Delta x, \Delta y) = H^{-1} \nabla \Phi \quad (3.20)$$

which leads to the best estimate for the x and y coordinates of the terrain cell as

$$(\hat{x}, \hat{y})_{\text{optimum}} = (X_{INS} + \Delta x, Y_{INS} + \Delta y) \quad (3.21)$$

where X_{INS} and Y_{INS} are the coordinates supplied by the INS system.

In order to be able to compute the optimum values of Δx and Δy , the values of the k's in Eq. (3.17) have to be computed. Equation (3.17) can be rewritten as

$$k = [A^T A]^{-1} A^T \Phi^T = B \Phi^T \quad (3.22)$$

where

$$B = [A^T A]^{-1} A^T \quad (3.23)$$

To avoid repeated lengthy matrix computations, Ref. 10 shows B to be

$$B = \begin{bmatrix} -1/9 & 2/9 & -1/9 & 2/9 & 5/9 & 2/9 & -1/9 & 2/9 & -1/9 \\ -1/6 & 0 & 1/6 & -1/6 & 0 & 1/6 & -1/6 & 0 & 1/6 \\ 1/6 & 1/6 & 1/6 & 0 & 0 & 0 & -1/6 & -1/6 & -1/6 \\ 1/6 & -1/3 & 1/6 & 1/6 & -1/3 & 1/6 & 1/6 & -1/3 & 1/6 \\ -1/4 & 0 & 1/4 & 0 & 0 & 0 & 1/4 & 0 & -1/4 \\ 1/6 & 1/6 & 1/6 & 1/3 & -1/3 & -1/3 & 1/6 & 1/6 & 1/6 \end{bmatrix} \quad (3.24)$$

b. Grid Search Method

For this search method, the assumption is made that the criterion function, Φ , is definitely reduced when a sufficiently small step is taken in the direction of the actual terrain cell. Applying this to the terrain cell mask in Table 3, the distances between each cell will be assumed to start at one standard deviation, σ , of the INS system estimate of x and y instead of a unit displacement. After each computation of the Φ 's in the 3×3 mask, the least value Φ is moved to the center of the mask and the coordinates of that terrain cell become the terrain cell of interest and the 3×3 mask and the associated Φ 's are again calculated with each cell being displaced by σ . Once the least value of Φ remains in the center cell for two successive searches, then the displacement of

each cell, Δ , becomes

$$\Delta = \left(\frac{1}{2^n}\right) \sigma \quad n=1,2,3 \quad (3.25)$$

The advantage of this type of search is that it does not depend on the cell of interest and its surrounding neighbors being able to be fitted with a quadratic function and thus can be applied to a larger variety of terrains.

E. SUMMARY

This chapter has presented the necessity for an accurate terrain altitude map before a vehicle can become autonomous. After the data from an optical radar scanner is converted into a form that can be used for route planning, there are various ways to compute the optimal altitude values and store them in the proper terrain cell. One method for computing the optimal altitude values was presented along with two ways to determine the proper terrain cell to store the altitudes. In the next chapter, the simulation model is presented that will be used to test these data manipulation methods.

IV. EXPLANATION AND JUSTIFICATION OF SIMULATION MODELS

A. INTRODUCTION

In Chapters II and III, various ways of handling data with INS and optical scanner errors were discussed. In order for these schemes to be tested, it is necessary to make use of either the ASV or similar vehicle, or an accurate computer simulation. The obvious choice for this thesis is the computer. In this chapter, a simulation model is presented that will allow simulation of terrain scanning by the optical scanner and then data manipulation using the schemes presented earlier.

B. TERRAIN AND OPTICAL SCANNER MODELS

The computer chosen for the simulation studies of this thesis is an ISI Optimum V Workstation. This is a graphics workstation with an UNIX4.2BSD operating system. The workstation has a high-resolution display for a two-dimensional, black and white graphics display. The display is 1280 pixels wide and 1024 pixels high with the upper left corner of the screen functioning as the reference point. The operating system has installed libraries of graphics routines which are accessed by the computer language C, which is the language chosen for this thesis. These libraries provide graphics tools which make it easy to run a visual simulation. [Ref. 13]

In order to simulate the operation of the optical scanner, a simulated terrain is needed. The method for generating and drawing this terrain is the same as that presented in Ref. 10. An *elevation oblique* projection is used in this thesis to display a three-dimensional model of the terrain [Ref. 14: pp.272-316]. In the elevation oblique representation, lengths and angles of lines in a vertical plane are preserved while others are distorted. Figure 2 shows the coordinate systems used for the Cartesian coordinate system, the ISI screen coordinate system, and the image coordinate system. Using these coordinate systems, the following equations apply:

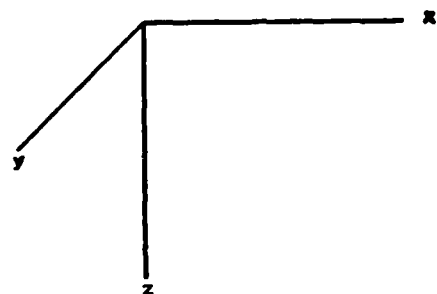
$$u = x - .5y \quad (4.1)$$

$$v = -z - .5y \quad (4.2)$$

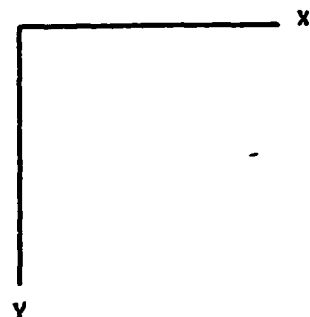
$$X = X_{origin} + u \quad (4.3)$$

$$Y = Y_{origin} - v \quad (4.4)$$

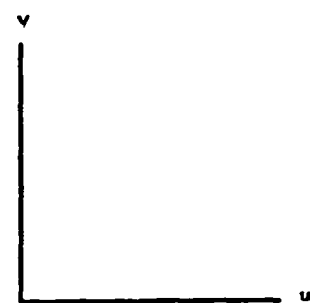
The use of the multiplier value of 1 in Eqs. (4.1) and (4.2) means that each pixel in x and z corresponds to one inch in physical units. The use of a scale factor of .5 for the y coordinate in both of the equations means that y is *foreshortened* by a factor of .707 relative to the scale of x and z. With these scale factors, the terrain shown in Figure 3 is of dimension 64 ft. x 64 ft. With Eqs. (4.3) and (4.4), the reference of the terrain is set in the lower left corner of the terrain.



(a)



(b)



(c)

Figure 2
Coordinate Systems
a) Cartesian , b) Screen , c) Image

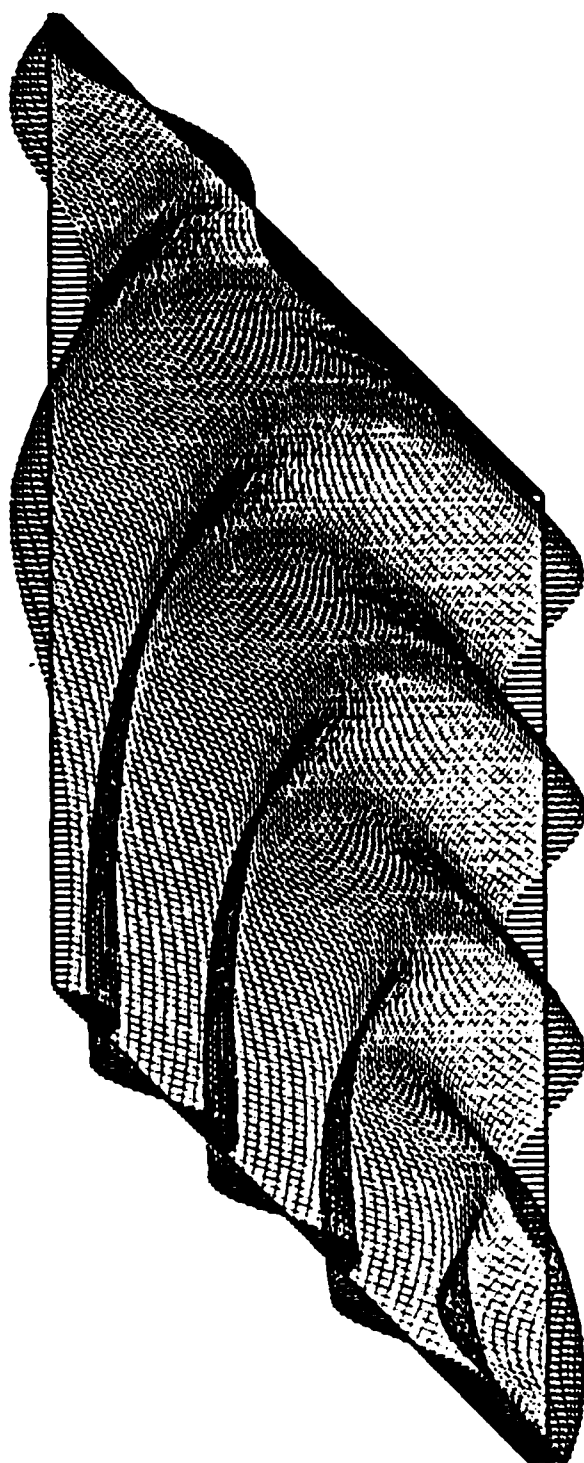


Figure 3
Simulated Terrain

The terrain data is generated by

$$z = r4 \cos\left(\frac{\sqrt{(x^2 + y^2)}}{50} 2\pi r5\right) \quad (4.5)$$

where r3 and r4 are random numbers and x and y are the horizontal coordinates of the altitude map measured in inches. Figure 3 is a picture of typical terrain generated by Eq. (4.5).

Figure 4 is a flow chart for the simulation program. The computer code in Appendix A is commented for further explanation of each segment of code. Figure 5 is a flow chart of the simulation of the optical scanner scanning the terrain. In scanning the terrain, the optical scanner's beam length is increased one pixel (one inch) at a time. After each increase of the beam, the x, y, and z coordinates of the end of the beam are calculated using the D-H transformation discussed earlier. The calculated value of z is then compared to the value of the altitude stored at the x and y coordinate of the actual terrain data. If the calculated value of z is greater than the actual terrain value, the beam length is increased and the process starts again. If the calculated value is less than the actual terrain value, then the optical scanner's range is set to

$$R = R_{beam} - .5 \quad (4.6)$$

where R_{beam} is the range to the z value that is less than the actual terrain value and the .5 adjustment brings the range back to a value closer to the actual range.

The actual ERIM optical scanner scans the terrain in 128 increments, both in elevation and azimuth. However, due to the time requirements of simulating the

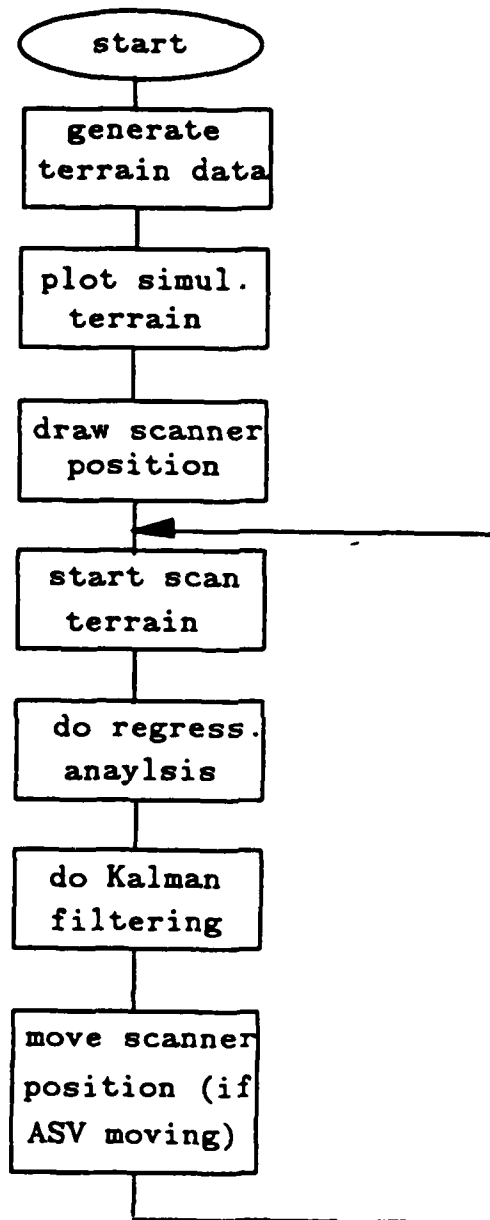


Figure 4
Simulation Flow Chart

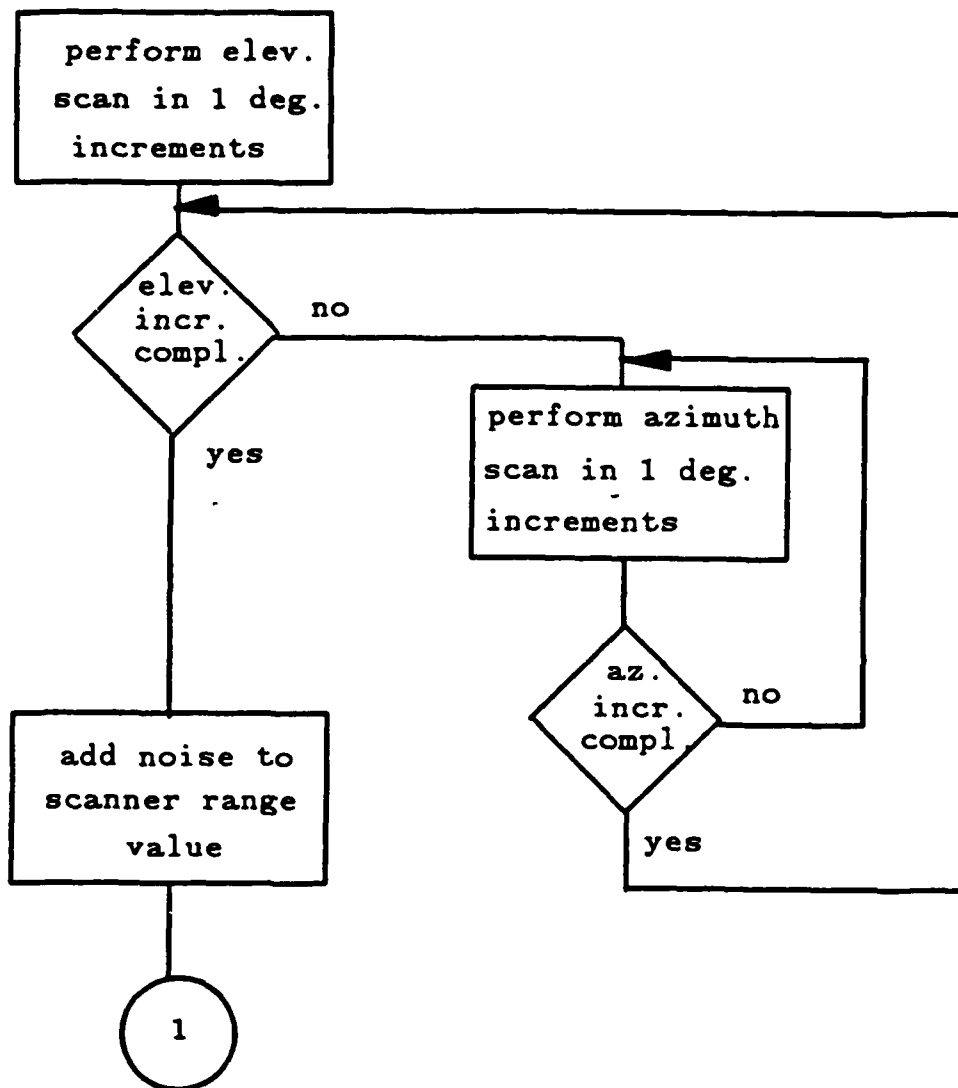


Figure 5
Scanning Flow Chart

scanning process, the simulation of this thesis uses twelve increments of one degree each in both the azimuth and elevation.

It is assumed that the error associated with the the optical scanner is range dependent; that is, the greater the range to the surface, the greater the error of the returned range value. For the purposes of this thesis, this error will be represented by

$$\sigma_{scan}^2 = \sigma_0^2 + \sigma_1^2 R^2 \quad (4.7)$$

where σ_{scan}^2 is the total variance of the scan and R is the range from Eq. (4.6).

Also, lacking any information about the scanner errors, σ_0^2 and σ_1^2 are set at

$$\sigma_0^2 = .02 \text{ in}^2 \quad (4.8)$$

and

$$\sigma_1^2 = .04 \quad (4.9)$$

After the variance in the range measurement is computed, the new value of the range, $\tilde{R}$, is computed by

$$\tilde{R} = R + n\sigma_{scan} \quad (4.10)$$

where n is the output of a gaussian random noise generator with zero mean and unit variance and

$$\sigma_{scan} = \sqrt{\sigma_{scan}^2} \quad (4.11)$$

C. DATA HANDLING DURING SIMULATION

Once the terrain has been scanned by the optical scanner, regression analysis is applied to compute any errors in the INS values for the x and y coordinates. Figure 6 is a flow chart of the gradient search method of regression analysis. Using the INS values of the x and y coordinates, ϕ_0 is computed using Eq. (3.10). The INS value of the x and y coordinates are then changed by the values in Table 3 and the associated ϕ 's are computed. To compute the ϕ 's, the terrain is scanned, but this time without noise added to the range value. In the actual ASV, this would be done internally with the pre-stored terrain map of the area and not an actual scan of the terrain. From these values of ϕ , Eqs. (3.22) and (3.19) are used to compute the k 's and the H matrix. With these, Δx and Δy are determined by Eq. (3.20). With these values of Δx and Δy , the INS values for x and y can be adjusted by Eq. (3.21).

The second regression analysis method is the grid search method. Figure 7 is the flow chart for this search method. It is much the same as the gradient search method except that the INS values of the x and y coordinates are changed by the standard deviation, σ , of the INS estimate of x and y . After each cell's ϕ is computed, the minimum value of ϕ is determined, and if it is not in the center cell, then it is moved to the center along with its associated value of x and y . If the minimum ϕ is already in the center cell, it is left there. If the minimum ϕ was not in the center cell, then after it is moved to the center, the x and y values are again changed by σ . This process continues until the minimum ϕ stays in the

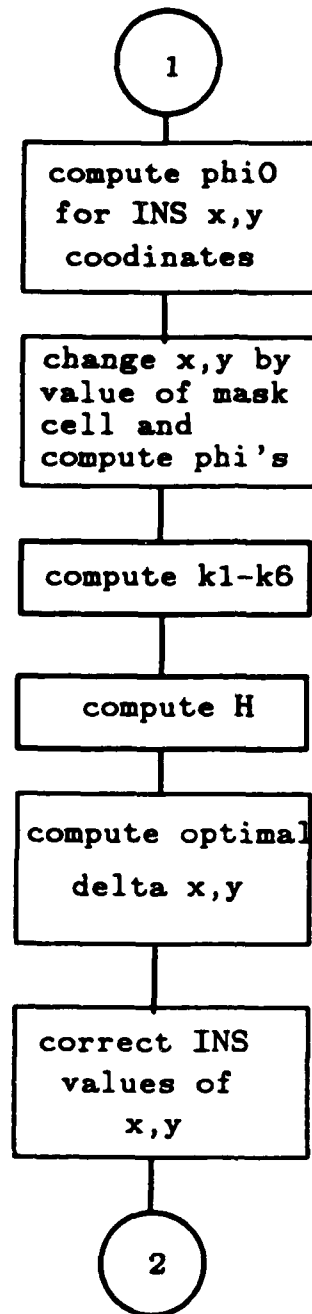


Figure 6
Gradient Search Flow Chart

center cell for two successive iterations. Once this condition is satisfied, the x and y values are changed by $\sigma/2^*$ and the Φ 's are again computed. This process continues for $n=1,2,3$. Upon completion of the search method, the cell with the minimum Φ has the new INS value of x and y to replace the old values.

After completion of the regression analysis, the Kalman filter is used to store the optimal values of the altitude in the appropriate altitude map cell. The cell indices are determined using the new INS values for x and y determined by regression analysis. Figure 8 is the flow chart for the Kalman filtering. For initialization of the Kalman filter, if a cell in the altitude map does not have a stored value for σ_{old}^2 , then the σ_{old}^2 of that cell is set to be $(120)^2$, which is the square of the vehicle's height. From Eq. (3.7), the partial derivatives with respect to x, y, and z are dependent on the nature of the terrain. Eqs. (4.12) and (4.13) show that σ_z^2 and σ_y^2 are given by

$$\sigma_z^2 = \sigma_R^2 \left(\frac{\partial z}{\partial R} \right)^2 \quad (4.12)$$

$$\sigma_y^2 = \sigma_R^2 \left(\frac{\partial y}{\partial R} \right)^2 \quad (4.13)$$

For purposes of this thesis, the values of σ_z^2 and σ_y^2 are set to 1, but in reality, they are dependent on range as well as azimuth and elevation scan angles.

From the terrain data generated for this simulation, an average value for $\left(\frac{\partial z}{\partial x} \right)^2$ was determined from the difference in the altitude value in the x direction

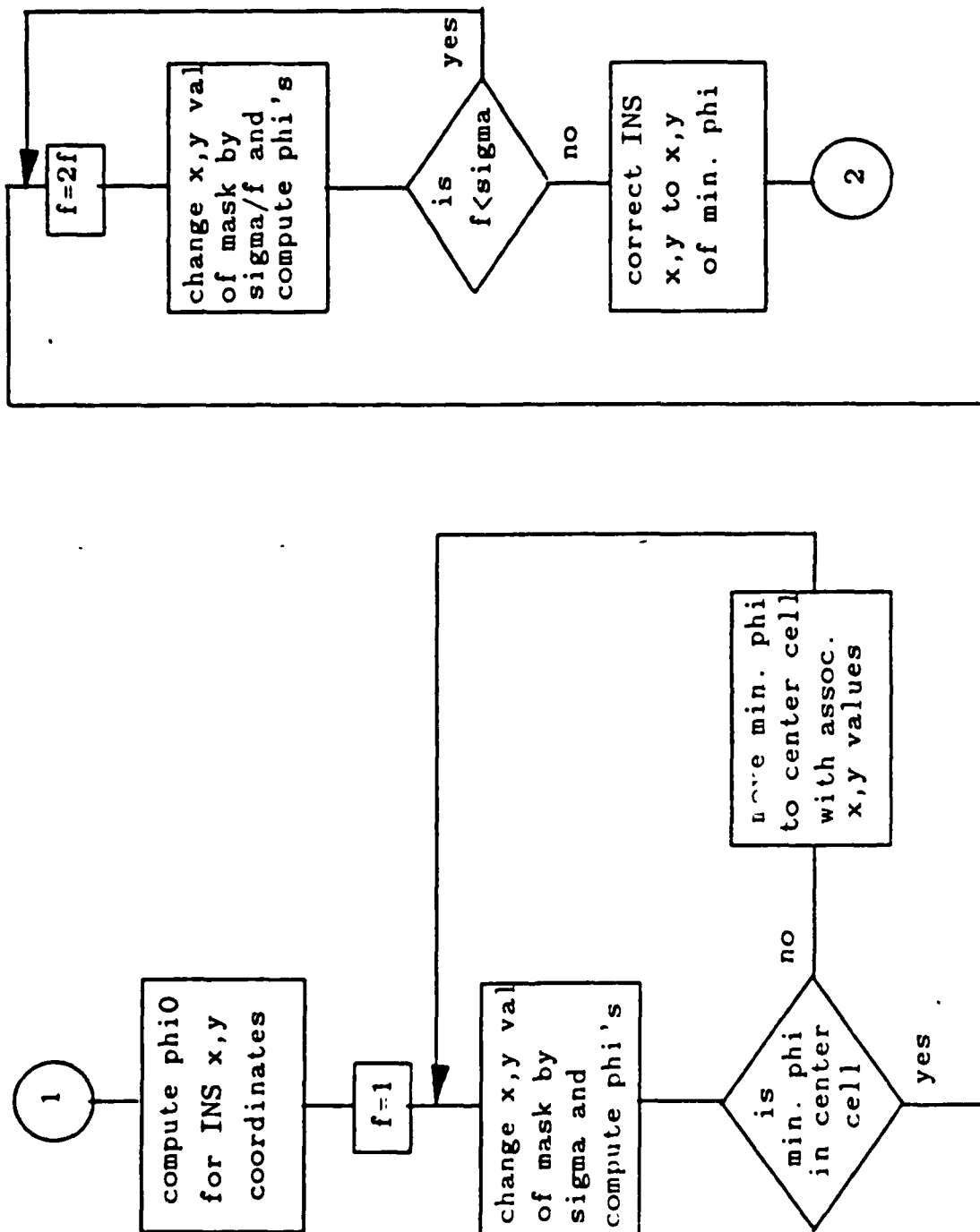


Figure 7
Grid Search Flow Chart

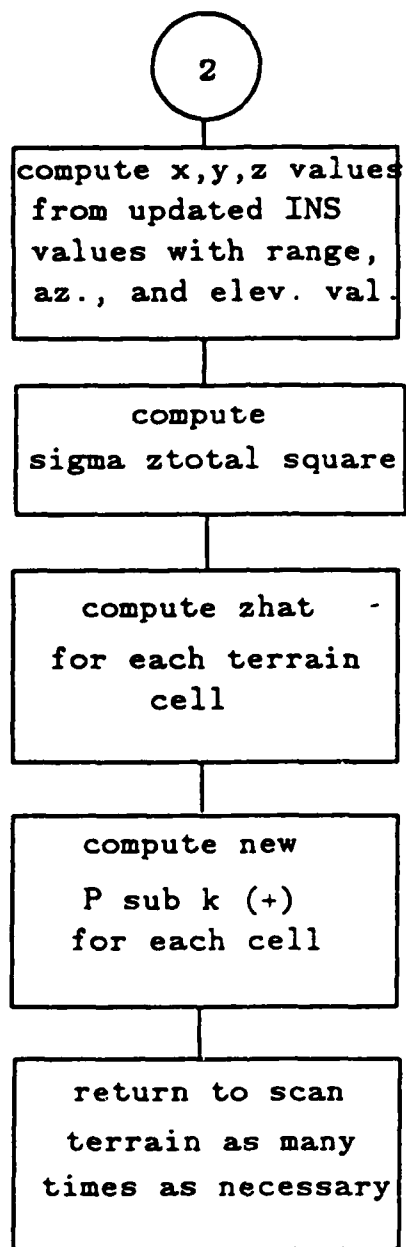


Figure 8
Kalman Filtering Flow Chart

with the y value held constant. From these calculations, the average value of this quantity is

$$\left(\frac{\partial z}{\partial x}\right)^2 = .116 \text{ in}^2 \quad (4.14)$$

The average value for $\left(\frac{\partial z}{\partial y}\right)^2$ was calculated in the same manner as that for $\left(\frac{\partial z}{\partial x}\right)^2$ with the results the same as those in Eq. (4.14). Next, σ_z^2 is set to be

$$\sigma_z^2 = \sin^2 \Theta_T \sigma_{scan}^2 \quad (4.15)$$

With the value of σ_{total}^2 computed, Eqs. (3.5) and (3.9) are applied to each cell of the altitude map to store the optimal value of the altitude. Upon completion of the filtering, the program returns to scanning the terrain.

D. SUMMARY

In this chapter, the computer simulation was discussed. The creation and scanning of the simulated terrain and the scanning of it was presented along with the two regression analysis schemes and the Kalman filtering scheme for handling the data obtained from the scanning. In the next chapter, the results of the simulation runs will be discussed.

V. SIMULATION PERFORMANCE

A. INTRODUCTION

In the previous chapters, methods for correcting INS drift errors were presented that might contribute to an effort to make the ASV an autonomous vehicle. After development of a computer simulation model in Chapter IV, numerous simulation runs were needed to validate not only the model but the various schemes for handling data so that an accurate terrain altitude map can be developed for future use in possible route planning. In this chapter, simulation runs are planned and run for a stationary walker and a walker with movement involved. Along with the results, some conclusions about these results are presented.

B. SIMULATION RESULTS

1. Stationary Walker Results

The first test of this thesis was to run the simulation using a stationary ASV. In this case, the walker is in a permanent position and it scans the same terrain over and over while performing Kalman filtering on the altitude information. Figure 9 shows the simulated terrain with the area being scanned indicated on it. In this figure, the optical scanner is represented by a small square and there is a line drawn from the scanner down to the terrain to give the viewer

an idea of the scanner's relative position over the terrain. Also, the INS origin is set to be the lower left hand corner of the terrain. Due to the assumptions made in Chapter III, Section C, the x coordinate increases positive to the right from the INS origin along the horizontal axis, and the y coordinate increases negative along the other (diagonal) axis.

To evaluate the effectiveness of the Kalman filtering presented in this thesis, three terrain cells are looked at to see how the optimal altitude value changes as the terrain is scanned in a stationary position. To get a representation of how the filtering is working at different ranges, the three terrain cells are selected at scanner elevations of -26, -41, and -61 degrees. Figure 10 is a plot of the three terrain cells altitudes stored after each scan after Kalman filtering has been applied. To check that the filtering works no matter what the noisy range values returned by the optical scanner are, a second simulation was run on the same terrain in the same position. Figure 11 shows the results obtained for the same three terrain cells.

The next test made on the stationary walker was done with the scanner in a different position on the terrain and pointing in a different direction. Figure 12 shows the scanned terrain for this case. Figure 13 shows the results of the Kalman filtering on three terrain cells at the same elevation angles used in the first case. Figure 14 shows the same three terrain cells with different noisy optical scanner range values.

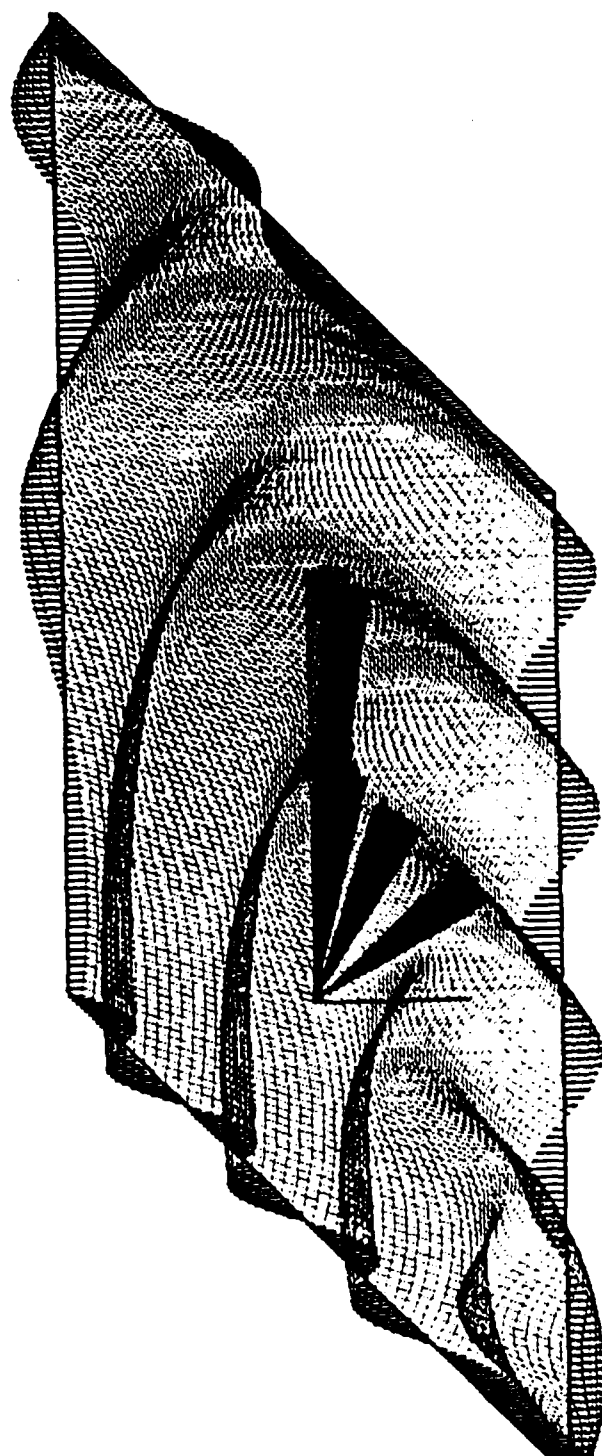
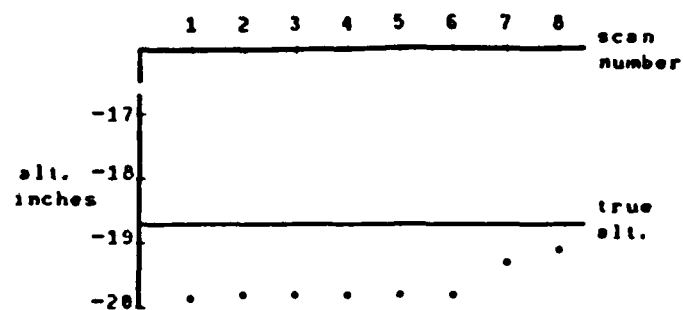
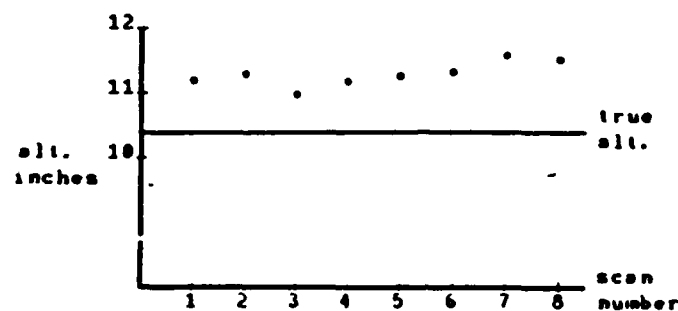


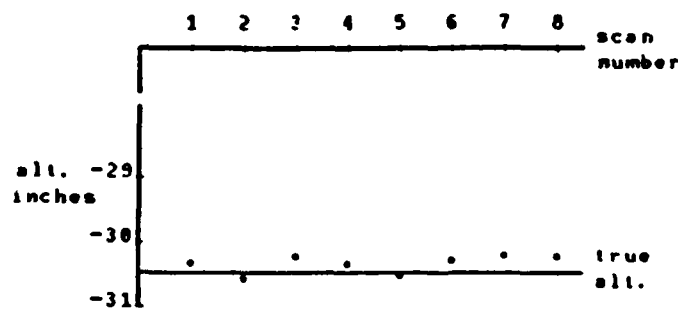
Figure 9
Scanned Terrain 1



(a)

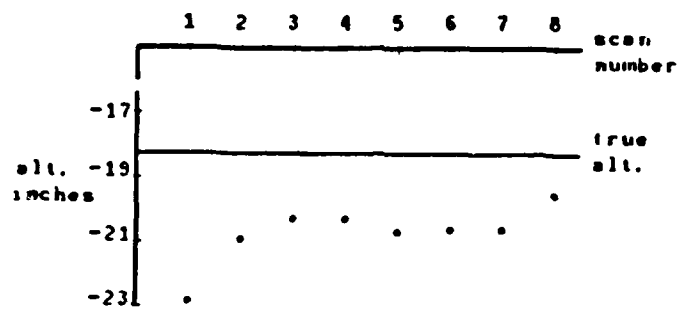


(b)

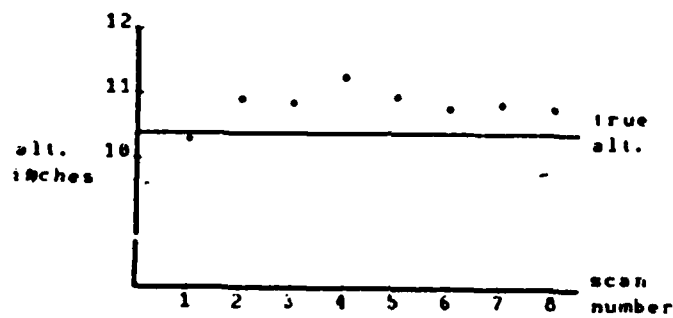


(c)

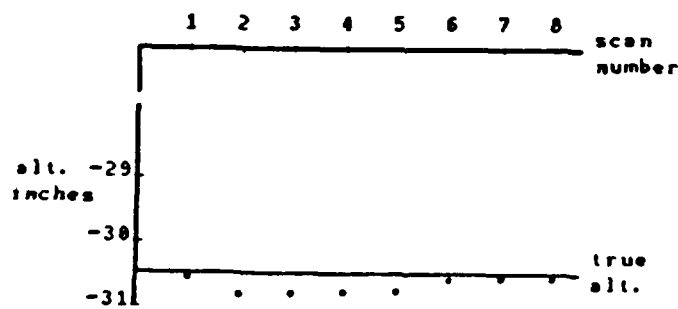
Figure 10
Kalman Filtering Results, Run 1
Stationary Walker on Terrain 1



(a)



(b)



(c)

Figure 11
Kalman Filtering Results, Run 2
Stationary Walker on Terrain 1

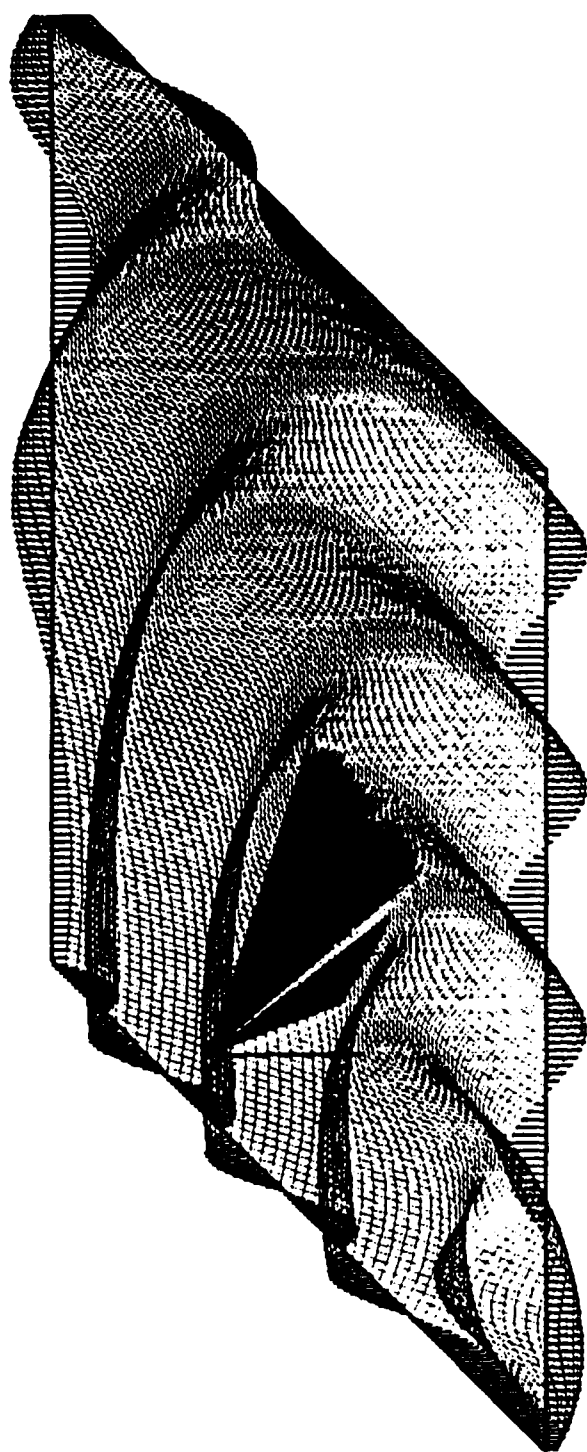
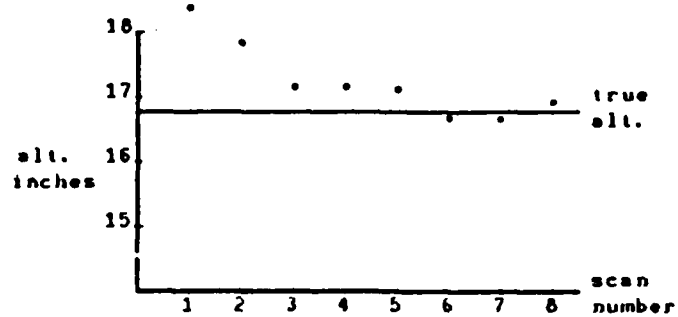
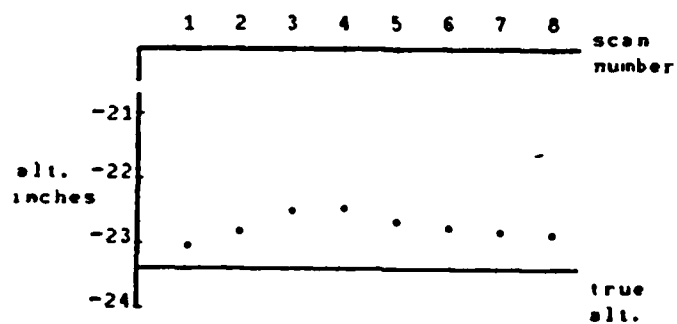


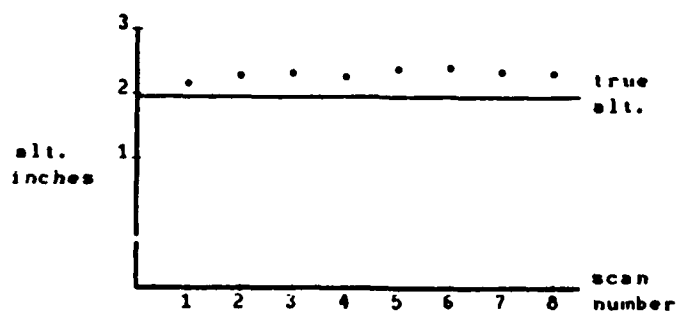
Figure 12
Scanned Terrain 2



(a)

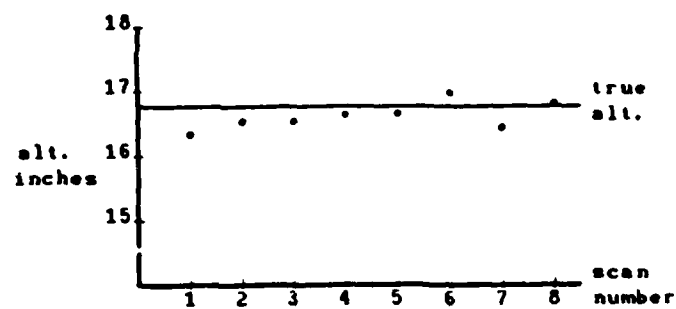


(b)

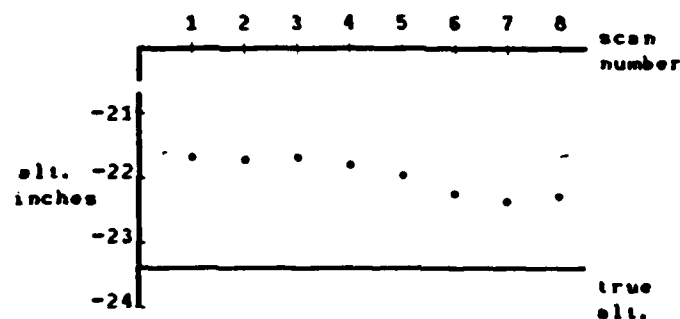


(c)

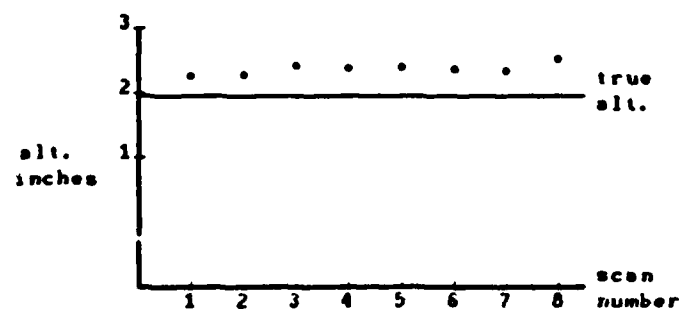
Figure 13
Kalman Filtering Results, Run 1
Stationary Walker on Terrain 2



(a)



(b)



(c)

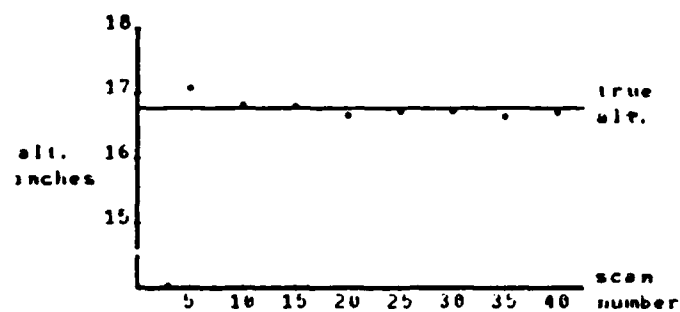
Figure 14
Kalman Filtering Results, Run 2
Stationary Walker on Terrain 2

The last simulation conducted for the stationary walker case was to scan the same terrain shown in Figure 12 and let the scanner scan it for 40 scans. Figure 15 shows the results of the 40 scans for the same terrain cells looked at in Figure 13.

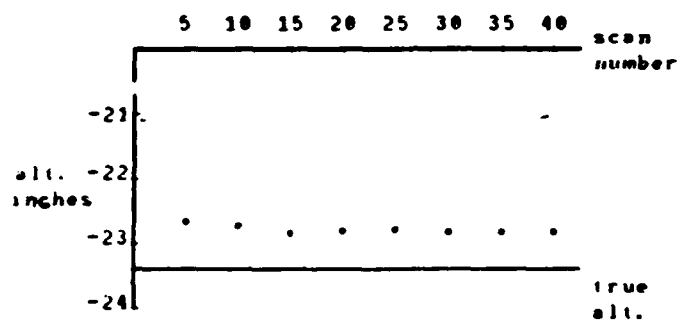
2. Moving Walker Case

The next case to look at was the walker with motion involved. To get the optimal altitude value for each terrain cell, errors in the INS values of the x and y coordinates must be corrected for so the altitude value can be stored in the correct cell. For the purposes of this thesis, the net effects of walker motion are represented by an INS initialization error or offset.

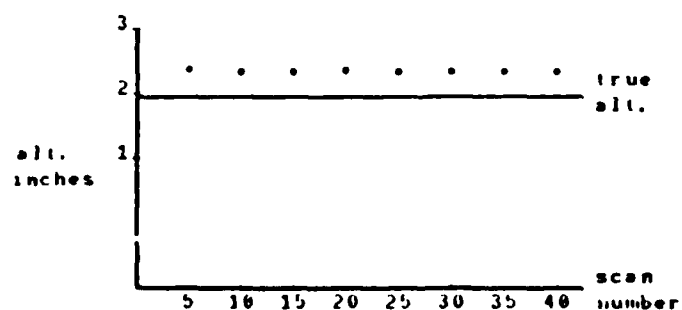
The first method used to correct for the INS errors was the gradient descent method. The scanner was placed in the same position as represented in Figure 9 and given an initialization error of $\Delta x = 5$ inches and $\Delta y = -4$ inches. After the first scan, gradient descent was applied using Eq. (3.20), and the optimum values for Δx and Δy were 10 and 5 respectively. A second run was conducted with different INS errors and the results were again not close to the value they should have been. The reasons for this failure are not known, but appear to be associated with the inadequacy of a quadratic approximation to the squared error function, Φ . This problem might be solved by a more elaborate gradient descent method as discussed in Ref. 8, but this approach was not investigated in this thesis.



(a)



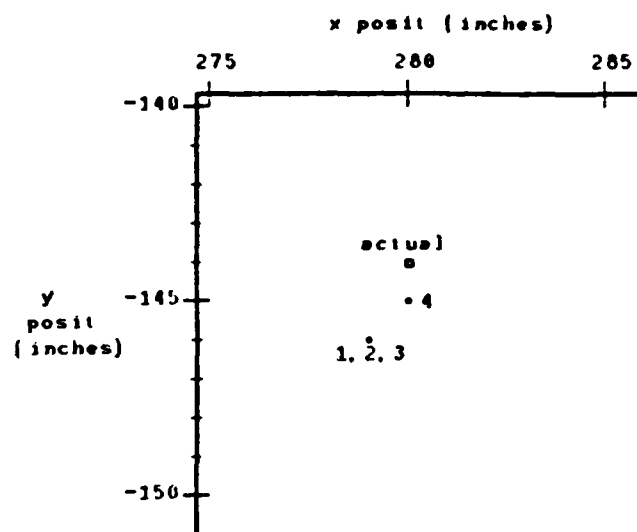
(b)



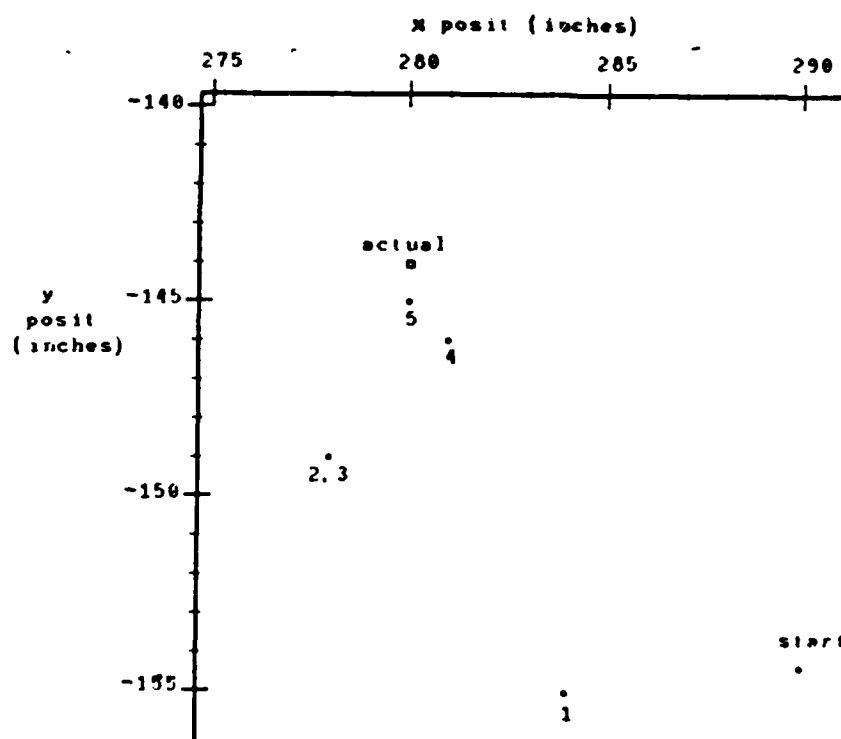
(c)

Figure 15
Kalman Filtering Results After 40 Scans
Stationary Walker on Terrain 2

After the apparent failure of the gradient descent method, the grid search method was applied to the moving walker case. The first simulation run using the grid search method was with the scanner in the position indicated in Figure 9 and with the standard deviation of the INS offset given by, $\sigma = 6$ inches. This σ was an assumption and is used in the terrain cell mask described in Chapter III, Section D. To test the effectiveness of the grid search method, a run was conducted with the INS error less than σ , another run with the error between σ and 2σ , and a third run with the error greater than 2σ . Figure 16a shows the results of the regression analysis with the INS error less than σ . The plot shows the actual x and y values, the starting point of the INS x and y coordinates and the numbers indicate the steps of the grid search. The final value is the value returned to be used in the Kalman filtering by the grid search at the end of the first actual terrain scan. As Figure 4 shows, the regression analysis is repeated each time a new frame of noisy range values is obtained from the simulation. Figure 16b shows the results of the grid search after the first scan with the INS error between σ and 2σ . After the second scan, the data showed that the grid search method locked in on the actual x and y coordinates and stayed there for subsequent scans. Figure 17 shows the results of the grid search after the first scan with the INS error greater than 2σ . The data collected also shows that the grid search method would not lock in on the actual x and y values on the succeeding scans when the error was greater than 2σ .



(a)



(b)

Figure 16
Grid Search Results With $\sigma = 6$ inches
a) $\text{INS Error} < \sigma$, b) $\sigma < \text{INS Error} < 2\sigma$

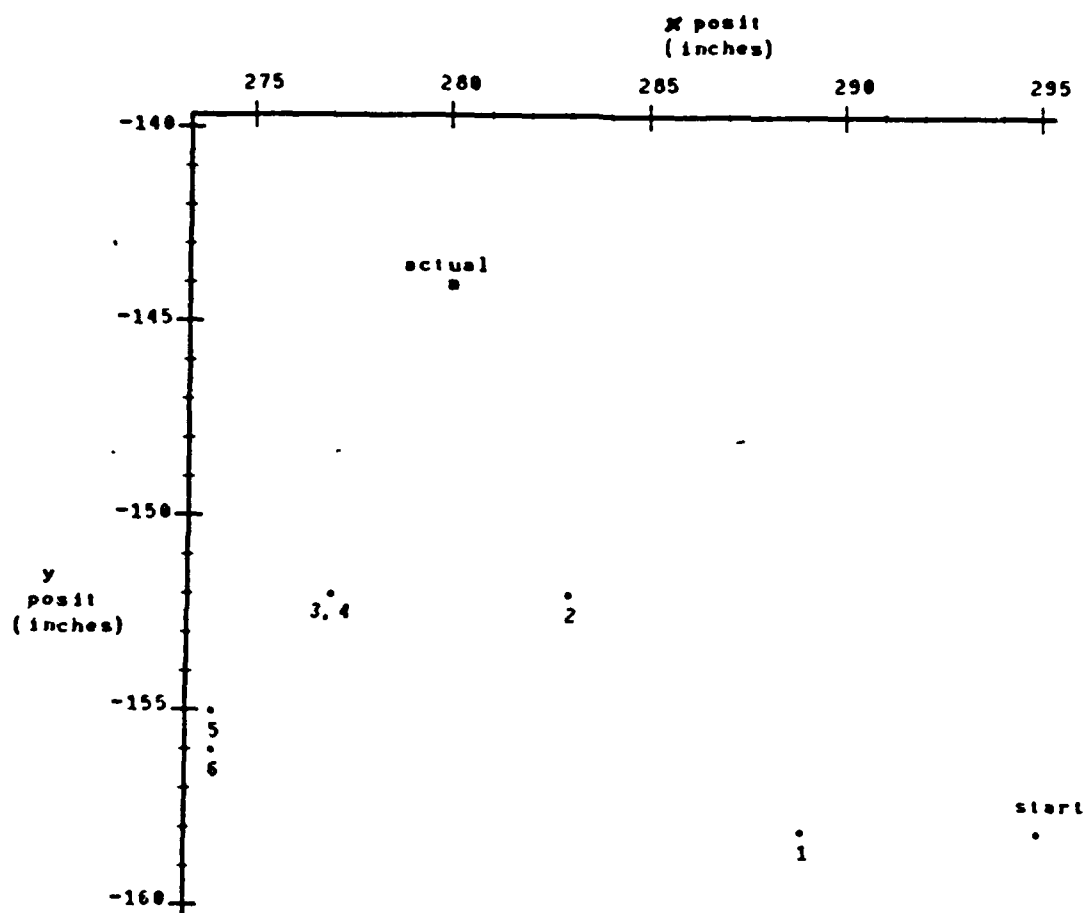
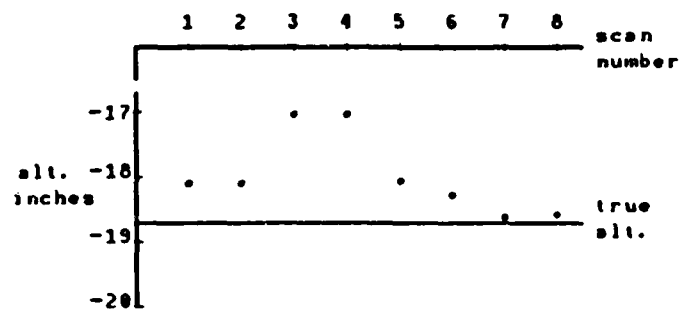


Figure 17
Grid Search Results With $\sigma = 6$ inches
INS Error $> 2\sigma$

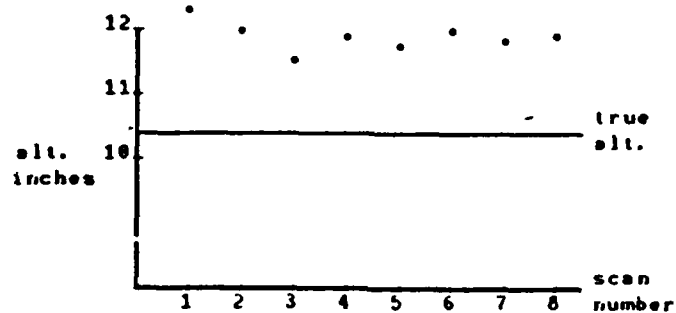
Upon completion of each grid search, Kalman filtering was applied to obtain the optimal altitude value for each terrain cell. Figure 18 are the results of the Kalman filtering after 8 scans with the INS error less than σ . The data also shows that the results of the Kalman filtering when the INS error is between σ and 2σ are almost exactly the same as those shown in Figure 18. Figure 19 presents the results of the Kalman filtering when the INS error is greater than 2σ .

The next test of the grid search method was to set the value of $\sigma = 12$ inches. With this value of σ , the same type of runs were conducted as when $\sigma = 6$ inches. Figure 20 shows the results when the INS error is less than 2σ . Figure 20a shows the grid search locks in on the actual x and y values with zero error after the first scan. Figure 20b indicates that the grid search did not lock in on the actual values after the first scan. To further test the grid search with the INS error between σ and 2σ , another run was conducted with different errors in that range and the results are shown in Figure 21. The data showed that the grid search locked in with zero error on the actual position on the first scan and stayed locked in on all succeeding scans. The final run was conducted with the INS error greater than 2σ and those results are shown in Figure 22. As with the results when $\sigma = 6$ inches, the grid search would not lock in on the actual position when the INS error was greater than 2σ . When the Kalman filtering data was reviewed, the results were very much like that received when $\sigma = 6$ inches.

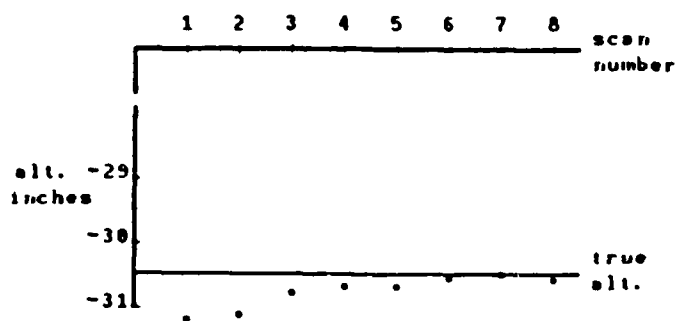
So far all of the results for the moving walker case involved using a pre-stored terrain altitude map. One final simulation conducted was to see if the grid



(a)



(b)



(c)

Figure 18
Kalman Filtering Results With $\sigma = 6$ inches
INS Error $< 2\sigma$

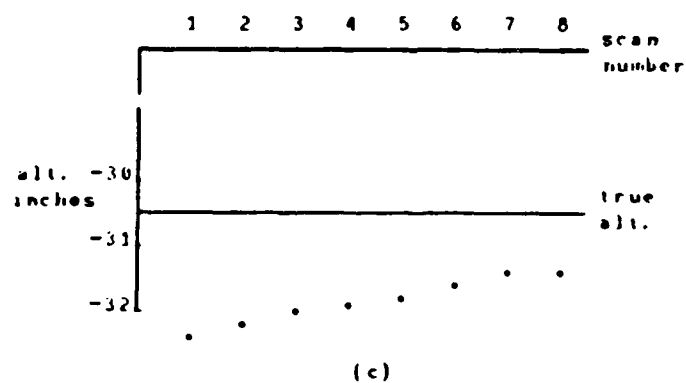
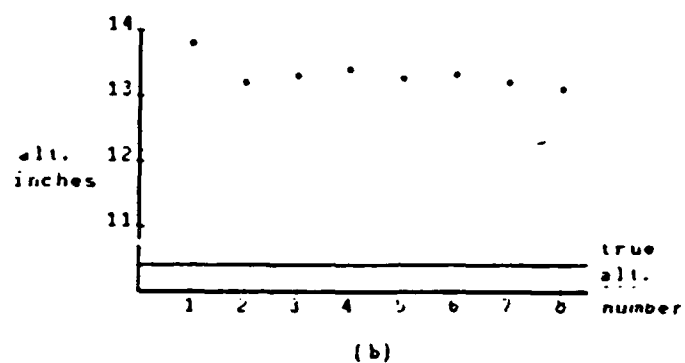
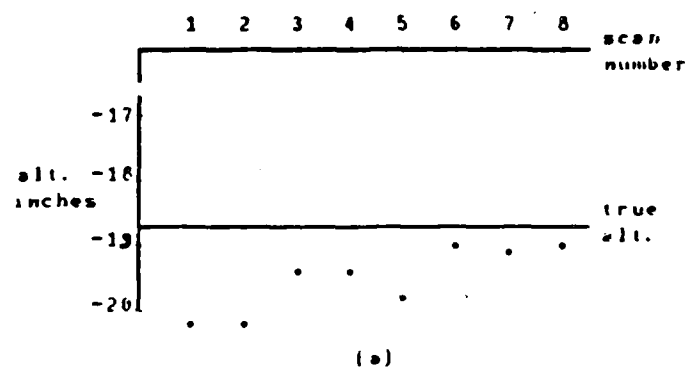
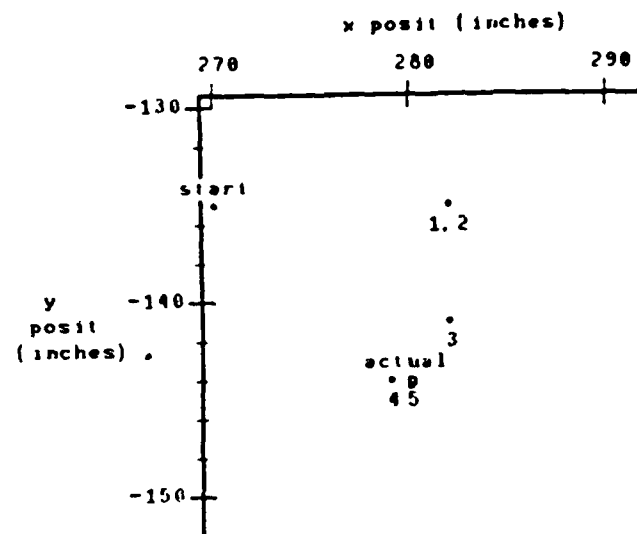
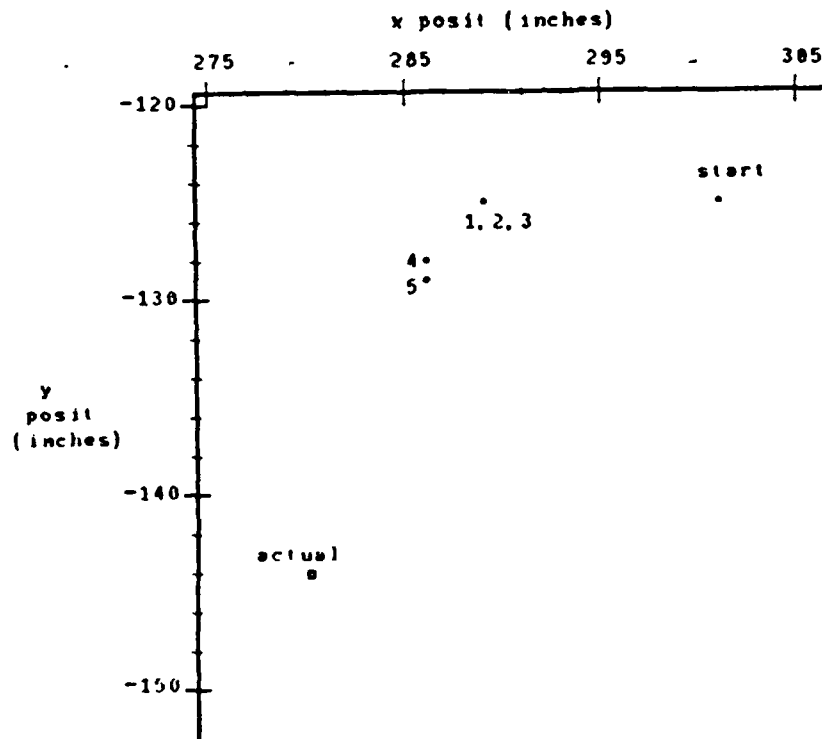


Figure 19
Kalman Filtering Results With $\sigma = 6$ inches
INS Error $> 2\sigma$



(a)



(b)

Figure 20
Grid Search Results With $\sigma = 12$ inches
a) INS Error $< \sigma$ b) $\sigma < \text{INS Error} < 2\sigma$

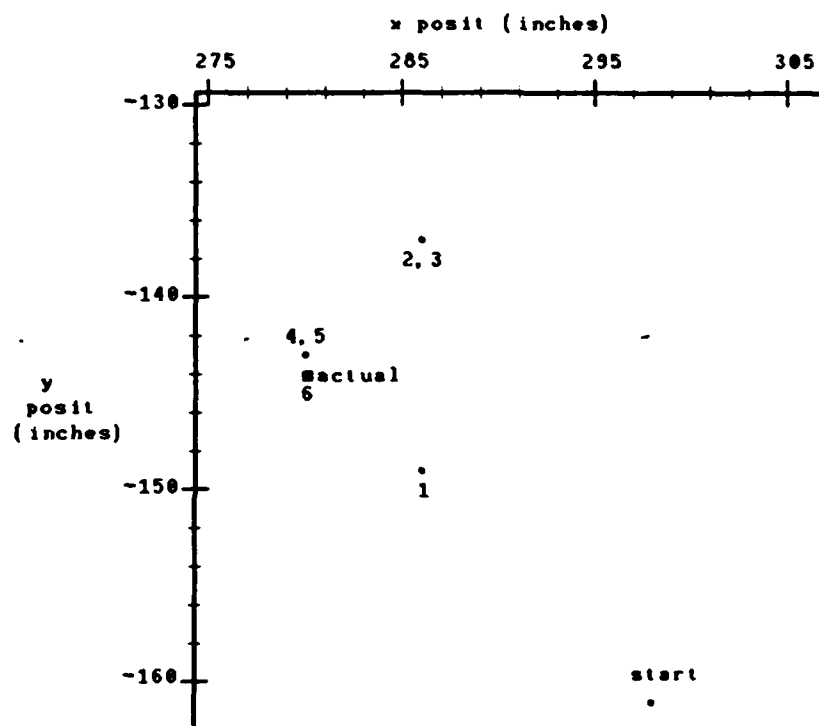


Figure 21
Grid Search Results With $\sigma = 12$ inches
 $\sigma < \text{INS Error} < 2\sigma$

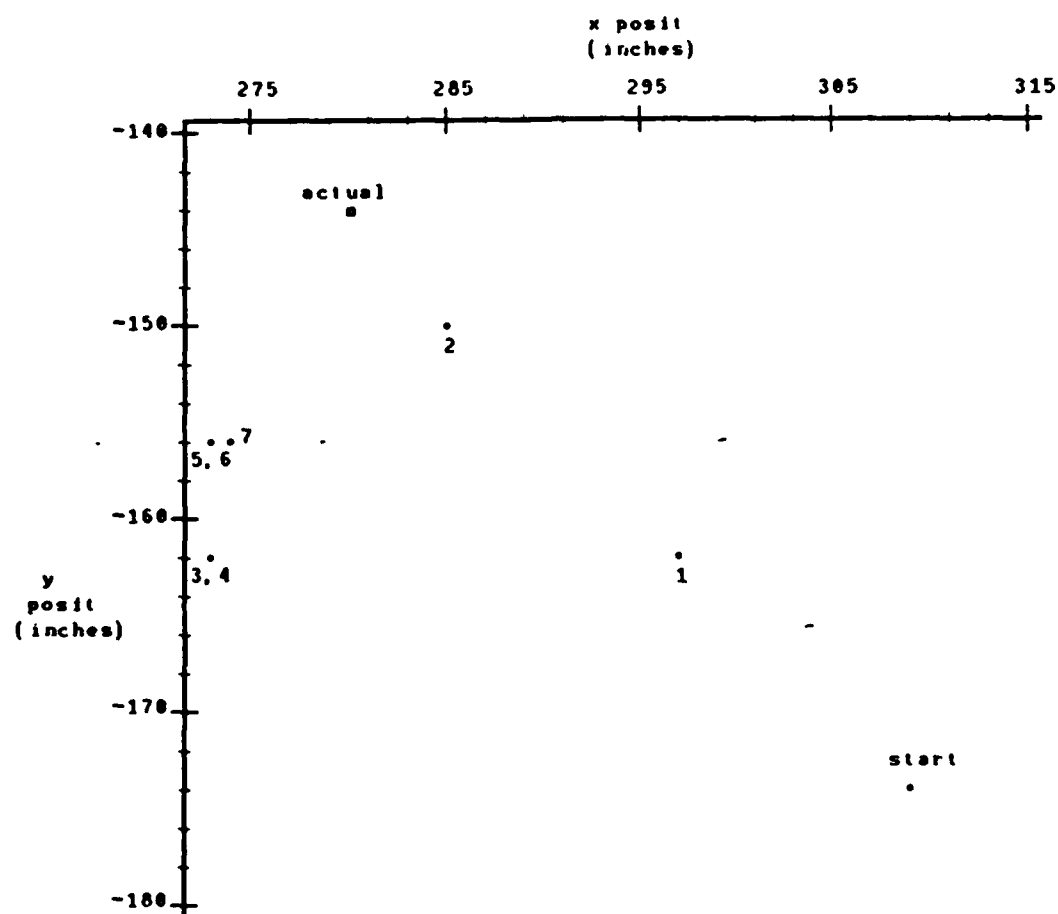


Figure 22
Grid Search Results With $\sigma = 12$ inches
INS Error $> 2\sigma$

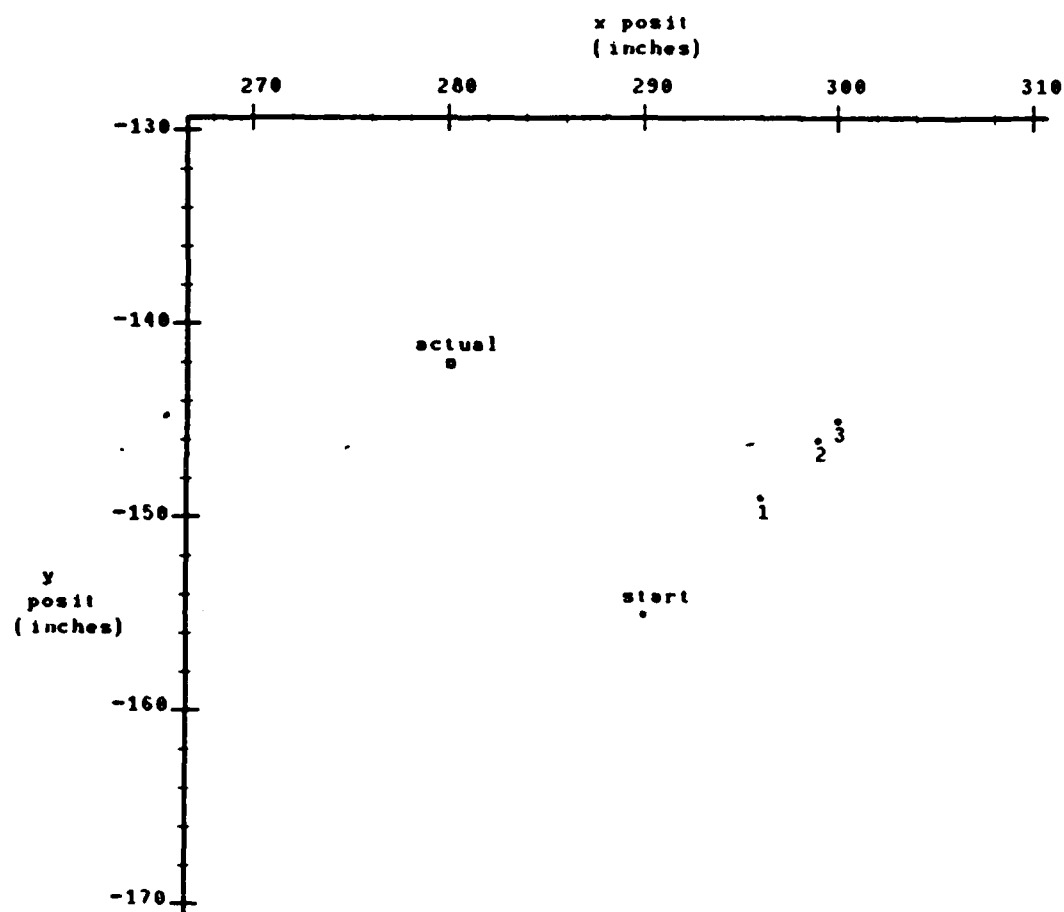


Figure 23
Grid Search Results
No Pre-stored Altitude Map

search method would work without a pre-stored map. In this case, the scanner was placed in the position represented in Figure 9 and the terrain was scanned 20 times to obtain an altitude map. From previous results, it is known that the map stored for the area scanned was very close to the actual map and the data for this case also showed that to be true. Upon completion of the 20 scans, the scanner was given an offset and then regression analysis and Kalman filtering was applied for the next 8 scans. Figure 23 shows the results of the grid search after the first scan. As can be seen, the grid search went farther away from the actual position. Actually, the grid search only conducted 3 steps because of the safety factor built into the program so it would not look forever if it was not converging. Also, the data indicated that on the succeeding scans, the grid search actually got worse. This run was conducted with an $\sigma = 6$ inches and the offset between σ and 2σ . It was run again with the offsets less than σ and the results were the same.

C. CONCLUSIONS

The first general conclusion to be made from the results presented in this chapter is that the Kalman filtering is working. As can be seen from the results of the stationary walker case, the stored altitude values were very close to the actual values. For the terrain cells where the scanner was at -26 and -61 degrees, the stored values were within .25 inch of the actual values while the one at -41 degrees was within about 1.1 inches. To evaluate the significance of these results, data was obtained concerning the standard deviation of the raw terrain elevation

values obtained from the noisy range values. The observed standard deviation had a value of 2.95 inches when averaged over all cases, much greater than the filtered values. One possible reason why the stored values at -41 degrees had a greater error was the arbitrary range adjustment from Eq. (4.6) and the $\sin^2$ effect of Eq. (4.15). In any case, these results would probably be good enough for some route planning program to use.

Referring to Figure 15, an apparent bias in terrain altitude estimates is revealed. From the magnitude and general behavior of this bias, it is believed that it results from the quantization of range into increments of one inch. This quantization is modulated by the $\sin^2$ term of Eq. (4.15), leading to the differences between the three curves of Figure 15.

The next conclusion is that the gradient descent method of regression analysis was not effective with this simulated terrain. The reason for this failure is not known except it may have something to do with the terrain and the resulting "lumpy" error criterion function, Φ .

The overall conclusion about the grid search method is that it is effective. How well it works depends on the σ selected and the amount of INS error relative to that σ . As Figure 20b showed, with a INS error of $\Delta x = 21$ inches and $\Delta y = 19$ inches, the grid search would not lock in on the actual position but as seen in Figure 21, with an error of $\Delta x = 18$ inches and $\Delta y = -17$ inches, the grid search locked in on the actual position. This would indicate that there might be a limit to the amount of INS error involved before the grid search can correct for the

error which is independent of the value of σ . With this method though, the INS initialization errors or offset can be corrected for, within limits, and thus allowing the Kalman filtering to provide an accurate terrain altitude map when a pre-stored altitude map is available.

Finally, using the grid search method without a pre-stored altitude map did not work. In analyzing the results, a possible reason for its failure was because the altitudes in all of the surrounding terrain cells was initialized to zero so when the terrain mask was applied and the stored map scanned, the altitude values stored in the resulting Cartesian coordinate map would probably be less than they should have been, thus giving a minimum criterion function in a false cell during regression analysis. With this false minimum, the grid search would chase after the wrong cell.

VI. SUMMARY AND CONCLUSIONS

A. RESEARCH CONTRIBUTIONS

In this thesis, a first approach was presented in providing an optimal terrain altitude map that could be used for route planning for an autonomous walking machine. Toward that end, an effective Kalman filtering scheme was developed and tested. The results of those tests show that the altitude values stored in the terrain map are good enough for a walking machine to walk across.

Another important part of providing an optimal altitude map is to ensure that the altitude values are stored in the right terrain cell locations associated with the proper x and y coordinates of the terrain area being traversed. If the x and y coordinates are in error because of errors in the inertial navigation system, then it is necessary to correct for those errors before storing the output of the Kalman filter. To solve this problem, a regression analysis approach called the grid search method was presented. This search method proved to be effective in correcting for these land navigation errors on the simulated terrain within limits. These limits appear to be within about 18 inches of the actual x and y coordinate values with the terrain used in this thesis. For a first approach, this type of accuracy is good and provides for a satisfactory starting point for refining this method.

Another contribution of this thesis was to provide a simulation for terrain and an optical terrain scanner scanning that terrain. This simulation was developed for the ISI graphics workstations and could be helpful for any further work that is conducted on these or similar workstations.

B. RESEARCH EXTENSIONS

Since the topics presented in this thesis represent a first attempt at optical radar data filtering with the ASV in mind, there are many areas that can be expanded upon.

First, further research is needed into why the gradient descent method did not work. It is very possible that the reason has something to do with the simulated terrain used. Another extension that could be pursued would be with the grid search method. Along with having a decreasing mask in the grid search, an expanding mask could be developed if the minimum criterion function cannot be brought into the center of the grid. Also, further research is needed to determine the optimal value for the grid size to use in the mask and what limits there are on the errors that can be corrected. Along with this, the grid search method should be applied to completely different terrain and the optimal value of grid size determined and then compared to the one for this terrain to see if there is a relationship. Next, investigation into ways to use the grid search method to correct for errors introduced by the accelerometers while walking would allow the vehicle to continuously walk while the grid search method corrects its position.

Currently, under the scheme presented in this thesis, the vehicle would have to take a few steps, stop and correct its position, and then take a few more steps.

Another important area that needs to be studied is the one of applying these techniques to the moving walker case where no pre-stored altitude map is available. As seen in Chapter V, the first attempt involving this situation did not work. Research into why it did not work and how to make it work will be invaluable to making the walking machine autonomous.

The last area that should be studied is to apply the techniques that have proven successful in simulation to actual terrain with actual optical scanner data.

LIST OF REFERENCES

1. McGhee, R. B., and Waldron, K. J., *An Experimental Study of an Ultra-Mobile Vehicle for Off-Road Transportation*, Final Technical Report for DARPA Contract MDA903-82-0058, College of Engineering, Ohio State University, Columbus, Ohio, 1984.
2. Raphael, B., *The Thinking Computer-Mind Inside Matter*, W. H. Freeman, 1976.
3. Flynn, A. M., *Redundant Sensors for Mobile Robot Navigation*, M.S. Thesis, Massachusetts Institute of Technology, September 1985.
4. Klein, C. A., Olson, K. W., and Pugh, D. R., "Use of Force and Altitude Sensors for Locomotion of a Legged Vehicle Over Irregular Terrain", *The International Journal of Robotics Research*, vol. 2, no. 2, Summer 1983.
5. Lowrie, J., *The Autonomous Land Vehicle Program Report*, Martin Marietta Denver Aerospace, Denver, Colorado, December 1985.
6. Nitao, J. J., and Pardi, A. M., "A Real-Time Reflexive Pilot for an Autonomous Land Vehicle", *IEEE Control Systems Magazine*, February 1986.
7. Zuk, D., Pont, F., Franklin, R., and Dell'Eva, M., *A System for Autonomous Land Navigation*, Technical Report, Environmental Research Institute of Michigan, Ann Arbor, Michigan, November 1985.
8. McGhee, R. B., *Identification of Nonlinear Dynamic Systems by Regression Analysis Methods*, Ph.D. Dissertation, University of Southern California, Los Angeles, California, June 1963.
9. Gelb, A., *Applied Optimal Estimation*, The MIT Press, Cambridge, Massachusetts, 1986.
10. Poulus, D., *Range Image Processing for Local Navigation of an Autonomous Land Vehicle*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1986.

11. Richbourg, R., *Solving a Class of Spatial Reasoning Problems: Minimal-Cost Path Planning in the Cartesian Plane*, Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 1987.
12. Fu, K. S., Gonzales, R. C., and Lee, C. S. G., *Robotics: Control, Sensing, Vision and Intelligence*, McGraw Hill, 1987.
13. Anon, *Programmers Reference Manual for Graphics Software*, Integrated Solutions Inc., San Jose, California, September 1985.
14. Foley, J. D., and Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, 1984.

APPENDIX

PROGRAM LISTING

```

/*****
This is a computer simulation of an optical
scanner scanning simulated terrain using
regression analysis and Kalman filtering.
This program is written in C for the ISI
graphics workstations.
This program is compiled by

cc scan.c -ltools -lbm -lvt -lm

by Mark D. Rickenbach
1 August 1987
*****/

#include <math.h>
#include <vt.h>
#include <stdio.h>
#include <bitmap.h>
#include <tools.h>
struct BMD *bmd;
int seed1;
float seed,seed2,a0_9[4][4],seed3;
main()
{
static int u,v,u2,v2,scan,ii,jj,v21,u21,rr,f,init,x,y,z,k;
static int rand1,rand2,rand3,rand4,xint,yint,safe,counter;
static int bmy1,bmx1,bmx,bmy,fd,ij,kk,xx,yy,s,rang,l;
static float sigscan,sig02,sig12,array[128][128],d9,r[456];
static float n.sigold2[128][128],noise(),sigscan2,elev,dx[9],dy[9];
static float xx1,yy1,zz1,d3min,gmin,zhat[128][128],sz[128][128];
static float w,t1,celev,melev,sigold2[128][128],sigz2[128][128];
static float th4,th5,th6,d1,d2,d3,th7,th8,g[8],rhat[456],d2min;

/*****
Open window, allocate bit map and set line discipline to graphics
for graphics simulation ASV scanning terrain.
*****/

fd=OpenWindow(15,63,1220,664,"Terrain Map");
SetLineDisc(fd,TWSDISC);
if ((bmd=BM_Allocate(1220,664))==0)
{ printf(" unable to allocate bit map");
}
BM_SetAddressing(bmd,1);
BM_SetColor(bmd,1);
BM_SetBColor(bmd,0);

```

```

/*****
    Initialize various arrays to zero.
*****/

```

```

for(i=0;i<128;++i)
{for(j=0;j<128;++j)
{sigold2[i][j]=0;
  sz[i][j]=0;
  maps[i][j]=0;
  sigs2[i][j]=0;
}
}

```

```

/*****
    Compute array of altitude values for terrain map.
*****/

```

```

counter=0;
do {
  seed=0.0;
  seed1=0;
  seed2=0.0;
  rand1=ran(21);
  rand2=ran(21);
  rand3=ran(1);
  rand4=ran(121);
  for(i=0;i<128;++i)
  { for(j=0;j<128;++j)
    {x=rand1-i;
     y=rand2-j;
     t1=(x*x) + (y*y);
     w=(sqrt(t1)/30.)*6.2832*rand3;
     array[j][i]=rand4*cos(w);
    }
  }
  counter=counter + 1;
} while (counter < 0);

```

```

*****
    Draw the base of the terrain map.
*****/

```

```

xint=30;
yint=542
BM SetPosition(bmd.xint yint);
BM SetThickness(bmd.3);
BM PaintLine(bmd.792.542);
BM PaintLine(bmd.1176.161);
BM PaintLine(bmd.414.161);
BM PaintLine(bmd.xint yint);
BM SetThickness(bmd.1);

```

```

/*****
Draw terrain to bit map using scheme developed by
Poulus in his thesis. Each terrain cell will be
represented by a 6 in. x 6 in. square.
*****/

```

```

i=0;
j=0;
loop:
x=i*6;
y=j*6;
z=array[j][i];
bmx=xtrfunc(&xint,&x,&y);
bmy=ytrfunc(&yint,&y,&z);
loop1:
if(i==127 && j==127)
{ BM_SetThickness(bmd,2);
  bmy1=bmy+z;
  BM_PaintLine(bmd,bmx,bmy1);
  BM_SetPosition(bmd,bmx,bmy);
  BM_SetThickness(bmd,1);
  BM_DisplayBitmap(fd,PAINTRASTER,bmd,0,0,0,0,1220,664,0);
  goto loop2;
}
if(j==127)
{if(i==0)
{ x=0;
  y=j*6;
  bmx=xtrfunc(&xint,&x,&y);
  bmy=ytrfunc(&yint,&y,&z);
  BM_SetPosition(bmd,bmx,bmy);
}
  BM_SetThickness(bmd,2);
  bmy1=bmy+z;
  BM_PaintLine(bmd,bmx,bmy1);
  BM_SetPosition(bmd,bmx,bmy);
  BM_SetThickness(bmd,1);
  i=i+1;
  x=i*6;
  y=j*6;
  z=array[j][i];
  bmx=xtrfunc(&xint,&x,&y);
  bmy=ytrfunc(&yint,&y,&z);
  BM_PaintLine(bmd,bmx,bmy);
  goto loop1;
}
if(i==127)
{ BM_SetThickness(bmd,2);
  bmy1=bmy+z;
  BM_PaintLine(bmd,bmx,bmy1);
  BM_SetPosition(bmd,bmx,bmy);
  BM_SetThickness(bmd,1);

```

```

j=j+1;
y=j*6;
x=i*6;
s=array[j][i];
bmxl=xtrfunc(&xint,&x,&y);
bmyl=ytrfunc(&yint,&y,&s);
BM_PaintLine(bmd,bmxl,bmyl);
i=0;
goto loop;
}
if(i==0 || j==0)
{ if(i==0 && j==0)
{ BM_SetThickness(bmd,2);
  BM_PaintLine(bmd,bmx,bmy);
  goto loop4;
}
x=i*6;
y=j*6;
s=array[j][i];
bmxx=xtrfunc(&xint,&x,&y);
bmy=ytrfunc(&yint,&y,&s);
BM_SetPosition(bmd,bmxx,bmy);
BM_SetThickness(bmd,2);
bmyl=bmy+s;
BM_PaintLine(bmd,bmxx,bmyl);
loop4:
BM_SetPosition(bmd,bmxx,bmy);
BM_SetThickness(bmd,1);
i=i+1;
x=(i-1)*6;
y=(j+1)*6;
s=array[j+1][i-1];
bmxl=xtrfunc(&xint,&x,&y);
bmyl=ytrfunc(&yint,&y,&s);
BM_PaintLine(bmd,bmxl,bmyl);
BM_SetPosition(bmd,bmxx,bmy);
x=i*6;
y=j*6;
s=array[j][i];
bmxx=xtrfunc(&xint,&x,&y);
bmy=ytrfunc(&yint,&y,&s);
BM_PaintLine(bmd,bmxx,bmy);
goto loop1;
}
i=i+1;
x=(i-1)*6;
y=(j+1)*6;
s=array[j+1][i-1];
bmxl=xtrfunc(&xint,&x,&y);
bmyl=ytrfunc(&yint,&y,&s);
BM_PaintLine(bmd,bmxl,bmyl);
BM_SetPosition(bmd,bmxx,bmy);

```

```

x=i*6;
y=j*6;
s=array[j][i];
bmx=xtrfunc(&xint,&x,&y);
bmy=ytrfunc(&yint,&y,&s);
BM_PaintLine(bmd,bmx,bmy);
goto loop1;
loop2:

.....

    Enter the starting LANS information from keyboard.
.....

printf(" enter x-coord. of ASV 0);
scanf("%f",&d2);
printf(" enter y-coord. of ASV 0);
scanf("%f",&d3);
printf(" enter s-coord. of ASV 0);
scanf("%f",&d1);
printf(" enter scanner body azimuth angle 0);
scanf("%f",&t4);
printf(" enter scanner body elevation angle 0);
scanf("%f",&t5);
printf(" enter scanner body roll angle 0);
scanf("%f",&t6);
scan=0;
xx1=d2;
yy1=d3;
sz1=d1;
d2=d2+29; /* Introduce LANS initialisation errors if any */
d3=d3-30;

.....

    Start scanning terrain.
.....

do {
    scan=scan+1;
    printf(" scan %d ",scan);
    u2=(xx1-0.5*yy1+30);
    v2=(sz1+.5*yy1+542);

.....

    Draw a rectangle to indicate position of the ASV scanner.
.....

    BM_SetPosition(bmd,u2,v2);
    BM_SetAddressing(bmd,0);
    BM_PaintRectangleInterior(bmd,8,8);
    BM_DisplayBitmap(fd,PAINTRASTER,bmd,0,0,0,0,1220,664,0);
    BM_SetAddressing(bmd,1);
    BM_SetThickness(bmd,2);

```

```

u21=xx1-.5*yy1+30;
v21=.5*yy1+542;
BM_PaintLine(bmd,u21,v21);
BM_SetPosition(bmd,u2,v2);
BM_SetThickness(bmd,1);

```

```

/*****
Scan terrain with one full azimuth scan for each elevation
increment starting at elevation represented by elev and
scanning down. For init=1, this simulates scanner operation
returning a noisy range value, r[rr]. For init=2, this
simulates the ASV's computer internally scanning its
pre-stored terrain map and returning noiseless range values,
rhat[rr]. For the purposes of this thesis, the scanner
scans in three elevation segments of 12 degrees each at
-20, -35, and -55 degrees.
*****/

```

```

for(init=1;init<3;init++)
{
rr=0;
i=1;
do {
if(i==1)
{elev=(-1)*20.;
}
else if (i==13)
{elev=(-1)*22.;
}
else if (i==25)
{elev=(-1)*30;
}
th7=elev-i;
tt=1;
j=1;
do {
rr=rr+1;
th8=6-j;
if(tt==1)
{rang=0;
do {
rang=rang+1;
d9=rang;
computematrix(&xx1,&yy1,&zz1,&th4,&th5,&th6,&th7,&th8,&d9);
l=(a0_9[1][4]/6);
k=(-1)*a0_9[2][4]/6;
celev=(-1)*(a0_9[3][4]);
melev=array[k][1];
} while(celev>melev);
}
else
{

```

```

do {
    rang=rang-1;
    d9=rang;
    computematrix(&xx1,&yy1,&zs1,&th4,&th5,&th6,&th7,&th8,&d9);
    l=(a0_9[1][4]/6);
    k=(-1)*a0_9[2][4]/6;
    celev=(-1)* (a0_9[3][4]);
    melev=array[k][l];
    } while(celev<=melev);
do {
    rang=rang+1;
    d9=rang;
    computematrix(&xx1,&yy1,&zs1,&th4,&th5,&th6,&th7,&th8,&d9);
    l=(a0_9[1][4]/6);
    k=(-1)*a0_9[2][4]/6;
    celev=(-1)* (a0_9[3][4]);
    melev=array[k][l];
    } while(celev>melev);
}
d9=d9-.5; /* back up range by .5 inch */
u=(a0_9[1][4]-0.5*a0_9[2][4]+30);
v=(0.5*a0_9[2][4]+a0_9[3][4]+542);
BM_SetPosition(bmd,u2,v2);
BM_PaintLine(bmd,u,v);
BM_DisplayBitmap(fd,PAINTRASTER,bmd,0,0,0,1220,664,0);

/*****
Compute the noisy range value for init=1.
*****/

sig02=0.02;
sig12=0.0001;
n=noise();
sigscan2=sig02 + sig12*d9*d9;
sigscan=sqrt(sigscan2);
if(init ==1)
{
    n=noise();
    d9=d9+n*sigscan ;
    r[rr]=d9;
}
else
{
    rhat[rr]=d9;
}
tt=tt+1;
j=j+1;
} while(j<12);
i=i+1;
} while(i<36);
}

```

```

/*****
Start regression analysis. Set value of sigma (s) to
be used in the cell masks.
*****/

s=12;
g[0]=0;
f=1;
for(ii=1;ii<433;ii++) /* compute initial criterion function value,g[0] */
{g[0]=g[0]+(r[ii]-rhat[ii])*(r[ii]-rhat[ii]);
}
d2min=d2;
d3min=d3;
gmin=g[0];
safe=1;
kk=1;

/*****
Change x and y coordinate values and simulate terrain
scanning to obtain rhat[rr] to be able to compute
criterion function values for each grid position.
*****/

do {
d2=d2min;
d3=d3min;
g[0]=gmin;
for(ii=1;ii<9;ii++)
{if(ii==1)
{d2=d2-s/f;
d3=d3-s/f;
}
else if (ii==2)
{d3=d3-s/f;
}
else if (ii==3)
{d2=d2+s/f;
d3=d3-s/f;
}
else if (ii==4)
{d2=d2-s/f;
}
else if (ii==5)
{d2=d2+s/f;
}
else if (ii==6)
{d2=d2-s/f;
d3=d3+s/f;
}
else if (ii==7)
{d3=d3+s/f;
}
}

```

```

else if (ii==8)
{d2=d2+s/f;
d3=d3+s/f;
}
rr=0;
i=1;
do {
if(i==1)
{elev=(-1)*20.;
}
else if (i==13)
{elev=(-1)*22.;
}
else if (i==25)
{elev=(-1)*30.;
}
th7=elev-i;
tt=1;
j=1;
do {
rr=rr+1;
th8=6-j;
if(tt==1)
{rang=0;
do {
rang=rang+1;
d9=rang;
computematrix(&d1,&d2,&d3,&t4,&t5,&t6,&th7,&th8,&d9);
l=(a0_9[1][4]/6);
k=(-1)*a0_9[2][4]/6;
celev=(-1)* (a0_9[3][4]);
melev=array[k][1];
} while(celev>melev);
}
else
{
do {
rang=rang-1;
d9=rang;
computematrix(&d1,&d2,&d3,&t4,&t5,&t6,&th7,&th8,&d9);
l=(a0_9[1][4]/6);
k=(-1)*a0_9[2][4]/6;
celev=(-1)* (a0_9[3][4]);
melev=array[k][1];
}while(celev<=melev);
do {
rang=rang+1;
d9=rang;
computematrix(&d1,&d2,&d3,&t4,&t5,&t6,&th7,&th8,&d9);
l=(a0_9[1][4]/6);
k=(-1)*a0_9[2][4]/6;
celev=(-1)* (a0_9[3][4]);

```

```

        melev=array(h)[i];
    } while(celev > melev);
}
d9=d9-5;
rhat[rr]=d9;
tt=tt+1;
j=j+1;
} while(j<12) .
i=i+1;
} while(i<36).

```

.....

Compute criterion function for the appropriate
cell mask

...../

```

g[ii]=0;
for(jj=1;jj<433;jj++)
{ g[ii]=g[ii]+(r[jj]-rhat1[jj])*(r[jj]-rhat1[jj]);
}
dx[ii]=d2;
dy[ii]=d3;
if(ii==1)
{ d2=d2+s/f;
  d3=d3+s/f;
}
else if (ii==2)
{ d3=d3+s/f;
}
else if (ii==3)
{ d2=d2-s/f;
  d3=d3+s/f;
}
else if (ii==4)
{ d2=d2+s/f;
}
else if (ii==5)
{ d2=d2-s/f;
}
else if (ii==6)
{ d2=d2+s/f;
  d3=d3-s/f;
}
else if (ii==7)
{ d3=d3-s/f;
}
else if (ii==8)
{ d2=d2-s/f;
  d3=d3-s/f;
}
}

```

```

/*****
Determine the minimum value of criterion function
and its associated x,y coordinates.
*****/

```

```

for(jj=1;jj<9;jj++)
{if(gmin < g[jj])
{d2min=d2min;
d3min=d3min;
gmin=gmin;
}
else
{d2min=dx[jj];
d3min=dy[jj];
gmin=g[jj];
}
}

```

```

/*****
If minimum value of criterion function is not
in center cell, keep f=1 and increment safe by 1.
If minimum is in center cell or safe=4, start
increasing f by 2 times until its is > s.
*****/

```

```

if(gmin != g[0] && kk==1 && safe !=4)
{f=1;
kk=1;
safe=safe+1;
}
else if (gmin == g[0])
{f=f*2;
kk=kk+1;
safe=4;
}
else if (safe == 4)
{f=f*2;
kk=kk+1;
}
} while(f<s);
d2=d2min;
d3=d3min;

```

```

/*****
Start computation of x,y, and z values for each
azimuth and elevation value.
*****/

```

```

rr=0;
i=1;
do {
if(i==1)

```

```

        {elev=(-1)*20.;
        }
        else if (i==13)
        {elev=(-1)*22.;
        }
        else if (i==25)
        {elev=(-1)*30.;
        }
        th7=elev -i;
        j=1;
        do {
            th8=6-j;
            rr=rr+1;
            d9=r[rr];
            computematrix(&xx1,&yy1,&ss1,&th4,&th5,&th6,&th7,&th8,&d9);
            xx=a0_9[1][4]/6;
            yy=(-1)*a0_9[2][4]/6;
            ss[xx][yy]=(-1)*a0_9[3][4];

/*****
        Perform Kalman filtering for each terrain
        cell scanned.
*****/

        sigs2[xx][yy]=(sig02 + sig12*d9*d9)*sin(th7*3.14159/180)
            *sin(th7*3.14159/180)+.232;
        if(sigsold2[xx][yy]==0)
        {sigsold2[xx][yy]=144000.;
        }
        shat[xx][yy]=shat[xx][yy]+(sigsold2[xx][yy]/(sigsold2[xx][yy]+sigs2[xx][yy]))
            *(ss[xx][yy]-shat[xx][yy]);
        sigsold2[xx][yy]=sigs2[xx][yy]*sigsold2[xx][yy]/(sigs2[xx][yy]+sigsold2[xx][yy]);
        j=j+1;
        } while(j<12);
        i=i+1;
        } while(i<36);
        } while(scan<8);
        printf(" program complete ");
        while (1)
        {}
    }

/*****
        End of the main program
*****/

/*****
        Subroutine to perform the coordinate transformation
        using the D-H transformation. Only the values of the
        matrices needed are computed.
*****/

```

```

computematrix(x1,y1,s1,tth4,tth5,tth6,tth7,tth8,dd9)
float *x1,*y1,*s1,*tth4,*tth5,*tth6,*tth7,*tth8,*dd9;
{
float c4,c5,c6,c7,c8,thact4,thact5,thact6,th7,th8,d9;
float s4,s5,s6,s7,s8,xx1,yy1,ss1;
int a7;
xx1=(*x1);
yy1=(*y1);
ss1=(*s1);
thact4=(*tth4);
thact5=(*tth5);
thact6=(*tth6);
th7=(*tth7);
th8=(*tth8);
d9=(*dd9);
c4=cos(thact4*3.1416/180 +3.1416);
c5=cos(thact5*3.1416/180 -1.5708);
c6=cos(thact6*3.1416/180 +3.1416);
c7=cos(th7*3.1416/180 -1.5708);
c8=cos(th8*3.1416/180 -1.5708);
s4=sin(thact4*3.1416/180 + 3.1416);
s5=sin(thact5*3.1416/180 -1.5708);
s6=sin(thact6*3.1416/180 + 3.1416);
s7=sin(th7*3.1416/180 - 1.5708);
s8=sin(th8*3.1416/180 - 1.5708);
a7=(-1)*5;
a0_9[1][4]=xx1+(c4*c5*c6+s4*s6)*(d9*c7*s8+a7*c7)+(c4*s5)*(d9*s7*s8+a7*s7)
+(c4*c5*s6-s4*c6)*(c8*d9);
a0_9[2][4]=yy1+(s4*c5*c6-c4*s6)*(d9*c7*s8+a7*c7)+(s4*s5)*(d9*s8*s7+a7*s7)
+(s4*c5*s6+c6*c4)*(c8*d9);
a0_9[3][4]=ss1+(s5*c6)*(d9*c7*s8+a7*c7)-(d9*s8*s7+a7*s7)*(c5)
+(s5*s6)*(c8*d9);
}

```

/******

Subroutines to do x and y transformations used in the
terrain drawing portion of the program.

*****/

```

xtrfunc(xint,x,y)
long int *x,*y ;
int *xint;
{
int xx;
xx=(*xint)+(*x)+(*y)*.5;
return xx;
}

```

```

ytrfunc(yint,y,z)
long int *y,*z ;

```

```

int *yint;
{
    int yy;
    yy=(*yint)-(*y)*.5-(*z);
    return yy;
}

```

```

/.....
Random number generator used in creating terrain.
...../

```

```

ran(r)
int r;
{
    int ran;
    seed=(seed2+3.1) * (seed2+3.1);
    seed1=seed;
    seed2=seed-seed1;
    ran=(r)*(seed2) + 1;
    return (ran);
}

```

```

/.....
Subroutine used to create a random gaussian number
with mean=0.
...../

```

```

float
noise()
{
    float b,n,seed,x,xx;
    int i,seed1;
    b=0.5;
    xx=0;
    for(i=1;i<13;++i)
        {seed=(seed3+3.141634)*(seed3+3.141634);
         seed1=seed;
         seed3=seed - seed1;
         x=2*seed3 - 1;
         xx=xx + x;
        }
    n=b*xx;
    return(n);
}

```

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library (Code 0142) Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman (Code 62) Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
4. Curricular Officer (Code 32) Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
5. Prof. Robert B. McGhee (Code 52Mr) Department of Computer Science Naval Postgraduate School Monterey, California 93943	20
6. Prof. Roberto Cristi (Code 62Cx) Department of Electronics and Computer Engineering Naval Postgraduate School Monterey, California 93943	1
7. Prof. Kenneth J. Waldron Department of Mechanical Engineering 206 W. 18th Avenue School Ohio State University Columbus, Ohio 43210	1
8. Russel L. Werneth (Code 69Wh) Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1

- | | | |
|----|--------------------------------------|---|
| 9 | William J Butler | 1 |
| | Naval Sea Systems Command SEA 90G | |
| | Washington, D.C. 20362-5101 | |
| 10 | Cdr Bart Everett | 1 |
| | Naval Ocean System Center (Code 442) | |
| | San Diego, California 92152 | |
| 11 | Research Administration (Code 012) | 1 |
| | Naval Postgraduate School | |
| | Monterey, California 93943 | |
| 13 | Center for Naval Analyses | 1 |
| | 2000 N. Beauregard St. | |
| | Alexandria, Virginia 22311 | |
| 13 | Mrs. Barbara Rickenbach | 2 |
| | 221 N. 20th | |
| | Pasco, Washington 99301 | |