

AD-A173 882

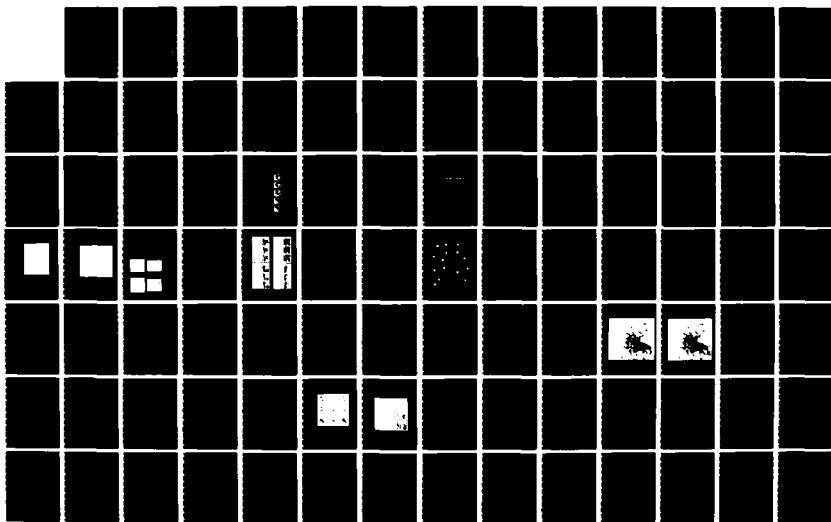
IMAGE-BASED NAVIGATION(U) LOCKHEED MISSILES AND SPACE
CO INC PALO ALTO CA RESEARCH AND DEVELOPMENT DIV
M T NOGA 21 DEC 85 LMSC-D060772 DAAK70-81-C-0098

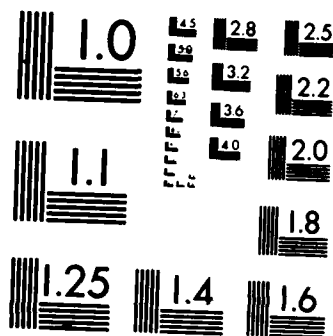
1/1

UNCLASSIFIED

F/G 17/7

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AD-A173 882

12

DTIC FILE COPY

DTIC
ELECTE
NOV 5 1986
S A D

 **Lockheed Missiles & Space Company, Inc.**
SUNNYVALE, CALIFORNIA

86 11 4 130

IMAGE-BASED NAVIGATION

M. T. Noga

Research & Development Division
Lockheed Missiles & Space Company, inc.
3251 Hanover Street
Palo Alto, CA 94304

December 1985

Final Report

Prepared for
ADVANCED RESEARCH PROJECTS AGENCY
1400 Wilson Boulevard
Arlington, VA 22209

FORT BELVOIR RESEARCH & DEVELOPMENT CENTER
U. S. Army Electronics R&D Command
Vint Hill Farms Station
Warrenton, VA 22186-5120

DTIC
NOV 5 1986
A

86 11 4 130

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) LMSC-D060772			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Research & Development Division Lockheed Missiles & Space Co. Inc.		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Ft. Belvoir Research & Development Center U.S. Army Electronics R&D Command		
6c. ADDRESS (City, State, and ZIP Code) 3251 Hanover Street Palo Alto, CA 94304			7b. ADDRESS (City, State, and ZIP Code) Vint Hill Farms Station Warrenton, VA 22186-5120		
8a. NAME OF FUNDING, SPONSORING ORGANIZATION Advanced Research Projects Agency		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAK70-81-C-0098		
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22209			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO	PROJECT NO	TASK NO
			WORK UNIT ACCESSION NO		
11. TITLE (Include Security Classification) Image-Based Navigation					
12. PERSONAL AUTHOR(S) M.T. Noga					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM 8/11/81 TO 9/27/85		14. DATE OF REPORT (Year, Month, Day) 12/21/85	
15. PAGE COUNT 90					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Passive Navigation, Image Understanding, Autonomous Navigation, Stereo Processing, Landmark Detection, Backtrack Algorithms, Image Analysis		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
This report describes add-on work involving the Passive Navigation Study an investigation to determine if a small, low-flying vehicle can be navigated using passively sensed images of the ground. The seminal work was carried out under contract no. F33615-78-C-1612 from 9/15/78 to 12/15/80, during which a stereo bootstrapping approach for position determination was developed. A landmark subsystem was employed to correct the calculated position of the vehicle after a number of bootstrap iterations. An error analysis of the approach and suggestions for further work were included. In this report, an attempt is made to integrate a set of landmark detector schemes with mission task information to locate the vehicle with respect to known ground coordinates. The navigator can be thought of as a program that processes various subsystem information and makes intelligent decisions to determine vehicle position. An approach is outlined which attempts to simulate a human pilot who navigates by visual correction. In this approach the autonomous aircraft have redundant and independent navigational subsystems and					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION		
22a. NAME OF RESPONSIBLE INDIVIDUAL			22b. TELEPHONE (Include Area Code)		22c. OFFICE SYMBOL

SUMMARY

The goal of the Image-Based Navigation project was to use an image-based computer system to simulate the heuristic techniques humans employ when attempting to navigate a small, slow-flying, low-altitude aircraft for reconnaissance, jamming, or weapon delivery. Such navigation techniques include (1) dead reckoning extrapolation of vehicle location and (2) periodic "fixing," in which the extrapolated location is corrected by using known landmarks on the ground.

This final report describes the various Image-Navigation studies that have been carried out over the past four years:

1. Systems definition: defining the mission, the mission tasks to be described to the system, the overall system, onboard planning requirements, and the nature of the onboard reference data.
2. Bootstrap extrapolation of location: a technique based on stereo analysis in which initially known points on the ground are used to locate the aircraft, and then the known location of the vehicle is used to locate new points on the ground.
3. Position finding: techniques for relating easily recognizable objects as shown on a map to ground objects viewed by the sensor. A particularly promising technique involving crossings of roads and boundaries was developed, and experiments were carried out.
4. Image-based velocity meter: an image-based approach for the determination of ground velocity.

Some of the questions that have been examined in these areas are:

- o How can the "common sense" used by an experienced pilot be incorporated into a computer executive?
- o How can position extrapolation be accomplished automatically?
- o What is the nature of the stored landmark maps?

- o How can reference and sensed landmark descriptions be efficiently compared?
- o How can shape and topographic descriptions be effectively derived from an image?
- o How can map data be automatically interpreted by a computer system so that landmarks can be extracted from an image?
- o How can crossings of roads and boundary transitions be utilized?

Our basic conclusions from these investigations are:

1. An automatic system should emphasize position fixing, rather than trying to concentrate on accurate dead reckoning.
2. The "bootstrap" stereo method of dead reckoning, Section 7.2, suffers from many operational difficulties, a major one being the initialization procedure and the need to repeat this when frames are missed because the ground is obscured.
3. For general landmark detection, the sensor should have a wide field of view, and should be capable of looking in all directions, similar to a person. A narrow field of view requires that the vehicle fly over landmarks, and this is often not possible due to errors in dead reckoning. A sensor incapable of looking in all directions is restricted to searching for landmarks, and cannot take bearings to a landmark. On the other hand, the "crossings" approach can utilize a narrow field of view.
4. In general, expected landmarks should be selected from the onboard database, and the system should then try to find the objects on the ground. A reference list can be used to restrict analysis of the sensor imagery to areas where landmarks with unique signatures exist. The opposite technique of classifying the large number of objects in the sensed images and attempting to match these to the small number of reference objects is quite difficult.
5. The vehicle should fly in straight-line segments so that if the ground is obscured, it can extrapolate the path and determine landmarks that will be in the field of view when the obscuration clears.

6. A landmark system based on transitions such as roads and regions has been conceptualized and is capable of utilizing multiple frames of imagery. Initial experiments have been run, and results are quite promising. We regard this part of the study as the approach having the greatest near-term potential. The present algorithm cannot deal with extraneous (erroneous) transitions, but we have some initial approaches which we feel can solve this problem.
7. The problem of determining relative altitude is one that arose throughout the passive navigation study, since the movement of a pixel in the image plane depends on V/H , the ratio of the vehicle velocity to relative altitude. Several ways of dealing with this problem are suggested in Section 4.1.

CONTENTS

Section		Page
	SUMMARY	iii
1	INTRODUCTION	1
2	SIMULATING THE HUMAN PILOT	2
	2.1 The Human Navigator	3
	2.2 Computer-Based Navigation	7
3	SYSTEM ASPECTS	9
	3.1 Describing a Mission to an RPV	9
	3.2 The Executive Program	9
	3.3 Image-Based Navigation	11
4	DEAD-RECKONING SUBSYSTEM	13
	4.1 Instrument-Based Dead Reckoning	13
	4.2 Image-Based Dead Reckoning	16
	4.3 The Image-Velocity Meter Concept	18
5	POSITION FIXING	23
	5.1 Landmark Determination Using Intensity Images	23
	5.2 Contour Determination Using Symchains	24
	5.3 Symbolic Matching	26
	5.4 Experimental Results	28
	5.5 The Landmark Data Base	32
6	A NEW APPROACH TO LANDMARK NAVIGATION	37
	6.1 Introduction	37
	6.2 Constraining the Problem	38
	6.3 Data Preparation	40
	6.4 Position Fixing Algorithm	42
	6.5 Sample Run of the MPP Algorithm	43
	6.6 Missed Crossings	52
	6.7 Extraneous Crossings	56
	6.8 MPP Microprocessor Feasibility	57

Section		Page
7	STEREO STUDIES	58
	7.1 Stereo Techniques	58
	7.2 Error Experiments	64
8	SUMMARY AND DISCUSSION	69
	8.1 Systems Aspects	69
	8.2 Stereo Subsystem	69
	8.3 Landmark Subsystem	72
9	REFERENCES	76
Appendix		
A	Flight Simulation Log	A-1
B	LOL Data Structure	B-1

ILLUSTRATIONS

Figure		Page
2-1	Navigation by Projection of Position, and Correction Using Landmarks	2
2-2	Components of the Vehicle Ground Velocity	5
2-3	Using Multiple Landmarks to Fix Position	6
2-4	A "Running Fix" Using a Single Landmark	7
3-1	Components of the Navigation Expert	12
4-1	Overall Navigation System	14
4-2	Velocity Determining Device	15
4-3	Altitude Measuring Devices	16
4-4	The Bootstrap Navigation Concept	17
4-5	Movement of New Points in Bootstrap Navigation	19
4-6	Bootstrap Stereo Computation	20
4-7	Histograms of Velocity Computed for Various Misalignments of Sensor	22
4-8	Block Diagram of the Instrumentation Setup	22
5-1	Extracting and Matching Sensed Line Segments to Reference Line Segments	24
5-2	System for Boundary Matching Using Symchains	26
5-3	Symchain Extraction for NVL Terrain Table	29
5-4	Symchain Extraction From Map and Image Data	30
5-5	Matching Reference and Sensed Symchains	31
5-6	Matching of Reference and and Sensed Symchains for Bridge Reference	33
5-7	Landmark Database	34
5-7	Example of Lineal Sketch and Reference Image Maps Used for Position Fixing	36
6-1	Sketch Map Data	38
6-2	Accumulation of Crossing Information	39
6-3	Displacing a Line Segment to Create a Trapezoid of Uncertainty	44

Figure		Page
6-4	Reference Map Line Segment Database	45
6-5	Confirming the Feasibility of Path (1 4)	48
6-6	Box of Uncertainty Resulting From Displacement of Line Segment 4	49
6-7	Box of Uncertainty Resulting From Displacement of Line Segment 7	50
6-8	Final Path (1 4 7 8 11)	51
6-9	Line Segment Database Superimposed Over Image	53
6-10	Path Calculation for Database and Image of Figure 6-9	54
6-11	Boxes of Uncertainty Corresponding to Removal of Crossing Data for Line Segments 4 and 8	55
7-1	Stereo Image Processing	58
7-2	Techniques for Interesting Point Determination	60
7-3	Interesting Point Determination Using Three Different Measures	62
7-4	Hierarchical Modeling Through a Reduction Hierarchy	63
7-5	Error Curves for Various Parameters	66
7-6	Error as a Function of Match Accuracy	67

TABLES

Table		Page
3-1	Types of Descriptions Required for RPV Mission	10
8-1	Image-Based Navigation	70

1. INTRODUCTION

Automatic navigation of a subsonic vehicle (50 to 100 knots) flying at low altitude (1500 to 5000 ft above the ground) is of great interest for unmanned flight applications, such as small aircraft used for reconnaissance, jamming, or weapons delivery. For a considerable period of time during such a mission, and preferably for its entire duration, it is desirable to use minimum communications between the ground station and the vehicle. This requirement necessitates an autonomous mode of operation, including navigation to and over the target areas.

A simple preprogrammed flight plan is subject to gross errors due to unpredictable winds and turbulence. These cannot be sensed onboard the vehicle because such inexpensive reconnaissance vehicles are not equipped with inertial guidance systems. However, these vehicles do contain attitude-sensing devices, such as a vertical gyro, a gyrocompass, and sometimes, also, rate gyros. In addition, there is always a barometric altimeter and an airspeed sensor in the onboard instrumentation package. Microprocessors are also becoming standard items, providing an ever-growing computational facility onboard the vehicle.

Recently, there has been interest in using images of the terrain as the basis for an autonomous navigation system. The interest has been sparked by the fact that the human pilot uses visual sensing to a large degree, and it is hoped that the availability of small solid-state sensors and microprocessors, will enable an autonomous system to simulate the human's behavior. To that end, a study of autonomous navigation of a small, slow-speed, low-altitude vehicle using passively sensed images has been carried out at the Research & Development Division of Lockheed Missiles & Space Company, Inc.

This final report describes and summarizes the Lockheed Image Navigation studies over the past four years.

2. SIMULATING THE HUMAN PILOT

A pilot flying a small plane at low altitudes depends very strongly on his vision for information about the plane's precise location. His instruments present him with airspeed, altitude above sea level, and heading. In using these instruments to estimate aircraft position, errors will occur over time, since the instruments are not perfect, and there is no low-cost instrument to directly measure the effects of wind on ground speed and course. A reference map is crucial for correcting these errors: the pilot must visually correct his estimated position by relating landmarks such as rivers, highways, and towns (shown on a map) to objects that he views. This flight path estimation and correction procedure is shown in Fig. 2-1.

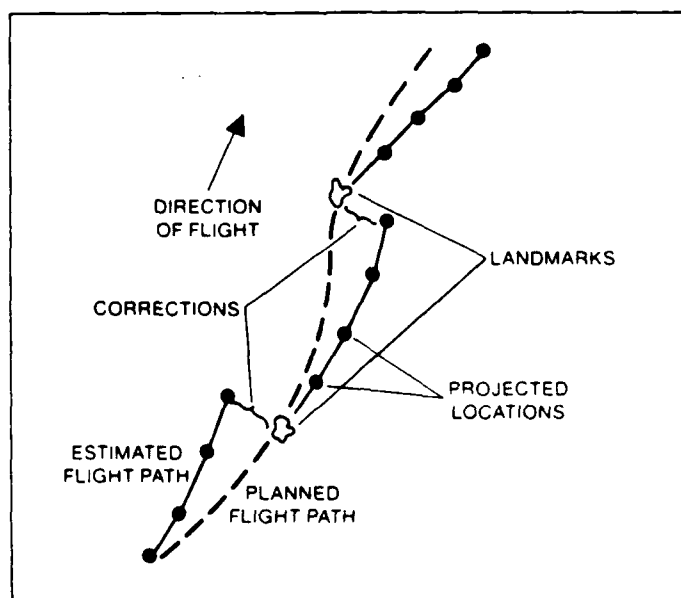


Fig. 2-1 Navigation by Projection of Position, and Correction Using Landmarks

The reference map can be thought of as a two-dimensional array of symbols, specially prepared for a geographic region, that has a special meaning for the pilot. For example, there are symbols that denote railroads, highways, rivers, and towns. A match between an object viewed and a landmark shown on the map gives the pilot a position "fix." The pilot's visual system, combined with the information from instruments, leads to a reliable system because the pilot can select measurements that are most appropriate for a given situation. When the ground scene does not contain recognizable landmarks, the pilot relies primarily on his instruments and performs dead reckoning. When there are clear landmarks that can be readily related to his map, he may choose to disregard the instruments.

Suppose we want to devise an autonomous aerial vehicle that uses images taken by an onboard camera as well as measurements taken from conventional instruments to navigate as the human pilot does. We then call on techniques in the field of "artificial intelligence" (AI), a discipline that attempts to model a subset of human intelligence and implement this model using a computer. We would want the autonomous aircraft to have redundant and independent subsystems so that the partial or complete failure of a subsystem doesn't cause the entire system to fail and a database that contains information about the application environment. In contrast to conventional database systems, AI systems require a knowledge base with diverse kinds of entries. These include, but are not limited to, knowledge about objects, knowledge about processes, and hard-to-represent common sense knowledge about goals, motivation, causality, time, actions, etc.

2.1 The Human Navigator

We can gain insight into the requirement for an automatic device by examining the procedures used by a human navigator. The rules of thumb, "heuristics," used by the human navigator in dead reckoning and location fixing are particularly important.

2.1.1 Dead Reckoning. In dead reckoning, there are five basic measures with which the navigator is continually concerned:

- True Airspeed: Speed of an aircraft through a mass of air (TAS)
- True Heading: The direction from true north in which the aircraft is pointed (TH)
- Wind Direction and Velocity: Wind direction with regard to true north, and speed in knots
- Ground Speed: Speed in relation to a fixed point on the earth (GS)
- True Course: The intended or actual path over the ground measured from true north (TC); the actual path is also called Track (TR)

The navigator's job is to give values to these quantities by measuring, computing, and sometimes by little more than guessing. The "wind triangle" shown in Fig. 2-2 gives the true course as the sum of the wind vector and the true airspeed vector. In actual flight, the true course is usually found from the estimate of the wind vector and the true airspeed vector. The magnitude of the airspeed vector is obtained from the airspeed indicator and the direction is obtained from the magnetic compass heading. The pilot can estimate the wind velocity at low altitudes by watching smoke plumes, or by noting the "drift" of terrain features with respect to the aircraft.

2.1.2 Location Fixing. Though basic to all navigation, dead reckoning is not a self-sufficient system. First, the errors of dead reckoning are cumulative. The farther an aircraft flies, the greater is the likelihood that the plotted position is in error. Errors arise from instrument inaccuracies, errors in estimating wind velocity, etc. The navigator must therefore have some reliable method of providing a continuing check on the accuracy of his dead reckoning. In a low-flying slow aircraft, the problem of dead reckoning error can be solved by watching for, and correcting back to, ground landmarks. This is known as "pilotage," landmark finding, or map reading.

If the pilot is heading toward a known landmark in a straight line, then he knows his line of position (LOP), a line connecting all possible geographic

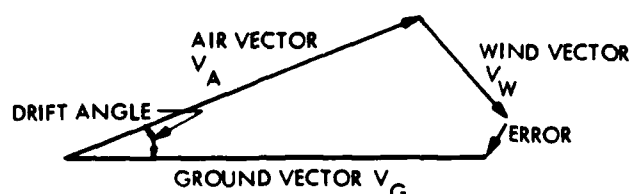


Fig. 2-2 Components of the Vehicle Ground Velocity

positions of an aircraft at a given instant. It is possible to establish a fix, an established geographic position of the aircraft at a given instant of time by crossing two or more LOPs independently determined. Two or more LOPs can rarely be obtained at the same instant of time. It is therefore necessary to rectify the time difference by advancing or retarding LOPs. An example of this is given in Fig. 2-3.

A major problem is finding known landmarks on the ground, either cultural features (cities, roads, bridges, etc.) or terrain features (rivers, mountain peaks, coast lines, lakes) that correspond to landmarks indicated on a map.

Good navigational practice consists of definitely locating the aircraft at a given instant, and then preparing to establish another definite fix at some instant of time later. This varies from 5 to 30 min, depending on many factors. In good weather, where visibility is clear and where wind patterns

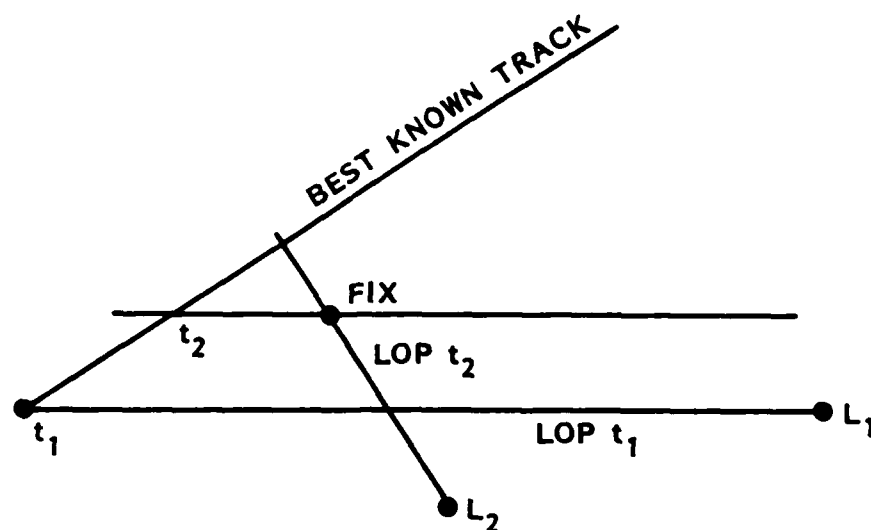


Fig. 2-3 Using Multiple Landmarks to Fix Position

are expected to remain constant, fixing every 20 to 30 min is usually adequate. More frequent fixing is necessary if the winds are variable and the visibility is bad.

A navigator prepares for his next fix by (1) selecting a good check point some distance ahead of the present position, (2) estimating the time of arrival, and (3) watching for the check point and noting the time of passage. Using the time of passage, he can update his groundspeed.

Three different bearings on one landmark point can be used to establish a fix, generally referred to as a "running fix." Bearings are taken to the object, e.g., 45, 90, and 135 deg relative to the right of the aircraft heading. The procedure is shown in Fig. 2-4.

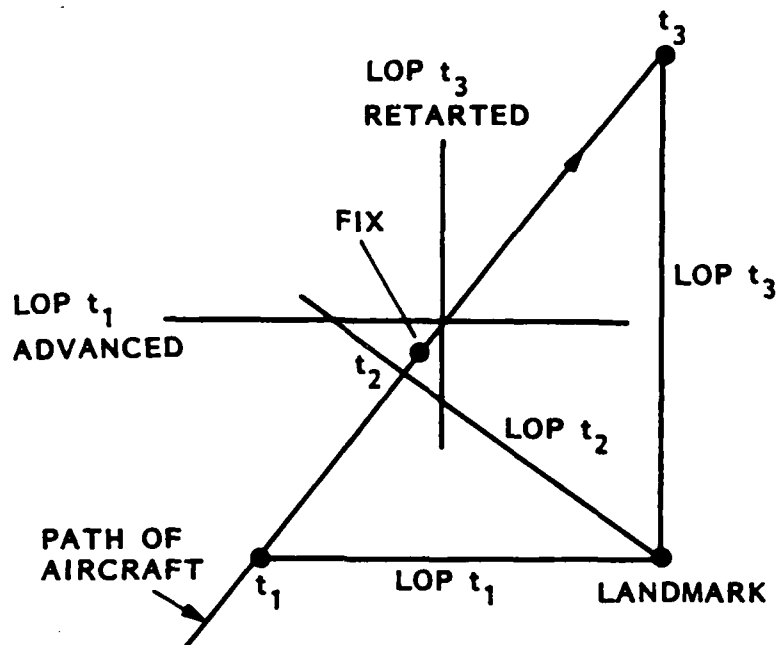


Fig. 2-4 A "Running Fix" Using a Single Landmark

A heuristic used by pilots for finding landmarks that could be important in the automated case is: "Select a feature on the map and then try to find it on the ground, rather than to work from ground to chart."

Another heuristic used when the navigator becomes lost, or after a period of flying over an overcast which has obscured the ground is: "Search only in that portion of the map where the aircraft could possibly be. This area is roughly a circle with a radius in miles equal to about 10% of the distance travelled since the last fix." This is a good reason for flying a steady course, rather than wandering.

2.2 Computer-Based Navigation

An AI approach to aircraft navigation uses a sensor to view the terrain as the human pilot does, and an Executive program that makes the decisions. The

Executive has available specialist subsystems, some of which are capable of analyzing images of the terrain. Each subsystem is called on by the executive for a specific task and reports back to the Executive with the results and the confidence for each result. Just as in the case of the human executive, the program Executive must be able to decide between conflicting reports from the subsystems.

2.2.1 Sensing the Terrain. If we imagine a pilot forced to fly wearing blinders, we immediately realize that his ability to locate landmarks would be severely limited. Thus, it is important that the sensor be capable of looking in all directions and of zooming when an object captures its attention. The sensor could be similar to the one used in the SKYEYE R4E30, manufactured by Development Sciences, Inc., that can zoom from 5.6 to 56 deg in 5 s, and has an azimuth range of plus or minus 90 deg (see Military Electronics/Countermeasures, March 1983).

2.2.2 The Navigation Expert. The navigation expert must provide two basic navigation capabilities: (1) dead reckoning, which starts from an initial known location and integrates the measured ground velocity as the aircraft moves, and (2) landmark-finding or fixing, in which location of the vehicle can be found because the relation of the vehicle to one or more known landmarks on the ground is known. The dead-reckoning and position-fixing subsystems work cooperatively, with the position obtained by dead reckoning corrected by observing known landmarks along the way.

3. SYSTEMS ASPECTS

This section discusses the systems aspects of an autonomous vehicle, such as an RPV. First, the requirements for describing a mission to an RPV are indicated. Then the navigation subsystem is described.

3.1 Describing a Mission to an RPV

An autonomous RPV must be given reference maps indicating identified landmarks for use in navigation, and a description of the mission tasks to be carried out. The mission description enables the RPV to make the necessary plans and carry out the necessary actions. One can derive the nature of these instructions by considering the briefing required for a human pilot for such a mission. Some examples are:

1. Take off, fly at random in Map Grid 28 and return with images of "interesting" targets.
2. Fly north along the river to the first bridge, find an emitter with characteristics C, and destroy it.
3. Fly the course given by a marked-up map, capture images at points shown, and return.

These instructions consist of overall goal statements, the initial location, the desired vehicle path, and additional knowledge concerning the environment. Table 3-1 indicates the type of descriptions required; the * indicates a difficult task for an automated system.

3.2 The Executive Program

The Executive program carries out the mission directives and interacts with the image-based navigation system.

3.2.1 Planning in the Executive. Planning is the process by which an intelligent agent determines and executes a plan of action. In the case of

Table 3-1 TYPE OF DESCRIPTIONS REQUIRED FOR RPV MISSION

Overall Goal Statements

Take photos (specify targets, time, or locations)
Transmit image
Transmit vehicle location or target location
Station-keep over target (possibly indicate path pattern)
Attack target
Jam target
Designate target

Initial Location

By coordinates
*By landmarks

Vehicle Path

Go to coordinates (x,y), act, and return
Go to coordinates (x,y) using the path XX, act, and return
Go to landmark X, using path YY, act, and return via ZZ
Go to general region R, reconnoiter, act, and return

Additional Knowledge Concerning Environment

Special navigation information: look for X as an important landmark
Estimated time of encounter of something of interest
*Dangers: there is an X to avoid
*Opportunities: if you come across an X let us know
Instrument information: wind magnitude and direction; barometric pressure; ground elevation

the autonomous vehicle, the Executive may be given an initial plan, but for true autonomy, the Executive should have the ability to develop a new plan based on the actual situation relative to the mission description. Thus, if cloud cover prevents the vehicle from reaching the target, there may be an alternative target that can be reached, and the mission description should provide such alternatives. Another type of planning occurs when the vehicle is lost, and a strategy for finding the path back to the origin must be developed.

Both high-level and low-level planning is carried out. On the high level, the system deals with problems of alternative targets, or of setting up a plan for a flight pattern to be followed when the vehicle is lost. On the low level, the Executive plans path changes or sensor orientations required to be able to view landmarks, and develops a plan for sensor movement to find these landmarks. Hence, the low level can be considered to be the normal operational mode and the high level a recovery mode.

3.3 The Image-Based Navigation System

The Executive computer program weighs evidence, makes decisions, and controls the vehicle's flight, with the aid of a group of specialist subsystems that are called on for advice. As shown in Fig. 3-1, systems that have been identified include an Instruments Subsystem, a Dead Reckoning Subsystem, a Stereo Subsystem, and a Landmark Subsystem. Each of these provides information on its specialty along with confidence measures so that the Executive can weigh contradictory results from two or more subsystems.

The Instruments Subsystem (IS) is the simplest of the specialists. It keeps track of instrument readings such as air speed, heading, and altitude, and reports results to the Executive. The IS will also be responsible for controlling the image sensor.

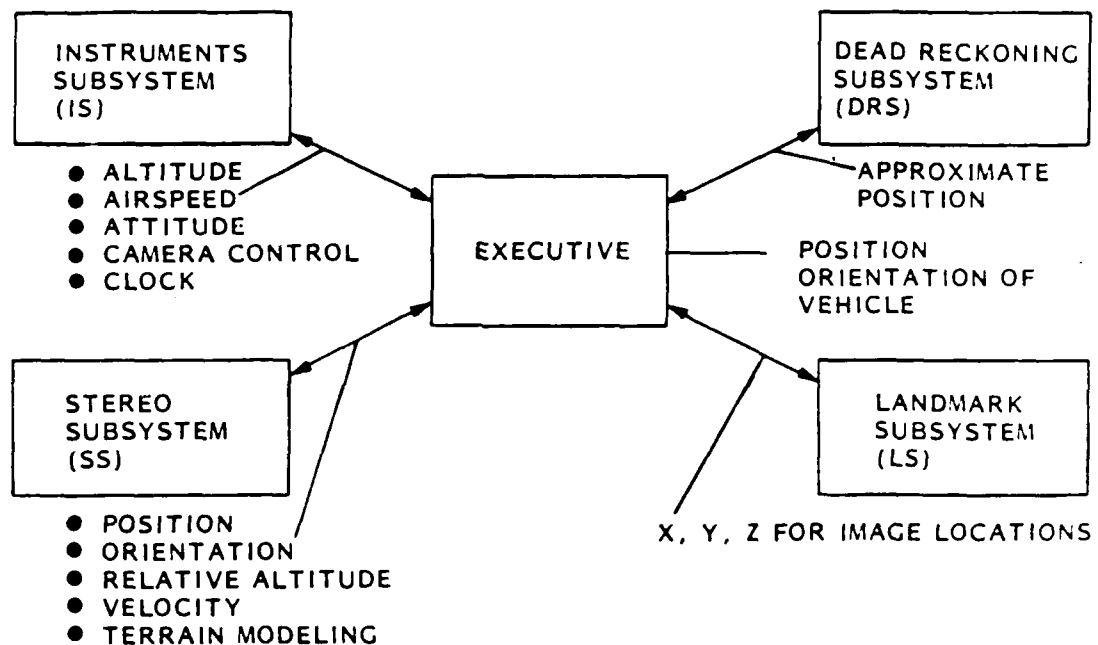


Fig. 3-1 Components of the Navigation Expert

The Dead Reckoning Subsystem (DRS) is also rather simple. Its job is to extrapolate the current position and course from the last several known true positions, course estimates, velocity estimates, flight times, etc.

The Stereo Subsystem makes measurements on images of the underlying terrain to determine the actual position of the vehicle from the positions of the selected terrain points. It does this by using a bootstrapping approach, described in Section 7.2.

The Landmark Subsystem (LS) has the duty of recognizing landmarks along the desired course. The LS performs the position fixing; these are required since the dead reckoning extrapolations diverge with time due to various estimation and convergence errors.

4. DEAD-RECKONING SUBSYSTEM

There are several methods of dead reckoning possible in the automatic system: (1) dead reckoning using instruments to compute the wind triangle, (2) use of bootstrap stereo, and (3) use of an image-based ground-velocity meter. The sources of the altitude, velocity, and position measurements obtained from the various instruments and modules are shown in Fig. 4-1, and are described below.

4.1 Instrument-Based Dead Reckoning

In instrument-based dead reckoning, we compute the wind triangle as indicated in Fig. 4-2 and use the airspeed indicator and the compass heading to obtain the air velocity. (The compass accuracy is plus or minus 1.5 deg.) The wind velocity is estimated by (1) computing the difference between an estimated landmark location and the actual location, and/or (2) by noting the "drift" of an object on the ground from frame to frame.

The drift computation requires that we know the distance on the ground represented by a pixel in the image, and this requires a knowledge of the relative altitude. Two approaches are possible, as indicated in Fig. 4-3. The first approach is stereo analysis, discussed in Section 7. The stereo analysis requires knowledge of points on the ground or the distances between stereo look points. If a passive absolute barometric altimeter is used, an accuracy of about 1% can be achieved. However, this reading must be converted to relative altitude by subtracting prestored terrain data. This requires that we have a topographic map of the region stored inside the onboard computer memory. The uncertainty in terrain elevation data and accurate knowledge of vehicle location may further decrease the accuracy, so that a 2% to 3% error in relative altitude is obtained. One problem with this method is that topographic maps usually contain a significant amount of data. This means that their storage and recovery aboard a flying vehicle with low-cost computer equipment remains infeasible.

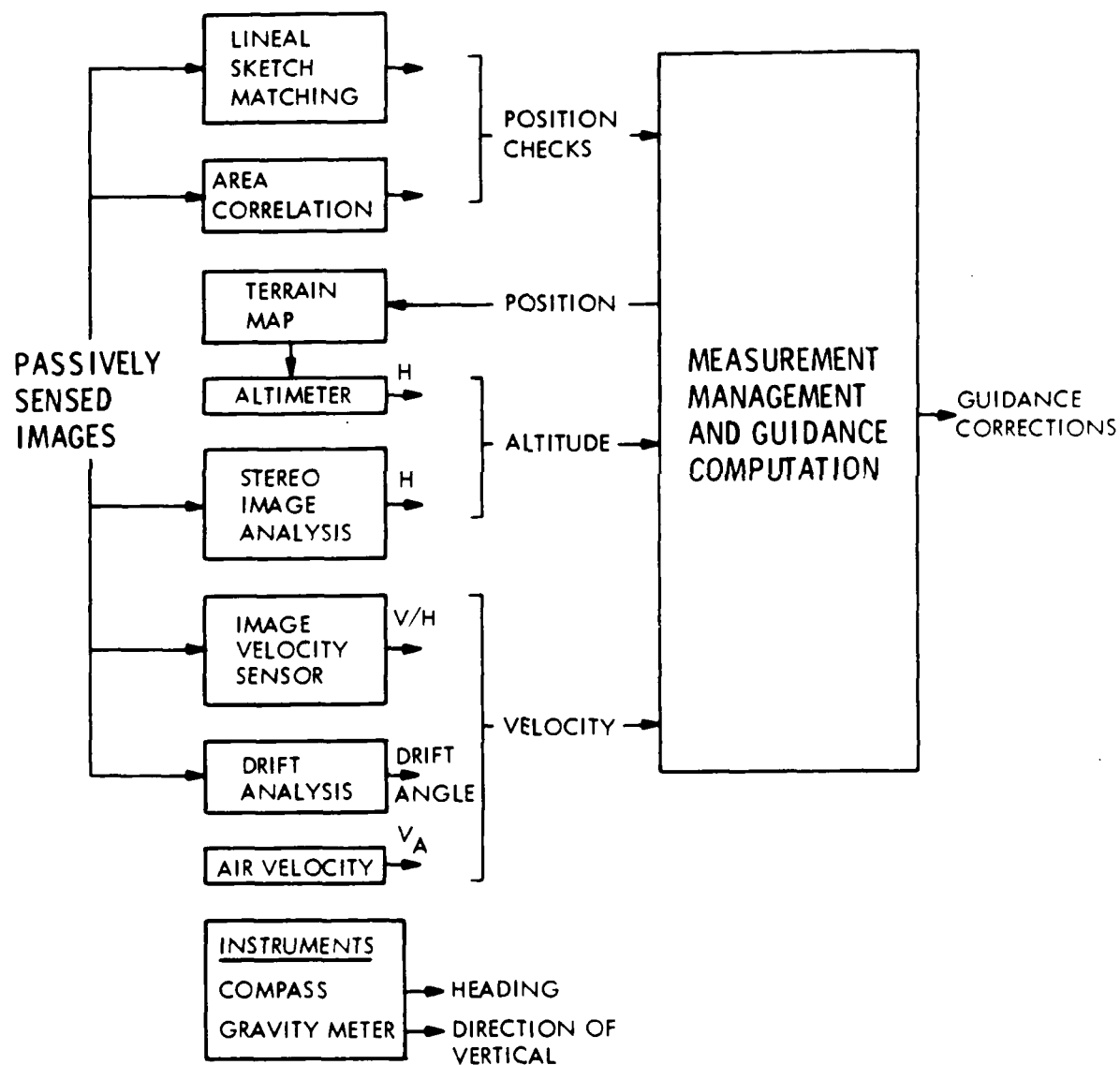
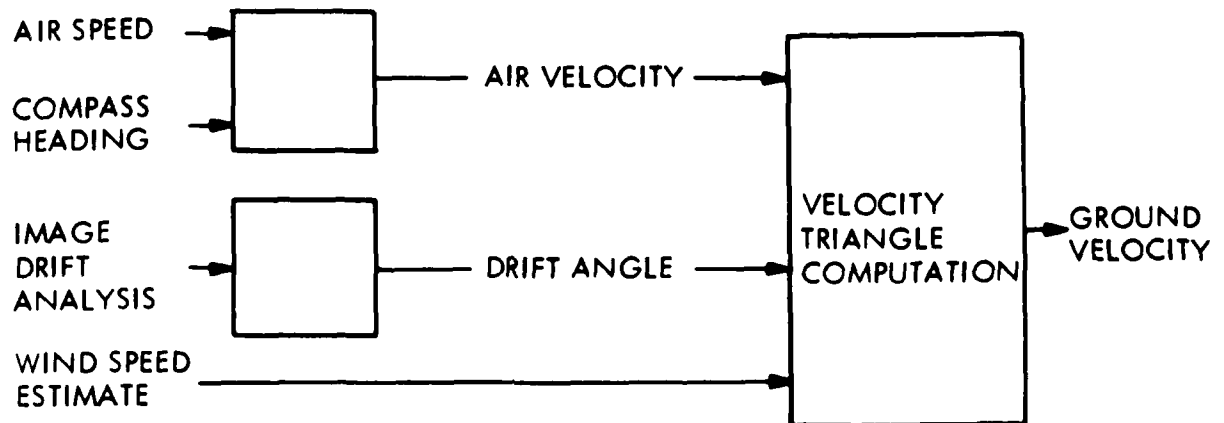
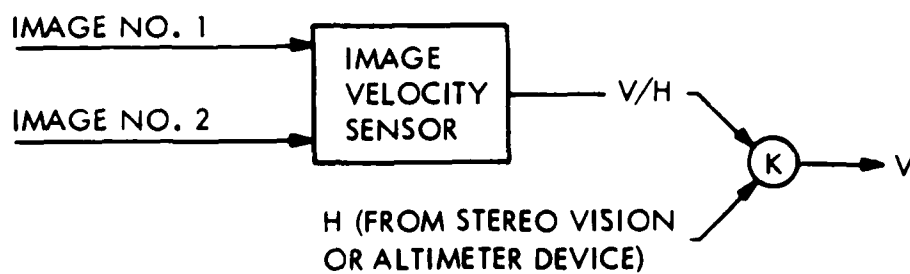


Fig. 4-1 Overall Navigation System



a. Velocity Device No. 1: Air/Wind Velocity Approach

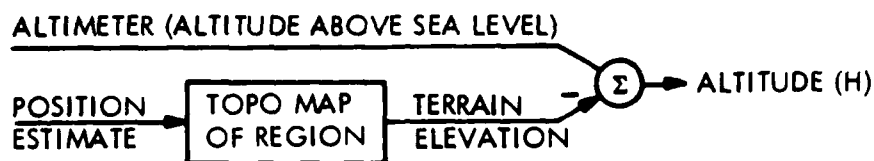


b. Velocity Device No. 2: Image Velocity Sensor

Fig. 4-2 Velocity Determining Device



a. Altitude Device No. 1 Stereo Image Analysis

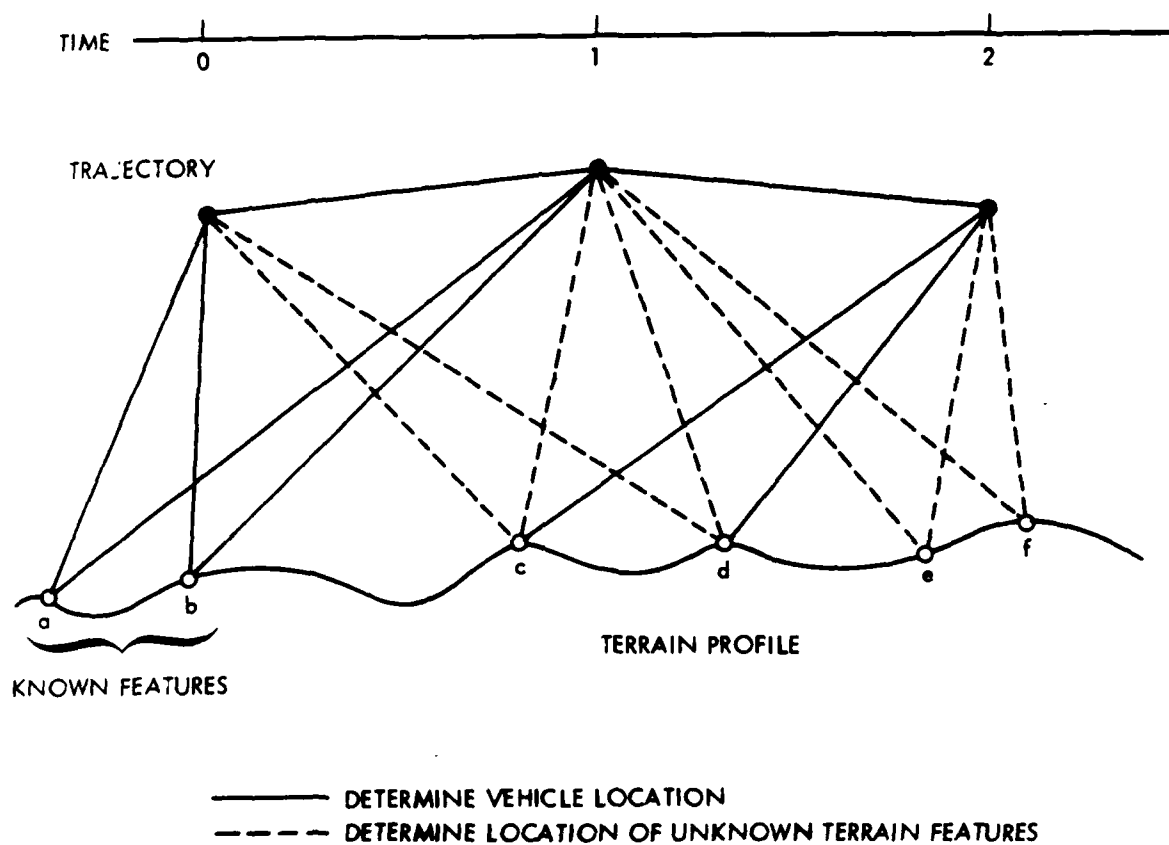


b. Altitude Device No. 2: Altitude

Fig. 4-3 Altitude Measuring Devices

4.2 Image-Based Dead Reckoning

Given a set of ground control points with known real-world positions, and given the location of the projections of these points onto the image plane, it is possible to determine the position and orientation of the camera which collected the image, a process known to photogrammetrists as "space resection." Conversely, given the positions and orientations of two cameras and the location of corresponding point-pairs in the two image planes, the real-world locations of the viewed ground points can be determined by a process known as "space intersections." Combining these two techniques iteratively produces the basis for "bootstrap stereo." Figure 4-4 shows an aircraft which has obtained images at three points in its trajectory. The bootstrap stereo process begins with the set of points a and b, whose real-world coordinates are known. (In reality, at least four points would be needed; only two are shown here to simplify the diagram.) From these, the



TIME	POSITION OF CRAFT DERIVED FROM	POSITION OF POINTS DETERMINED
ϕ	a, b	—
1	a, b	c, d
2	c, d	e, f

Fig. 4-4 The Bootstrap Navigation Concept

camera position and orientation are determined for the image frame taken at time 0. Standard image-matching correlation techniques are then used to locate these same points in the second, overlapping frame taken at time 1. This permits the second camera position and orientation to be determined.

Because the aircraft will soon be out of sight of the known landmarks, new landmark points must be established whenever possible. For this purpose, interesting points - points with a high likelihood of being matched - are selected in the first image and matched to the second image. Successfully matched points have their real-world locations calculated from the camera position and orientation data, then join the landmarks list. In Fig. 4-4, landmarks c and d are located in this manner at time 1; these new points are later used to position the aircraft at time 2. Similarly, at time 2, new landmarks e and f join the list; old landmarks a and b, which are no longer in the field of view, are dropped from the landmarks list.

Figure 4-5 presents an example of bootstrap processing. The data set is a part of a sequence of images from the Night Vision Laboratory.

At time 1, known points in Region a are used to find the camera location. This allows the (x,y,z) coordinates of points in Region b to be determined. At time 2, known points in Regions a and b are used to locate the camera at time 2. The coordinates of points in Region c are then determined. The mechanics of bootstrapping are complex, as shown in Fig. 4-6, since lists of points must be manipulated as the system goes from known points in one frame to unknown points in another.

4.3 The Image Velocity Meter Concept

A basic problem in dead reckoning is the determination of the velocity with respect to the ground. One of the Passive Navigation investigations carried out was the use of an image-based "velocity meter." This device would indicate the direction and magnitude of the ground velocity, given a sequence

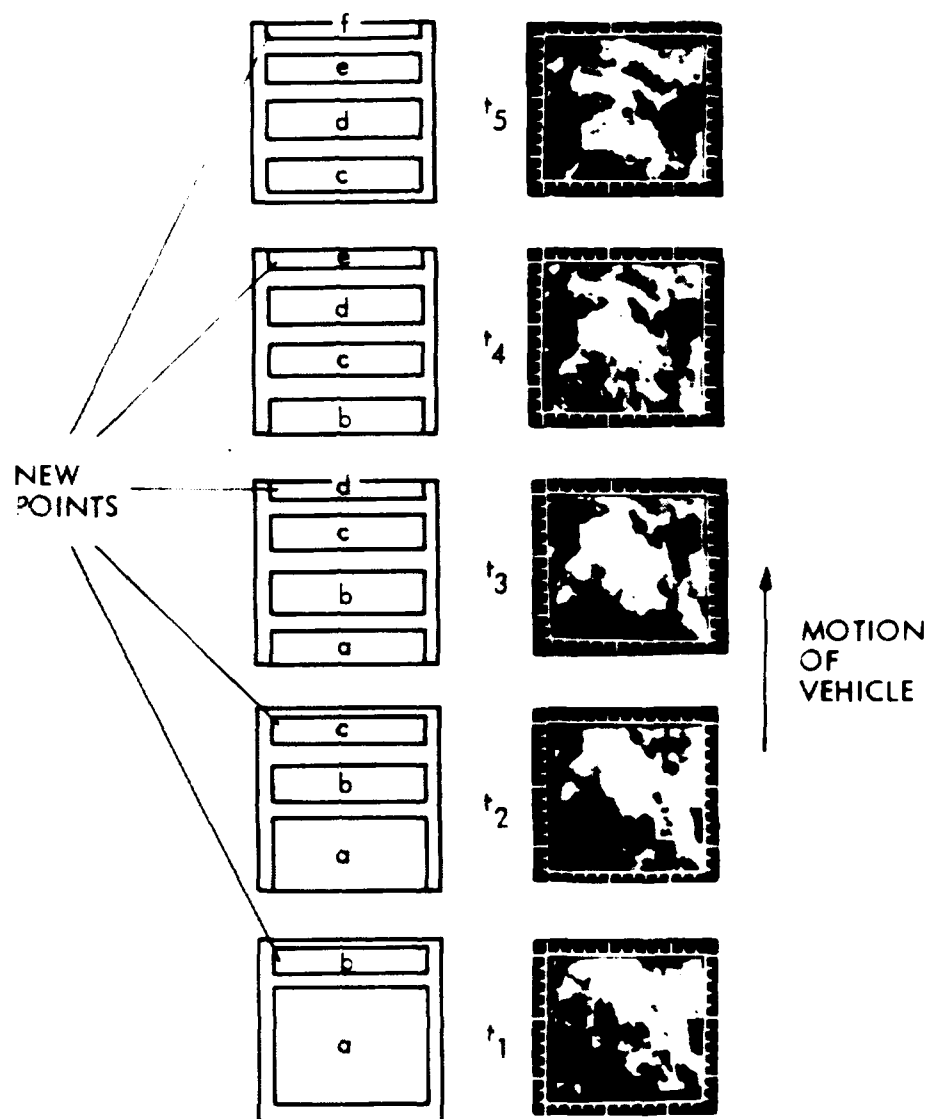


Fig. 4-5 Movement of New Points in Bootstrap Navigation

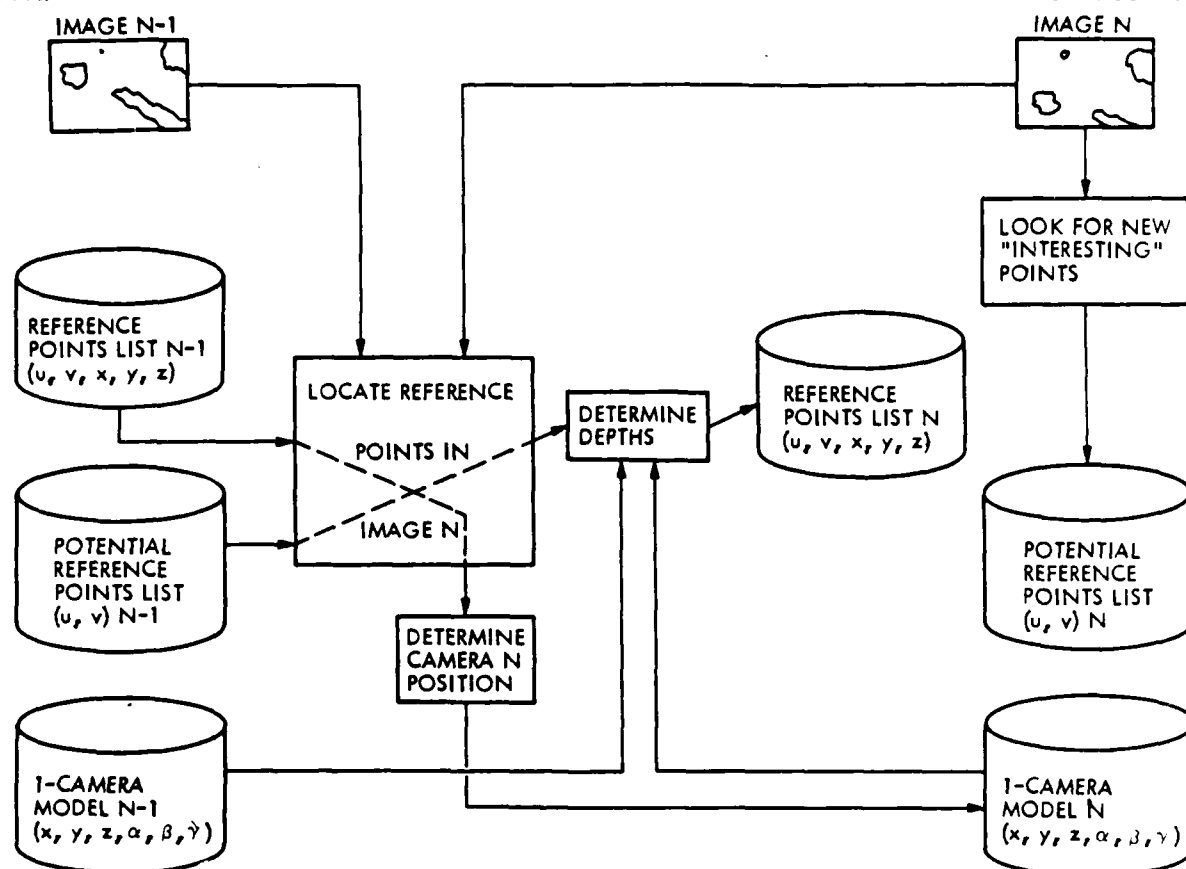


Fig. 4-6 Bootstrap Stereo Computation

of image frames and the relative altitude of the aircraft. The basic concept, related to optic flow, is described below.

4.3.1 The Basic Concept. The concept uses a solid-state electro-optical line sensor, such as the linear array charge-coupled device (CCD). Since this sensor is amenable to electronic compensation of attitude changes of an autopilot-controlled aircraft, as shown in Ref. 4-1, the necessity for electromechanical and gyroscopic stabilization of the optical axis is eliminated, thus avoiding a cost and weight penalty.

For simplicity, assume that the sensor is aligned exactly along the ground velocity vector. Every t milliseconds the sensor generates M pixels or readings of intensity which are discrete samples of a two-dimensional brightness function $f(x,y)$ taken along the ground velocity vector. As shown in Ref. 4-2, the velocity along the x axis is then:

$$V_x = (df/dt)/(df/dx)$$

Thus, we are dealing with a time derivative which is the change of intensity at a pixel from frame to frame, and a space derivative which is the local variation of $f(x,y)$ for a frame. The basic idea is to compute values of V_x for each pixel in the sensor and take the mean of the values.

An important aspect of the approach is that the electro-optical sensor is mounted on a rotational holder so that the sensor remains aligned with the velocity vector. Reference 4-2 indicates how the appropriate signals are derived from the images to obtain this direction-following.

4.3.2 Experiments Carried Out. In the experimental investigation described in Ref. 4-2, a rural scene was used, and histograms of ground velocity were computed for various misalignment angles of the sensor. Figure 4-7 summarizes nine systematic runs which were performed. The histograms show a predicted behavior, the one for an aligned sensor displays a sharp symmetric peak, while the others are skewed with a pronounced decrease in peak height, sharpness, and symmetry.

4.3.3 Conceptual Mechanization. The block diagram of the instrumentation setup is shown in Fig. 4-8, and is described in detail in Ref. 4-2. The sensor is mounted on a special holder which will be rotated in the focal plane around an axis of rotation which is perpendicular to the surface of the earth and parallel to the z axis of the aircraft. The rotational motion will be provided by a small stepping motor which need not be faster than 100 Hz (10 ms per step) or more accurately positioned than 1 deg. The system will analyze 5 deg to each side of this direction. About 40 different sets of V computations, each consisting of more than 10,000 valid values will be obtained, and the angle for which the most Gaussian-like and symmetric histogram is obtained will be selected as the correct angle. Mounting the electro-optical sensor in this manner can also aid in compensating for sudden changes in yaw, as discussed in Ref. 4-2.

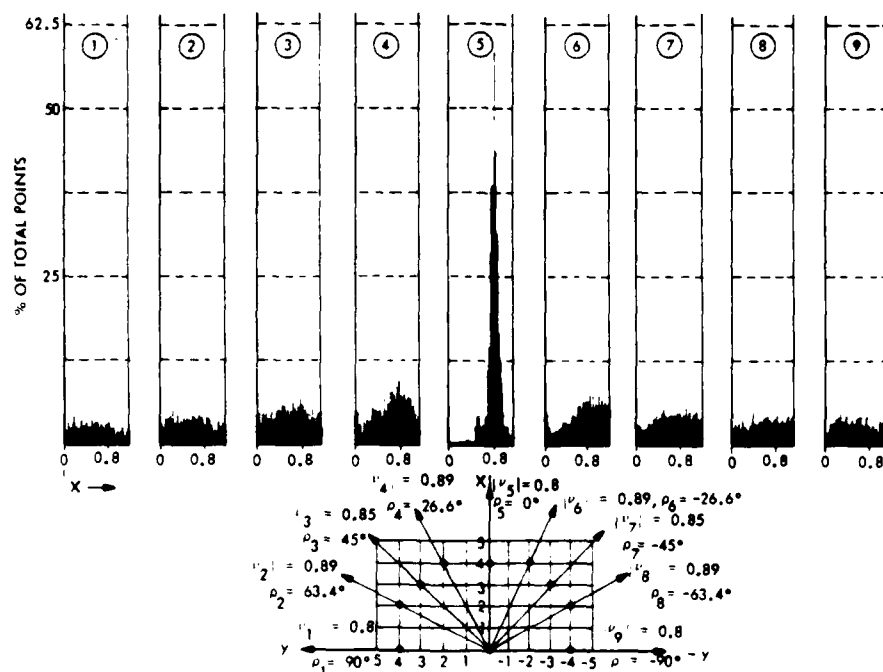


Fig. 4-7 Histograms of Velocity Computed for Various Misalignments of Sensor

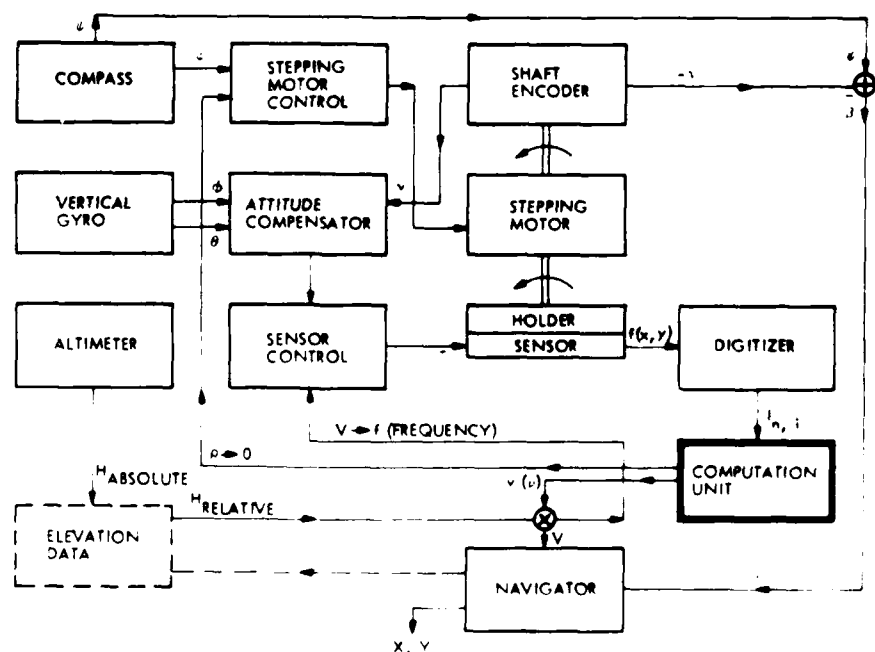


Fig. 4-8 Block Diagram of the Instrumentation Setup

5. POSITION FIXING

Position fixing is performed by analyzing sensed images, looking for landmarks expected at this time in the flight. The Landmark Database describes each landmark symbolically, as well as the image processing method for finding it in an image. The sequence in position fixing is as follows:

1. The Executive indicates to the Position Fixing Subsystem that a particular landmark is expected.
2. The landmark database provides the symbolic description of the landmark and the method of processing to be used to find it.
3. The Landmark Analysis Subsystem examines frames currently being sensed until the landmark is found.
4. If no landmark is found for N frames, then the Executive is notified.

5.1 Landmark Determination Using Intensity Images

In many image identification environments, edges (locations of high intensity gradients) characterize significant attributes of a scene. However, initial processing by edge operators provides very local (pixel-level) information about these points of high contrast. This local information is not in a form amenable for identification or matching of the major constructs in the image. There have been many approaches described in the literature which attempt to form intermediate or higher level data structures which can then serve as more reasonable primitives for the identification or matching process. For example, edge elements can be linked based on proximity and orientation and then the linked elements can be approximated by piecewise linear segments. Relaxation has also been applied, using iteration to refine the probabilistic interpretation of individual pixels as edges based on their neighbor's interpretations. Many of these approaches share a common defect; the decision to join edges or bridge gaps is made on the basis of local information within the gap. Our approach joins edges and links across gaps only if the two candidate points appear to bound the same region. Thus edge points in the vicinity of a corner (whose directions differ by 90 deg or more) are still

associated as part of the same boundary. Using this idea, it is possible to filter the linking process and extract robust boundaries (strands) of significant length. As shown in Fig. 5-1, the landmark matching consists of extracting prominent edges from an image and forming a symbolic description of the segments and the angles between them. This description is then matched against a database of symbolic descriptions. The material below summarizes our original investigations of the utility of using these strands for locating in actual imagery landmarks derived from maps, which then can be used for position update. For a more complete discussion, see Ref. 5-1.

5.2 Contour Determination Using Symchains

Low-level edge processes are frequently used to identify discontinuities. Because of noise, shadows, etc., in real-life images, these local edges do not always fit into extended, nonconflicting boundaries. One type of tracking

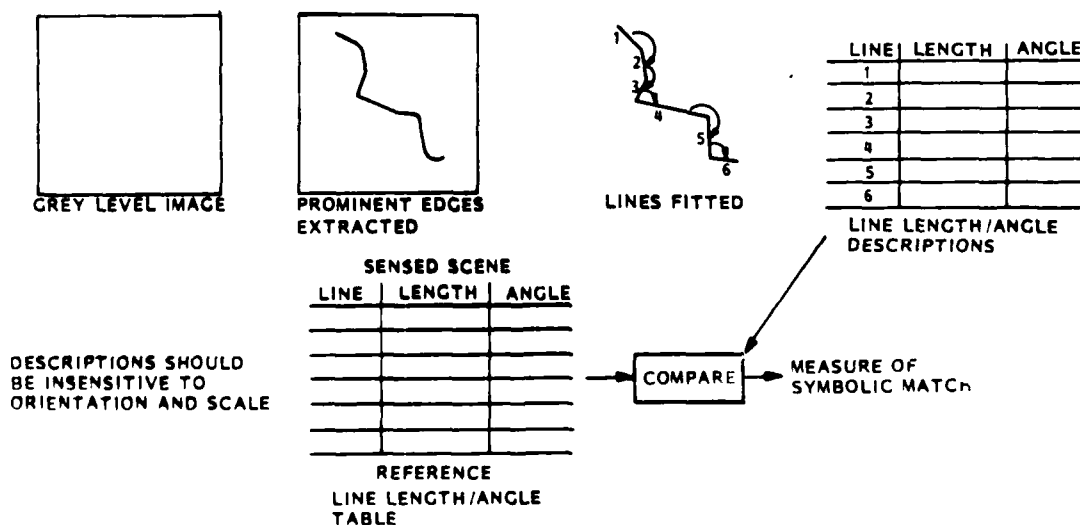


Fig. 5-1 Extracting and Matching Sensed Line Segments to Reference Line Segments

approach ranks the edge sequences formed by the low-level process using a confidence measure and then considers edge sequence associations in an ordered fashion to determine whether separated edge sequences can be combined into a more extended boundary. Another set of techniques, e.g., the Hough transform, transforms the edges into an alternate domain and associates clusters in the transformed space with long boundaries.

The original "symchain" approach consisted of the following steps:

1. Image elements that are the edges of objects are identified.
2. Boundaries are tracked, resulting in a list of edge pixels and their neighbors on the boundary.
3. The best left and right neighbors for each pixel are then selected as link elements.
4. The resulting connected set of links are formed into symchains.
5. Symchains are represented symbolically in terms of link element length and angle which are called strands.
6. This description is then compared with reference strand descriptions in the landmark data base.

When completed, the linking process has selected one clockwise associate and one counterclockwise associate, and has computed their figure of merit. Note, however, that the association is not necessarily mutual (symmetric). This is reasonable since it is possible for an edge-point to be in the vicinity of a corner at which three or more surfaces meet. It may also result from the breaking of ties. Nonetheless, for images in which the edge extraction process has produced thin (1-pixel wide) edges, the great majority of linkings turn out to be mutual, providing additional evidence of their correctness. Such links are called "symlinks."

The symlinks can be aggregated into sequences of maximal length called symchains (Ref. 5-1). The symchain associations can be extended to include as sequences chains for which the associate of an edge is part of another symchain, although the pixel's association is not mutual. This permits

alternative strongly supported chains to be included as competing interpretations. Only the resulting symchains exceeding a minimum length (currently, eight) are retained. A subsequent test determines whether edges of these long symchains exist within prespecified distances, and, if so, they are also combined into single entities. These are called strands. The strands are polygonally approximated using a split-and-merge technique. These polygonal approximations become the basic match components (Ref. 5-1). The basic steps in this process described above are illustrated in Fig. 5-2.

5.3 Symbolic Matching

Local Matching. The matching procedure uses as input the polygonally approximated strands determined from the two images being compared. The following method is used. Each polygonal boundary segment is represented as a sequence of lengths and angle values (Fig. 5-5). A match is found between two

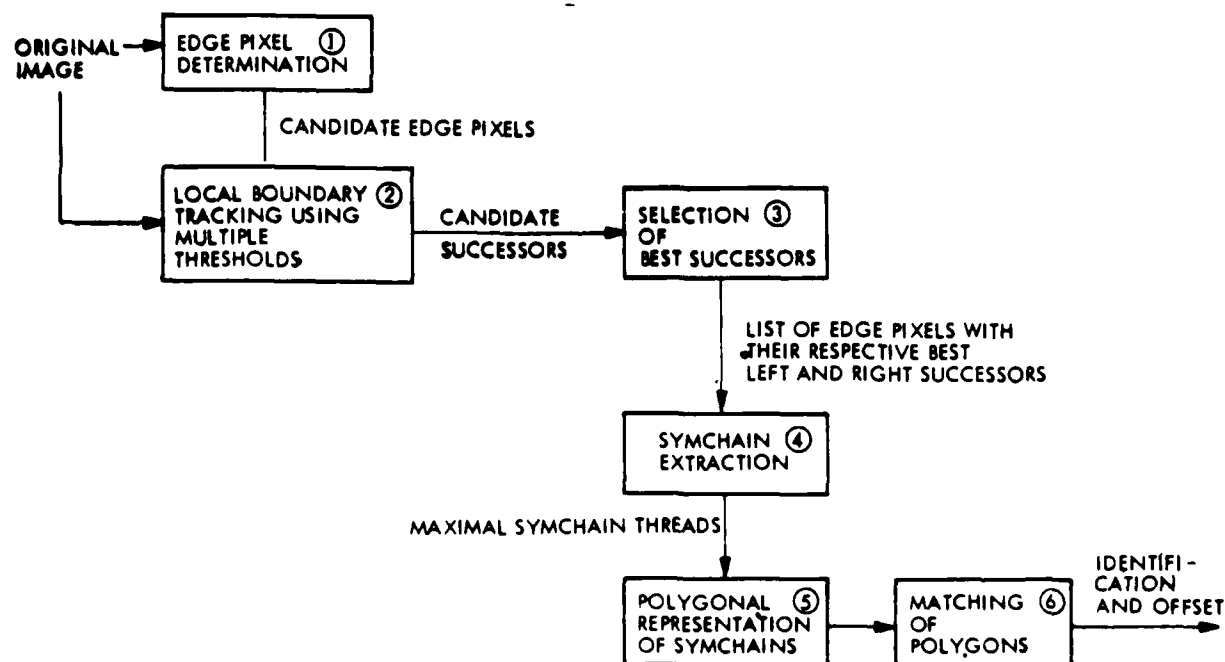


Fig. 5-2 System for Boundary Matching Using Symchains

boundary segments b_1 and b_2 whenever both of the following conditions are satisfied:

1. b_1 and b_2 are of the same type i (either length or angle).
2. The difference (b_1, b_2) is less than VAL_i , where VAL_i is based on type i (for the example case, where b_1 and b_2 are angles, b_1 and b_2 were considered a match if $b_1 - b_2 = 30$ deg).

A maximum match value is computed for each pair of strands as the maximum number of consecutive matches between two polygonal boundary segments. This is called the match length. Note that the match constraints allow a fair degree of size and angle variability. This maximizes the ability of the algorithm to obtain valid matches even in the presence of scale, rotation, and perspective effects.

Global Matching. The output of the local match algorithm is a set of reference/sensed pairs of strands which have been associated. Although identifying many local matches may imply that there is a correspondence between reference and sensed scenes, a more convincing test is whether some subset of the local matches can be combined to form a consistent set of global solutions, since this can provide stronger evidence that the sensed image is viewing the same scene as the reference. Briefly, the global match determines the mapping relationship between the reference and sensed data. The global match program checks pairs of reference/sensed local matches, associates them if they are globally consistent, and groups them into consistent entities. The group containing the largest number of consistent matches, weighted by the match lengths of each match pair, is chosen to be the global match. However, if this group does not contain matches from differing parts of the images, it is not accepted as a viable match.

The algorithm which identifies consistent global match pairs is detailed in Ref. 5-1.

5.4 Experimental Results

The NVL terrain model was used as source of imagery. Three studies were made. They were:

- Determine whether the extracted strands represent significant features of an image
- Determine whether strands similar to ones located in sensed imagery can be obtained from a map representation
- Determine whether the existence of global matches can be associated with the determination of a match in a sequence of images

5.4.1 Strands as Features. Figure 5-3 illustrates some of the steps which were performed on the original image to obtain the polygonal representation (strands) of the significant edge features. Section a of both figures shows the original digitized image. Section b illustrates the polygonal approximations of the strands. Note that many significant edge features are captured.

5.4.2 Association With Map Features. In the second test, the outline of a lake was traced from the topographic map of the artificial terrain model. This was then digitized and is illustrated in Fig. 5-4a. (The terrain map itself was not digitized, as it contained a confusion of ancillary markings.) This image was processed in the same manner as the actual terrain imagery. The polygonally approximated strands of the lake map are shown in Fig. 5-4b. These strands are compared to those which have been obtained from the sensed imagery (Fig. 5-3). Match lengths of value 4 or greater between the polygonal strands are shown in Fig. 5-4c. Note that significant portions of both images seem to be matched even though size and perspective differences between the images are substantial. The lake image was also compared to a sensed image taken from a different view. The results are shown in Fig. 5-5. Note that there are match segments of significant extent. These matched strands, after consistency checking, can then be used as input to a procedure which provides position error information.

DERIVED
SYMCHAINS

SYMCHAINS
GREATER THAN
8 PIXELS

ORIGINAL
IMAGES



Fig. 5-3 Symchain Extraction for NVL Terrain Table

WATCHED
SEGMENTS

POLYGONAL
APPROXIMATIONS
TO SYMCHAINS



MAP DATA

IMAGE DATA

Fig. 5-4 Symchain Extraction from Map and Image Data

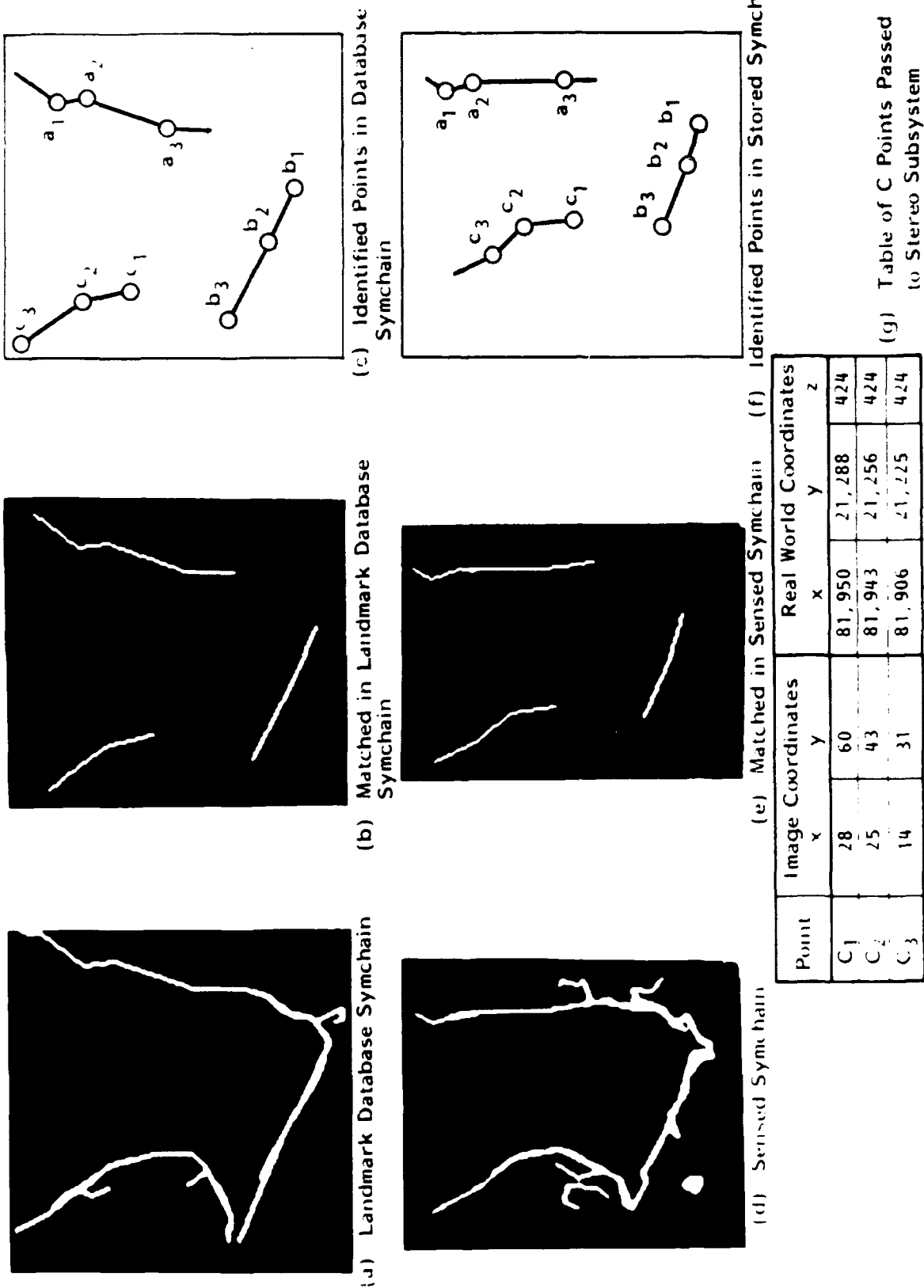


Fig. 5-5 Matching Reference and Sensed Signals

5.4.3 Global Matching for Landmark Identification. A sequence of four images of the NVL terrain model illustrates the steps in matching (Fig. 5-6). Images 33 and 34 contain a bridge whose reference representation appears in the landmark database; images 32 and 35 do not contain the bridge. For each image, the reference strands for the bridge are shown on the left and the strands extracted from the image appear to the right. The images labeled "a" show the original strands for the reference and sensed image, "b" images show the strands that matched locally, and the "c" images show the best match satisfying positional relationships. It should be noted that global matches (consistent matches in different parts of each image) were obtained only in the bridge images. Note also that global matches were obtained in both sensed images even though the forward-look angle at which the bridge was seen in the image ranged from 50 deg in image 33 to about 30 deg in image 34.

5.5 The Landmark Database

Simulating the map used by the pilot, the landmark database is an onboard database covering a particular geographic region, and indicating objects that might be of interest to the navigation or to the mission of the vehicle. The onboard landmark database can represent knowledge about landmarks in many ways:

1. A landmark database can consist of a list of point features, such as road intersections, small buildings, storage tanks, etc., described by their three-dimensional world coordinates and characteristics for each. Linear landmarks, such as roads and coastlines, can be represented as curve fragments, e.g., symchains, with associated ordered lists of world coordinates, as shown in Fig. 5-7. Ground coordinates can be expressed in a standard reference frame, the UTM grid, with elevations expressed in meters above sea level. This type of database can be accessed by location or by entity type.
2. Because the preparation of the symbolic type of database described above is time-consuming, and offers many operational limitations, it would be desirable to use a two-dimensional map representation, such as is currently used by a human pilot. If the map were to scale,

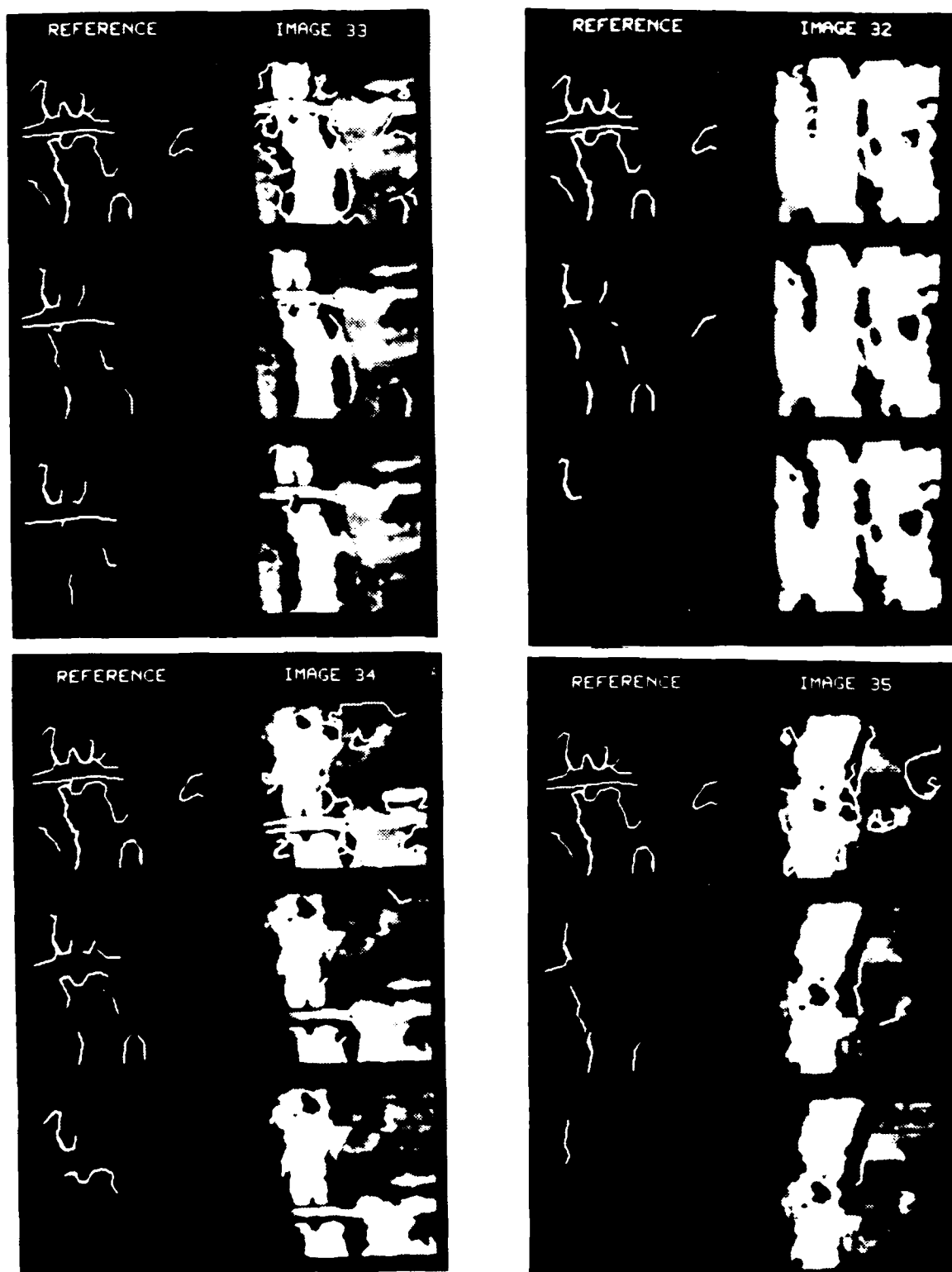


Fig. 5-6 Matching of Reference and Sensed Symchains for Bridge Reference

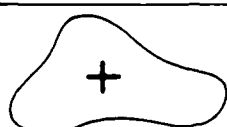
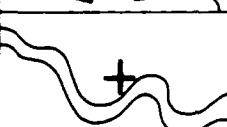
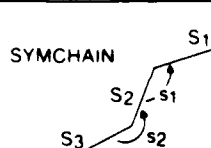
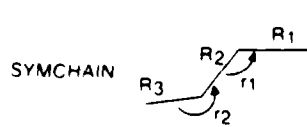
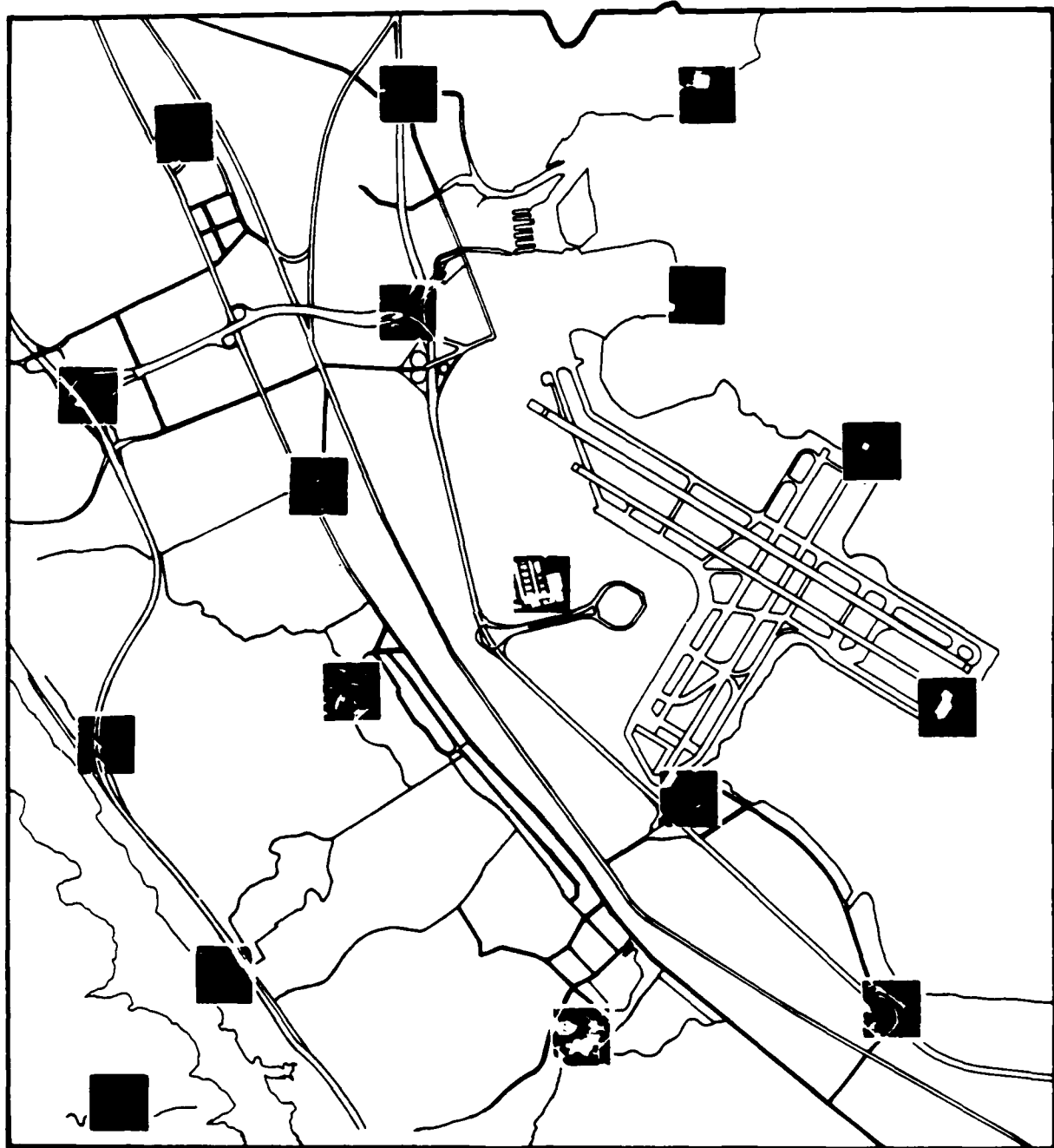
GENERAL LOCATION		DESCRIPTION												
LATITUDE	LONGITUDE	IMAGE	STORED DESCRIPTION	TYPE										
23 40'	120 10'		LAKE	HIGH LEVEL										
26 10'	120 30'		TOWN											
28 04'	121 10'		RIVER LENGTH (FT) ANGLE (DEG) 137 22 409 123 221 114	LOW LEVEL										
29 13'	121 30'		HILLS LAT LONG ELEV 29 12' 50 1" 121 29' 30 2" 1520' 29 12' 38 2" 121 29' 10 7" 2204' 29 12' 55 7" 121 29' 55 8" 1147'	TOPO-GRAPHY										
A LANDMARKS DATA BASE														
		SENSED	<table><tr><th colspan="2">DESCRIPTION</th></tr><tr><th>LENGTH</th><th>ANGLE</th></tr><tr><td>S1</td><td>s1</td></tr><tr><td>S2</td><td>s2</td></tr><tr><td>S3</td><td></td></tr></table>		DESCRIPTION		LENGTH	ANGLE	S1	s1	S2	s2	S3	
DESCRIPTION														
LENGTH	ANGLE													
S1	s1													
S2	s2													
S3														
		REFERENCE	<table><tr><td>R1</td><td>r1</td></tr><tr><td>R2</td><td>r2</td></tr><tr><td>R3</td><td></td></tr></table>		R1	r1	R2	r2	R3					
R1	r1													
R2	r2													
R3														
B SYMCHAINS OF A REFERENCE AND SENSED LANDMARK SHOWING SYMBOLIC DESCRIPTIONS														

Fig. 5-7 Landmark Database

then the automated system could plot its path and estimate its location the way the pilot does. A combined line-sketch and photo-chip map is shown in Fig. 5-8. Significant landmarks that are somewhat invariant to vehicle orientation might be represented in the form of images, and a more conventional map-matching might be used. One possible way of doing this is to develop a database preparation system capable of scanning maps used by a pilot, and able to recognize the graphic symbols used. The computer map might be stored as a two-dimensional line image.

3. Sometimes people are given a mission in terms of a roughly drawn "sketch map" that indicates the landmarks to be found. Some research has been reported in the literature on systems that automatically interpret freehand geographical sketch maps. For example, Mapsee2, developed at the University of British Columbia by Macworth, interprets the sketch maps to produce a hierarchical structural description of the scene. Mapsee's cartographic world is composed of a number of geographic systems, which are, in turn, composed of combinations of river systems, road systems, mountain ranges, shorelines, and towns. Each of these, in turn is composed of simpler subschemata, finally terminating in the primitive input sketch lines, called "chains." These descriptions can be used to guide the segmentation of an image to locate objects in a scene, such as rivers, bridge crossings, etc. This work is still in the preliminary research stage, but it points the way toward more sophisticated automated analysis of landmark database maps.



b. Lineal Sketch of Region With Photo Chips Inserted

Fig. 5-8 Example of Lineal Sketch and Reference Image Maps Used for Position Fixing

6. A NEW APPROACH TO LANDMARK NAVIGATION

6.1 Introduction

The ability to navigate autonomously while flying at low altitude has become increasingly important given the Eastern European scenario and the present (and future) sophistication of military hardware. Simply put, a vehicle must maintain a low altitude as it navigates to avoid ground-based radar and/or the possibility of inclement weather. The implication is that the vehicle will not be able to sense (or see) much of the terrain it flies over at one time. Thus, the old idea of taking pictures from a high-flying vehicle and correlating digitized images of these pictures with an onboard reference map must be abandoned. The scenario suggests that we explore ways of navigating a low-flying vehicle by the detection and matching of landmarks with a reference map. Given the present state of landmark detection, we feel this approach bears investigation.

As a low-flying vehicle senses the scene below, its sensing mechanism(s) can determine when a road or boundary of a region (a landmark) has been crossed. For example, the sensor(s) might report that a crossing was made at time = 5 min, angle = 80 deg, and crossing type = water to land. This so-called crossing information may be fully interpreted as follows: After being airborne for exactly 5 min, a vehicle with a known velocity (plus or minus some error factor) crossed from water to land. At the crossover point, the path of the vehicle made an angle of 80 deg with a line segment representing the orientation of the crossing. The path (a straight line) and crossing line segment angles are computed with respect to a common coordinate system. As the vehicle progresses, a list (sequence) of crossing information is obtained and cataloged into the memory of an onboard computer (Fig. 6-1).

If a database of line segments representing the orientation of roads and regions on a reference map is stored onboard the vehicle, then it should be possible to locate the vehicle with respect to the reference map as it flies over terrain which corresponds (in an approximate way) to the reference map.

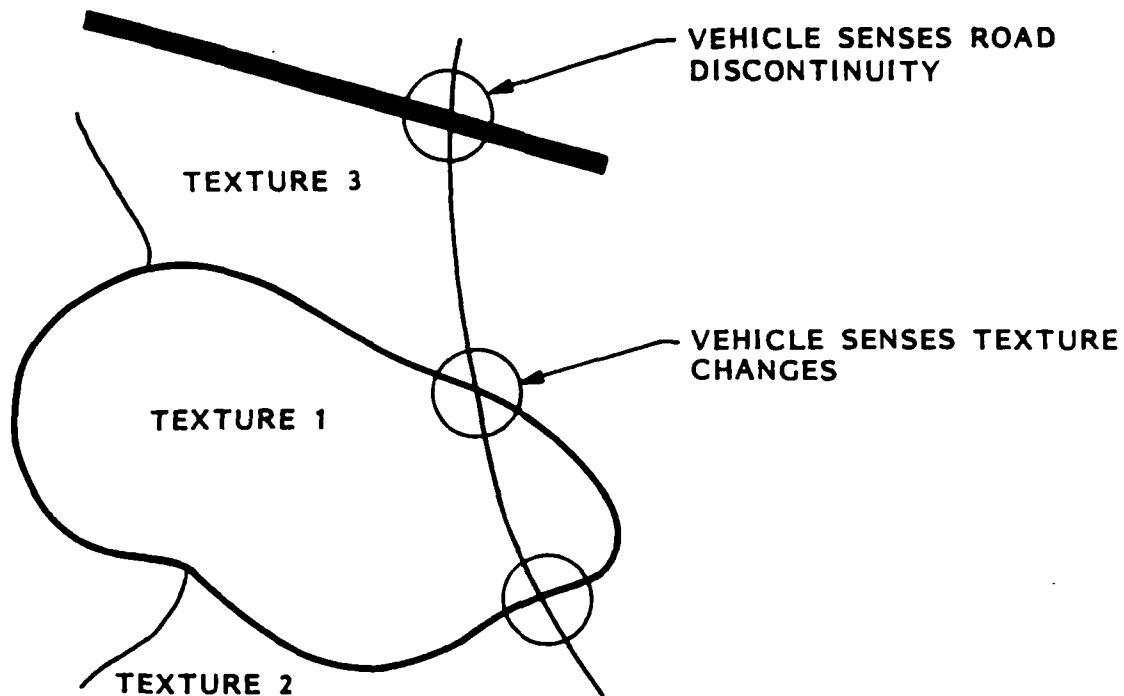


Fig. 6-1 Sketch Map Data

Our work has involved the investigation of efficient algorithms for computing the most probable path (MPP) through a line-segment data base, given a list of time, angle, and crossing type. Because uncertainty of location grows with time, an important objective of any MPP algorithm is to obtain confirmation of a consistent set of crossings as soon as possible.

6.2 Constraining the Problem

The passive navigation problem can be recast as follows: given a set of line segments S in Euclidean 2-space, and a set of intersection times and angles A derived from drawing an arbitrary time-ray through S , is it possible to reconstruct the approximate path taken by the ray based only upon the set of intersection times and angles A (Fig. 6-2). We define a time-ray to be a ray (in the geometric sense) parameterized by time.

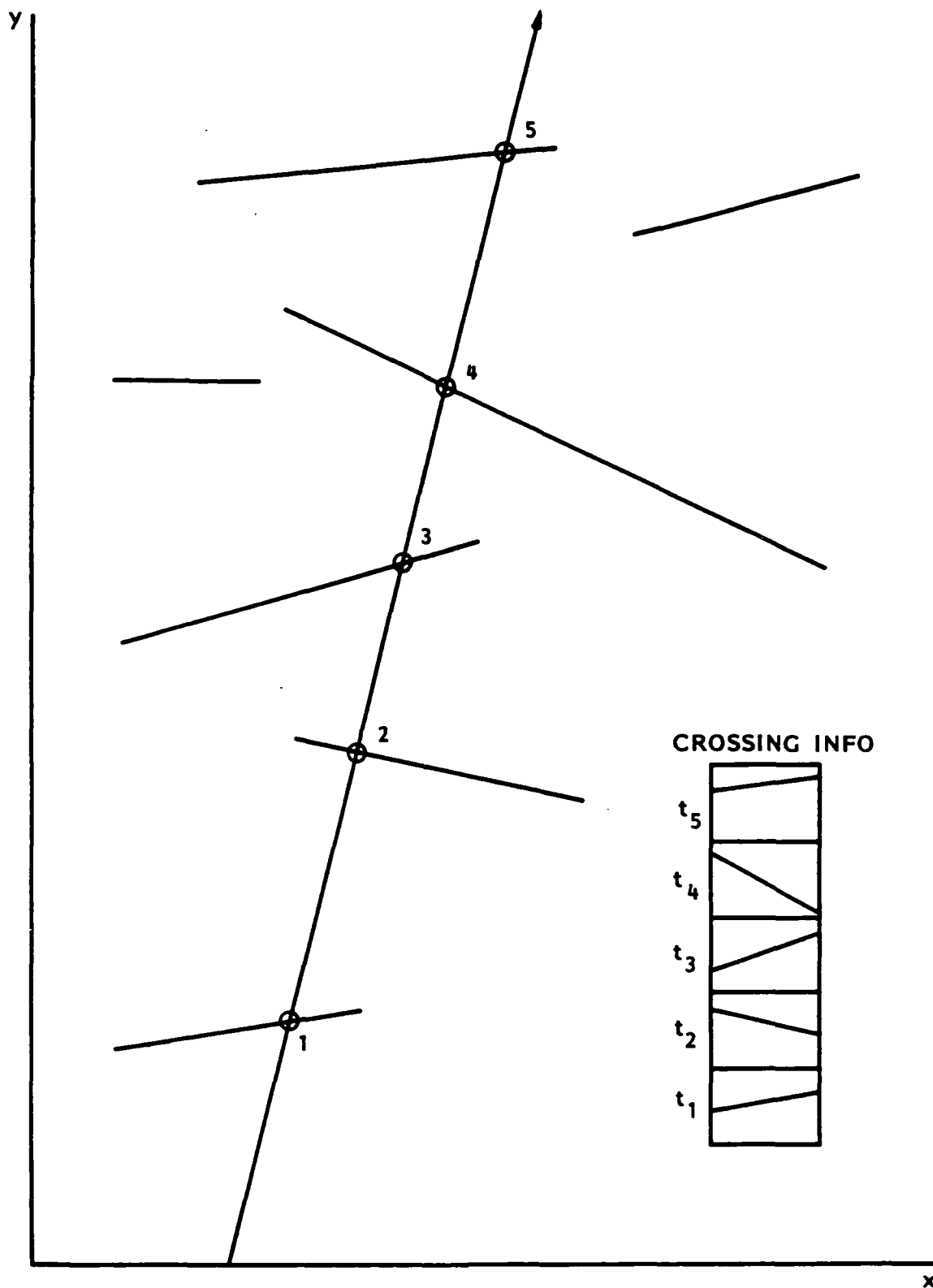


Fig. 6-2 Accumulation of Crossing Information

Clearly, this geometric problem relates back to landmark navigation because the ray can be viewed as representing the path of a low-flying vehicle, the line segments S as representing the reference map, and the set A as representing the crossing information obtained by the vehicle as it travels over terrain corresponding to the reference map.

Note that the problem just introduced is somewhat more constrained than can be inferred from the scenario of an actual low-flying vehicle navigating by landmark detection. For one, the vehicle would not always fly along a straight-line path. Two, the speed and heading of the vehicle could vary somewhat due to variations in expected flying conditions (wind, barometric pressure, etc.). Third, crossings could be missed (not sensed by the vehicle), erroneous crossing information might be recorded, or extra crossing information (no correspondence in the reference map) might be obtained. We initially deal with the simpler problem. However, it will become clear that the procedure devised to handle this problem can also handle missed crossings, as well as some variation in path velocity.

6.3 Data Preparation

Before giving the details of the position-fixing algorithm we digress to discuss the experimental procedure employed to create a set of line segments, crossing times, and angles corresponding to an actual image.

The hardware system used is a VICOM image-display system interfaced to a VAX-11/780 computer running under Digital's VMS operating system (version 3.0).

First, an image file corresponding to actual terrain is displayed on a monitor attached to the VICOM system. The lower left corner of this image is assumed to be the origin of a Cartesian coordinate system which all computations use as a reference point. A PASCAL program is then called which allows an operator to move a cursor displayed on the VICOM monitor via the keypad on a VT100 computer terminal. (A more sophisticated system would use a graphics tablet or light pen.) The cursor is used to mark the endpoints of line

segments that correspond to (possible) crossings sensed by a low-flying vehicle. Curved crossings can be approximated by a series of straight-line segments.

After endpoints are marked, the system displays a colored line connecting the endpoints and then prompts the user for the crossing type (i.e., river, road, land_water, water_land) of the line segment. After the user replies, the coordinates of the endpoints are entered into the line-segment database (a file) along with the chosen crossing type and angle that each segment makes with a half-line parallel to the x-axis emanating from its endpoint with smallest y-coordinate.

Once the line-segment database has been entered, the data preparation program prompts the operator to input a hypothetical flight path. The operator is instructed to mark the endpoints of a vector with positive y direction. The length of this vector corresponds to the distance traveled by the vehicle in one time unit (1 s); the direction of the vector gives the heading of the vehicle. The program then extends the vector so that it intersects with as many of the displayed line segments as possible.

Crossing times and angles are then computed and entered into another file which serves as input to the MPP on-line algorithm.

A related program has been written which takes as input a set of line-segment endpoints from a file, prompts the user for a hypothetical flight path, then computes crossing times and angles as described above. Although this program is less interactive, it does allow predetermined databases to be instantiated into the framework of the existing software system with relative ease.

Note that there is nothing about either scheme that cannot be generalized to handle curved flight paths, an important consideration if a low-flying vehicle has the ability to maneuver quickly to avoid possible threat areas. A curved path can be approximated by a sequence of (connected) straight-line segments.

The intersection of the path line segments with the line segment database could then be used to generate the crossing times and angles.

6.4 Position Fixing Algorithm

The algorithm we have developed is an on-line algorithm that processes inputs dynamically. Thus, after receiving the time and angle of each crossing, the most probable path (paths) of the vehicle with respect to the reference map is (are) updated. This would be in contrast to an algorithm which takes all the crossing information as initially input before attempting to compute the most probable path(s). An important feature of on-line algorithms is their ability to simulate real-time behavior. As each new input is received, a simulation log can be updated showing the steps used to delimit the number of feasible paths.

Perhaps the most important issue in designing an on-line position fixing algorithm is the desire to carry along more than one possible path at a time, yet avoid the possibility of combinatorial explosion. This can be accomplished by employing a recursive data structure called a list-of-lists, somewhat similar to a binary tree (see Appendix B). Storage for this structure can be dynamically allocated and deallocated as the algorithm first considers and then prunes nonfeasible paths. Another advantage is that a recursive data structure allows for recursive list processing thereby simplifying the coding of the algorithm.

The gist of the algorithm is as follows: For each crossing (time, angle, and type) all line segments in the line-segment database are added to the path list data structure to form a new set of paths that contain one more line segment than the previous set of paths. (Line segments are identified by numbers.) If, during this process, a path is found to have a repetition (the same line segment twice) it is immediately dropped from the path list.

A probability measure is computed for each new path based upon the difference between the observed crossing angle and the heading vector of the vehicle with

the new path line segment. Since probabilities are normally constrained to fall in the range $[0,1]$ the difference between the line segment and the vehicle heading vector is mapped (linearly) into this range. This measure is multiplied by the probability stored in the tail of the new path's base. We define the tail of a path to be the last added line segment and the base of a new path to be the new path minus the last added line segment. Paths with insufficient probability (less than some threshold) are then pruned from the data structure path list.

Next, the tail of each remaining path is examined to see if its type (road, river, waterland, landwater) coincides with the crossing type of the observation. Those which do not coincide, result in the pruning of their respective paths. (Note that this last step is optional, we could have just as well been carried on under the assumption that we have no crossing type information.)

The final step consists of examining all paths to see if they are feasible with respect to the approximate velocity of the vehicle. This is accomplished by displacing the line segment stored in the tail of the base of the path by a distance which is a function of the vehicle's heading and the time that has elapsed since the last crossing (Fig. 6-3). A trapezoid (or box) of uncertainty is created around the displaced line segment, the size of which is a function of the elapsed time. This box allows for uncertainty in the direction and speed of the vehicle. If the tail of the new path intersects the box of uncertainty, then the new path is retained along with that part of the line segment which is contained in the box of uncertainty. This would be the line segment used in any future box of uncertainty calculations.

6.5 Sample Run of the MPP Algorithm

In this section the reader is taken through a sample terminal session involving the MPP algorithm. Hopefully this will make clear some of the ideas presented in the previous section. Throughout the discussion please refer to Figs. 6-4 through 6-8 and the simulation log (Appendix A). The figures are

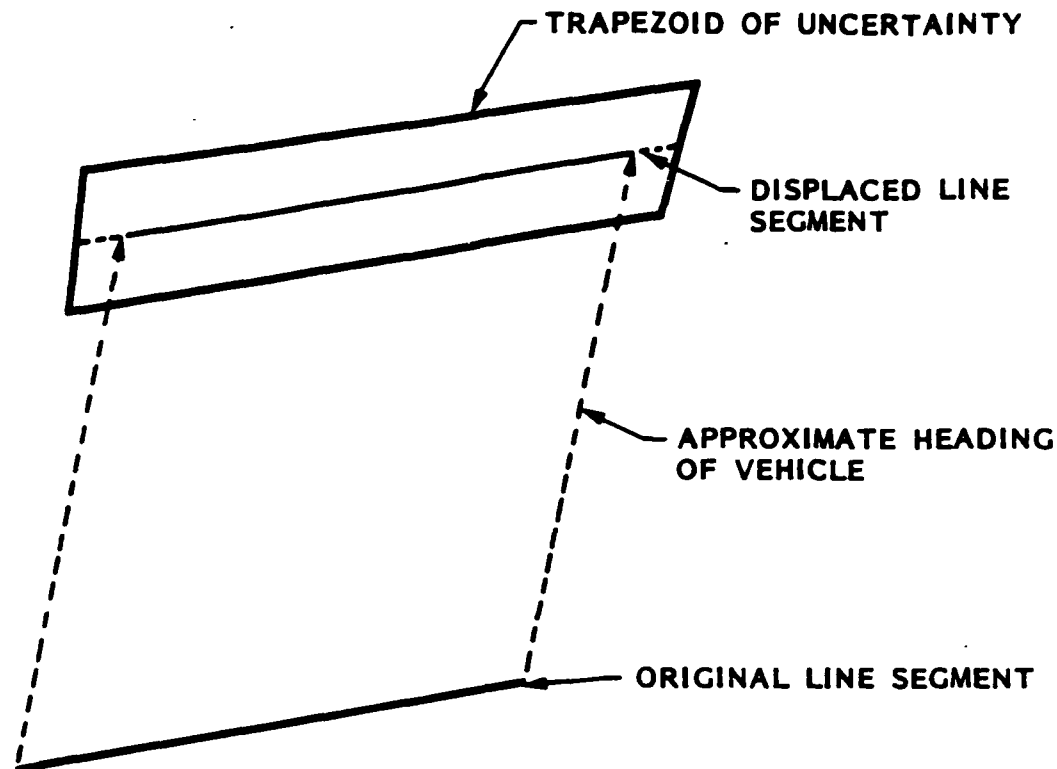


Fig. 6-3 Displacing a Line Segment to Create a Trapezoid of Uncertainty

facsimiles derived from hardcopy photographs of the VICOM monitor taken during different stages of the algorithm's progress.

At the beginning of the simulation, the user is asked to input several pieces of information. First, the name of a file which contains the velocity and starting point of the vehicle with respect to the line-segment database (reference map). The starting point is never actually used by the MPP algorithm, instead it is used to draw the true path the vehicle took through the line segment data base (Fig. 6-4). Thus, it serves as a check of the MPP algorithm as the computation for the most probable path continues.

Second, the user is prompted for the name of a file containing the line-segment database. The MPP program then displays the line-segment database and true path of the vehicle with respect to the data base. Each line segment is given an identifying number along with the known crossing type (RI = river, RO = road, LA = land to water, WA = water to land).

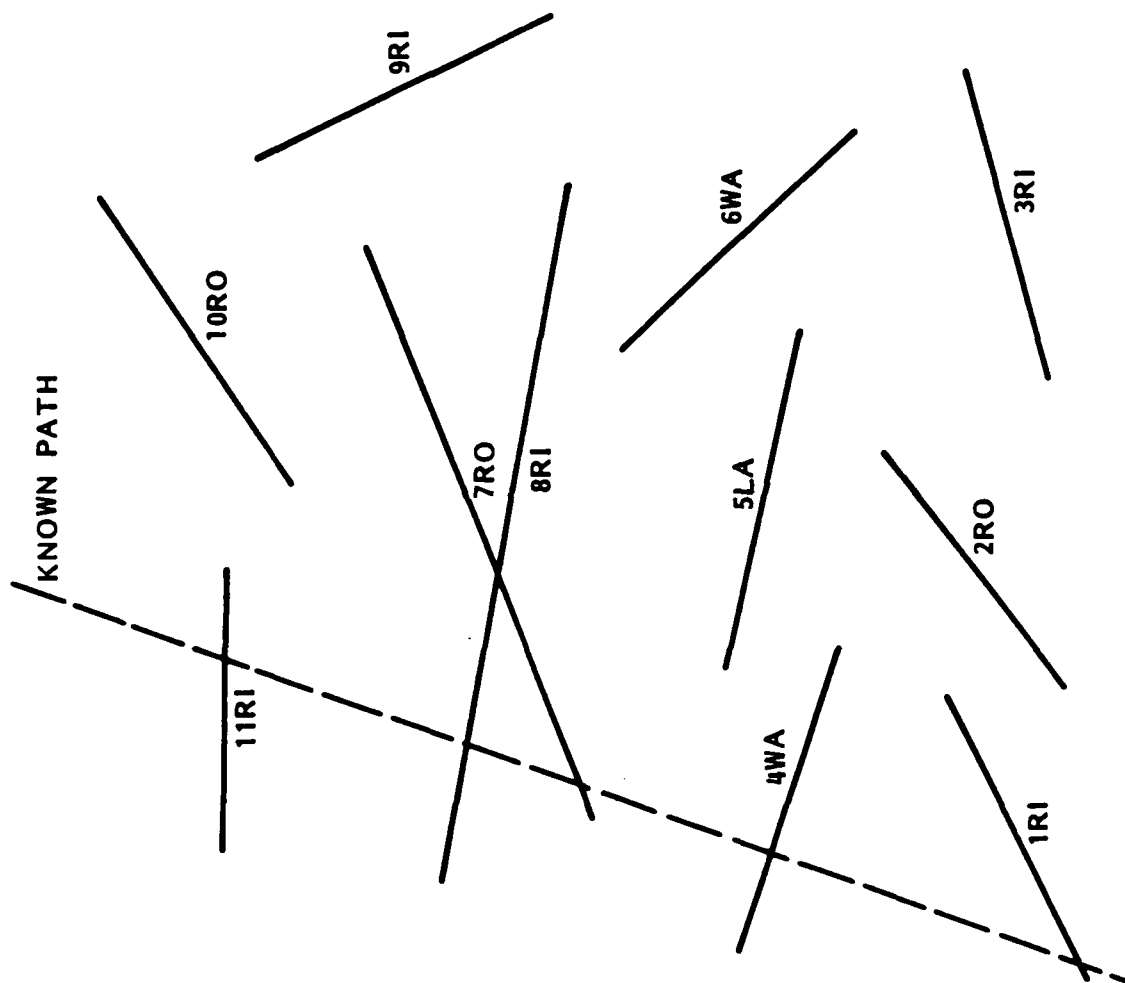


Fig. 6-4 Reference Map Line Segment Database

Third, the user is asked to input the name of a file which contains crossing times, directions, and times.

Fourth, the user is asked to input the size of a window in the reference data base. Essentially, the window delimits the number of segments that are considered for possible path inclusion (as a function of approximate position). The starting point of the vehicle is assumed to be anywhere inside the window at time zero. That is, we do not assume an exact starting point with respect to the line-segment database. This step is included to simulate a reference database derived from a set of time-sequenced imagery. That is, depending on the approximate position of the vehicle, the algorithm considers adding to the path list only those line segments taken from one part of an image or set of images.

At this point the simulation is ready to commence. Typing any character (and hitting the RETURN key) starts the simulation. The time, direction, and type of the first crossing (observation) are placed into the simulation log as well as the line segments that fall within the window. The first crossing, a road intersection, is satisfied by line segments 1 and 3. The window is computed by moving a line centered at the bottom of the reference database ($x = 256$, $y = 0$) perpendicular to the heading vector in the positive y direction. The distance moved is a linear function of the crossing time. All line segments less than a "window size" number of units away from this line are then added to each path. The simulation log shows the probability and type of the tail of each new path as it is computed.

An attempt is then made to prune all paths whose tails have less than the threshold probability (in this case 0.8). Those paths which remain feasible are printed into the log. In a similar fashion, those paths whose tails have the incorrect crossing type are deleted from the path list. Because this is the first crossing, the remaining paths are not checked for uncertainty with respect to velocity considerations.

Crossing information for the second observation, a water-to-land crossing, is read in, and the process of computing the window and pruning paths with insufficient probability and crossing type is carried out again. This time however, paths are examined to see if they are feasible with respect to the heading and speed of the vehicle. Uncertainty boxes are drawn on the VICOM monitor by displacing line segments 1 and 3 (Fig. 6-5). Note that line segment 4 is in the box derived from line segment 1 which means that (1 4) is a feasible path. That part of 4 which is in the box will be carried along (stored in the path-list data structure) for future box of uncertainty calculations. Note also that 4 is not in the box derived from 3, therefore (3 4) cannot be a feasible path.

The computation of each remaining crossing (observation) is handled in a similar fashion as the second crossing. The third observation is a road crossing. Figure 6-6 shows the box of uncertainty resulting from the displacement of that part of line segment 4 contained in the previous box of uncertainty. Line segment 7 intersects this box, so the updated path is (1 4 7).

For the fourth observation, a river crossing, line segment 7 propagates, resulting in a box of uncertainty which contains line segment 8 (Fig. 6-7). The updated path is therefore (1 4 7 8). Note that the width of the box of uncertainty corresponding to the fourth crossing is quite small because the elapsed time between the third and fourth crossings is relatively small.

The fifth observation is also a river crossing. Line segment 11 is contained in the box uncertainty resulting from the displacement of line segment 8, giving the final path of (1 4 7 8 11) (Fig. 6-8).

The log shows the steps taken to compute new paths and prune those which are infeasible with respect to the various criteria. At the end of the simulation the final path is printed into the log along with the endpoints of the clipped line segments which resulted from the box-of-uncertainty calculations. (These

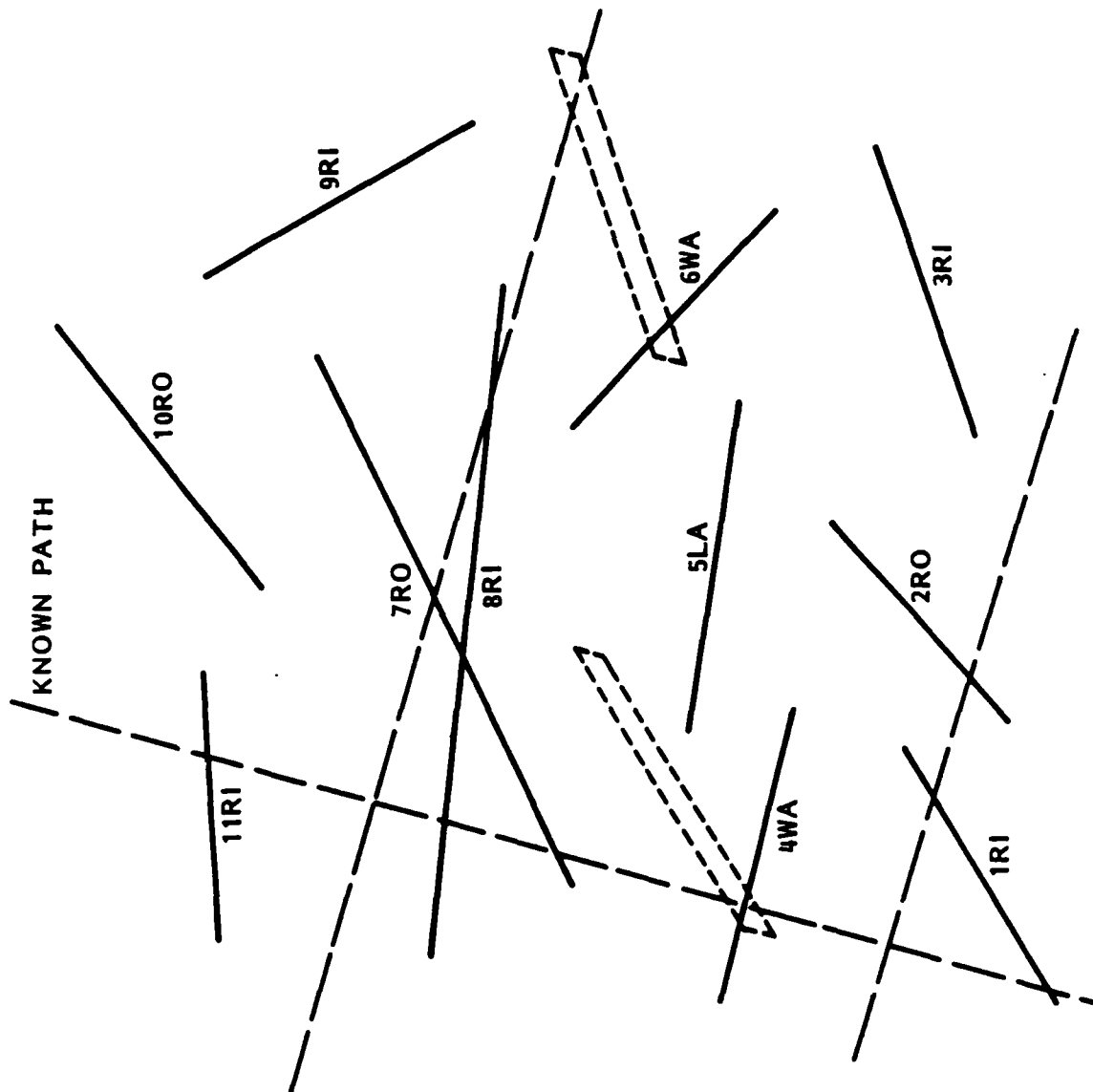


Fig. 6-5 Confirming the Feasibility of Path (1 4)

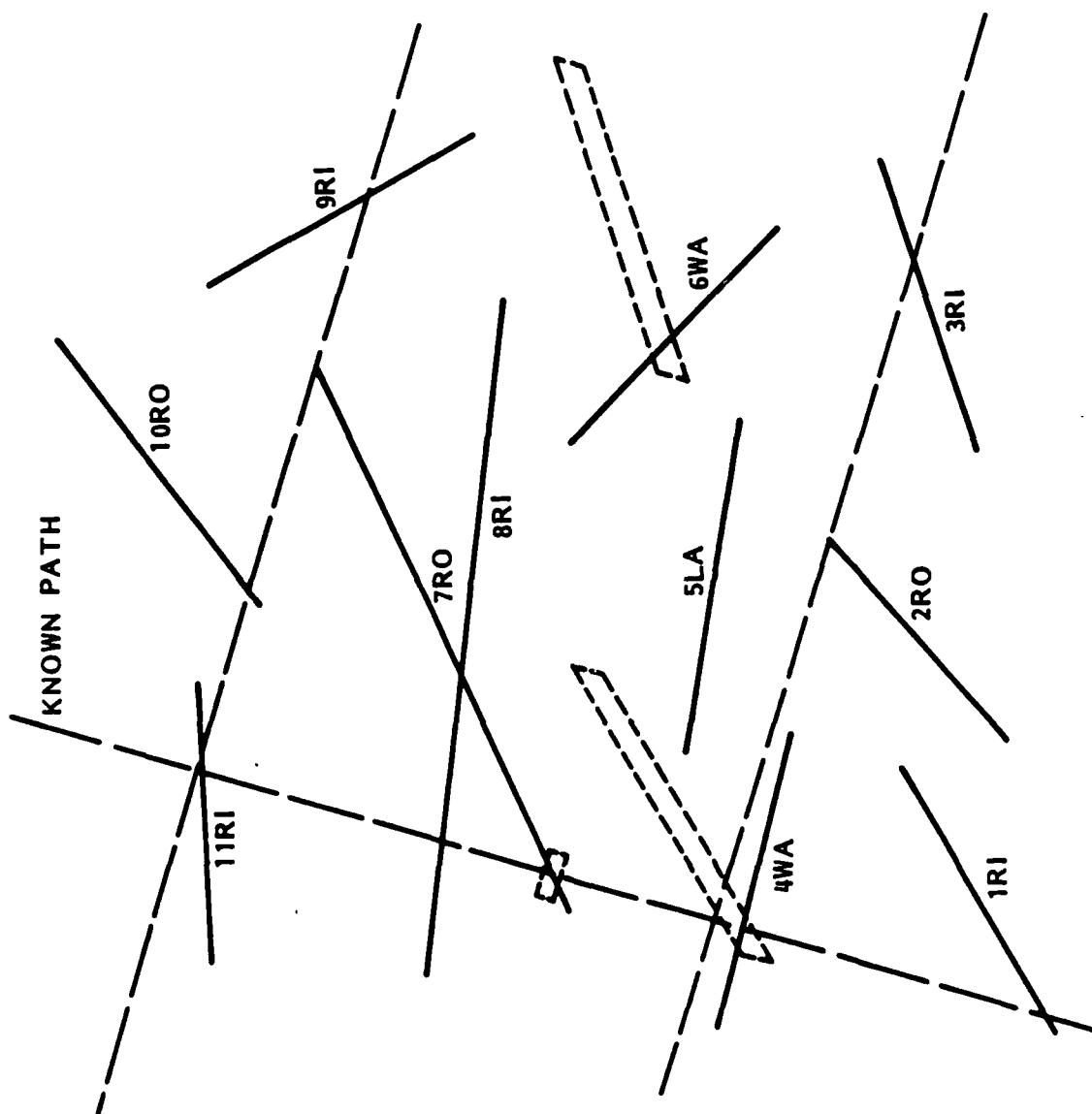


Fig. 6-6 Box of Uncertainty Resulting from Displacement of Line Segment 4

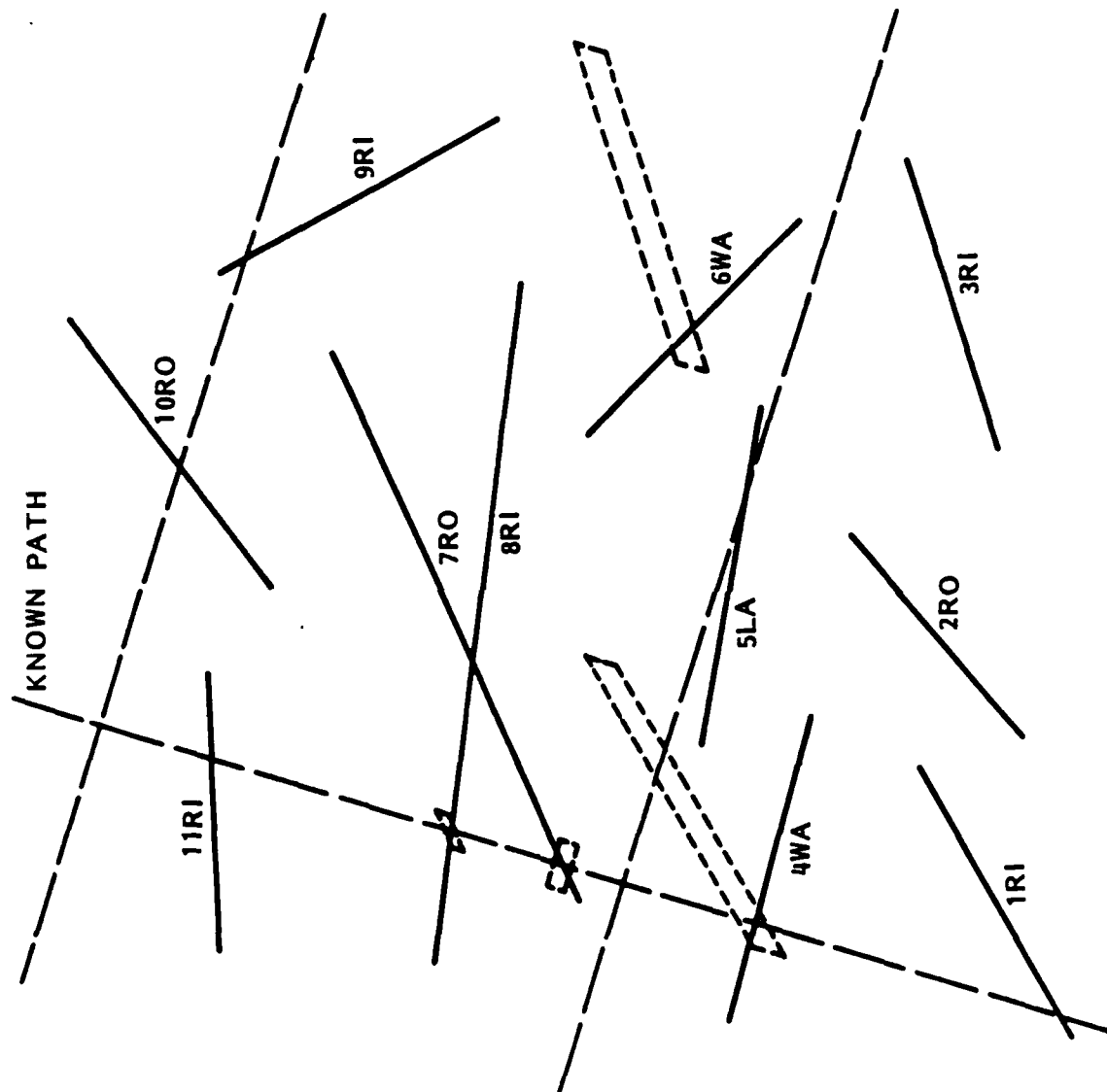


Fig. 6-7 Box of Uncertainty Resulting From Displacement of Line Segment 7

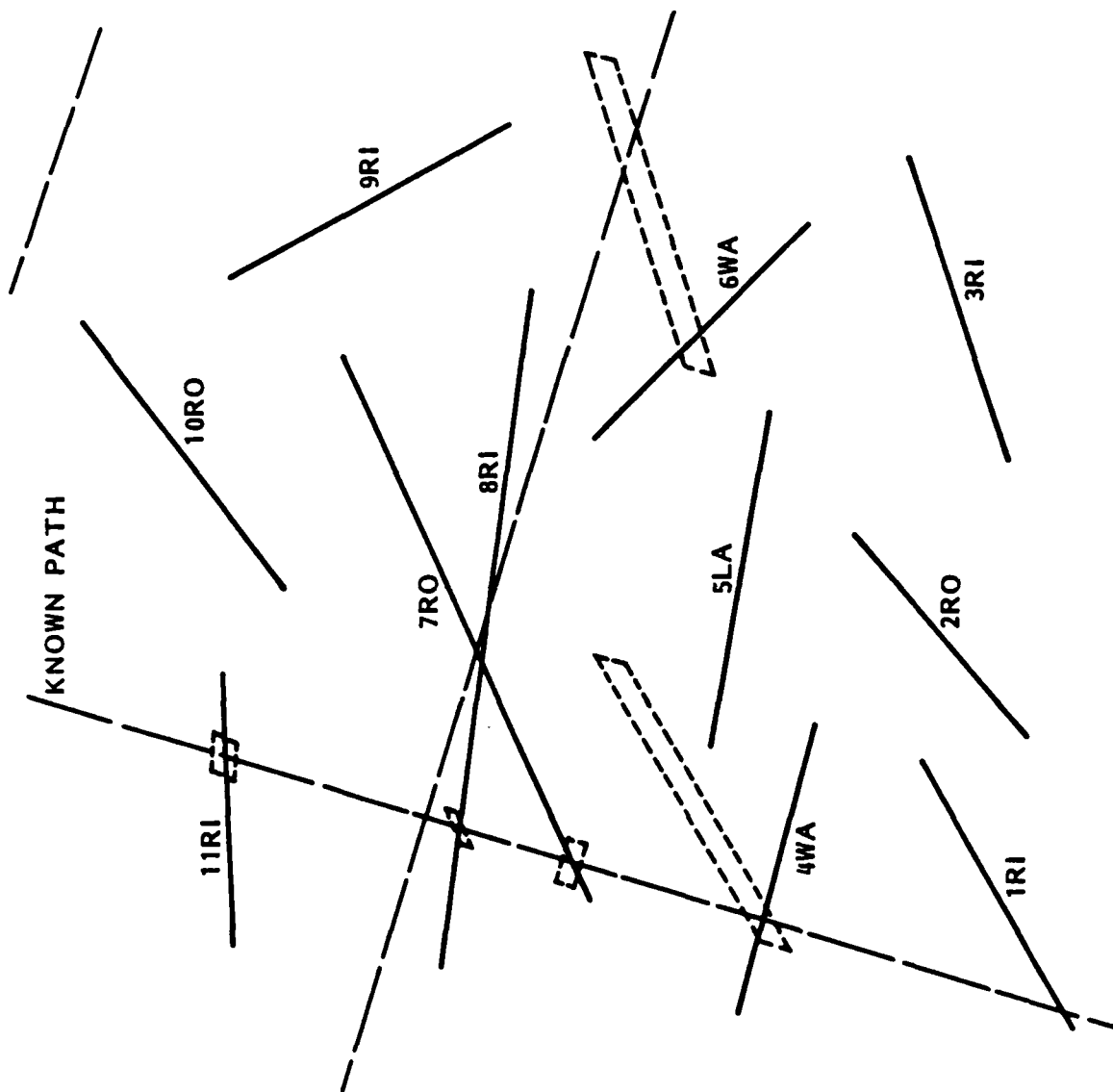


Fig. 6-8 Final Path (1 4 7 8 11)

line segments can be used to reconstruct the corridor-like path the vehicle must have taken through the line-segment database.)

The example used in this section to demonstrate the MPP algorithm is somewhat contrived. A more realistic scenario is presented in Fig. 6-9.

This is a LANDSAT image of the East Watsonville, CA, area with the line-segment database superimposed over the image. Following the MPP algorithm through to completion produces Fig. 6-10 and path (1 10 13 20 21). We chose not to include a simulation log for this case because of its size. Furthermore, the progression of line segments superimposed over the image is somewhat difficult to follow. As an expository device, the simpler example better serves to illustrate the fundamental details of the MPP algorithm.

6.6 Missed Crossings

Missed crossings are those observation times and angles which the sensor fails to detect while flying over terrain. This would reduce the amount of information available to the MPP navigation program. However, this usually does not cause the algorithm to fail, because the algorithm operates on the premise that the observation data is accurate. Thus, as long as a few observations have been accurately sensed, the probability is high that the correct correspondence can be found in the line-segment reference map.

Several tests were carried out which verify this performance. One, two, and three observations were removed from the set of five observations used in the navigation problem of Section 6.5. In each case the MPP algorithm reported a path corridor with respectively one, two, or three fewer line segments. The line segments of these path corridors were in general longer (or wider) because that part of the line segments contained in the boxes of uncertainty grew larger when more time elapsed between crossings. Figure 6-11 shows the boxes of uncertainty used to compute the final path (1 7 11) which correspond to the removal of crossing data corresponding to line segments 4 and 8 in Figure 6-4.

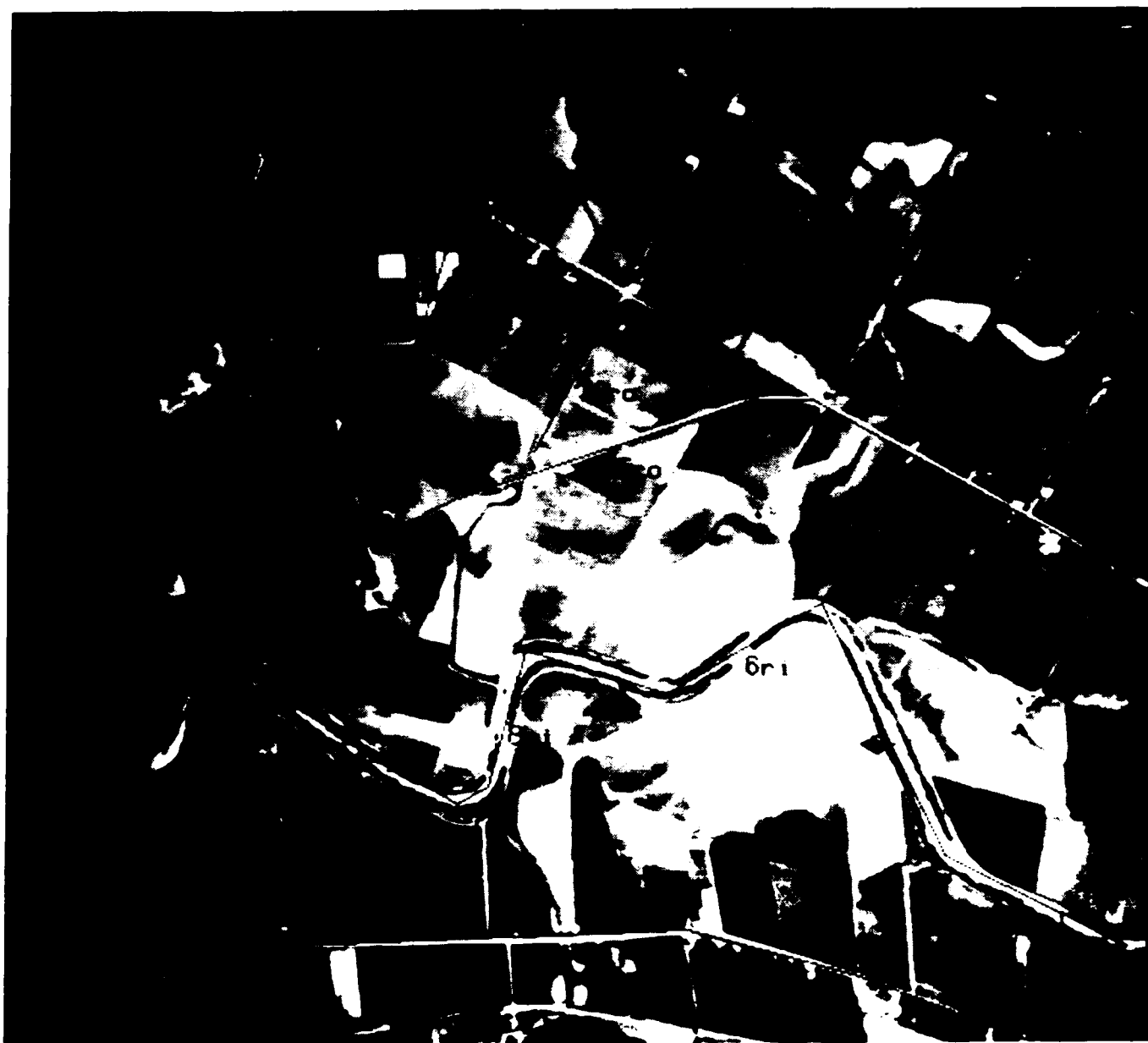


Fig. 6-9 Line Segment Database Superimposed Over Image

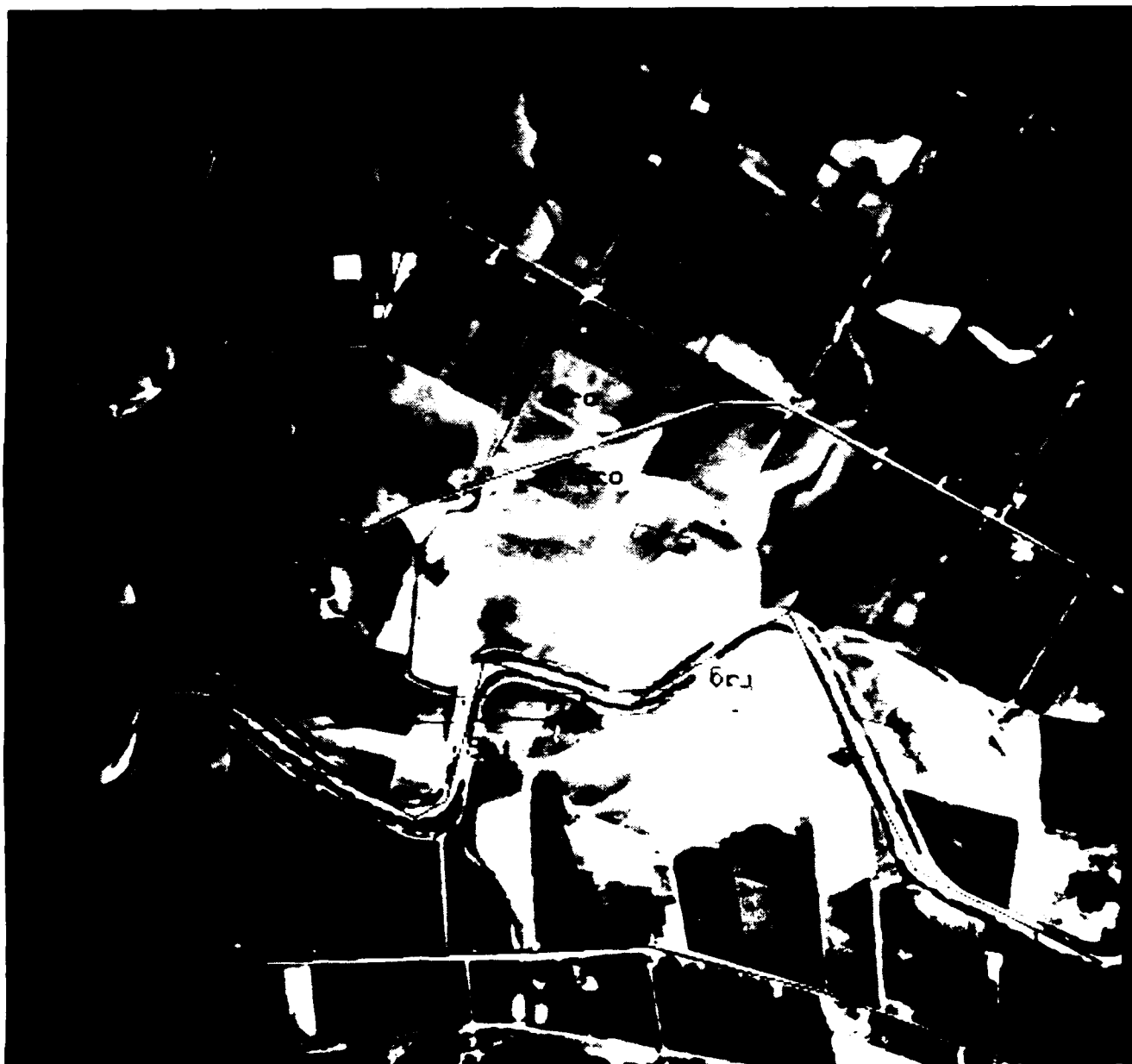


Fig. 6-10 Path Calculation for Database and Image of Figure 6-9

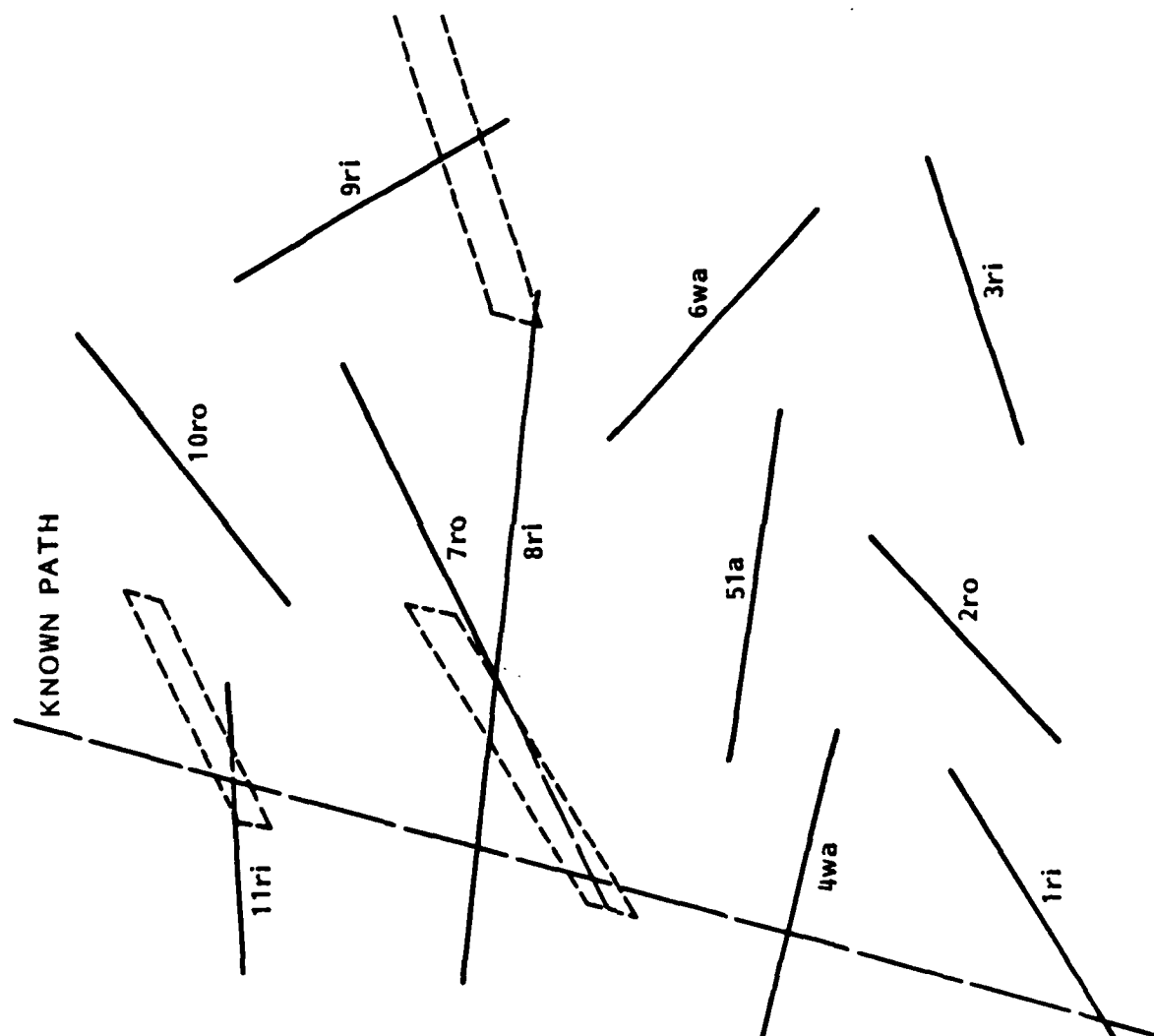


Fig. 6-11 Boxes of Uncertainty Corresponding to Removal of Crossing Data for Line Segments 4 and 8

6.7 Extraneous Crossings

Extraneous crossings (or incorrect crossings) are those which have no clear correspondence in the reference map. This type of sensor information can be fatal to the present MPP algorithm since the computed path is based upon a consistent set of crossing data. For example, given a new crossing time and angle, the MPP algorithm may not be able to find any line segment intersecting any box of uncertainty. Or, the angle of a line segment may correspond rather poorly to the observed crossing angle. Ultimately, this leads to the maintenance of incorrect paths, or the possible deletion of paths which contain mostly correct matches (versus a path with all correct matches). One can discriminate against extra crossings by only looking for crossings when one expects to find a match (based upon velocity) in the line-segment database. Unfortunately, this still does not exclude the possibility of finding an incorrect match. An improved MPP algorithm would have a mechanism for retaining the most important entries in each path. One idea, which has its foundations in the field of artificial intelligence, is a method called backtrack. Essentially a (heuristic) set of rules is chosen which determine how to prune the list of available solutions. In the context of the problem at hand, one such rule would be: An incorrect entry e will most likely lead to a rather severe decrease in the accumulated probability. Thus, the backtrack procedure could proceed by reconstructing a path based on the removal of e . If the accumulated probability of this path fell above some threshold, we would accept it. Otherwise, we would repeat the procedure with the entry which gave rise to the secondmost severe decrease in accumulated probability, then the thirdmost, and so forth. This procedure could also be extended to remove the two most severe candidates at the same time.

A nice feature of this heuristic is that it can be extended to the problem of path size. That is, at some point in time the number of entries in a path may grow to become quite large. The extra expense of maintaining a large list of path line segments could become prohibitive. Again, the probability rule could be used to prune a path to a manageable size. We believe that the ideas outlined in this section merit further study.

6.8 MPP Microprocessor Feasibility

Because microprocessors are relatively cheap in comparison to the hardware needed to sustain controlled low-level flight, we have examined the question of whether the MPP program developed on the VAX-VICOM system could be run on an available Intel 8086-87 microprocessor system.

A serial line connecting the two systems was installed and the PASCAL code was copied from the VAX to the Intel system. The code was then stripped of all sections which involved the display of line segments, boxes of uncertainty, etc., on the VICOM system. Slight differences in PASCAL syntax involving the definition of procedure modules were hand translated into Intel's PASCAL-86.

The program was then compiled, linked, and debugged. The execution time for the example of 11 line segments and 5 observations used in Section 6.5 was approximately 0.72 s. The 8086 processor and 8087 math compressor ran at 5 MHz. The program required approximately 20 kilobytes of memory for code and 10 kilobytes for data. This included code and utility routines for reading the line-segment database and interfacing with an operator.

While the use of the present MPP algorithm inside an actual low-flying vehicle is highly improbable, the microprocessor experimentation at least gives some feeling for the time required by that part of a hardware/software system employing a modified version of the MPP algorithm (e.g., such as that outlined in Section 6.7). Given the present state of the art in sensing hardware, it would seem that the time to process sensory data and recover crossing information could be somewhat greater than the running time of the MPP algorithm.

7. STEREO STUDIES

Stereo analysis is a crucial part of the bootstrap dead-reckoning procedures, and also offers the possibility of finding topographic landmarks. This section describes the nature of the stereo problem, and indicates the studies that were carried out.

7.1 Stereo Techniques

The basic requirement for stereo computation is a stereo pair of images, i.e., two distinct views of an object. In the Image-Based Navigation System, a sequence of overlapping images is obtained by a single sensor flown over the scene; these images are processed in pairs as motion stereo.

There are four basic techniques used in the stereo system, as shown in Fig. 7-1:

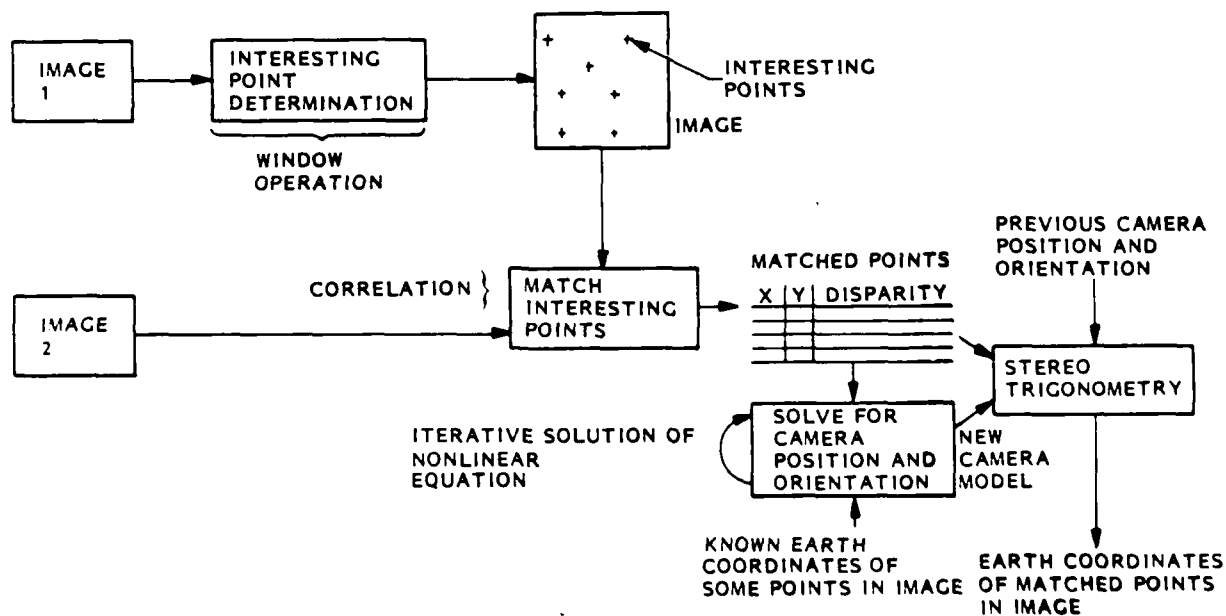


Fig. 7-1 Stereo Image Processing

1. Camera calibration - determining the camera position and orientation from known ground control points
2. Interesting point selection - choosing potential new control points
3. Point matching - pairing a point in one image with its corresponding point in a second, overlapping image
4. Control point positioning - locating points on the ground, given their positions in two images and the relevant camera positions and orientations

These techniques will be discussed in the following sections.

7.1.1 Camera Calibration. Given a set of ground control points with known real-world positions, (X_i, Y_i, Z_i) , and given the perceived locations of these points on the image plane (U_i, V_i) , it is possible to determine the position (X_0, Y_0, Z_0) and orientation (HEADING, PITCH, ROLL) of the camera that sensed the images. This is accomplished by minimizing the mean of the errors between each image plane point (U_i, V_i) and the projection (U_i', V_i') of that point's real-world location (X_i, Y_i, Z_i) onto the image. Because the equations are highly nonlinear, a solution is usually sought by iterating on a linearized version of the problem (Ref. 7-1).

7.1.2 Interesting Point Selection. The basis of stereo processing is the generalized matching of corresponding points between the two images of a stereo pair. Experience has shown, however, that some image points will match better than others, based largely on the intensity information surrounding the point. For this reason, interesting points - points with a high likelihood of being matched (Ref. 5-1) - are selected as the points to be matched.

Matching is done on the basis of the normalized cross correlation between small windows of data (typically 11×11) around the two points in question. If the window to be matched contains little information, it can correlate reasonably well with any other area of similar low information. To avoid mismatches from attempting to use such areas, the simple statistical variance of the image intensities over the window was used as an early measure of

information (Ref. 5-1), with only areas of high information being acceptable candidates for matching.

Matching also has trouble with strong linear edges, since an otherwise featureless area containing a strong edge will match equally well anywhere along the edge. To reject such areas, the notion of directed variance was introduced (Ref. 5-1). Points with poor visual texture will have low directed variance because adjacent samples differ little in any of the directions. Points with linear edges will show low directed variance in the direction of the edge. Conversely, points with high directed variance should avoid these defects. Thus, "interesting points" were defined to be local maxima in this "interest operator" directed variance, Fig. 7-2.

We have developed another indicator of the presence of an edge in the window - the ratio of the directional variances, taken in perpendicular pairs. This measure takes advantage of the fact that a window with a strong edge will have much greater information content across the edge than along it. Because this

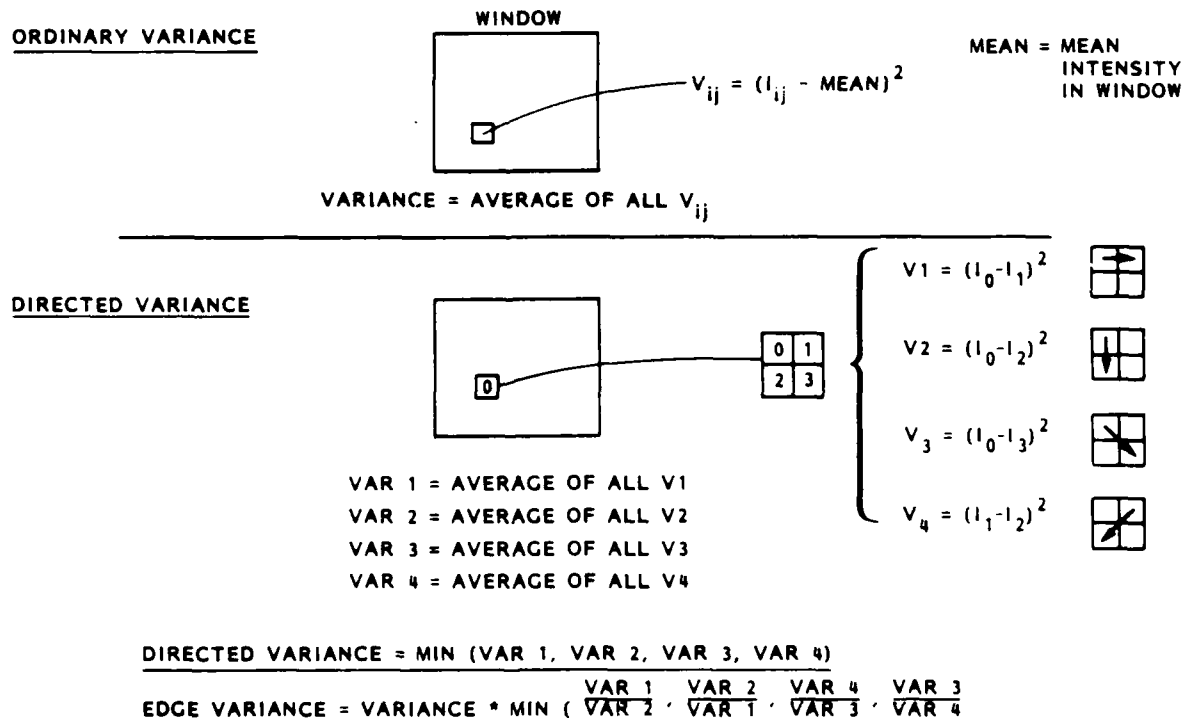


Fig. 7-2 Techniques for Interesting Point Determination

measure does not give an indication of low information, we have combined it with ordinary variance to form an interest measure we call edge variance.

Interesting point determination uses ordinary variance, directed variance, and edged variance, as shown in Fig. 7-3. Note that there are significant differences in interesting points for each technique.

7.1.3 Point Matching. The actual matching of points in an image pair is done by maximizing normalized cross correlation over small windows surrounding the points. Given an approximation to the displacement which described the match, a simple spiraling grid search is a fairly efficient way to refine the precise match. To provide that initial approximation, we have employed a form of reduction matching (Ref. 5-1).

As shown in Fig. 7-4, a hierarchy of N-ary reduction images is first created.

Each $N \times N$ square of pixels in an image is averaged to form a single pixel at the next level. (For the example shown, $N = 3$). This reduction process is repeated at each level, stopping when the image becomes approximately the size of the correlation windows being used.

Matching then begins at the smallest images. A window centered on the image is matched via the spiral search, beginning at the center of the second image. Thereafter, each matched point spawns four points around itself, offset by half a window radius along the diagonals of the window.

These are mapped down to the next level of images, carrying their parent's displacement (suitably magnified) as their suggested match approximation.

These points then have their displacements refined by a spiraling search before spawning new points. This process continues until the largest images are reached, effectively setting up a grid of control points for matching.

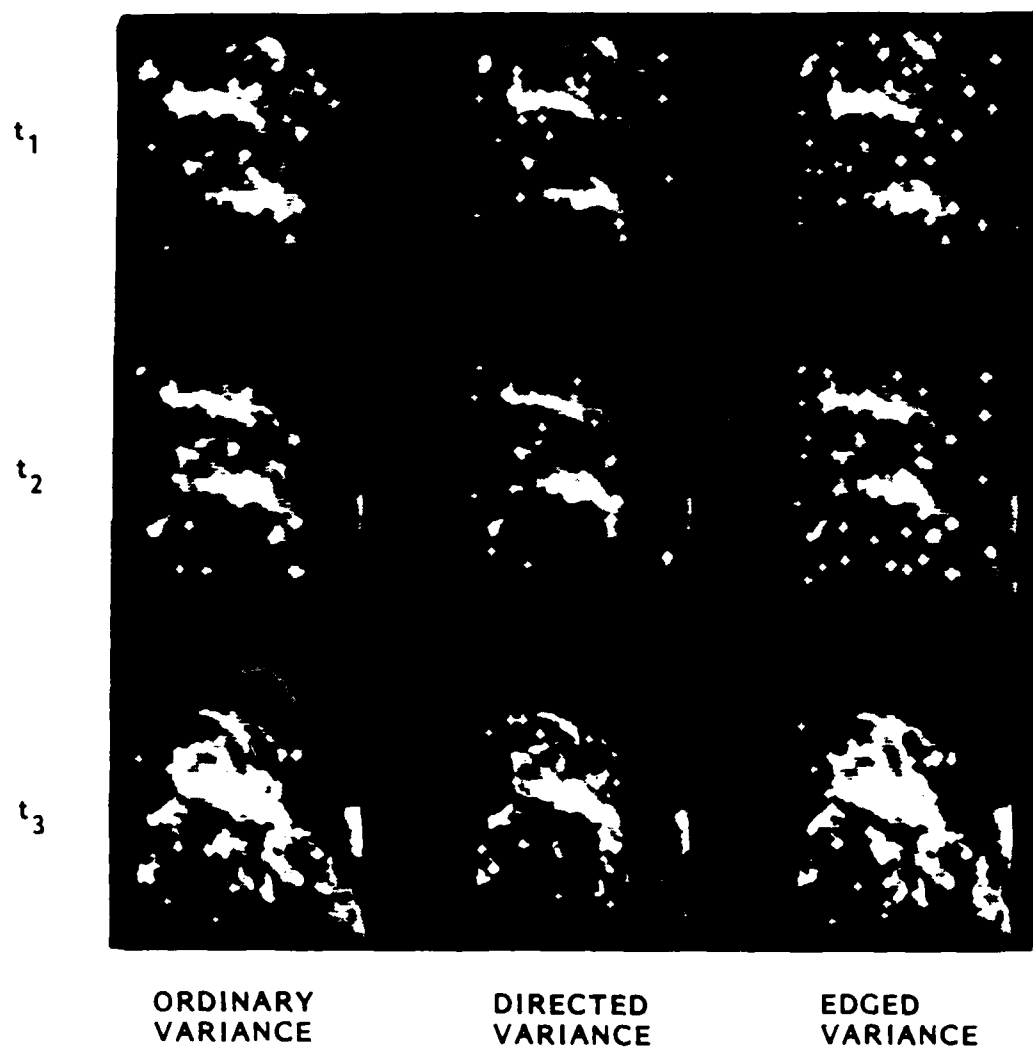


Fig. 7-3 Interesting Point Determination Using Three Different Measures

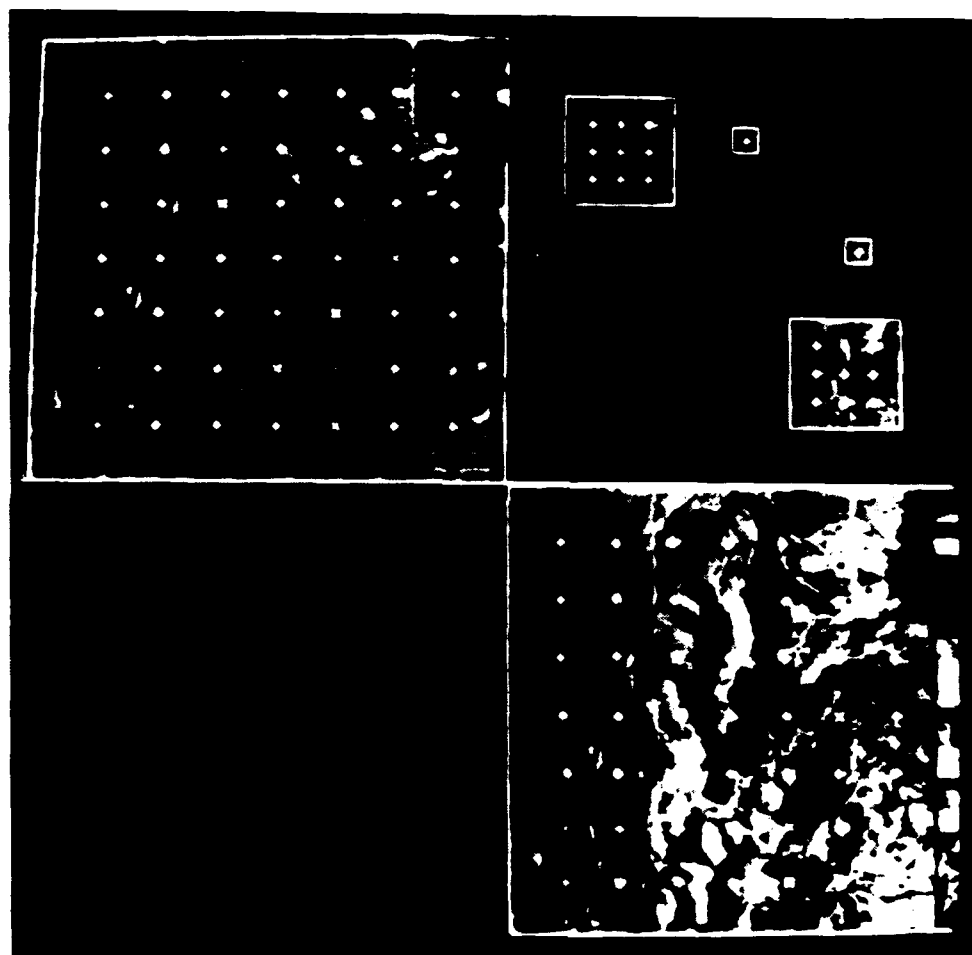


Fig. 7-4 Hierarchical Modeling Through a Reduction Hierarchy

7.1.4 Control Point Positioning. Given the positions and orientations of two cameras and the locations of corresponding point-pairs in the two image planes, the real-world locations of the viewed ground points can easily be determined. The vectors from the focal points of the cameras through the respective image plane points are simply projected into space. Since these rays rarely intersect exactly, we find their points of closest approach and average them.

If the difference is large, or the real-world point is unreasonably different from its neighbors, the point is rejected as having resulted from a bad match. Otherwise, this point joins the list of control points for future processing. The resulting positions can be expressed either in real-world coordinates (absolute position) or in camera-relative coordinates (relative position).

7.2 Error Experiments, Simulation, and Analysis

In the course of developing the Bootstrap Stereo extrapolation system, described in Section 4.2, it became obvious that we could not obtain a data set with known camera geometry and known ground landmarks. Despite this, it would still be necessary to document the buildup of error in the camera and ground point positions as the bootstrap progressed. The only solution was to program a means for simulating a flight, thus creating data on which the pieces of code could operate as they would for bootstrapping.

We began by reexamining what we needed to simulate. The major sources of error in bootstrapping come from errors in point matching between images and the manner in which these perturb the numerical analysis and projective geometry of the camera model calculations and the ground point positioning. If we could reasonably simulate the errors in the matched image plane points, we could do away with the need for gray-level imagery.

Given this simplification, we proceeded with our simulation. We first created a digital terrain model by constructing a plane grid below the general swath

of the flight path, then using a random-number generator to create elevations at each point of the grid. We then devised a flight path by flying our hypothetical aircraft along a given vector, taking its position at intervals, and introducing random perturbations in the position and orientation of the aircraft (hence the camera) at each step along the way.

For each camera position, we did the necessary projective geometry to see where each of the terrain grid points fell in the image; those which fell outside of the field-of-view of the camera were discarded. Each image point was perturbed by a random amount and/or rounded (i.e., to the nearest pixel or 1/10 pixel), to simulate a match error. Image points were tagged with the ID of the grid point which generated them, so that points in two images could be matched symbolically.

We then ran the bootstrapping programs on this data set. The programs for locating interesting points and matching them were replaced with a single program to retrieve interesting (i.e., visible) points from the image point files and match them from file to file by their ID numbers. The camera position calculation program and the ground-point positioning program were used without changes.

The general conclusions from these simulations are that the distance traveled before path error becomes unacceptable can be increased by: (1) increasing the camera field-of-view, Fig. 7-5a, (2) increasing the platform stability, Fig. 7-5b, (3) increasing the match accuracy, Fig. 7-6, and (4) using backward-looking imagery. The first three of these are fairly obvious. Increasing the field-of-view increases the distance which can be traveled between camera shutterings while still maintaining 75 percent overlap in the data. Increasing the platform stability makes it less likely that wild swings in the pointing angle of the camera will decrease the ground coverage overlap, which increases camera positioner inaccuracy. Increasing the match accuracy decreases the uncertainty about camera and ground-point positions, allowing error to build up more slowly.

ERROR AS A FUNCTION OF FIELD OF VIEW

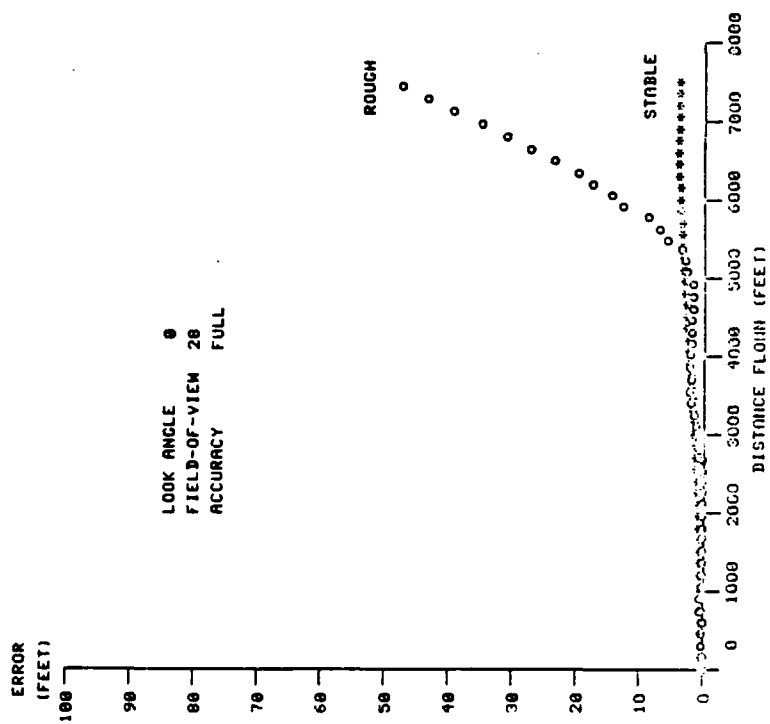
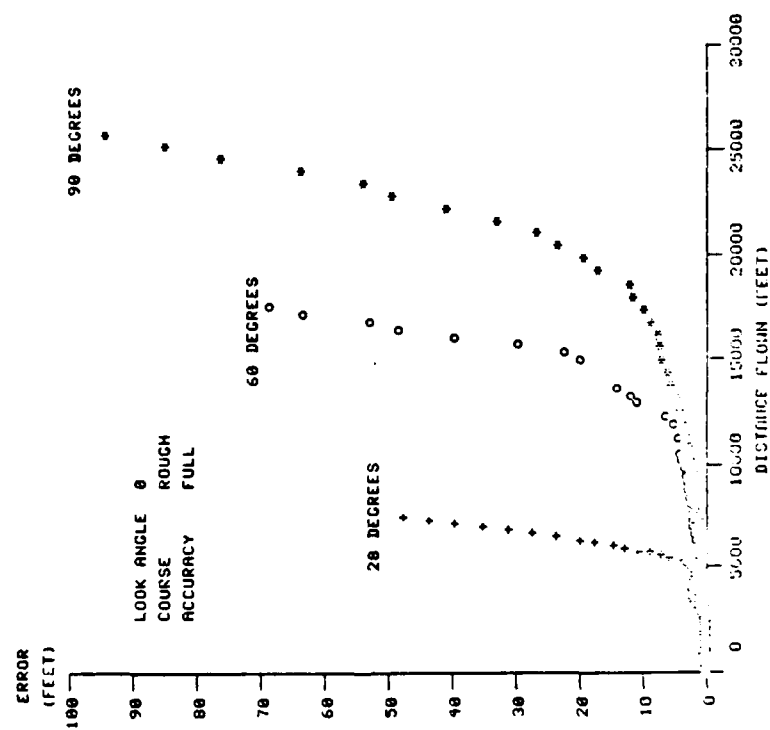


Fig. 7-5 Error Curves for Various Parameters

ERROR AS FUNCTION OF MATCH ACCURACY

ERROR AS FUNCTION OF MATCH ACCURACY

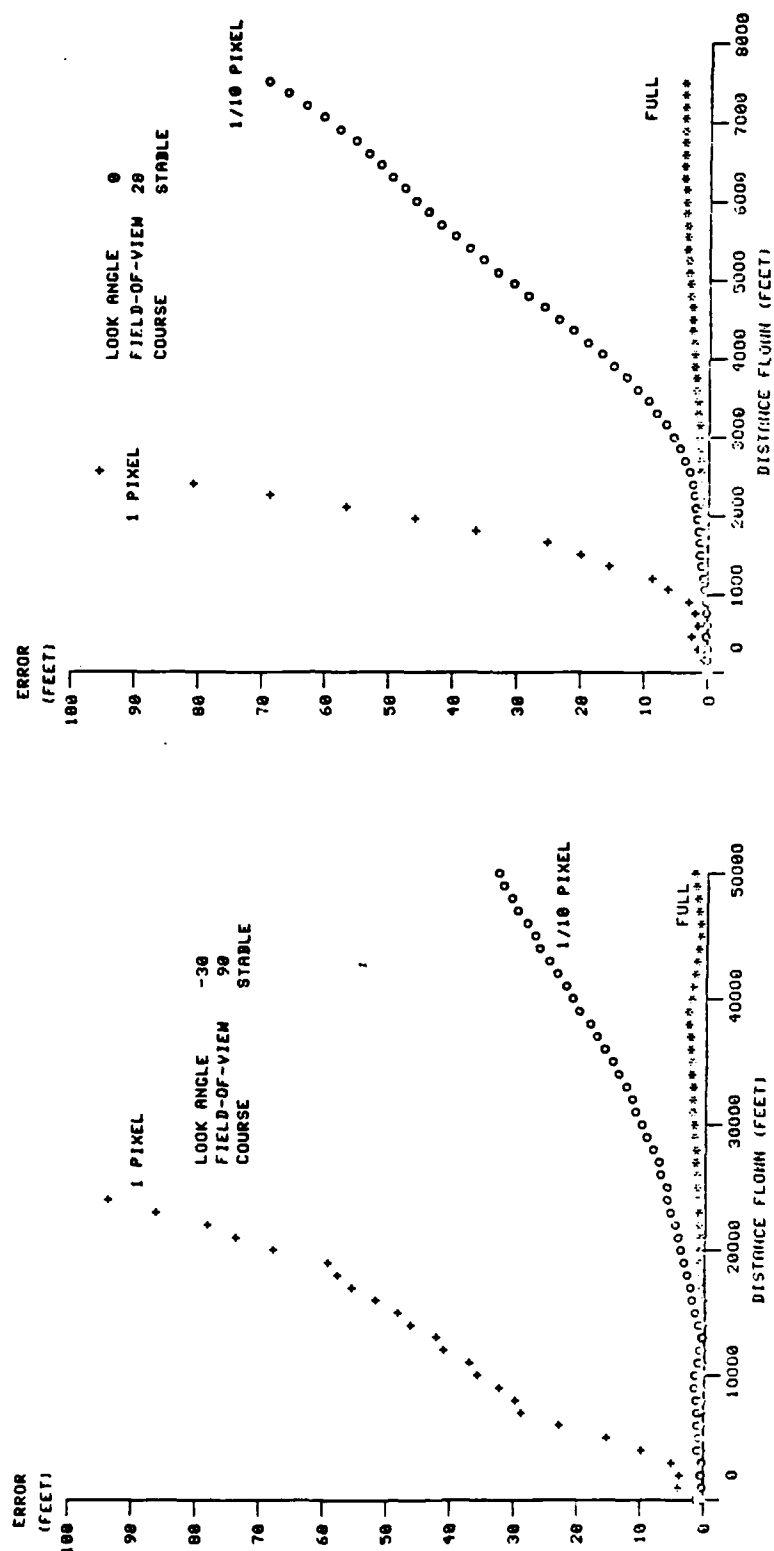


Fig. 7-6 Error as a Function of Match Accuracy

The obvious tactical question is how far can the bootstrap technique fly before its errors become unacceptable. That, of course, will depend on the flight parameters. We ran one simulation in which all parameters were favorably set - after flying 100,000 ft (almost 20 miles), the position was off by about 25 ft, and error was still accumulating slowly. We do not know how far this flight could have gone before the errors became unacceptable, as this is the longest flight we have simulated.

It should be mentioned that most of our simulations did not include any of the instrumentation on our simulated aircraft. Of course, any reasonable system which is flown will have attitude and altitude instruments whose readings can be obtained at the times the camera is shuttered. This has the effect of constraining the orientation solution even further. A simulation using postulated instrument readings showed a 5-fold increase in distance traveled before the position became inaccurate, when compared to a similar run without the instrumentation and constraints.

8. SUMMARY AND DISCUSSION

The basic AI/image analysis problems in dead reckoning and position fixing are given in Table 8-1. This section provides a brief summary of the efforts for each subsystem and discussed important aspects of each.

8.1 System Aspects

The effort in the overall system has been conceptual, with the interaction of the various subsystems defined, as given in Section 2. Of particular interest is the redundancy in vehicle location obtained from both the stereo and the dead-reckoning subsystems. Both subsystems provide a confidence measure for the location estimate, with the stereo estimate based on computation residuals and the dead reckoning estimate based on the "age" of the wind velocity estimate.

8.2 Stereo Subsystem

The stereo system described in Section 7 performs the bootstrap navigation, and also can obtain the ground velocity and the relative altitude of the vehicle. The stereo subsystem has the advantage of being independent of aircraft orientation, since its camera model solver can deal with a camera that is not aimed at the nadir point. During the course of the initial 2-year study, the automatic handoff of points from frame to frame, required in the bootstrap concept, was completed and demonstrated. Because calibrated imagery was not available, the error demonstrations were limited to simulations using synthetic data. From the simulation, we determined the best combinations of look angle and field-of-view so as to maximize the total distance that can be bootstrapped before a landmark fix is required.

The stereo bootstrap approach suffers from several operational difficulties:

1. A set of 5 to 10 known ground points in the image must be provided initially. This can be obtained by knowing the precise location and

Table 8-1 IMAGED-BASED NAVIGATION

Subsystem	Technique	Basic AI/Image-Analysis Problems
Dead Reckoning	Bootstrap Stereo	● Accurate (Subpixel) Matching ● Buildup of Errors
	Image Driftmeter	● Finding Corresponding Points Along Flight Path; Perturbations of Vehicle
	Velocity Meter	● Buildup of Integration Errors
Position Fixing	Intensity-Based Landmarks	● Extracting Invariants Shapes ● Matching Invariant Shapes
	Topographic Landmarks	● Obtaining Topographic Map Via Stereo ● Matching Topographic Features in Reference and Sensed Maps

orientation for an image, or by having known marks on the ground that the system is able to recognize, e.g., cross marks in a known pattern.

2. The matching of interesting points must be carried out to subpixel accuracy, a difficult and time-consuming task.
3. When the ground is obscured, the bootstrap procedure cannot continue. Resumption of bootstrapping requires initialization, i.e., 5 to 10 points in an initial image must be known ground points. This is very difficult to accomplish in an operational setting.

The bootstrapping process is far from infallible. The errors to which it is vulnerable fall into roughly four categories:

1. Loss of overlapped imagery
2. Errors in matching control points
3. Errors in camera calibration
4. Errors in control-point positioning

The bootstrapping process could lose its needed overlap in imagery if the terrain over which the vehicle is flying is obscured by clouds. If the terrain is essentially featureless (for example, a large body of water or a

desert), then the bootstrapper will be unable to find and match sufficient control points to continue. Similarly, several dropped frames resulting from a temporary equipment failure could cause the bootstrapping process to abort.

A human navigator faced with clouds or featureless terrain would simply shift "mental gears" and fly on instruments and dead reckoning until more favorable terrain was found. He would then attempt to locate new landmarks from which to reorient himself. Mimicking this, when stereo bootstrapping loses its overlapped imagery, the Stereo Subsystem reports failure to the Navigation Expert, which then proceeds to rely on its Dead Reckoning Subsystem until its Landmark Subsystem can recognize some new landmarks from which to reorient the Stereo Subsystem for bootstrapping. Until then, the Stereo Subsystem processes any available image sequences to extract ground velocity for the Dead Reckoning Subsystem.

The other three problems (2, 3, and 4) rarely cause the bootstrapping process to fail completely. Instead, they interact to create errors in the vehicle positions determined by bootstrapping. Since these errors are somewhat inevitable, it is anticipated that the Landmark Subsystem will be invoked periodically to search for checkpoint landmarks. Course corrections will then be determined from these checkpoints, and the bootstrapper will be reinitialized.

Gross errors in match, which can occur due to repetitive textures or moving objects, are likely to be caught by the autocorrelation thresholding or by the depth consistency or camera model consistency requirements. Small errors in match, such as would result from improper subpixel registration of the data, can slip through; these will bias the camera calibrations and control point locations slightly.

Errors in camera calibrations result either from errors in the data or from insufficient precision in the calibration calculations. The latter can be designed out of the system by ensuring that the processor has sufficient word-length and/or floating-point precision to handle matrix inversion. Errors in the data are most likely to affect the camera position, as its

orientation is fairly well known from the vehicle orientation reported by the Instrument Subsystem. Techniques exist for the adjustment of the image data points, along with the camera parameters, to produce more consistent results.

Errors in the positions of the original landmarks will be propagated through the bootstrapping chain. Such errors should be static, however, and should only result in small perturbation in the vehicle location. Errors in control point positioning are interrelated with errors in the match point location and the camera calibration. Using the redundancy inherent in multiple images can resolve some of these uncertainties.

8.3 Landmark Subsystem

A landmark identification is required:

- Periodically, say every 5 to 15 min.
- When the bootstrapper starts to diverge, say every 5 to 10 miles. The stereo can tell the landmark system that a specific landmark will be in a specific frame.
- When the bootstrapper has lost a sequence of frames due to clouds. Position must be extrapolated using the wind and the airspeed, and the landmark subsystem must look in many frames for the landmark.

8.3.1 The Symchain Approach to Landmarks. The role and interactions of the Landmark Subsystem were identified in the original 2-year study using an intensity-based landmark system developed under the Lockheed Independent Research Program. This approach identifies edges of objects and finds sets of connected edge points, called "symchains." The symchains are represented symbolically in terms of line segments and the angles between them, and this description is then compared with reference symchain descriptions in the landmark database. This edge-based approach, which was developed to deal with large perspective distortions:

- Is less sensitive to vehicle position/attitude errors than correlation-type approaches
- Reduces the amount of storage necessary to match against a reference

- Has the ability to obtain a match even when the reference is artificially obtained (as from a topographic map)

Characteristics which separate the approach from some other structural approaches are:

- Extended "strands" are used in the match strategy, rather than linear edges being matched. These strands thus embody shape information which can be very useful in match environments.
- Perspective effects can be compensated for, as the global match is performed in a later stage than the local matching. A camera transformation can thus be applied as a result of the matches in the different areas of the images.
- The edge elements combined to form the strands are created using the gray level information in the image and must also be symmetrically associated with one another. (Each edge element in the strand considers its right- and left-hand edge associate as the best ones.) This provides robust determination of high gradient areas initially simplifying the identification of edge structures.

This approach to landmark navigation suffers from several shortcomings:

- No consideration is given to the inherent shape constraints of well-chosen landmarks, i.e., the parallelism of river-banks and road-edges, the perpendicularity of crossroads, the anomalous height of water towers, transmission lines, etc.
- Since the modeled features are essentially linear, it is only in the global matching of several features that one accounts for the inherent spatial relationships.
- The global matching procedure which attempts to combine matched features does not use the available camera model information.

As indicated previously, position fixing, and not dead reckoning, lies at the heart of the image-based navigation problem. Landmark analysis then becomes crucial. Although much research has been carried out in image understanding to detect landmarks, the only robust landmarks appear to be roads and region

transitions. We therefore focussed our efforts on a landmark approach based on these, as described in Section 6. The approach takes advantage of the image frame sequences by using transitions found in one frame to verify those found in previous frames.

8.3.2 Operational Aspects of the Landmark Database. Before a vehicle using passive navigation can make an operational flight, it is necessary to load the landmark database with a symbolic description of landmarks that the vehicle will fly over. For this we require photographic imagery taken at some previous time from which landmarks on a 10-mile swath (representing the flight path) are chosen. It would be desirable to find suitable landmarks for every 5 miles along the swath so that the system could deal with the situation of obscured ground at any point in the flight path. A ground-based landmark analyzer processes the photographic data and converts the processed landmarks to symbolic form for loading into the vehicle memory. For each landmark, the processor indicates the best processing technique for the vehicle to use.

The Landmark Subsystem as conceptualized in the initial Passive Navigation study could not be like a barnstorming pilot who looks out the window to check on landmarks. Because of the low altitude (say 1500 ft), the narrow field-of-view, and the look angles of 0 to 30 deg from the nadir, the system is more like a pilot looking down through a hole in the floor of the aircraft. Note that looking at a swath 600 ft to 2400 ft in width has an important operational implication: We cannot set up a flight path and then look for landmarks adjacent to the flight path. Rather, we must set up the flight path so as to navigate from landmark to landmark.

8.4 Image-Based Velocity Meter

Simulations of the ground velocity meter described in Section 4.3 showed that a sufficient degree of directional sensitivity and overall accuracy can be attained with this method. The basic limitation is the need for relative altitude in order to obtain the true magnitude of the ground velocity.

8.5 General Discussion

The new approach to landmark navigation described in Section 6 is the most promising technique developed:

- It is ideal for a low-flying vehicle.
- The required transitions are readily determined from simple image processing.
- The mechanization is straightforward.
- It makes possible missions requiring complicated flight paths.

We recommend that this approach be extended to handle the problem of extraneous transitions. A continuation of the contract at a funding level of 1/2 man-year would cover the costs of investigating and implementing the backtrack procedure for extraneous crossings.

9. REFERENCES

- 4-1 M. Oron and M. Abraham, "Analysis, Design, and Simulation of Line Scan Aerial Surveillance Systems, Proc. Soc. Photo. Instr. Eng. (SPIE) 219, 77 (1980).
- 4-2 M. Oron and O. Firschein, "Airborne Ground Velocity Determination by Digital Processing of Electro-Optical Sensor Signals," Optical Engineering, March/April 1982, Vol. 21, No. 2.
- 5-1 O. Firschein, M. J. Hannah, D. L. Milgram, and C. M. Bjorklund, "Passive Imagery Navigation: Final Report," LMSC-D767313, Lockheed Palo Alto Research Laboratories, Palo Alto, CA, Dec. 1980.
- 7-1 D. G. Gennery, "A Stereo Vision System for an Autonomous Vehicle," Proc. of the 5th Int. Joint Conf. on Artificial Intelligence, Cambridge, MA, 1977.

Appendix A
FLIGHT SIMULATION LOG

Input velocity-start file name?
dblvs. dat

Input line segment data base file name?
dblls.dat

Input crossing times-direction file name?
dblts.dat

window size?
50.

please view VICOM monitor ...
(type any character to continue simulation

c

time, direction, and type of observation 1 = 0.81, 0.481, RIVER

Activated line segments at time 1 are as follows:

1
2
3
4
5

First, the active segments are inserted into the path list.

5	0.568	LANDWATER
4	0.508	WATERLAND
3	0.863	RIVER
2	0.885	ROAD
1	1.000	RIVER

Second, we delete those paths with insufficient probability; less than 0.8.

3	0.863	RIVER
2	0.885	ROAD
1	1.000	RIVER

Third, we delete those paths with incorrect crossing type.

3	0.863	RIVER
1	1.000	RIVER

time, direction, and type of observation 2 = 5.57, 0.972, WATERLAND

Activated line segments at time 2 are as follows:

1
2
3
4
5
6
7
8

First, the active segments are inserted in the path list.

3	8	0.795	RIVER
	7	0.490	ROAD
	6	0.584	WATERLAND
	5	0.812	LANDWATER
	4	0.863	WATERLAND
	2	0.339	ROAD
	1	0.439	RIVER
1	8	0.921	RIVER
	7	0.568	ROAD
	6	0.677	WATERLAND
	5	0.941	LANDWATER
	4	1.000	WATERLAND
	3	0.646	RIVER
	2	0.393	ROAD

Second, we delete those paths with insufficient probability; less than 0.8.

3	5	0.812	LANDWATER
	4	0.863	WATERLAND
1	8	0.921	RIVER
	5	0.941	LANDWATER
	4	1.000	WATERLAND

Third, we delete those paths with incorrect crossing type.

3	4	0.863	WATERLAND
1	4	1.000	WATERLAND

Fourth, we delete those paths which cannot be connected due to velocity considerations.

checking path (1 4) for vector connectivity
please view VICOM monitor...
(type any character to continue simulation

c

checking path (3 4) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

1 4 1.000 WATERLAND

time, direction, and type of observation 3 = 8.55, 0.540, ROAD

Activated line segments at time 3 are as follows:

3
 5
 6
 7
 8
 9
 10
 11

First, the active segments are inserted into the path list.

1	4	11	0.745	RIVER
		10	0.869	ROAD
		9	0.060	RIVER
		8	0.647	RIVER
		7	1.000	ROAD
		6	0.245	WATERLAND
		5	0.627	LANDWATER
		3	0.922	RIVER

Second, we delete those paths with insufficient probability; less than 0.8.

1	4	10	0.869	ROAD
		7	1.000	ROAD
		3	0.922	RIVER

Third, we delete those paths with incorrect crossing type.

1	4	10	0.869	ROAD
		7	1.000	ROAD

Fourth, we delete those paths which cannot be connected due to velocity considerations.

checking path (1 4 10) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

checking path (1 4 7) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

1 4 7 1.000 ROAD

time, direction, and type of observation 4 = 10.34, 0.894, RIVER

Activated line segments at time 4 are as follows:

5
 6
 7
 8
 9
 10
 11

First, the active segments are inserted into the path list.

1	4	7	11	0.901	RIVER
			10	0.516	ROAD
			9	0.413	RIVER
			8	1.000	RIVER
			6	0.599	WATERLAND
			5	0.981	LANDWATER

Second, we delete those paths with insufficient probability; less than 0.8.

1	4	7	11	0.901	RIVER
			8	1.000	RIVER
			5	0.981	LANDWATER

Third, we delete those paths with incorrect crossing type.

1	4	7	11	0.901	RIVER
			8	1.000	RIVER

Fourth, we delete those paths which cannot be connected due to velocity considerations.

checking path (1 4 7 11) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

checking path (1 4 7 8) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

1 4 7 8 1.000 RIVER

time, direction, and type of observation 5 = 14.00, 0.795, RIVER

Activated line segments at time 5 are as follows:

7
 8
 9
 10
 11

First, the active segments are inserted into the path list.

1	4	7	8	11	1.000	RIVER
				10	0.614	ROAD
				9	0.314	RIVER

Second, we delete those paths with insufficient probability; less than 0.8.

1 4 7 8 11 1.000 RIVER

Third, we delete those paths with incorrect crossing type.

1 4 7 8 11 1.000 RIVER

Fourth, we delete those paths which cannot be connected due to velocity considerations.

checking path (1 4 7 8 11) for vector connectivity
 please view VICOM monitor ...
 (type any character to continue simulation)

c

1 4 7 8 11 1.000 RIVER

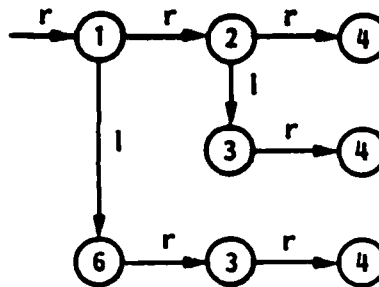
Final path 1

id	x1	y1	x2	y2
1	20.000	20.000	150.000	90.000
4	82.750	164.929	68.555	168.984
7	98.599	253.638	112.446	259.496
8	127.336	309.479	119.400	310.719
11	150.488	420.000	169.515	420.000

Appendix B

LOL DATA STRUCTURE

The list-of-lists (LOL) data structure is a linked list of linked lists. That is, every element of the list is itself a linked list. We define a canonical form which distinguishes a right list from a left list as follows: Each node in a left list has an ancestor which is the head node of the right list that seeded the left list. Essentially, what this means is that the left list starts a new path (or subpath), and the right list continues a path. Consider the diagram below (l = left, r = right);



the paths are (1 2 4), (1 3 4), and (6 3 4). Note that (3 4) is a subpath of (1 3 4) because 1 is in the head node of a list which gave rise to (3 4).

Storage for the left and right lists can be allocated the same way. Thus, a suitable definition of the LOL type in PASCAL is as follows:

```

lol = ^Node;
node = RECORD
    id, level : integer;
    x1, y1, x2, y2, prob : real;
    tipe : crossing_type;
    left, right : lol
END;
```

It may appear that the LOL data structure type definition is similar to that of a binary tree. However, the definition of the list implies that traversals, updates, and deletions must be carried out by indirect recursion. Direct recursion is usually sufficient for binary tree manipulation.

END

12-86

DTIC