FEATURE-BASED ROAD AND TRAIL DETECTION

By

CHRISTY L. CHISM

Bachelor of Science in Electrical Engineering

Oklahoma State University

1983

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1990

FEATURE-BASED ROAD AND

TRAIL DETECTION

Thesis Approved:

C. M. Bauer

Thesis Advisor

Keith A. Teague

Richard L. Cummins

Norma N. Durham

Dean of the Graduate College

PREFACE

The road and trail detection algorithms are part of ongoing work by Texas Instruments, Inc. in the area of intelligent weapon systems. The author is employed by the Defense Systems Electronics Group of Texas Instruments in the Multi-Sensor Autonomous Target Recognition Branch of the Image Processing Laboratory. The Image Processing Laboratory is currently addressing a variety of programs with the need for road and trail detection.

One of these programs is a demonstration of the technologies involved in using an autonomous air vehicle to locate camouflaged targets. This program successfully integrated the trail detection algorithm.

I wish to thank Texas Instruments for their cooperation and encouragement in completing my graduate studies, and especially my co-workers in the Image Processing Laboratory who contributed their ideas and time to this effort. I also wish to thank Dr. Charles Bacon, not only for his guidance, but for all the little things he did in helping me to complete my long distance graduate work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The identification of battlefield vehicles in real time is a difficult problem. An automatic target recognition (ATR) system must search large areas for relatively small targets. A target vehicle's appearance may be affected by sensor degradation, atmospheric effects, occlusion, and varying aspect angles. Furthermore, the inclusion of natural camouflage and camouflage nets has become commonplace in ATR scenarios. A knowledge base containing all relevant models could easily become unmanageable. Such diverse considerations place great demands on an ATR system.

Scene context information improves image understanding and helps to reduce processing requirements. In particular, road and trail detection can be useful. Many vehicles require roads to traverse difficult terrain. Others may leave tracks as an indication of their presence. Scene context processing as a top-down approach provides a means of reducing the target search area by focusing on and around roads. Scene context clues could also be used in a bottom-up fashion to increase the confidence of a nearby target candidate.

A general approach to road or trail detection consists of two stages, each of which may utilize feature knowledge. Segmentation is the first. If the road or trail boundaries are not correctly delineated, it is much more difficult to recognize their features. Knowledge of road and trail features can guide the segmentation process. The second stage is for the algorithm to identify the roads and trails by their features. Once an algorithm has obtained the optimal

segmentation of the roads or trails, features can be used to evaluate each scene object for road or trail-likeness.

This second stage process of identification is more complex for trails than roads. In this discussion, "road" refers to a vehicle path which appears in map data, while "trail" will refer to one that does not. In trail detection, the identification process must be substantially more powerful than in road detection. The area to be processed when searching for trails is very large, while road locations are restricted to within the error bounds of the map. Also, the availability of map data will allow road detection to restrict its consideration to a subset of all possible features. For example, the map can guide the algorithm to a section of the road that is known to be straight. In road detection, the features select the most road-like object, while in trail detection the feature evidence must be substantial enough to determine if a trail exists at all.

It is clear that when map data is available, the problem of detecting roads is greatly simplified. The availability of map data reduces the computational burden since the processing would not involve the entire image, but only the most likely areas. Being able to focus the search also helps to control false alarms. The digital map provides information about the location and size of many of the road features. Unfortunately, trails, tracks, and many secondary roads do not appear in map data.

In the area of road detection, much of what has already been accomplished applies to imagery viewed from an autonomous land vehicle. New algorithms are needed for aerial imagery at various ranges and resolutions.

Existing algorithms for trail detection were not discovered. Therefore, this research combines the general ideas of several knowledge-based

computer vision researchers and applies them specifically to trail detection. Many details of their approaches have been changed to make them applicable to real time systems. The majority of previous work lies in the area of applying knowledge to segmentation. This work goes beyond segmentation to address the recognition problem.

This research addresses the problems of finding both roads and trails for application in automatic target recognition scenarios. Chapter 2 provides a summary of existing road detection algorithms. It also discusses research in knowledge-based segmentation which contributed to this work. Chapters 3 and 4 describe the details of the road and trail detection algorithms, respectively. Considerable time was devoted to evaluting the trail detection performance and those results are presented in Chapter 5. Conclusions and suggestions for continued work are given in Chapter 6.

CHAPTER II

SURVEY OF ROAD AND TRAIL DETECTION TECHNIQUES

Road Detection Algorithms

Previous road finding algorithms have investigated related scenarios. McKeown and Denlinger (1988) have made significant progress on road tracking by combining two independent approaches. Their algorithm currently requires inputs of road starting location, width, and direction. Kuan, Phipps, and Hsueh (1988) have developed a road following algorithm for an autonomous land vehicle. The Duda road operator used by SRI (Fischler, 1981) applies only to low resolution imagery and is sensitive to road orientation.

For use in automatic target recognition, the road detection algorithm must be insensitive to orientation, and applicable to aerial imagery at a variety of ranges and resolutions. This eliminates the direct application of the Duda road operator and autonomous land vehicle approaches. This new approach to road detection differs from all of the above approaces by emphasizing the real time processing requirements. Processing time is reduced by focusing the search on straight road segments only.

Knowledge-Based Segmentation

Some researchers have investigated the application of knowledge to improve the segmentation process without addressing the recognition problem. The purpose of the generic geometric models proposed by Fua and Hanson (1987) is to provide an intermediate level of image representation. This intermediate level contains only semantically meaningful combinations of low-level features for input to a high-level reasoning system. There are three key elements to this approach. First, the generic model describes characteristics that belong to a wide class of objects. These characteristics include parallelism, collinearity, and surface uniformity. Second, the model utilizes both edge and area features. More than one object feature may contain information relevant to its identification. Therefore, it is important to have a technique for integrating features. A third aspect is that the model must suggest a means of predicting and verifying missing components. These components may be missing due to thresholds or occlusions. Fua and Hanson show examples of using these models to detect buildings, roads, and trees. The output of their algorithm identifies regions that exhibit the characteristics desired by the models.

The use of multiple features is also a primary theme in the work of Reynolds, Irwin, Hanson, and Riseman (1984). They describe a system in which feature extraction begins at a very course resolution. Next, the algorithm evaluates these features using models appropriate for that level of resolution. This process repeats at a finer level of resolution, but only for those areas with a high confidence of containing a goal object. Since the search area is reduced, the algorithm can apply more expensive evaluation procedures. The evaluation process employs the integration of line and region

features. This type of integration is similar to that suggested by Fua and Hanson. However, Reynolds is more specific about his approach. He makes use of the common pixel based representation of both regions and lines. For example, a set of lines can be superimposed on the boundary of a region. Thus, these lines form a group. A line could also group regions. Reynolds applies this approach to the detection of buildings. Rectangularity characterizes buildings. To detect this property, the orientation of lines are histogrammed. Two groups of lines 90 degrees apart indicate potential buildings. As described above, lines are combined with regions, which provide intensity and size information.

Fua and Reynolds each use knowledge about the goal object to improve segmentation. More general knowledge about vision is stated in the Gestalt laws. Nazif and Levine (1984) use rules to apply Gestalt laws. These laws state that proximity, continuity, closure, and other related factors play a role in vision. For example, regions that are in close proximity are more likely to have a relationship than those that are far apart. Nazif and Levine form a rule that combines proximity with statistical information to merge similar adjacent regions. They also use rules to integrate multiple features. Another of their rules inhibits the merging of two regions due to the presence of a line at the common boundary. The importance of the rule-based approach is that knowledge is applied only when appropriate conditions exist.

Application to Trail Detection

This new algorithm utilizes Fua's three basic premises: generic models, multiple features, and predicting missing components. However, only the generic model relating to roads, parallelism, was fully developed. The

technique for combining multiple features came from Reynolds (1984). Finally, a rule-based approach using Gestalt laws described by Nazif (1984) was used to fill in missing components. This research goes beyond Fua's intermediate level of representation and completes the recognition process for the detection of trails.

CHAPTER III

ROAD DETECTION ALGORITHM

Introduction

The first step in road detection is to prepare a subimage based on the information from a digital map. Next, the subimage is processed to enhance the road's characteristics. The two most important concepts behind this approach to road detection are the modified median filter and the Hough transform. The modified median filter is applicable because of its 1-dimensional size filtering capability. In the case of road detection, the width may be known within some degree of certainty while the length may be too great to be of practical use. In fact, when detecting straight road segments, it will be assumed that the longer the segment the more likely it is to be a road. The Hough transform will be used to provide information about large groups of pixels that fall in a straight line.

Digital Map

The digital map provides longitude and latitude of end points of straight line segments that define the median strip of roads. Even though the map provides information about the location and size of the features, there are inaccuracies in the data. In addition, there are also errors in the position of the aircraft relative to earth coordinates and errors in the gimbal angles that define the sensor line-of-sight. All of these errors result in the map features

being miss-registered with the image. This miss-registration must be accounted for in the search for the features. However, we can transform the map features from object space to image space and define an uncertainty region in the neighborhood of the projected points in which to conduct the search.

Figure 1 shows how the map errors combine to determine the size of the uncertainty region. The length of the required subimage is based on the road's predicted length, the tolerance on the predicted length (tol_length), and the tolerance on the location (tol_loc). Since the worst case must be accounted for, the length of the uncertainty subimage is the sum of these 3 values.

The subimage width is more difficult to calculate because of the worst case effect of orientation errors. The tolerance in orientation (tol_ornt) must be accounted for in both directions of rotation. Therefore, the amount that orientation errors contribute to the width of the subimage is length*sin(tol_ornt). The width of the road also contributes to the subimage width. Again accounting for both directions, (width+tol_width)/cos(tol_ornt) is added to the subimage width. Since the modified median filter requires a substantial number of neighbor pixels for its calculations, an additional border is added to the subimage. The filter requires half its size on each side. Therefore, the vertical filter width is added to the subimage width, and the horizontal filter width is added to the subimage length.

The subimage is extracted from the original image as indicated by Figure 2. As the pixels are selected from the original image, they are rotated by the predicted orientation. The centerpoint of the subimage is at the predicted location of the road.

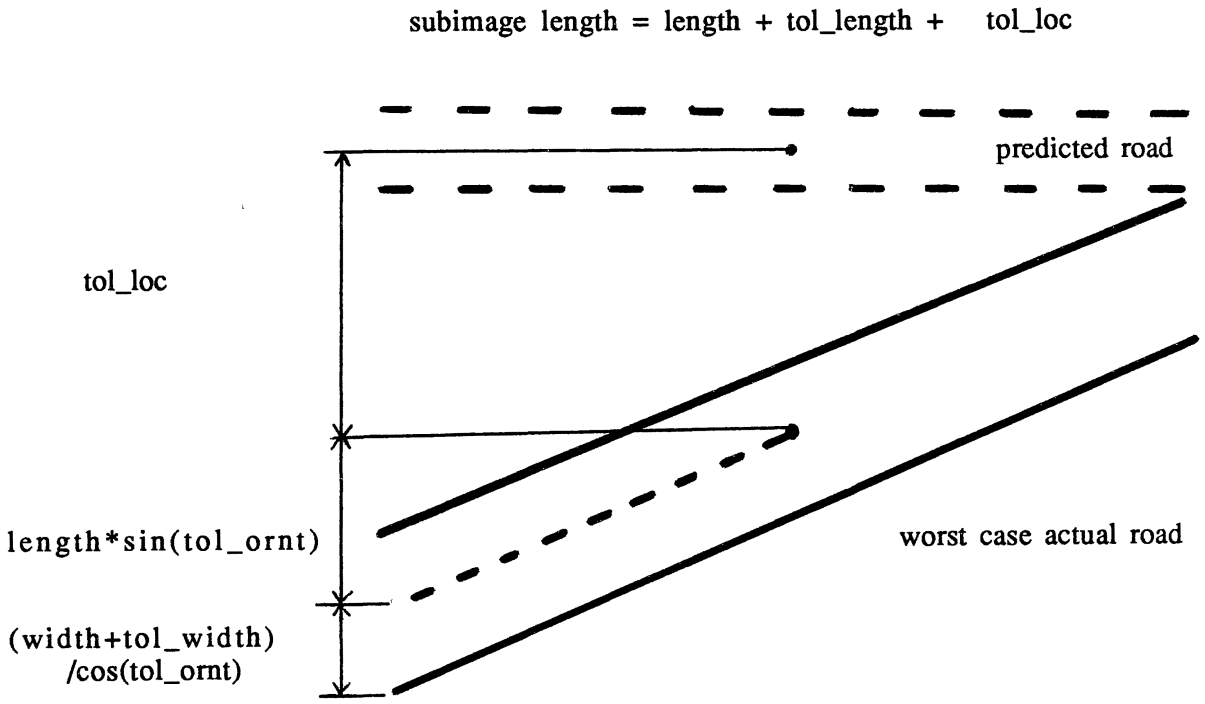


Figure 1. Calculating Subimage Size

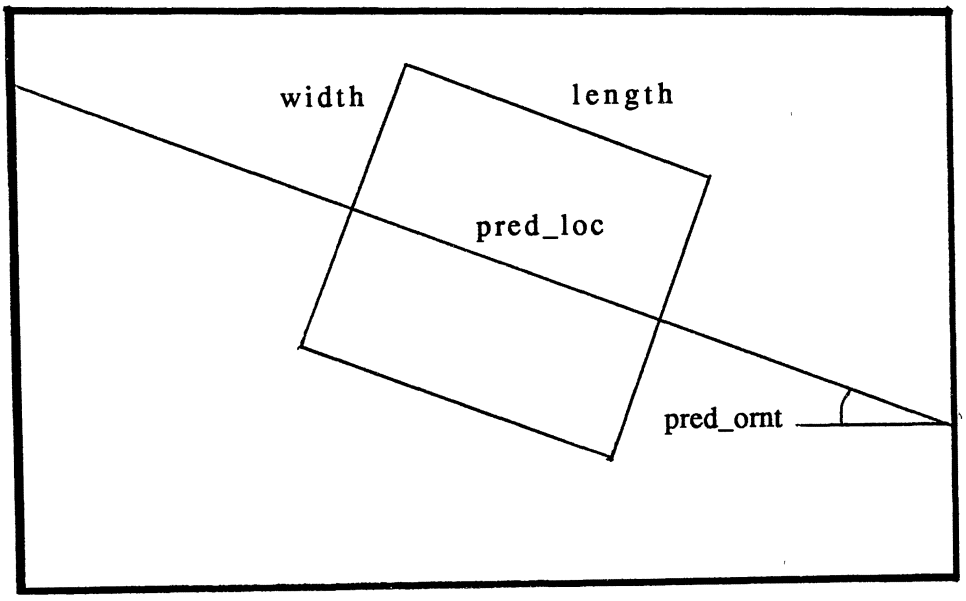


Figure 2. Subimage Location

Modified Median Filter

Modified median filtering is a background suppression technique which eliminates regions of improper size. It is actually a variation of the median filter, which finds the median value of the pixels that fall within a specified window as that window slides across the entire input image. The modified median filter can be thought of as an alternative to the contrast box screener. Rather than using only pixel values from the input image, the modified median filter uses the results of previous calculations to determine its next value. These differences are illustrated for a window of size three in Figure 3. The result of using a 1-dimensional filter of size 2N+1 is that everything smaller than N is eliminated from the scene.

The modified median filter can also be used to eliminate regions that are larger than a given size. The output of using a filter of size 2N+1 can be subtracted from the original input image to retain only objects that are smaller than N. The processes for rejecting small and large clutter can be combined as in Figure 4 such that only objects of a specified size are left. By applying the absolute value to the results of the subtraction, both targets of hot and cold contrast can be detected. The entire process is shown step by step in Figure 5.

Hough Transform

The Hough transform (Gonzalez, 1987) is a technique for mapping shapes in images into an associated parameter space. In its generalized form, which is difficult to implement and quite time consuming, the Hough transform is capable of detecting any shape whose characteristics are known. The Hough

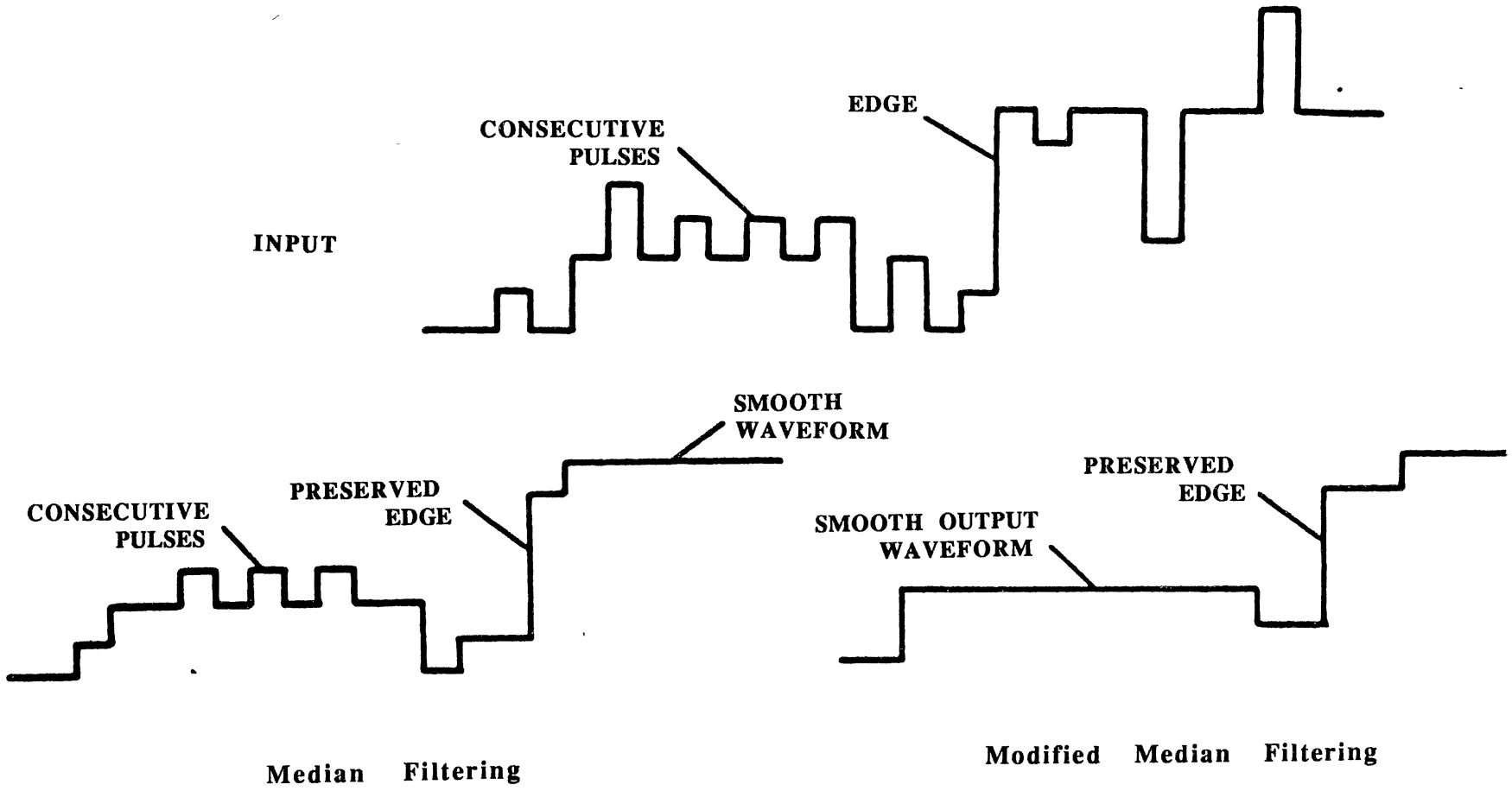


Figure 3. Modified vs. Median Filtering

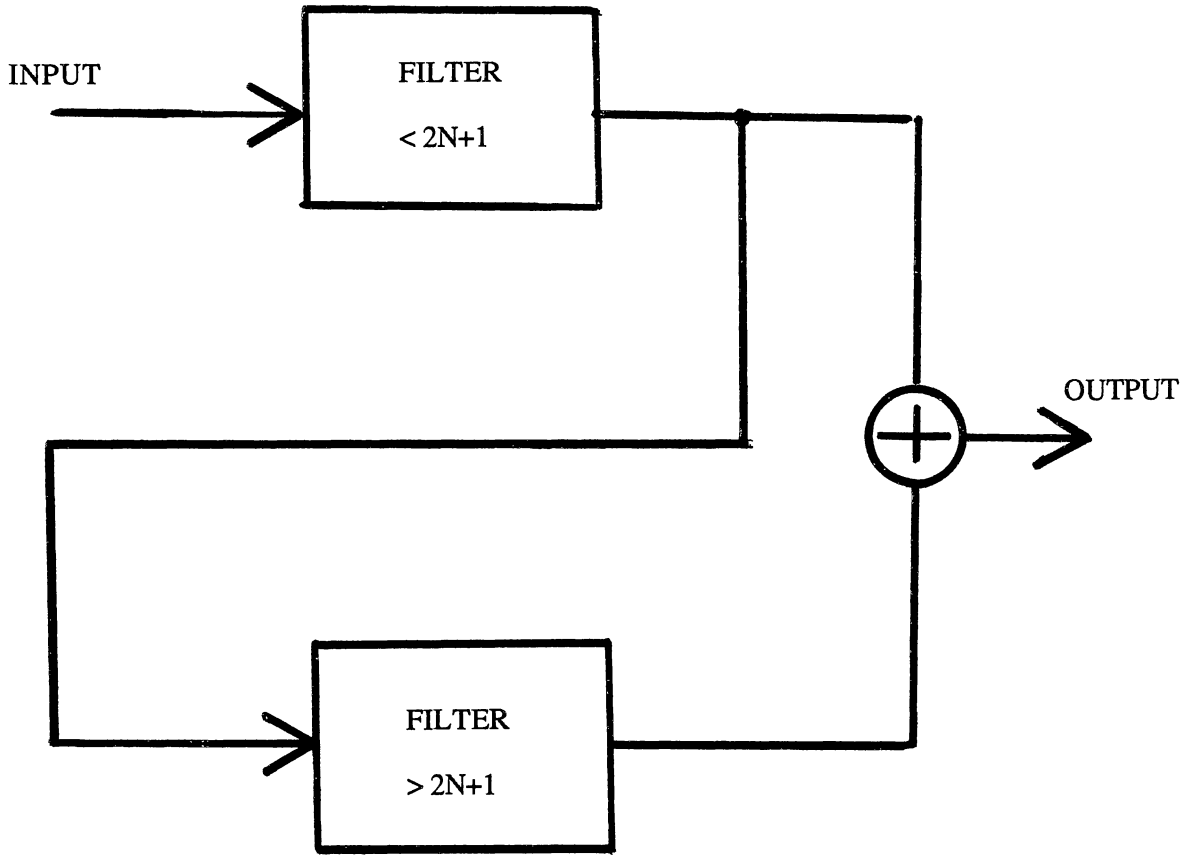


Figure 4. Block Diagram of Size Filtering

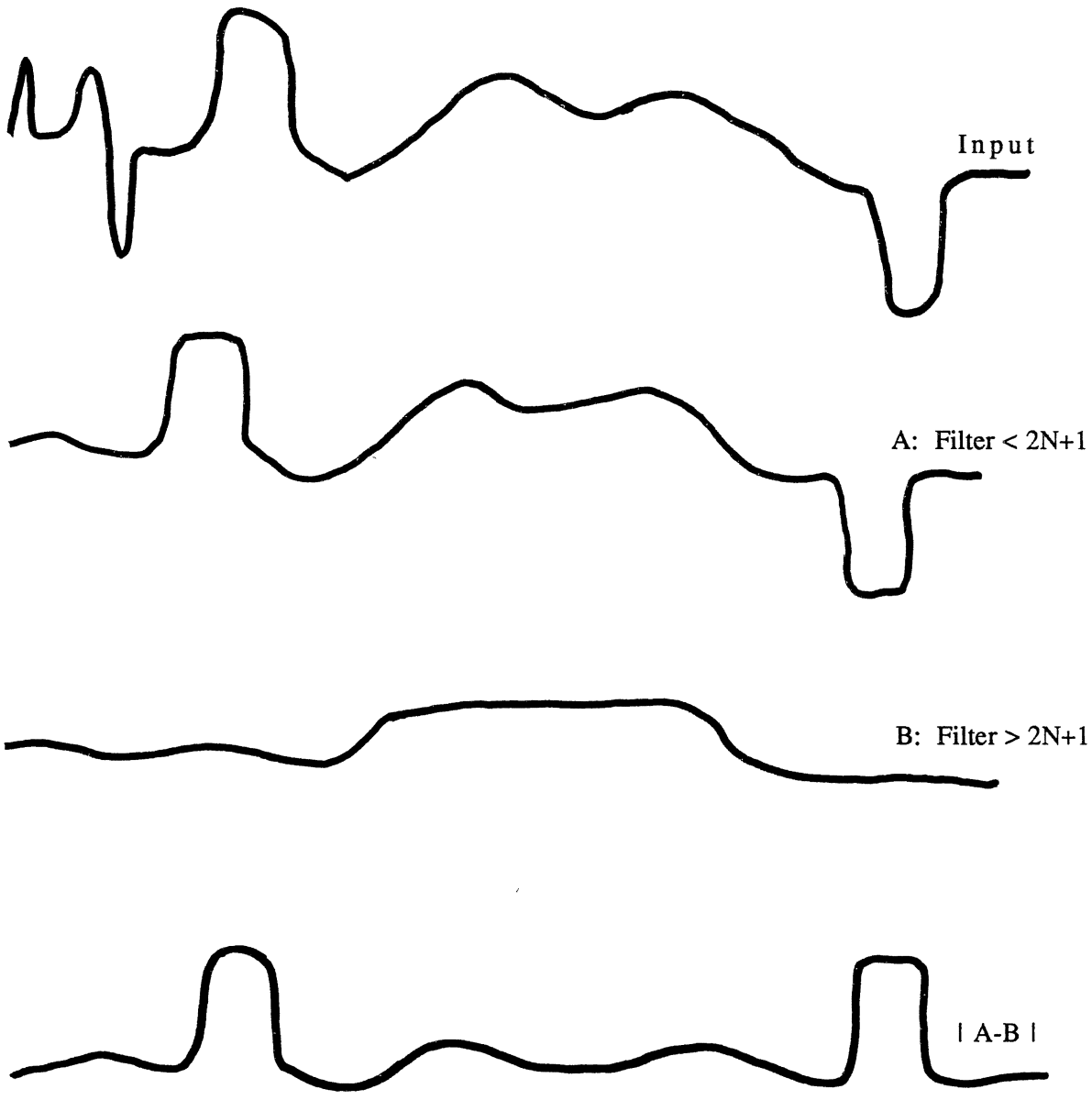


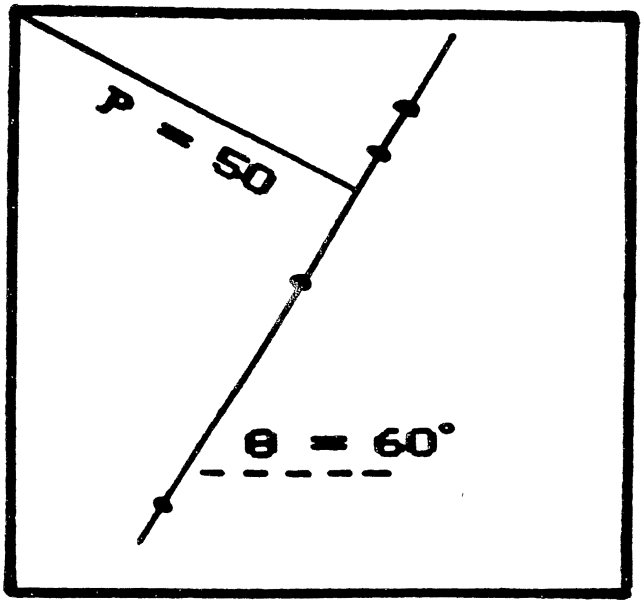
Figure 5. Multiple Filter Results

transform can be used to extract lines by replacing the problem of detecting collinear sets of points in an image with the problem of finding intersecting lines in the parameter space.

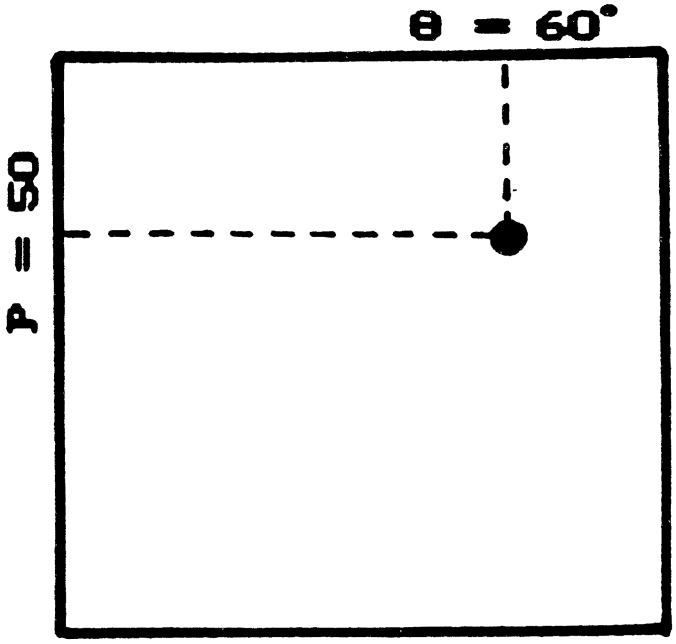
The slope-intercept form , $y = mx + b$, where m is the slope and b is the intercept of the line with the y-axis, can be used to describe a line in image space. All points (x,y) on a line will have the same values for m and b. A point can be transformed to parameter space using the equation $b = -xm + y$. Each point in image space produces a single line in parameter space. The intersection of lines in parameter space indicate values of m and b for collinear sets of points.

The problem with the above approach is that the parameter space for m,b is unbounded. This makes the implementation impractical. This is resolved using the normalized parameters ρ and θ, where θ is the angle of a line relative to a reference axis, and ρ is the distance from the origin normal to that line. The equation of a line becomes $\rho = y \cos(\theta) + x \sin(\theta)$. The value of θ is now restricted to 180 degrees, and ρ to twice the length of the diagonal of the image. The slope-intercept form gives a straight line in parameter space, while the normalized parameters result in a sinusoidal curve.

The computational implementation of the Hough transform space is an array whose dimensions are the quantized levels of ρ by the quantized levels of θ. The transform is applied to each point of interest in an image. The procedure involves varying θ through its quantized range, calculating a value for ρ with each θ, and then assigning the value for ρ to its nearest quantized level. The array element which corresponds to this ρ and θ is incremented. Peaks in this transform space indicate line equations that encompass the maximum number of points. A row of points in image space and their corresponding mutual location is Hough space is shown in Figure 6.



Road Segment



Hough Transform Space

Figure 6. Hough Transform

Road Detection Implementation

The digital map can provide an estimate of the road orientation and size. The greatest possible error from the map data must be incorporated in determining the median filter size. In order to get the maximum effectiveness from the filter in the least amount of time, the filter will be run once in the horizontal direction and once in the vertical direction. In the direction parallel to the length of the road, everything shorter than the expected size of the road will be eliminated. This will include a significant amount of clutter since the road is expected to be long. Everything larger than the width of the road will be eliminated in the direction perpendicular to the road. The filter could be run again in each direction to eliminate clutter thinner than the road in the direction perpendicular to the road, and anything known to be longer than the road in the direction parallel to the road. However, this is avoided because of the increase in computation time and the minimal improvement in the performance.

After the image is filtered, it is thresholded to reduce the total number of pixels, leaving only those of relatively high contrast. This threshold value is determined from the average contrast of the filtered image multiplied by some constant. The constant being used is 2. The resulting output is a binary image.

Another step to reduce the number of pixels, as well as improve the accuracy of the Hough transform, is to thin the regions remaining in the image. This is done by checking each non-zero pixel in the image, and if the one above it is 0 and the one below it is non-zero, then change that pixel to zero. If this process goes from top to bottom, the result is that the region is thinned to one pixel along the bottom edge of the region. This process can also be executed in either the horizontal or vertical direction, depending on the

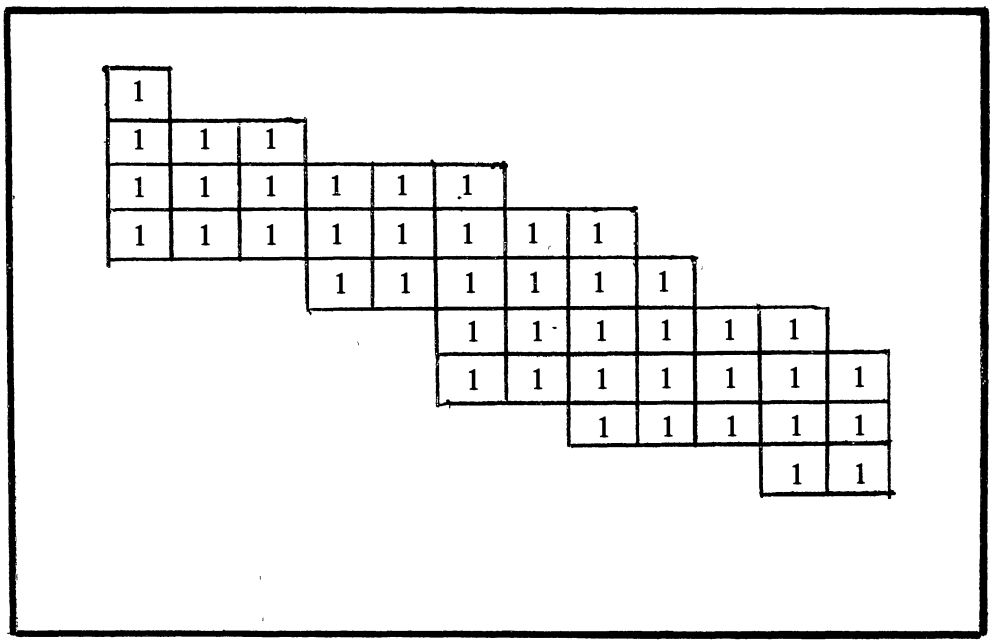
expected orientation of the road. If a pixel meets the above conditions and is changed to zero, its contents are added to the one below it. The remaining pixel will represent the width of the region at that point. Figure 7 shows how a region is thinned down to one pixel along the lower boundary representing the width of the road.

The next step is to create the Hough transform space as in Figure 6. In the transform space, θ is quantized by whole degrees, and the distance is quantized by whole pixels. Since the expected road orientation is known, only θ values that are within a certain tolerance of that orientation need be included.

The expected width of the road is also known. The ρ and θ location in the transform space will be incremented to reflect how close the width at that pixel is to the expected width. From the thinning algorithm, a pixel location contains the width of the region at that point. The value of the increment to the transform space should be at a maximum when the pixel value is equal to the expected width. The amount that the current pixel value differs from the expected width is subtracted from the expected width and added to the appropriate transform space location.

Once the Hough transform is completed, appropriate peaks must be selected from the transform space. The top peaks (currently 3 of them) must be local maxima and separated by a ρ distance greater than the expected width of the road. The ρ and θ values of these top peaks are used to determine exactly which pixels are part of that candidate road. A pixel is considered to be part of the road if its ρ value at the peak θ is within some tolerance of the peak ρ.

The lowest and highest row and column values for each pixel within the tolerance are kept for determining the length of the road. The difference in



Binary Image

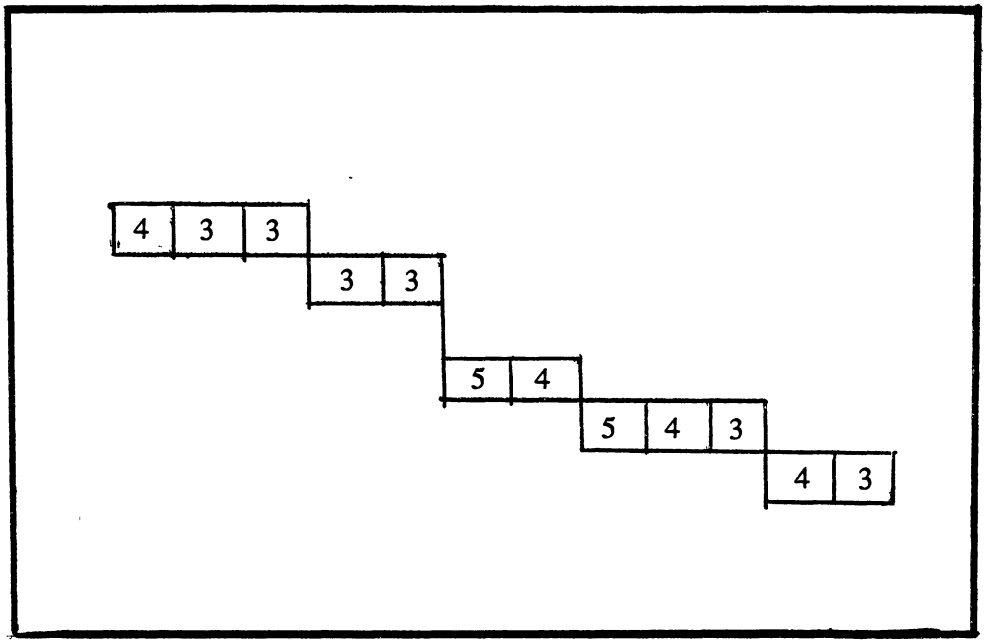


Figure 7. Thinning

the 2 row values is squared, the difference in the 2 column values is squared, and the square root of the sum of the squares is taken. This length is necessary for determining the road-likeness of a set of points.

Certainty Measurements

Two measures of certainty are used to evaluate the remaining sets of points. The first is based on the length and is calculated from the equation:

(1 - Expected width / Detected length) ** 2

The ratio of width to length is essentially a measure of elongation. The smaller this value is, the more road-like the set of points and therefore it is subtracted from 1, which is the maximum possible score. This result is squared to spread the values for better comparison. The expected width is used rather than the detected width since the detected width varies along the length of the segment. These variations in the detected width form the basis of an alternative certainty measurement.

The second measure of certainty counts the total number of pixels in the set forming a Hough peak that are above some percentage of the predicted width and divides it by the total number of pixels in the length.

(# pixels > % predicted width) / (total # pixels in Hough set)

This can be thought of as a density check on the road surface. This forces errors in width and length to be weighted more equally.

Road Detection Results

The road detection algorithm was tested on a set of 40 images. The algorithm requires as input the width, length, orientation and position of each road. For an error bound of 15% on position and no error in width, the

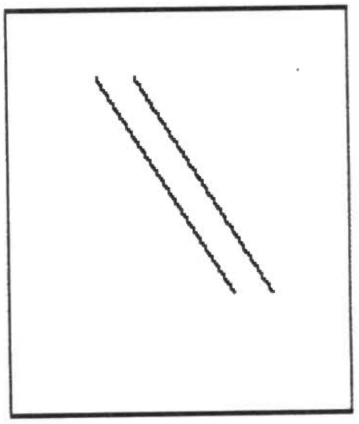
probability of detection was 85%. An example of a detected road and its corresponding elongation certainty value is shown in Figure 8.

Since road detection can use the map to reduce the search area, and less evidence in the form of features is required to verify the position, one of the most important considerations in the development of road detection algorithm was processing time. A careful evaluation was done to isolate the most time consuming steps in the algorithm and the code was optimized. It must be kept in mind that the input width and tolerance on the orientation both affect the required processing time. Therefore, the evaluation was performed with these values fixed. The worst case for road width in the image database was 19 pixels, and the worst case error in orientation, based on current map data, is 15 degrees.

The tests were performed on a VAX 785 which completes 1 operation per microsecond. The results for subimages of 2 different sizes are shown in Figure 9. The timing achieved both in terms of seconds and operations per pixel are well within the requirements of present ATR systems.

a) Original Image of Road

b) Extracted Subimage
with rotation



c) Detected Road
Certainty = 0.7476

Figure 8. Road Detection Results

Image Size	87x99	181x145
MMF (horizontal)	0.16 sec	0.50 sec
MMF (vertical)	0.10 sec	0.42 sec
Average/Threshold	0.10 sec	0.27 sec
Thinning	0.07 sec	0.21 sec
Hough Transform	0.18 sec	0.52 sec
Peak Detect	0.46 sec	1.04 sec
Confidence (3 peaks)	0.15 sec	0.46 sec
Total	1.22 sec	3.42 sec
VAX (1 OP/microsec)	141 op/pixel	130 op/pixel

Figure 9. Road Detection Timing Analysis

CHAPTER IV

TRAIL DETECTION ALGORITHM

Introduction

Since no map data is available, trails must be identified by their features. A trail can be broken into segments which are characterized by parallel lines. Linear features are useful for identifying parallel relationships and easy to parameterize. Parallelism can be represented by a generic geometric model. Curves more accurately describe the trail, but are difficult to parameterize. Therefore, the approach involves both features. Reynolds (1984) has described how features with a common pixel-based representation can be combined in a straight-forward manner.

Figure 10 depicts the general organization of the trail detection algorithm. The first step is to extract features from an input image. There are three features which combine to provide the necessary information for trail detection; they are lines, edges, and regions. The feature extraction subfunctions pass their results to the rule-based reasoning. The reasoning analyzes and combines features. The output from the reasoning is the precise location of edges that the algorithm has determined to be trails.

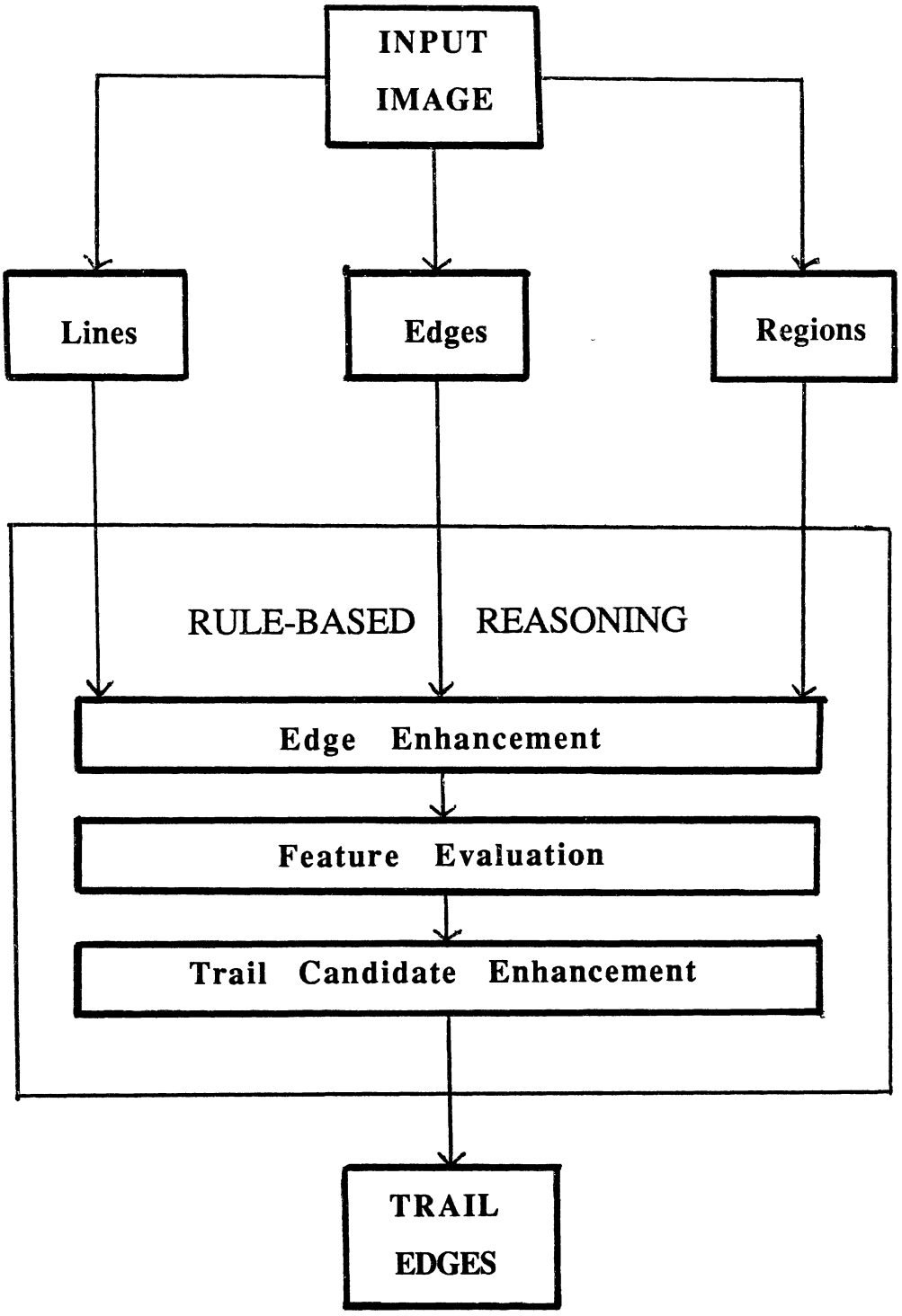


Figure 10. Trail Detection Method

Line Extraction

The purpose of line extraction is to detect linear features and identify relationships between them. The procedure used in trail detection for extracting lines is based on the Burns' line detection algorithm (Burns, 1988).

The most distinctive characteristic of a trail is the anti-parallel nature of its two sides. Anti-parallel lines are lines whose gradient phases are 180 degrees different; that is, a transition in an image from light to dark pixels is 180 degrees different from a transition from dark to light pixels. The information from the Burns algorithm makes it possible to identify anti-parallel lines.

The first step in the Burns' line detection algorithm generates gradient magnitude and gradient phase maps using a Prewitt edge operator. The gradient phase of a pixel computed by the edge operator has a value between 0 and 360 degrees. To group pixels into line support regions, the algorithm partitions gradient phase into 16 intervals of 22.5 degrees each. A second phase map is also generated; it is offset from the previous phase map by 11.25 degrees. One phase map is insufficient because lines become fragmented when their orientation is close to the partition boundary. The phase maps are then relabeled such that each group of similar phase is given a unique label. The algorithm has also been tested using 8 phase intervals for each of the two maps for a total of 16 phase values. When this smaller value is used, the effect is to create longer lines, but subtle changes and shorter lines are less detectable.

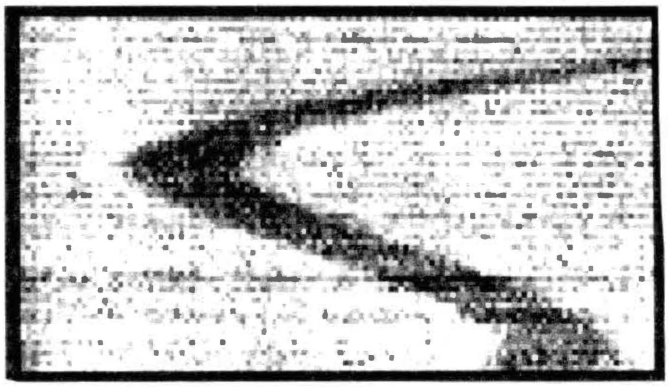
The algorithm combines the two resulting connectivity maps using a voting procedure. Each pixel in the image gets one vote, and it casts its vote for the larger of the two phase regions that overlaps it. After all votes have

been cast, each pixel location is labelled with the highest scoring region that overlaps it.

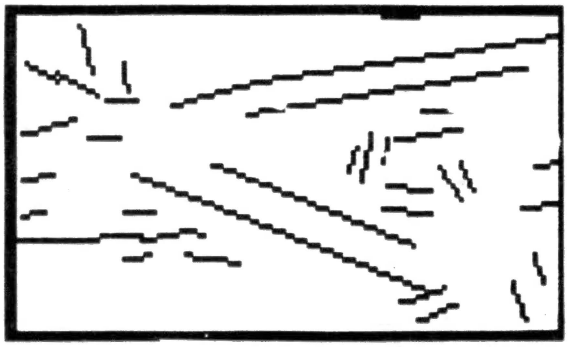
Lines are then fitted to the remaining regions. This is done using moments, with the gradient magnitude as the weighting. A line's orientation, equation, and endpoints are calculated. These parameters determine which lines are anti-parallel. The maximum and minimum distance between lines, and the maximum angular deviation are inputs to the parallel line detection portion of the algorithm. The output is an array flagging line pairs which are anti-parallel.

The algorithm used in this study deviates from the Burns approach by applying a filtering process to the phase regions before voting and again before lines are fitted. The purpose of this filtering is to eliminate small regions and gaps caused by the voting procedure. This helps to reduce the amount of information that must be stored. Filtering is important in the trail detection because it is the phase regions that are overlayed with edges. Phase regions are used as opposed to the fitted lines because they are wider and thus allow for a larger tolerance in the overlap. Information about the correspondence between lines and edges is improved if the phase regions are smooth. The filtering process works as follows: if the majority of a region's neighboring pixels are of one phase interval which differs from the initial region's phase by not more than two intervals, then the region's phase is changed to reflect that majority.

For the input scene in Figure 11a, the resulting fitted lines are shown in Figure 11b. The lines corresponding to the straight segments of the trail are the most prominent in the output image. This is the strength of the line extraction algorithm. However, there is no straight-forward method for



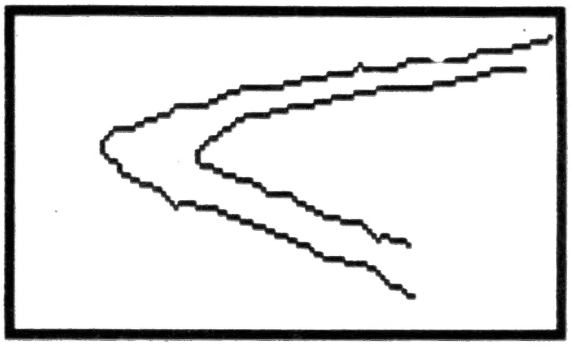
a) Trail image



b) Lines

c) Edges

d) Regions



e) Results

Figure 11. Trail Features

identifying a relationship between the two pieces of the trail using lines alone.

Edge Extraction

Edge extraction detects and highlights boundaries in an image. Unlike lines, edges are capable of accurately representing object boundaries which are curved. Lines can always be described by a first order equation and their parameters are derived primarily from the gradient phase map. Edges are typically represented by higher order equations and edge extraction focuses on manipulating the gradient magnitude map using local phase.

The edge extraction algorithm is based on the work of John Canny (1988). He describes three performance criteria which apply to a general purpose edge detection problem. The first of these is good detection. There should be a high probability of marking a pixel that is part of an edge, and a low probability of marking something that is not an edge. The second criteria is localization. That is, the pixel that is marked should be as close as possible to the center of the edge. The third criteria involves eliminating multiple responses. Multiple responses are really a subset of good detection. However, the mathematical form for good detection corresponds to maximizing the signal-to-noise ratio. Therefore, the multiple response criteria must be made explicit. Canny shows there is a direct tradeoff between good detection and localization of step edges based on operator width.

These criteria are used to derive optimal operators for step edges. When a Gaussian operator, which is much less computationally intensive, is used to estimate the optimal operators, there is a 20% reduction in performance. We have found the Prewitt filter to be the easiest to implement and suitable for

both edge and line extraction. Thus, the edge extraction uses the same magnitude and phase maps generated for use in the line extraction.

Canny (1988) also combines the concepts of adaptive thresholding and hysteresis to reduce the effects of noise. The present algorithm selects an upper and lower threshold from a histogram of the gradient magnitude map for the given input image. Hysteresis reduces streaking in the following way. The threshold selected from the lowest percentage of pixels in the histogram is applied to the entire image. After the lower threshold is applied, local minima in the magnitude map are suppressed. This results in a thinned magnitude map. The upper threshold selects the strongest peaks to be used as starting points for the edge tracing routine.

Tracing begins with a peak that is above the upper threshold, and subsequent pixels for that edge are found by looking 90 degrees from the phase for a pixel that is still on. All pixels which belong to the same edge are given the same label in the output image.

Edge extraction results for the input scene of Figure 11a are shown in Figure 11c. The edge extraction algorithm is capable of accurately representing curvature as well as straight segments. In fact, both characteristics may belong to the same edge. This is the key to joining the piecewise linear features.

Region Extraction

Although the reasoning process does not currently include regions, they are potentially useful for trail detection. However, the use of regions was considered, and several strengths and weaknesses were found.

Region extraction segments the image into homogeneous regions. The SCENESEG algorithm developed at Texas Instruments oversegments an input image and iteratively merges regions with similar statistics. Because of the statistical nature of the scene segmenter, it is capable of describing trail surface uniformity. Although a trail may contain some discontinuities, the algorithm can use this uniformity measurement to increase or decrease the confidence that an area actually contains a trail.

The image in Figure 11d shows the results of SCENESEG applied to the image in Figure 11a. This result illustrates that the scene segmenter handles curvature. The region that corresponds to a trail has a consistently narrow width compared with a much greater length. However, this elongation is difficult to parameterize. A common procedure for calculating elongation involving moments fails because it cannot be guaranteed that the pixels on a trail will be a unique function of x with respect to y or vice versa. It is also very difficult to describe the relationship between two regions for the purpose of reasoning. As shown in Figure 11d, it is not clear whether region 1 is above or below region 2. This becomes important when a decision is being made about merging regions.

Reasoning

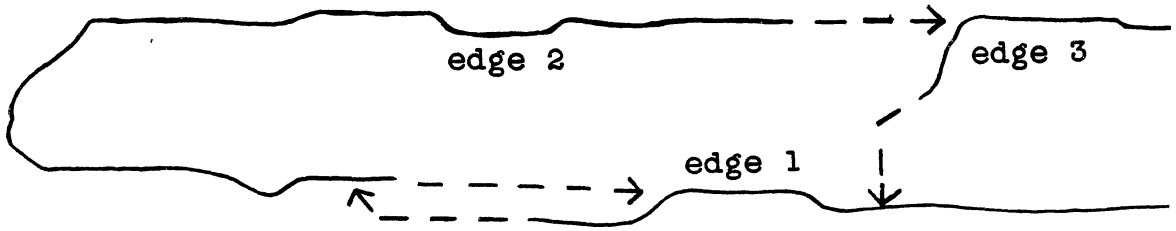
Rule-based reasoning utilizes scene features synergistically to identify trail-like areas in a scene. Each rule consists of a condition/action pair. When the features meet the conditions of a Gestalt law or the generic geometric model, then the rule updates those features. There are three steps in the reasoning process; they are edge enhancement, feature evaluation, and trail candidate enhancement.

The purpose of the edge enhancement is to obtain the best representation of edges before any feature evaluation occurs. The feature evaluation forms a ranked list which is used to focus attention on trail-like objects in the trail candidate enhancement stage. The three parts of the generic geometric model defined by Fua (1987) are general goal object characteristics, integration of multiple features, and prediction of missing components. Feature evaluation involves the goal object characteristic of parallelism and integration of features. The third aspect, prediction of missing components, is addressed in trail candidate enhancement.

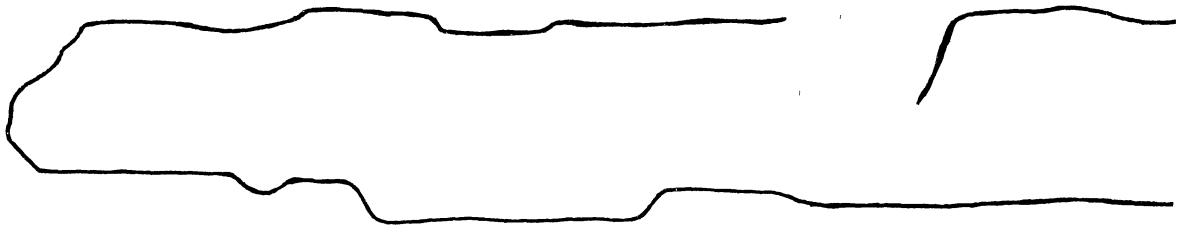
Edge Enhancement

A preliminary part of the reasoning process is edge enhancement. The edge enhancement rules either link or break edges and employ the Gestalt laws (Nazif, 1984) of proximity and continuity. Since Gestalt laws are a general principle of vision, these rules apply to all edges in an image.

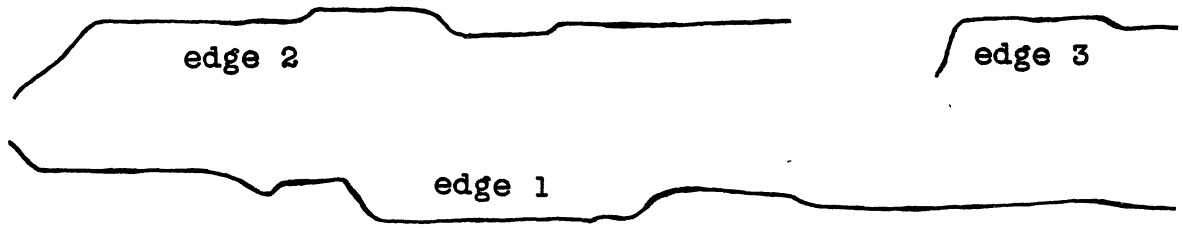
The results of the edge extraction algorithm often contain edges which are broken. The purpose of the edge linking routine is to fill those discontinuites according to the Gestalt law of continuity. The tracing method used for linking differs from the tracing method used to generate the initial edges in that it refers to the original magnitude and phase maps, rather than the ones which were thresholded and thinned. Tracing begins at the endpoint of an edge and continues until another edge is encountered, as shown in Figure 12a. If the edge that is encountered links to the edge that linked to it, then the rule will join these two edges. It does this by extending the edge which required the fewest pixels to reach the other edge. The rule erases any extraneous pixels from the second edge, and relabels the remaining pixels.



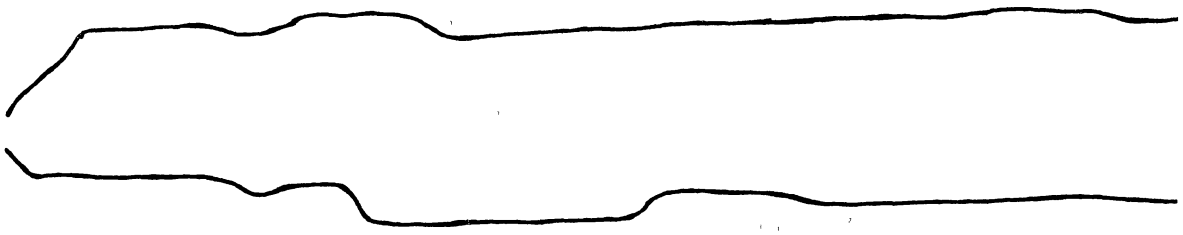
a) Reasoning input edges



b) Edge enhancement linking



c) Edge enhancement breaking



d) Trail candidate enhancement

Figure 12. Reasoning

Figure 12a shows disconnected edges and the course from their endpoints through the magnitude map. In this example, edge 2 links to edge 3, but edge 3 does not link to edge2. Therefore, they cannot be connected. Edge 1 and edge 2 link to each other. Figure 12b shows the resulting edge connected by the shortest link.

In certain cases, a rule is needed to split edges. The purpose of this rule is primarily to simplify the reasoning task. Occasionally, an edge is connected to both sides of a trail due to image anomalies or obscuration. Figure 12b illustrates this condition. The algorithm can avoid additional complications by breaking such an edge at an appropriate point. The rule must break any edge that overlaps two or more line support regions which are parallel. An edge that meets this condition is traced, keeping track of the line support regions that it passes through. The rule starts a new label when an edge enters a line support region that is parallel to one which has already been encountered. The results are shown in Figure 12c.

Feature Evaluation

The purpose of the evaluation process is to form a ranked list of edge pairs which are the most trail-like in the image. This list then functions as a focus of attention mechanism to give processing priority to trail-like objects.

Feature evaluation forms the list by calculating a score which is based on the amount of parallelism between edge pairs. The first step is to determine line phase regions which correspond with edges. This is similar to Reynolds' common pixel-based superimposing technique (Reynolds, 1984), but uses edges and lines instead of edges and regions. Line phase regions work well for this process since they are usually greater than one pixel thick, and thus an exact

match is not necessary. The algorithm stores the number of overlaps between each edge and line in an array.

Next, the information in the overlap array is combined with the parallel array output by the line extraction. If edge 1 overlaps line 1, and edge 2 overlaps line 2, and line 1 is parallel to line 2, then there is some parallelism between edge 1 and edge 2. The amount of parallelism, indicated by the overlap array, is accumulated in a score array. This process essentially connects together multiple parallel line sets overlapped by the same edge pair.

At this point, the feature evaluation could form the ranked list based on the accumulated score array. However, single long edges tend to dominate those scores. Instead, the score is the product of the minimum accumulated edge/line overlap and the average accumulated edge/line overlap. This tends to emphasize edges that are close to the same length rather than long edges paired with several very short edges. Figure 13 illustrates several example score calculations.

Trail Candidate Enhancement

Using the ranked list of edge pairs, the algorithm can now make a hypothesis about the location of a trail and try to find more evidence to substantiate that hypothesis. If the hypothesized trail is truly a trail, and one of its edges appears in the ranked list more than once, then it logically follows that there is a relationship between the other two edges with which it is paired. Figure 12c illustrates an example. If edge 1 and edge 2 form a pair, and edge 1 and edge 3 form a pair, then there may be a relationship between edge 2 and edge 3. If a relationship can be established, then the score of the resulting trail should be even higher than either of the two separate trails.

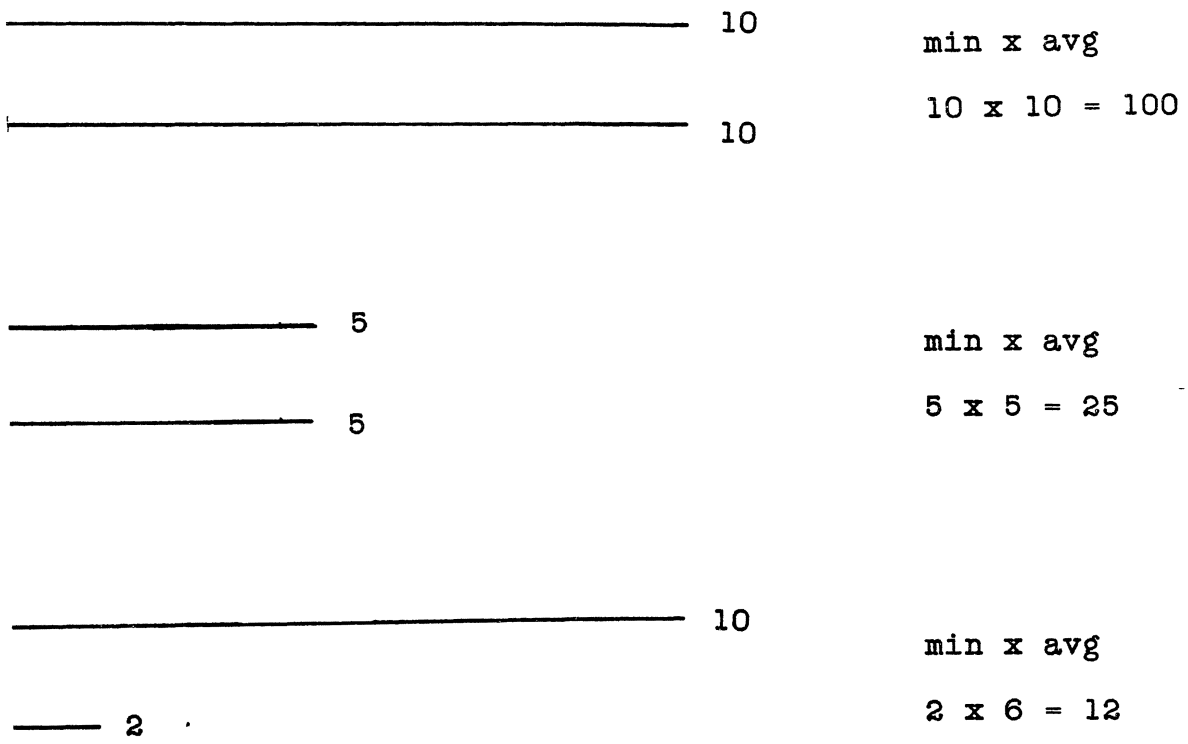


Figure 13. Sample Edge Pair Score Calculations

The initial trail hypothesis is the top-scoring pair; the algorithm then searches the ranked list to find other pairs which contain one of its edges. Only a certain number of the highest scoring pairs become the hypothesis, but the search list contains every pair of edges whose score is greater than zero. Once the algorithm finds a match, it must determine if there is a relationship between the two edges that are parallel to the same edge. The desired relationship is collinearity, although this description is used somewhat loosely since the trail can be curved. However, if the two edges already exist on the list as a pair, their relationship is known to be parallel rather than collinear, and the algorithm need not investigate further.

When two edges are found that are parallel to the same edge, two additional conditions must exist before the rule executes. First of all, the rule must establish a link between the two edges. The geometric model of parallelism suggests the location of this missing component. The rule searches the appropriate area in the magnitude map in a manner similar to that used in the edge linking routine. The difference is that only one of the edges has to link to the other. The second condition is that the score of the new trail must be greater than the score of any previous trail which included either of the new trail's edges. In order to find the necessary score, entries in the ranked list that are higher than the current hypothesis are searched for relevant edges. If a match is found, the corresponding score is used as the score that the new pair must beat. In the case of Figure 12c, the score of the new connected edge is sufficient. The new pair replaces the old as shown in Figure 12d.

After all changes are made for a particular hypothesis, the feature evaluation process repeats. The result is a new ranked list. A new hypothesis

is selected, and the process begins again. This continues until the algorithm has investigated the desired number of hypotheses.

A final application of the feature evaluation process results in a score for each of the remaining edge pairs which indicates their trail-likeness. The algorithm applies a threshold to these results; edge pairs above the threshold are output as trails, while those below the threshold are discarded. As a final step in the processing, a special subfunction removes from the ends of the edges all pixels which do not contribute to the score.

Summary of Trail Detection Algorithm

The first step in the trail detection algorithm is to generate the edge and line features from an input image. The line features form the basis of the generic model used: parallelism. Rules are used in the feature enhancement stage to combine edges based on proximity and closure. Feature evaluation combines edges and lines by measuring edge pairs against the parallel line model to select trail candidates. The trail candidate enhancement predicts and verifies missing trail components.

Once all of the desired feature enhancements have been completed, a final score is assigned to the trail candidate edge pairs. The calculations involved in this score, based on parallelism and length, are the criteria for identification. The final step is to apply a threshold to this score. The experimental results, discussed in the following section, are helpful in determining the threshold value.

CHAPTER V

TRAIL DETECTION EXPERIMENTAL RESULTS

Test Database

The test database for the trail detection performance evaluation contained 93 images. All trails were ground-truthed for their average width and length in pixels and for percent occlusion. Some images contain multiple trails. Figures 14 and 15 show width and length distributions for the test images. The width distribution indicates that trails with a width of less than 10 pixels may be over-represented in this test set. The length distribution has a more uniform population. This is relevant since the scoring mechanism is based on length. In the case of a very short trail, the algorithm may select the trail as its top candidate, but its length may not be large enough to put it above the score threshold.

The database consisted primarily of FLIR images, but includes some TV images, also. Natural terrain features are more apparent in the high spatial frequency information from TV, but that does not appear to be the case for the line or edge features used to detect trails. Therefore, no parameter adjustments are made based on the type of input imagery.

Test Parameters

Trail detection performance is affected by inputs and thresholds used within the algorithm. The program requires inputs of median filter and

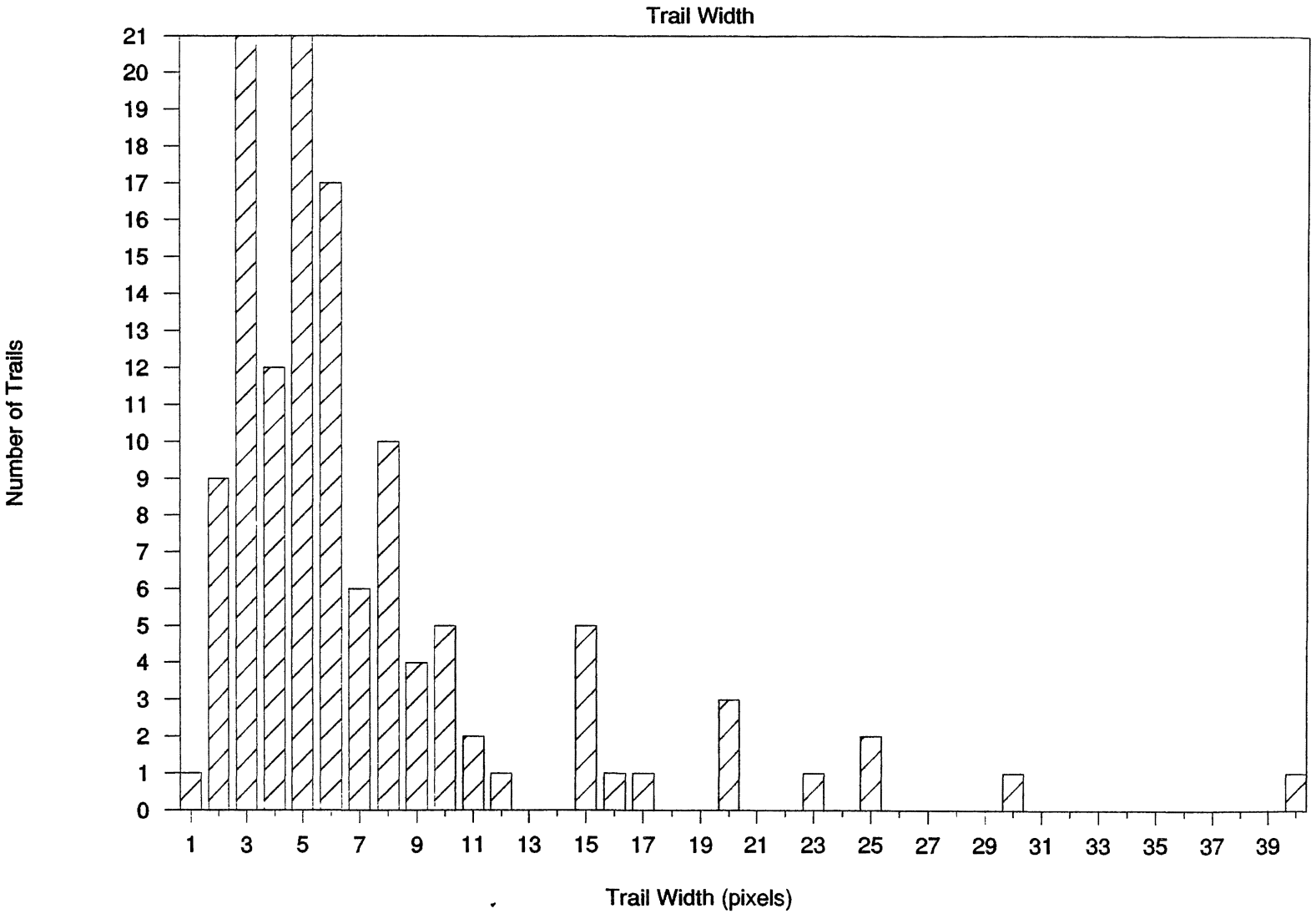


Figure 14. Database Width Distribution

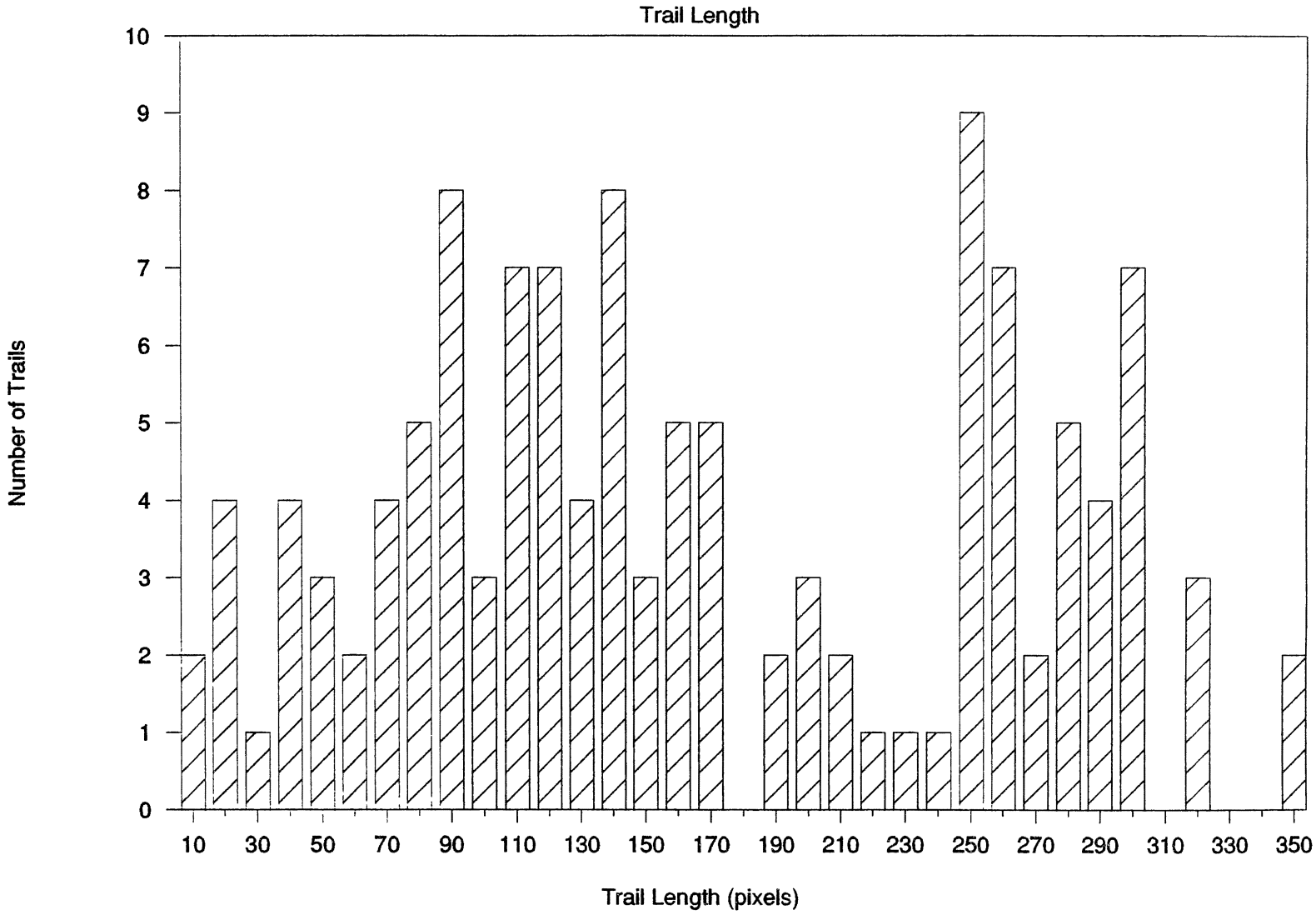


Figure 15. Database Length Distribution

Prewitt operator sizes, minimum and maximum separation between parallel lines, and a score threshold.

Input Parameters

Median filtering is used for preprocessing. Generally, this improves the quality of the features and smoothes out noise. However, if the filter size used is too large, it may remove trails with a small width. The trail detection algorithm usually responds well when a filter size of 5 is used, unless the trail width is 3 or less. In these cases, a filter size of 3 must be used.

The same filter size that was used for median filtering is used in the Prewitt filter. In edge extraction, a larger Prewitt filter size will result in a better signal-to-noise ratio while a smaller filter size gives better localization and more detail. The Prewitt filter size used may also affect the grouping of pixels into regions in the line extraction. The same restriction on filter size compared with trail width applies to the Prewitt filtering. In particular, if the filter size is larger than the width of the trail, both sides of the trail cannot be detected in either of the feature extraction algorithms.

The primary input to the line extraction function is the minimum and maximum separation between parallel lines. If these values are precisely known, the performance of the trail detection algorithm will be better. However, they can be entered with whatever degree of accuracy they are known. For our experiments, the values used were plus or minus 25% of the ground-truthed width.

The score threshold is the most critical input in determining if a trail will be detected. The tests were conducted using a very low score threshold of 100. This provides the data to compare false alarms per image with performance

based on detection percentages. In this way, an application could select a score threshold according to its own performance requirements.

Fixed Parameters

Image independent parameters are fixed within the trail detection algorithm. Edge extraction determines the upper threshold for selecting seed pixels from the top 20 percent of magnitude values. The lower threshold is set at 100 percent. This upper threshold was selected by experimentation. The lower threshold is used because the reasoning subfunction is going to allow tracing to any pixel regardless of its magnitude value.

A threshold on minimum line length is used to discard short lines and thus reduce the amount of data that must be stored. This threshold is set at 5 pixels. This value can be increased to reduce processing time, but will also have an affect on performance. If a curved trail is represented by several short lines, it is possible that those lines may be too short to be retained.

Another threshold that affects all trails, but particularly those that are curved, is the angle tolerance on anti-parallel lines. Lines may be broken into small pieces to fit the edge of a curving trail. However, the lines on the other side of the trail may be fit in such a way that no anti-parallel lines are detected. This angle tolerance may also affect relatively straight trails whose edges are vague or not straight. These are the reasons the angle tolerance has been set to a very liberal 10 degrees.

In the reasoning subfunction, a specified number of top scoring candidates are used as seeds for linking. The algorithm attempts to link each of these to some second, larger number of trail candidates. These values are set

at 10 and 50. This will affect the output of trail detection if the features break the trail into several short pieces none of which are in the top 10 scores.

Performance Measures

There are several difficulties in defining performance measurements because of the problems inherent in ground-truthing image features. It is not always clear whether a particular pixel is part of the trail or not. Sometimes the output contains extraneous edges while in other cases parts of the trail are missed. Decisions involving these situations were left to the operator. However, in most cases where any part of the trail was correctly detected, it was counted as a detection.

Another phenomena occurred frequently enough to address it directly. In many cases, only one of the output edges matched the edge of the trail. Two reasons for this became apparent. The first is that the features from one side of the trail may be strong enough to force an incorrect pairing into the output list even when the features from the other side of the true trail are weak. The second reason is that most trails have a visible shoulder. Keeping in mind the goals of scene context processing, it is just as useful to know the boundaries of the trail shoulder as it is the boundaries of the trail itself. Therefore, detection of the trail shoulder was counted as a correct detection.

Performance Analysis

Figure 16 shows the probability of detection versus trail width in pixels for a score threshold of 1000. The trail width axis was broken into intervals that contain roughly equal number of images. The graph shows poor performance for trails thinner than 4 pixels. The most obvious explanation

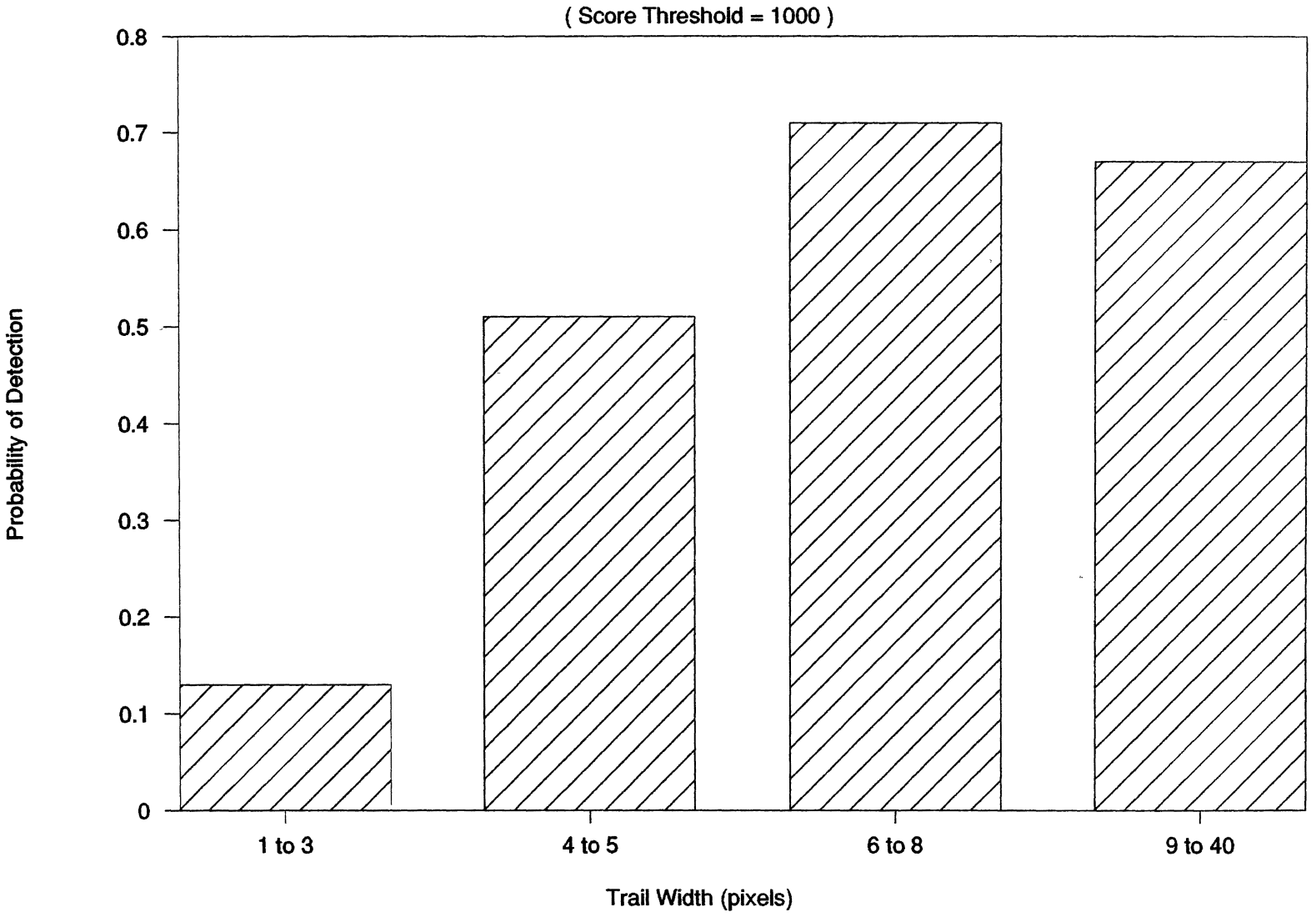


Figure 16. Probability of Detection

for this is that small widths are sensitive to median filtering. Another possible explanation is not related to width at all. Most of the trails with small widths have small widths because they are at a long range. This could cause the features to be degraded for the same reasons that it is difficult to identify anything at long range. The graph also shows a slight drop in performance for large widths. Again, range probably has a role. At close range, features may have a different appearance, and many new features have become visible.

Figure 17 shows false alarms per image versus trail width. This graph indicates that as the trail width increases, there is also an increase in false alarms. This is probably due to omitting the surface features. Also, the concept of proximity in the Gestalt laws has been violated.

The receiver operating characteristic for the entire data base is illustrated in Figure 18. It shows the relationship between probability of detection and false alarms per image for decreasing score thresholds. As expected, lowering the score threshold increases the probability of detection, but has a corresponding increase in false alarms. This plot is useful for determining a score threshold given the requirements of a particular ATR system.

Parameter Sensitivity and Detection Limitations

The image of Figure 19a pushes the limitations of the trail detection algorithm. Figure 19b shows the results for inputs of 2 to 15 for minimum and maximum separation between anti-parallel lines, and a score threshold of 2000. The distance of 2 to 15 will liberally cover the widths of all the trails in

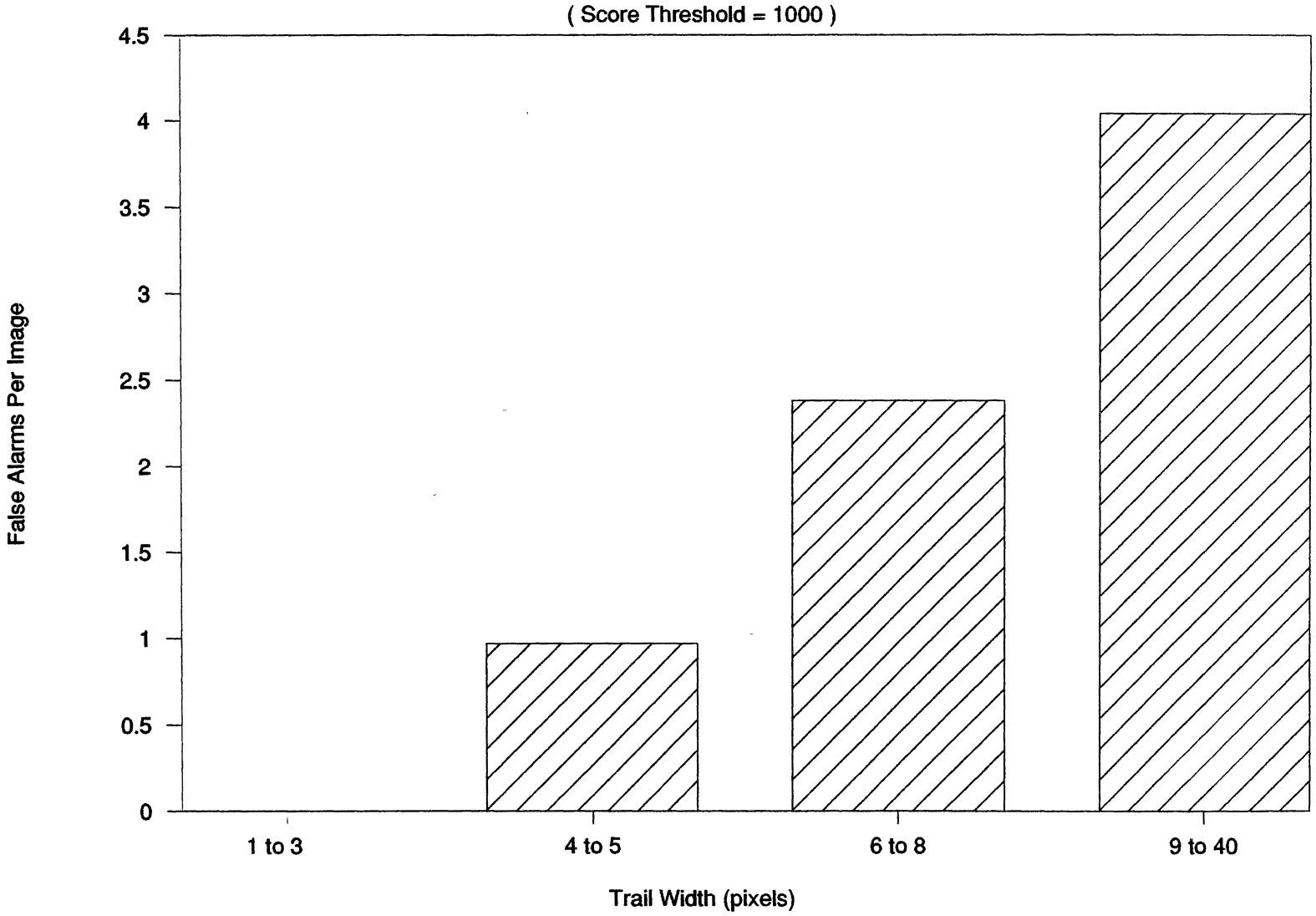


Figure 17. False Alarm Rate

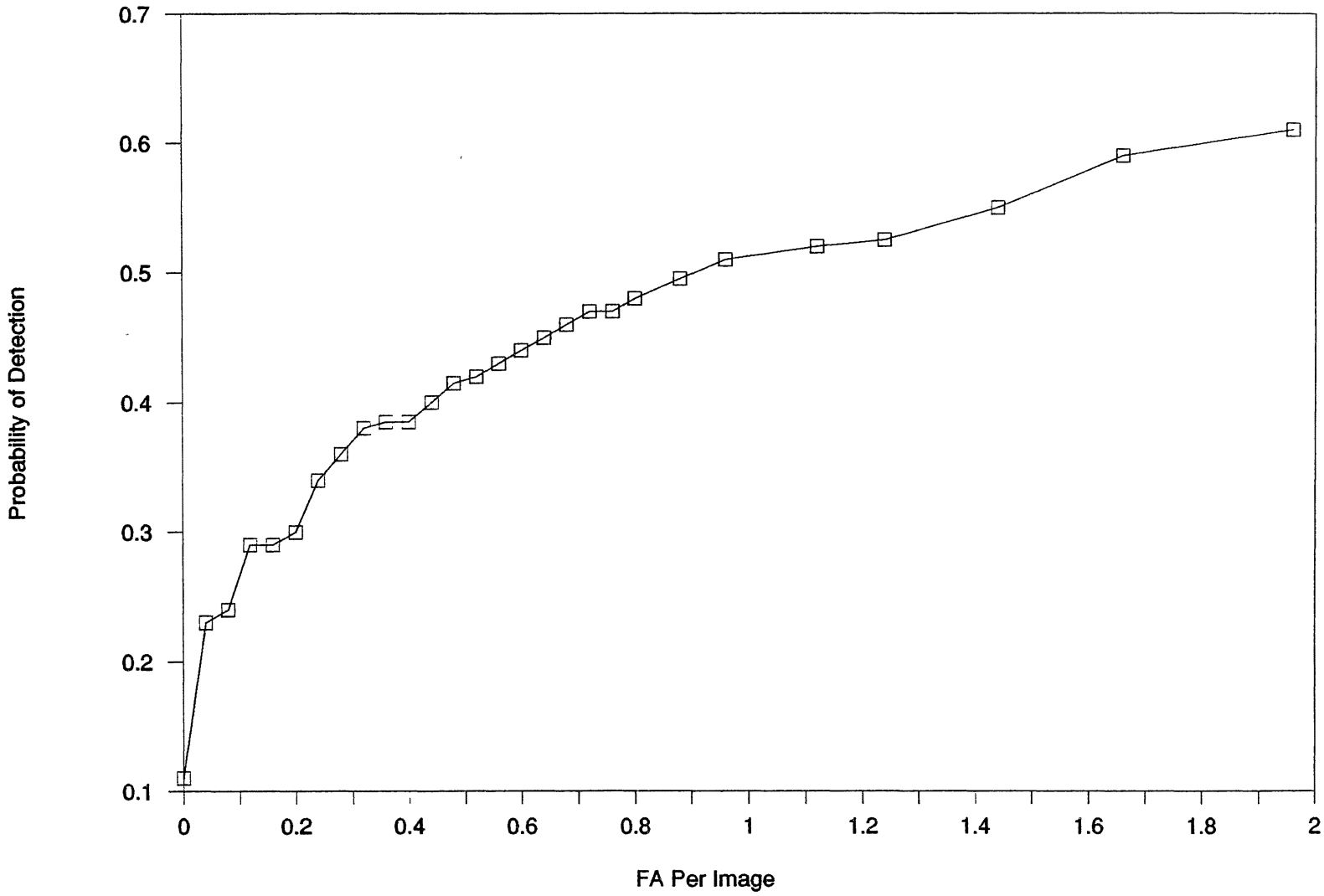


Figure 18. Receiver Operator Characteristic

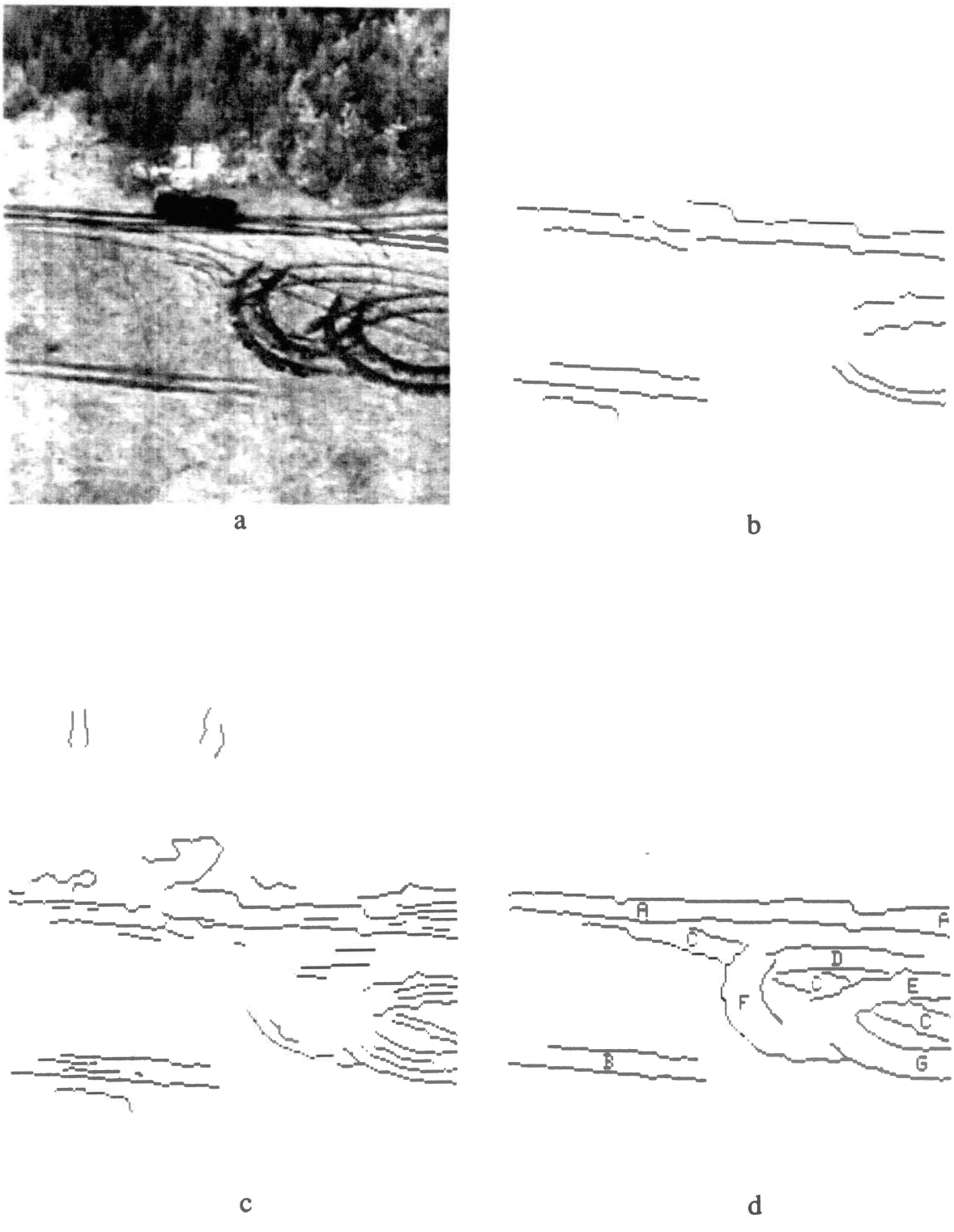


Figure 19. Trail Detection Limitations

this image. The score threshold of 2000 is relatively high, but was found to be reasonable for reducing the false alarm rate over a larger data base of trail images. The result for a score threshold of 100 is shown in Figure 19c. The desired trails for the image of Figure 19a are labelled in Figure 19d.

Trails A and B are easily detected even with the higher score threshold of 2000. Trails C through G illustrate one situation where the algorithm is limited. The difficulty with these trails is that many of them overlap one another. When the Prewitt filter passes over this area, multiple trails may fall within its window. The result of this is that either one trail dominates and the other is ignored or the output is random. A good example for this case is trail C. It intersects in several locations with tracks that are of higher contrast. Not only is trail C of low contrast, but its width is relatively smaller and therefore sensitive to filter size. When the filter size is small enough to detect trail C and D, it loses its smoothing effect on all trails, thus causing many of the edges to be broken. The threshold on seed trails comes into play here for a number of reasons. The edges are broken up which makes it difficult for a small piece to get into the top ten seed trails, especially when there are so many trails in the image and only ten are selected. Added to that is the condition that each trail consists of two tracks, and thus it is possible to form several different combinations of trail candidates for each one. A more critical factor affecting trails D through G is that they are curved. In this image, trails D through G are curved, and the lines are affected in the ways described above for curved trails.

Example Results

The final results for the infrared image of Figure 11a are shown in Figure 11e. One trail was selected for output. The algorithm used a median and

Prewitt filter size of 5. The range for parallel lines was 1 to 10. The results indicate that the reasoning process was successful in reconnecting several of the broken edges.

Figure 20 contains a sharply curving trail, partially obscured by trees. The parallel line separation input was 3 to 10. One trail exceeded the score threshold. Although its edges end abruptly due to obscuration by trees, the result is still a correct detection.

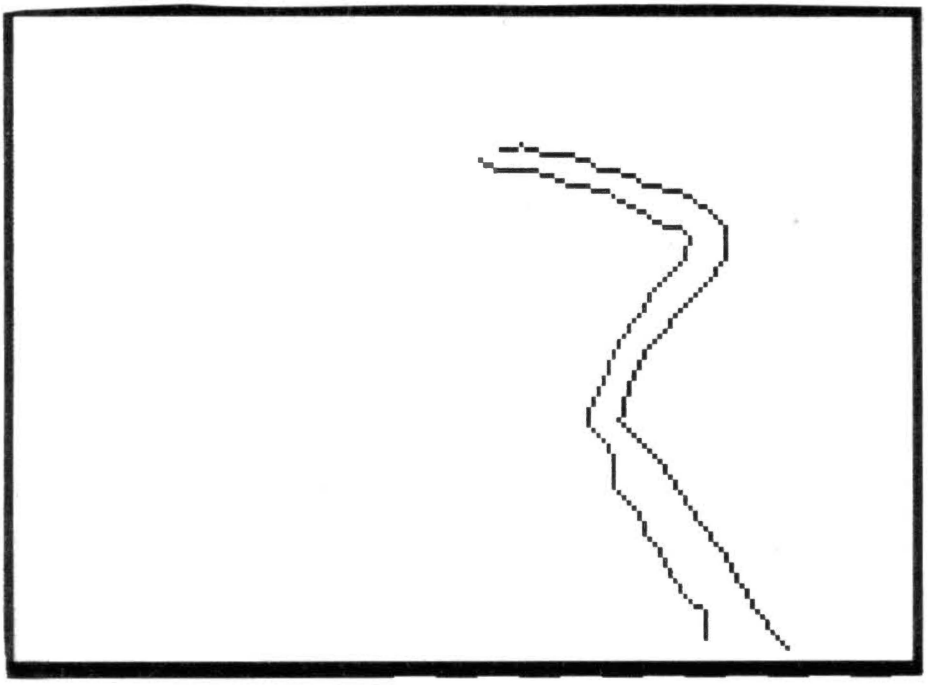
The image of Figure 21 appears to contain a plowed field. In this case, there is more than 1 trail to be detected. The input for parallel line separation was 6 to 10. The algorithm selected 5 trails. Of these, 4 are correct trails while the fifth is a single vehicle track or shoulder. Because of their trail-like characteristics and proximity to the actual trail, it is common for the algorithm to detect a road shoulder.

Figure 22a is a TV image of vehicle tracks. For a parallel line separation input of 3 to 7, the results are shown in Figure 22b. The algorithm detected the tracks in 3 separate pieces. When the parallel line separation input is changed to 10 to 20, only 1 trail is output which consists of the outer edges of the tracks. Its result is shown in Figure 22c.

Timing Analysis

The trail detection algorithm was implemented in FORTRAN on a VAX 6220 computer. Figure 23 contains information about 3 images which were among the most CPU intensive in our database. Because of the reasoning functions,

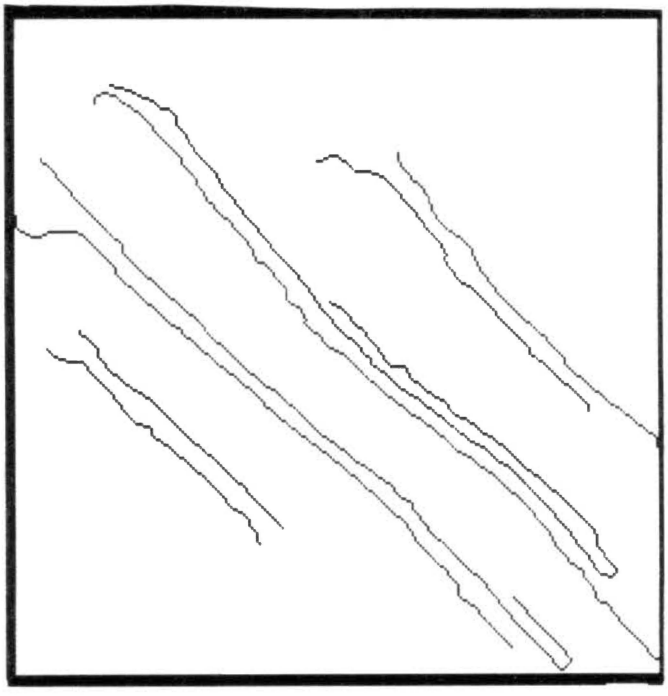
a) Input image with trail obscured by trees



b) Detected trail

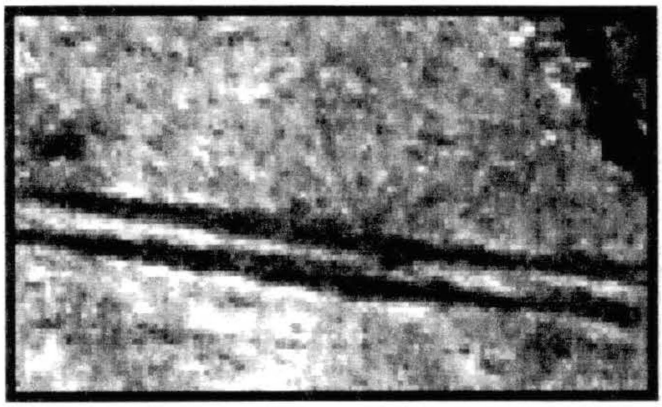
Figure 20. Curved Trail with Obscuration

a) Input image of field with multiple trails

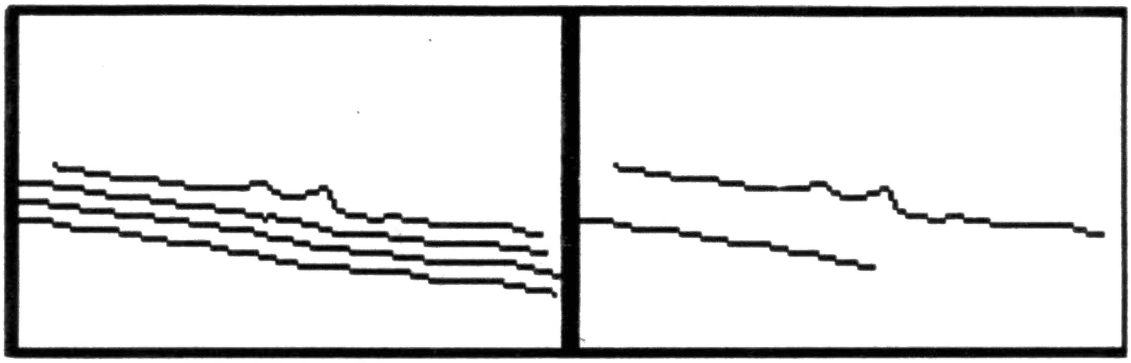


b) Five detected trails

Figure 21. Plowed Field



a) Input image containing vehicle tracks



b) Three detected trails for parallel separation input of 3 to 7

c) Detected trail for parallel separation input of 10 to 20

Figure 22. Vehicle Tracks

processing time is effected by the image contents as much as input parameters and image size. In general, it would seem that a smaller range of separation between parallel lines would reduce the number of features passed to reasoning, and thus reduce the processing time. This is not directly supported by the data in Figure 23 since a larger separation (image 2) took less time than a smaller separation (image 1). However, tests which varied that parameter would probably support that concept. With respect to image size, a 5% reduction in size (between 2 and 3) corresponds to a 14% reduction in processing size. Again, this reduced processing time may not be directly related to image size. What actually proved to be the case throughout the database is that the algorithm processing requirements are so dominated by image features that any rule relating processing time to anything else would always have exceptions.

The critical issue in this evaluation was maximum processing time. The images included in Figure 23 were not selected based on their size or trail widths (and thus input parameters), but because they indicate the worst cases for processing time from among the database. The goal was to keep processing time for a 256x256 image within 5 minutes. This was accomplished.

IMAGE	IMAGE SIZE	FILTER SIZE	MIN/MAX SEPARATION	TIME MIN:SEC
1	256x256	5	1 , 10	4:34
2	256x256	5	3 , 15	4:18
3	256x242	5	1 , 10	3:41

Figure 23. Trail Detection Processing Requirements

CHAPTER VI

CONCLUSIONS

We have shown that map information is helpful for detecting roads. Not only does it improve the probability of detection, but it also significantly reduces processing time. Map information allows our algorithm to be restricted to a subset of all possible features. By selecting only linear segments, the Hough transform can be applied in a straight-forward manner. The modified median filter enhances regions of the proper size. This results in an algorithm that is suitable for a variety of imagery containing roads at various ranges, and is substantially faster than existing road detection algorithms.

The trail detection algorithm is unique in its effort to recognize trails without the aid of map cues. This algorithm succeeds by combining several techniques from existing computer vision research. The rule-based approach and combination of multiple features are essential elements of this algorithm, and the content of the rules and choice of features were determined in this study. The trail detection algorithm goes beyond applying knowledge in the segmentation phase and addresses the identification phase.

The trail detection algorithm demonstrates effective performance for both TV and IR imagery. This characteristic can be particularly advantageous in an ATR system employing multiple sensors. Another strength of the algorithm is its ability to recognize trails despite occlusions and curvature.

These strengths result in a trail detection algorithm that is applicable in a variety of scenarios.

Future Work

The framework of the rule-based reasoning allows for knowledge expansion. There are several areas in which additional knowledge would improve trail detection performance. The algorithm needs a better understanding of how to deal with occlusions. Other scene regions characterized by anti-parallel lines could be mistaken for trails. A rule could incorporate knowledge about the subtle differences between vehicle tracks and trails.

For a scenario in which scene context processing is applicable, it is expected that the images would be highly cluttered with natural camouflage. Therefore, it is likely that the trails will be obscured in such a way that no actual link can be detected. The image of Figure 20 illustrates this case. The detected trail ends abruptly due to a tree, even though several pieces of the trail extend past that tree. At present, the algorithm makes changes in the edge image only when a link can be established in the magnitude map. A new rule could handle this case. In this rule, proximity would have to be a consideration as well as a strict collinearity requirement for both sides of the trail. The rule could use line parameters calculated in the line extraction to determine collinearity.

The trail detection algorithm could use knowledge to discriminate between trails and other scene regions with linear features. For example, rivers and plowed fields both have parallel edges. The ability to identify these areas would be useful not only for trail detection, but for the scene context

processing problem as a whole. Differences between rivers and trails would be difficult to detect without the use of lasar radar, but a rule could incorporate knowledge that multiple lines indicate a texture pattern rather than a trail. This type of knowledge would give the trail detection algorithm a better understanding of the image in Figure 21.

By utilizing range information, rules could expand the trail detection algorithm to detect vehicle tracks. Range information allows the algorithm to predict the width of the trail and also the width of vehicles of interest. As a result, the line extraction could restrict tolerances on the distance between anti-parallel lines. Two closely spaced pairs of anti-parallel lines at a distance equal to the width of the vehicle would characterize tracks. These characteristics are somewhat more complex than those of a trail; therefore, false alarms would be uncommon. Even the detection of a short section with these characteristics would be a strong indication of the presence of a vehicle.

REFERENCES

Burns, J. B., A. Hanson, and E. Riseman. " Extracting Straight Lines." IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (July 1986): 425-455.

Canny, J. "A Computational Approach to Edge Detection." IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (September 1986): 679-697.

Fischler, M., J. Tenenbaum, and J. Wolf. "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique." Computer Graphics and Image Processing. (1981): 201-223.

Fua, P., and A. Hanson. "Using Generic Geometric Models for Intelligent Shape Extraction." Proceedings of the Image Understanding Workshop. 1987.

Gonzalez, and P. Wintz. Digital Image Processing. 2nd ed. Reading, MA: Addison-Wesley, 1987.

Kuan, D., G. Phipps, and A. Hsueh. "Autonomous Robotic Vehicle Road Following." IEEE Transactions on Pattern Recognition and Machine Intelligence 10 (September 1988): 648-658.

McKeown, D., and J. Denlinger. "Cooperative Methods For Road Tracking in Aerial Imagery." Proceedings of the Image Understanding Workshop. April 1988.

Nazif, A., and M. Levine, "Low Level Image Segmentation: An Expert System." IEEE Transactions on Pattern Analysis and Machine Intelligence.6 (September 1984): 555-577.

Reynolds, G., N. Irwin, A. Hanson, and E. Riseman. "Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation," Representation and Control. Proceedings of the Workshop on Computer Vision. April 30 - May 2, 1984.

VITA

Christy Lynn Chism

Candidate for the Degree of

Master of Science

Thesis: FEATURE-BASED ROAD AND TRAIL DETECTION

Major Field: Electrical Engineering

Education: Graduated from Ponca City High School, Ponca City, Oklahoma, May 1979; received Bachelor of Science Degree in Electrical Engineering (Computer Option) from Oklahoma State University, Stillwater, Oklahoma, December 1983; completed the requirements for the Master of Science degree at Oklahoma State University, July 1990.

Professional Experience: Engineering Systems Analyst, Texas Instruments, May 1986 to present; Technician, Frontier Engineering, May 1985 to January 1986; Graduate Research Assistant, Department of Electrical and Computer Engineering, September 1983 to May 1985; Engineer, Texas Instruments, Summer 1983 and 1984.