OBJECT DETECTION IN INFRARED IMAGES

David L. Milgram*
Azriel Rosenfeld**

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

* Lockheed Palo Alto Research Labs, D52-53, B204, 3251 Hanover Street, Palo Alto, CA 94304.

**Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742.

ABSTRACT

This paper describes algorithms for detecting and classifying objects such as tanks and trucks in forward-looking infrared (FLIR) imagery. It summarizes research conducted in the course of a two-year project in the areas of image modeling, pre- and post-processing, segmentation, feature extraction, and classification.

1. Image models

The work on image modeling conducted under this project was concentrated in three main areas:

1) Modeling of the joint (gray level, edge value) statistics of FLIR scenes, as a basis for defining threshold selection techniques.
2) Modeling of thresholding and edge detection responses to background regions, as a basis for predicting false alarm rates.
3) Modeling edges in images as a basis for defining optimal edge detection operations and for evaluating edge detector output.

This work is briefly summarized in the following subsections. References are given to earlier project reports [1-4] in which detailed treatments can be found.

1.1 Model-based threshold selection

An approach to modeling FLIR imagery has been developed, based on the simplifying assumption that targets appear as homogeneous hot regions within a homogeneous cooler surround. This model describes the joint probability density of gray level and edge strength in such images, for various edge-detecting operations [1,2]. In brief, the model predicts that for low edge values (corresponding to points in the interiors of objects and background), there should be two relatively well separated probability peaks, of different sizes, representing the gray levels of object and background interiors, respectively. For higher edge values, corresponding to points on object/background borders, these peaks should merge together and become a single peak representing the

border range of gray levels.

The model just described can be used as a guide to segmenting FLIR images by thresholding. At low edge values, it should be easy to pick a threshold at a gray level in the valley between the two probability peaks, since these are relatively well separated. At high edge values, the peak gray level value itself, or perhaps the mean gray level, should be a good threshold, since this represents the "center" of the edges. For intermediate edge values, one can compromise between these two thresholds in various ways. A comparative study of threshold selection schemes based on this approach has been conducted [5]. This work will be discussed further in Section 4.

1.2 Operator response prediction

a) Predicting results of thresholding

Thresholding images is a common process and much work has been directed towards selecting the correct value at which to threshold and estimating the expected error. Normally, one thresholds only images which contain some signal. Thresholding pure noise is to be avoided when possible. Since there may be occasions when thresholding noise is unavoidable (e.g., a poor threshold was chosen), it is important to predict the expected results. The expected number of above-threshold regions that result when noise is thresholded is useful in planning for data structure storage allocation and in predicting false alarm rates. When a bad threshold "breaks up" an object, knowledge of the expected sizes and shapes of noise regions can be used to help discriminate object fragments from noise. No methods currently exist for predicting the number of connected components of thresholded spatially correlated signal (or noise). However, it has been found possible [6] to estimate the moments of regions, the density of border points, and lower bounds on the number of connected components in thresholded noise images. The input grayscale image is modeled as a two-dimensional random process (stationary random field) characterized by its mean and power spectrum. Tests with both synthetic data (smoothed noise) and actual data were conducted to compare the predicted and measured responses. The predictions are worst for thresholds at or near the mode of the noise distribution, but in general, the comparison showed reasonable agreement between the

predicted and measured values.

b) Predicting edge detector response

Statistical response prediction for edge operators can be used to determine the nature of further processing of the response. If edge detector output is to be thinned or thresholded, the false alarm and false dismissal rates depend on the statistics of the operator responses. A study has been conducted [7] which discusses the statistical properties of the outputs of some edge detectors operating on a general class of images.

The image model considered is the same as in (a) just above; this model is appropriate for predicting the response of edge detectors to background noise. The edge detectors analyzed are the Laplacian and its absolute value, and the absolute difference of averages over adjacent 2x2 and 4x4 neighborhoods. The response features which were measured are the mean edge response at each point, the variance, the auto-covariance, and the cross-covariance of gray level and edge value at a number of displacements. In addition, the density of the local maxima of edge values was computed. Tests using a set of synthetic background images showed good conformity to the predicted features.

1.3 Edge modeling

a) Optimal edge detection

Many optimality criteria have been proposed for edge detection. Among the most well known is that devised by Hueckel [8,9]. It involves fitting an ideal parameterized step edge to the image data so as to minimize the mean squared error. A new optimal detector has been designed [10] that simplifies several assumptions associated with the Hueckel detector and thereby solves an easier optimization problem. Specifically, by assuming that the local mean is zero and the local variance is unity, two Hueckel parameters can be eliminated. Further simplifications follow if the operator can be applied at each point with the edge assumed to pass through the center (or not to exist at all). The resulting formulation can be tuned to favor edges with known a priori probabilities. The computational effort involved in applying the operator may be reduced by solving the associated cubic equation using a simple iterative approximation, such as Newton's formula. Testing on actual data has verified that this approach provides

greater sensitivity than a previously proposed [11] simplification of the Hueckel operator.

b) Evaluation of edge operators

The Hueckel operator defined in [9] has been found to incorporate a theoretical flaw leading to eccentric behavior in textured images. An operator which is conceptually similar but apparently more dependable has been defined [12]. Comparative tests have been made of this and several other operators (including Hueckel's [9], its simplification [11], and the new optimal operator defined in [10], as well as the very simple Sobel operator), to evaluate their adequacy in obtaining the magnitude, direction, and reliability of the edge response at some set of image points, for both ideal and distorted images. The performances of these operators were, in general, closely related to their sizes (and hence, to their computational costs). All of the local operators were able to detect the directions of distorted edges on small (6x6) domains to an accuracy of about 10°, and their magnitudes to within about 10%. On larger (9x9) domains, angular resolution was improved, but ramps became significant as a source of spurious responses. The Sobel operator was judged to perform better than the operator of [11]. The operator of [10] was better able to reject ramps on larger domains, but it is more expensive to apply than the other local operators. The regional operators of [9] and [12] performed similarly; the latter was less affected by the presence of imperfections.

2. Preprocessing

Preprocessing refers to those transformations applied to the raw image data for the purpose of correcting, simplifying, and regularizing the imagery. The resulting images should therefore be more amenable to further processing and more alike in certain properties essential to subsequent algorithms. Thus, for example, sampling and windowing reduce the size of the image to be processed. Histogram transformation and requantization convert all image quantization levels to a range which facilitates feature extraction. Smoothing reinforces regional uniformity and decreases the effects of certain kinds of noise.

Preprocessing steps are best justified by the problem environment itself. A knowledge of the sensor characteristics and

geometry will suggest various kinds of radiometric and geometric corrections. For example, with FLIR data, the image is best understood as an array of thermal measurements. If these measurements reflect the ground truth, then much more subtle distinctions can be made; recognizing that a particular temperature is beyond the normal range for a vehicle can, perhaps, indicate that the vehicle is on fire. Similarly, a range map converting pixels to actual size/area measurements can allow a viewer or a program to gauge the size of a particular region and thereby discern its identity more reliably.

In the problem environment at hand it was not possible to acquire substantial information concerning the sensor or the imaging situation, due to classification problems. Thus the "intelligent" corrections of the previous paragraph were impossible. However, several preprocessing steps do make sense. The following subsections describe them.

2.1 Sampling

According to the sampling theorem, the spacing of the data points should be half the size of the smallest feature to be detected. Thus, to detect objects of one meter on a side, pixels should correspond to one half meter on a side. In practice, however, the presence of noise demands that the data be redundant to increase the reliability of the extraction process. A finer resolution can often provide this redundancy. Naturally, the price must be paid in additional processing time. This tradeoff is difficult to model analytically especially since many different features are extracted from an image and their relative importance is difficult to assess.

Two processes for region extraction are paramount for our work -- thresholding for whole region extraction and edge detection for region border verification. Of these two, edge detection is more sensitive to noise. The degree of sampling allowable for the image data set should therefore not be so great that reliable edge extraction is compromised. A 2-to-1 size reduction (eliminating every other row and column) was found to be compatible with reliable edge detection. In the unsampled images, the average edge ramp cross-section was found to be about 5 pixels wide; thus a 2-to-1 reduction gave about a 3-pixel edge ramp

which was consistent with the need to localize edges fairly accurately.

An alternative approach attempted to reduce high frequency noise by extracting windows based on 2x2 averaging rather than sampling. Thus, instead of discarding every other row and column, each pixel in the sampled image was the average over a (disjoint) 2x2 neighborhood in the original image. A smooth, less noisy image was produced and row dropouts were partially eliminated. However, the images seemed to have less constrast. Sampling followed by smoothing appears to be better than smoothing followed by sampling.

A major emphasis in the project has been the detection of small or faint targets. For this reason, the sampled images were also windowed so as to capture the target regions and to further reduce the computational load. Naturally, one must avoid techniques which assume that each window contains exactly one target in its central region. This dilemma asserts itself in subtle ways. Statistical properties of the window, e.g., histogram, central moments, etc. are good predictors of object presence, threshold, etc. However, they cannot be employed in practice unless window size is correctly estimated and window border situations are handled. Our approach does not depend on window size (or frame size) and therefore windowing is an appropriate preprocessing step. Note, however, that estimates of the false alarm rate cannot be reliably derived solely from target windows. For this reason, small noise windows (containing no targets) and large windows (consisting mainly of background clutter) were also processed.

2.2 Histogram transformations and adaptive quantization

Sensor output is related to actual phenomena according to physical laws. If this correspondence is well understood beforehand, it is possible to correct and transform the data to improve subsequent processing. Thus if FLIR data could be used to estimate reliably the temperature of objects, then quite stringent tests could be made to enhance recognition rates. Unfortunately, the analytic interpretation of FLIR data at long range is complicated by many effects such as sun-angle, wind, smoke, surface composition, etc. Furthermore, the sensor hardware itself is subject to

unpredictable electronic noise, disturbances and failures. Only some of these effects can be alleviated and then (due in part to the classified nature of the sensor) only statistically.

Among the conventional gray level modifications considered useful for producing more manageable imagery are the rather simple histogram mapping techniques. Figure 2.1a illustrates the gray level histogram of an unmodified image. The gray level range is defined by eight bits--256 gray levels--and can be seen to exhibit significant non-uniformities of response. Moreover, from a processing point of view, 256 gray levels do not effectively reflect the true gray level range and contrast. A simple 2-bit shift operation, converting 8-bit pixels to 6 bits, has the effect seen in Figure 2.1b of smoothing the histogram while reducing the gray level range to 64 gray levels. This technique if continued for further shifts would ultimately combine significant peaks corresponding to object/background contrast. However, the conversion from 8-bit to 6-bit was found to be justified, as it alleviated non-uniform sensor response without destroying target discriminability.

If one assumes that a scene consists of the juxtaposition of objects of uniform temperature taken from a small number of such temperatures, then it is possible to convert the image into one with only a few different gray levels present. An attempt at adaptive requantization is described in [13]. Briefly, an iterative process constructs a new histogram from the previous version by identifying gray level peaks and having them gain strength (i.e., points) from neighboring non-peaks while the non-peak areas are thereby depleted. The result is a mapping from the original gray level domain to a new sparse set of gray levels. The resulting quantized images (Figure 2.2) seem not to have lost object/background discriminability.

2.3 Image smoothing

In the previous section, preprocessing steps were described which contributed to the interpretation of a scene as a mosaic of uniform sensor responses. The techniques considered the gray level population only. Proximity was not involved. In this section, we discuss attempts to smooth the image spatially so that nearby points from the same region will have more nearly identical gray

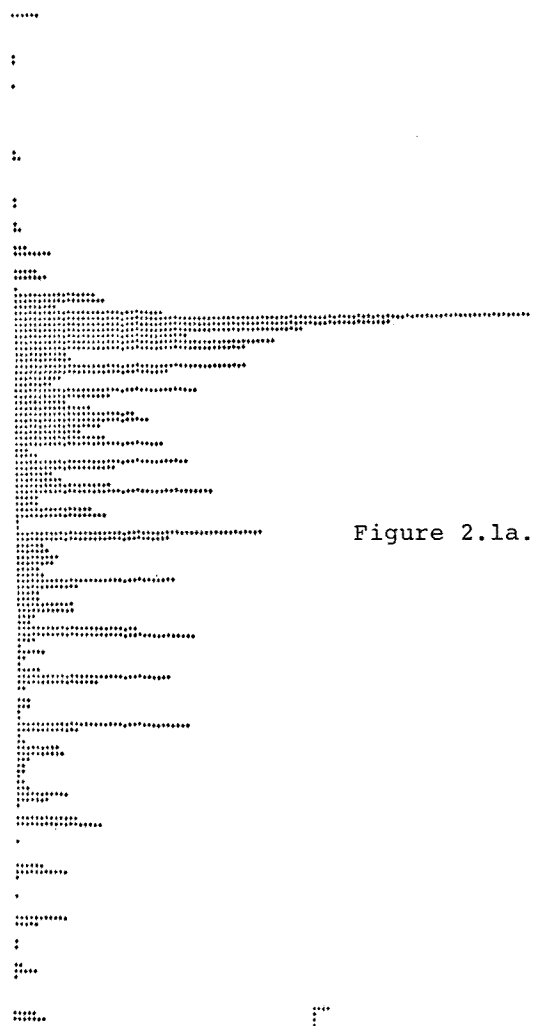


Figure 2.1a. 256-level histogram.

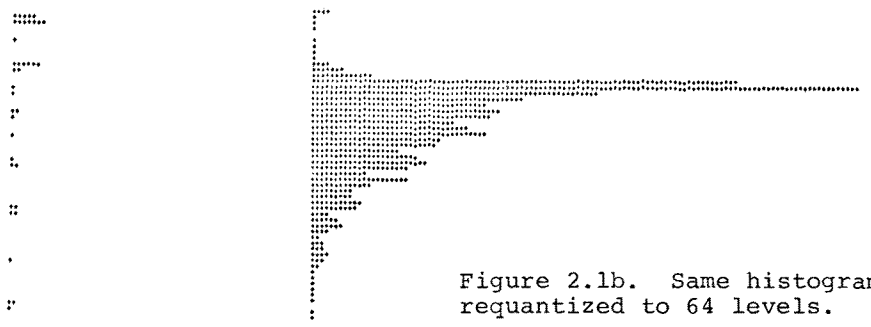
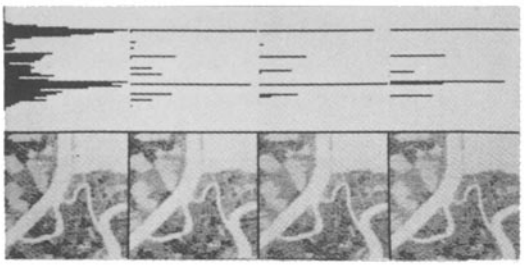
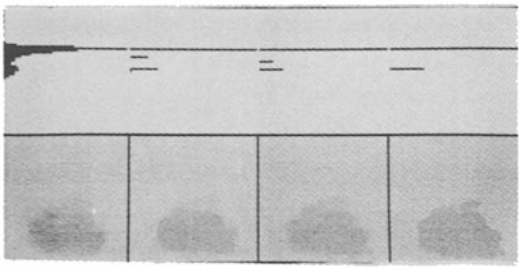


Figure 2.1b. Same histogram requantized to 64 levels.

a.



b.



c.

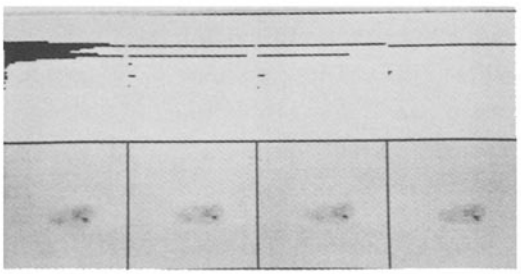
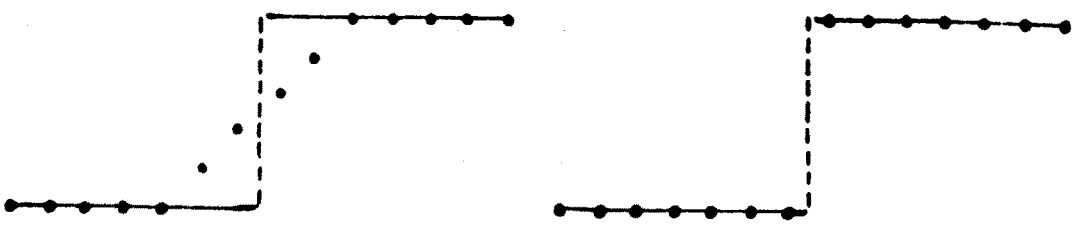


Figure 2.2. Result of four iterations of the peak sharpening process using neighborhood sizes of 2, 3, and 4 for (a-b), sizes 2, 4, and 5 for (c).

levels. There are a number of justifications for spatial image smoothing. First, by making the image spatially more uniform, one increases the probability that points belonging to the same region will be treated identically. Thus, the point sets extracted by thresholding will appear better defined with fewer pinholes and fewer isolated points. This is reasonable since the chosen image resolution is intended to cover any object with numbers of pixels. The second reason for smoothing is to eliminate insignificant local changes of contrast. Otherwise the output of edge detection operations based on differencing would contain many tiny spurious edges which tend to obscure the proper edge signals. Third, the statistical properties of a smoothed image are more representative of the true situation. Thus, many decisions based on the statistics of the smoothed image are more reliable.

Two methods of image smoothing were investigated. In a first attempt, the mean value of a fixed neighborhood about each point replaced the point's value. Figure 2.3a shows the effect of replacing each point of a step edge by its mean value (blurring). As is evident, blurring smears edges. Figure 2.4a-d illustrates blurring for several target windows, and also shows the histograms of these windows before and after blurring. Note that blurring tends to blend peaks in histograms, thus making thresholding more difficult. Also, small faint objects tend to become less distinct.

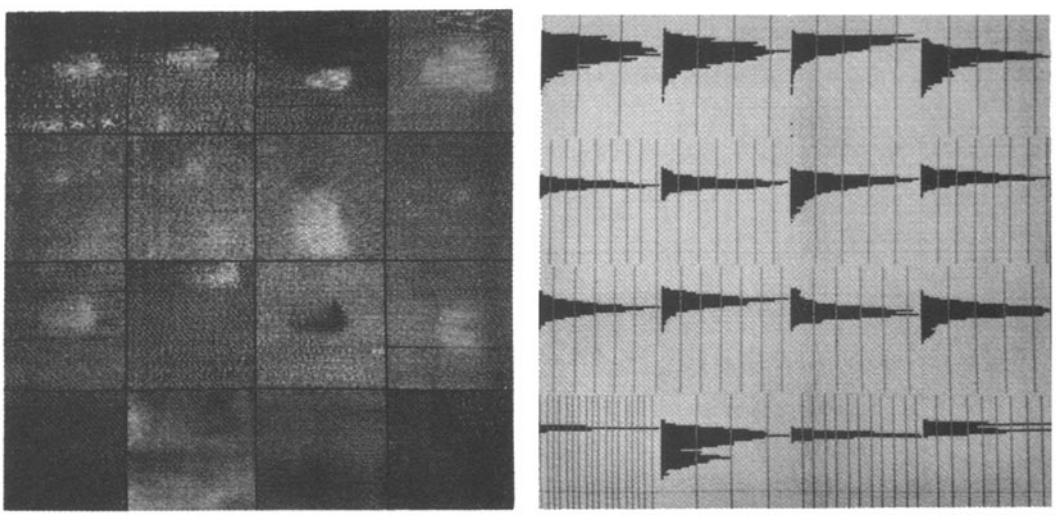
A second approach to image smoothing has the property of preserving edges. At each point of an image, the median value of the gray levels over a kxk neighborhood is computed. The value of k depends on the amount of local noise variation. For the original images, a 5x5 neighborhood size was chosen. Figure 2.3b illustrates the effects of median filtering on a step edge. Note that the median does not increase the ramp width. Thus edges do not smear. This is demonstrated in the two-dimensional case in Figures 2.5-2.7 for a tank image. Median filtering does, however, round off sharp corners. This was not a serious problem in this data base. Figure 2.4e-f illustrates a number of median filtered windows and their histograms. The general algorithm for median computation over k^2 points is of order k^2. However, better results may be obtained when evaluating a running median, by making use of the high autocorrelation of gray level in most images. The cumulative histogram of the k^2 data points is



a. Mean filtering

b. Median filtering

Figure 2.3. Effect of filtering on step edges using a five point neighborhood.



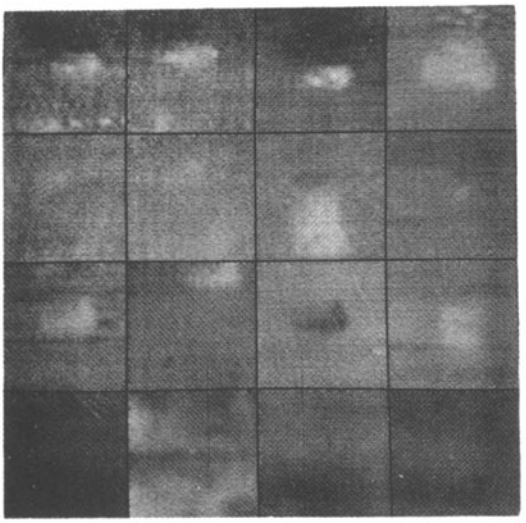
a. Originals.

b. Histograms of (a).

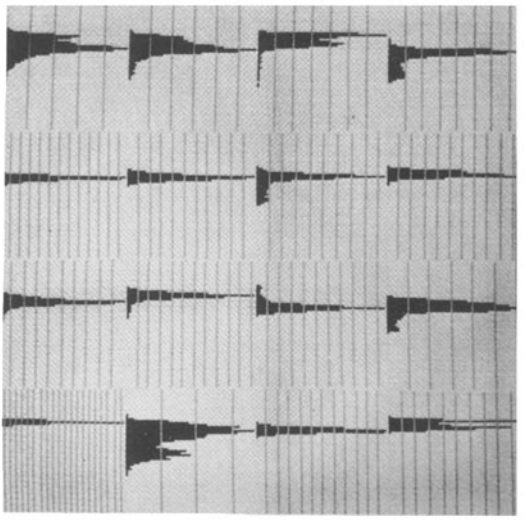
Image Reference:

3R	4T	6T	24T
34R	35R	41R	52R
21A	22A	23A	37A
14N	20N	26N	38N

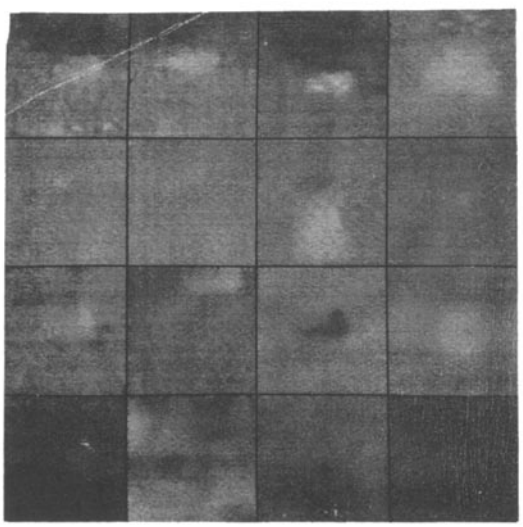
Figure 2.4. Comparison of mean and median filtering.



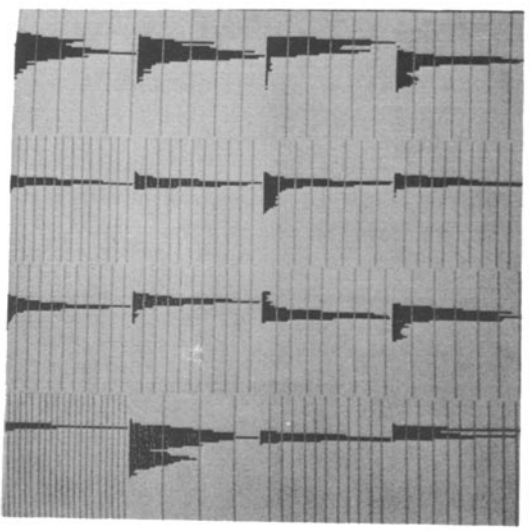
c. 3x3 mean filtered windows.



d. Histograms of (c).



e. 3x3 median filtered windows.



f. Histograms of (e).

Figure 2.4. (continued)

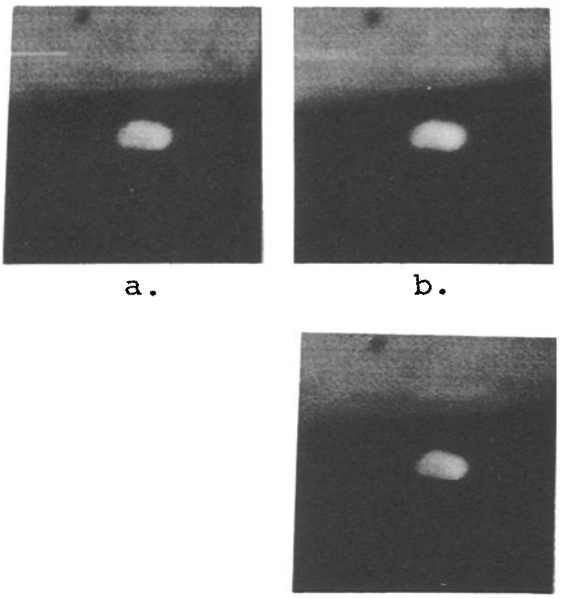
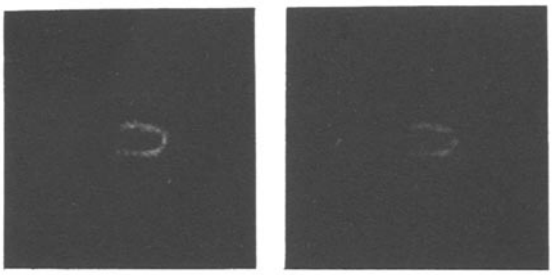


Figure 2.5. Gray level images.

a. Original FLIR image of a tank. Note the noise content and the presence of a thin noise line at the upper left.

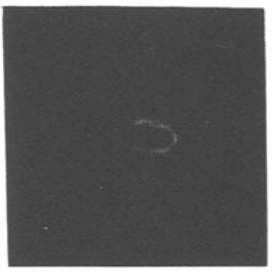
b. Mean filtered image using a 5x5 square neighborhood at each point. The tank appears blurred, as does the border between the road (dark) and the grass (light). The thin noise line is smeared into the background.

c. Median filtered image using a 5x5 square neighborhood. The tank contours appear sharper, while overall the image has been smoothed.



a.

b.



c.

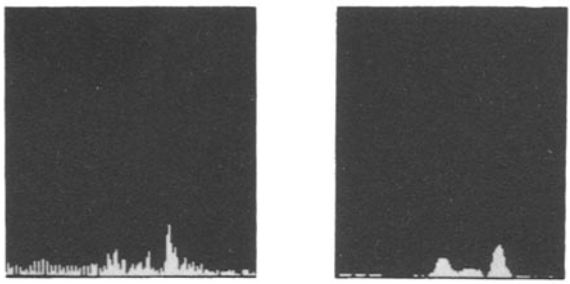
Figure 2.6. Results of Edge Detection

Each of the windows was subjected to an edge detection operation which detects the most significant edge at each point over four orientations. Note that edges surround the various regions in the image but that the edges in the median filtered image (c) are sharper and have more contrast than those in the mean filtered image (b).

a. Edge detection response for the original image.

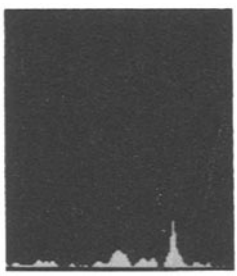
b. Same as above for the mean filtered image.

c. Same as above for the median filtered image.



a.

b.



c.

Figure 2.7. Edge Cross Sections

A single line of the edge detection image passing through the tank is displayed with height corresponding to edge value. Notice that the median filtered response exhibits a thinner, higher peak corresponding to a sharper, more contrasting edge than the mean filtered response.

a. A single line of edge response from Figure 2.6a.

b. Same as above for Figure 2.6b.

c. Same as above for Figure 2.6c.

maintained in a vector of length d (e.g., d=64). The k deletions and k insertions are interleaved in pairs. Each (deletion, insertion) pair isolates a region of the vector which must be modified. The smaller this region on the average, the less work to be done. If the deletion and insertion in a given pair affect the same bin, no change is necessary. The length of the region of change in the cumulative histogram is the expected gray level difference of points at distance k. This corresponds to a variogram value, v(k). After updating, the vector is binary-searched for the median. Thus the number of vector operations is $k \cdot v(k)$, followed by log d operations to binary-search the updated vector. The sum $k \cdot v(k)$ + log d should be quite small for relatively smooth images.

3. Edge detection

The extraction of edge features has proved useful in a number of project areas. Section 4 will describe threshold selection methods which utilize edge information. Edges are also used in the critical step of the Superslice algorithm (Section 6). In this section, we discuss the variety of edge operations investigated and a method of thinning edge response so as to locate the apparent edge.

3.1 Comparison of methods

Methods for edge detection abound in the literature (for a survey, see [14]). Some of the simplest methods involve convolutions of templates with an image. A number of these were considered in the current work. These include:

Laplacian: $|e - (a+b+c+d+f+g+h+i)/8|$, where the neighborhood of e is

```
    a b c
    d e f t
    g h i u
      v w
```

Roberts Gradient: $\max\{|a-e|, |b-d|\}$.

Three-by-three: $\max\{|a+b+c-g-h-i|, |a+d+g-c-f-i|\}$

2x2 Difference:

$1/4 * \max\{|d+e+g+h-f-t-i-u|, |b+c+e+f-h-i-v-w|\}$.

(In other words, the value corresponds to the maximum of the differences between 2x2 averages over adjacent pairs of

horizontal and vertical neighborhoods. This scheme extends to diagonals also.

4x4 Difference: This is the same as the 2x2 difference except that averages are taken over 4x4 neighborhoods.

8x8 Difference: The same as the previous except that averages are taken over 8x8 neighborhoods.

Experiments with these operators indicated that the Laplacian (which is a second difference), the Roberts Gradient and the 3x3 Gradient were too sensitive to minute changes in gray level. The differences of averages operators produced better output by virtue of the increased amount of smoothing on each side of the edge. Knowledge that typical edge width in the windows was three pixels suggested that the 4x4 operator could span the edge ramp (to give the maximum gradient value) while remaining sensitive enough to detect the edges of small faint regions.

3.2 Edge thinning

In the world of man-made objects, edges correspond to the juxtaposition of surfaces and shadows. In a well focused image, edges should appear sharp and should extend in some direction for some length. (In the natural world, the boundaries of regions are not necessarily as sharply defined, e.g., for trees, fields, etc.) The output of the operators described in the previous section, however, is generally smeared at or near the true edge location. Nonetheless, for certain types of image understanding it is necessary to localize the edge so that it lies along the object boundaries. Given a knowledge of the edge detector, it is possible to design a process which accepts the output of the operator and which produces a thinned representation of the edge at the location of maximum edge response.

It is not sufficient to consider simply those points of maximum response, since this would force adjacent points in the direction of the edge to compete. This problem can be alleviated by taking into account the computed local direction of the edge and by placing into competition only those points which are normal to the direction of the edge. In practice, a directional mask is associated with each edge point oriented normal to the direction of the maximum edge response at that point. The center point is then deleted (assigned zero response) if any point within the

mask has a greater response. There are four masks:

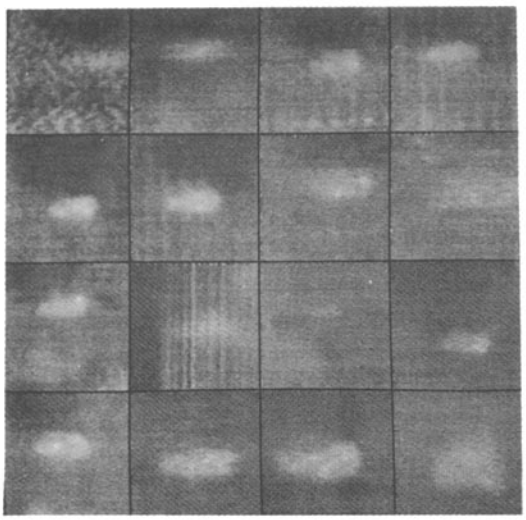
```
                  x x x       x                   x
  x       x         x       x x                   x x
  x x 0 x x   ,     0     ,     0       ,       0       ,
  x       x         x           x x       x x
                    x x x         x         x
```

one associated with each principal edge direction. The process, called "non-maximum suppression," operates simultaneously on all edge values to produce a "thinned" edge map. Figure 3.1 illustrates the process.

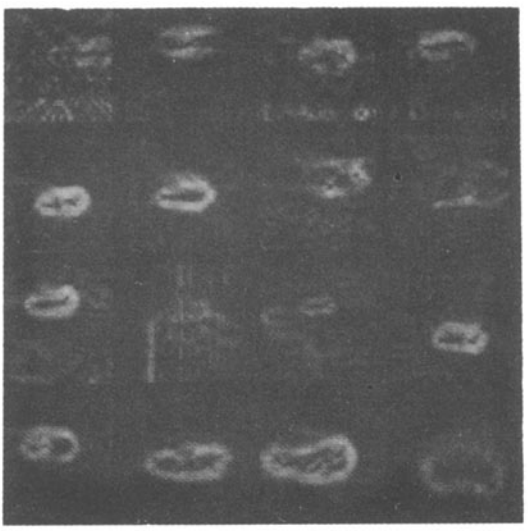
4. Threshold selection

The properties of a pixel in a single sensor image are its position and its gray level. Our knowledge of the imaging environment allows us to predict an object's gray level more accurately than its position. In fact, the whole point of cueing is to locate a target. Thus one has little a priori information about target position; however, inasmuch as gray level is related to thermal emission in FLIR data, there are some fairly powerful heuristics available to aid in target recognition. For example, we may choose to assume that operating vehicles are warmer than the immediate background and that they radiate uniformly over their surfaces. Naturally, such assumptions are not always possible. In cold weather, metal loses heat faster than the ground; at close range, fine thermal detail is visible and the uniformity assumption fails. Nor are these assumptions meant to be exclusive, e.g., we do not claim that every object region warmer than its surround is a target. The power of these heuristics is to suggest approaches which capture essential problem domain knowledge.

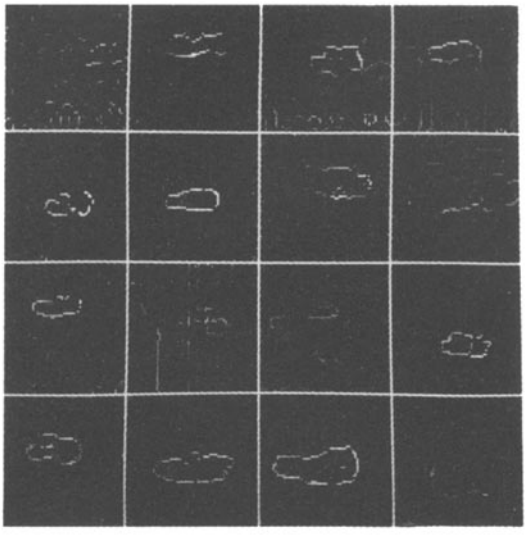
In this project, the force of the heuristics of the previous paragraph is to emphasize methods which isolate distinct gray level regions from their surrounds. The simplest of such methods is thresholding -- the assignment of all points whose gray levels are greater than a predetermined level (the threshold) into a single class of potential object points. The filtering, aggregation and ultimate classification of these points are the subjects of subsequent sections. In this section, we discuss our investigations of numerous methods for single and multiple threshold selection.



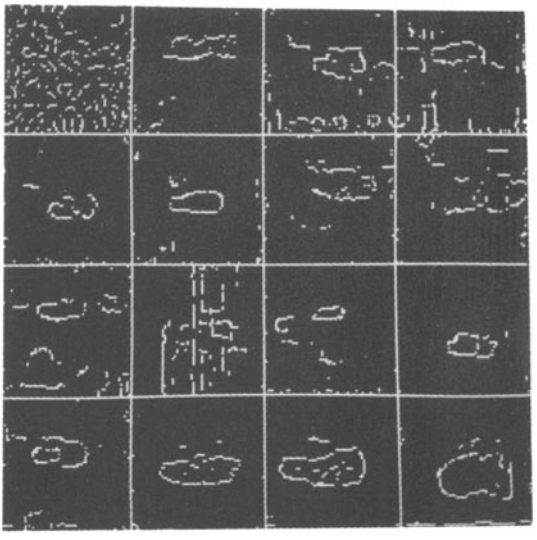
Originals.



Edge detector output.

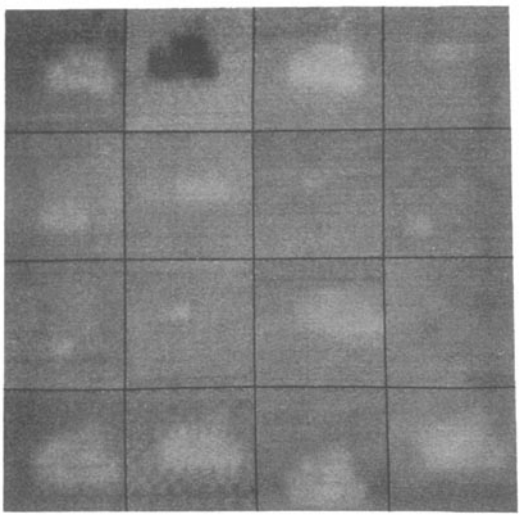


Thinned edge map.

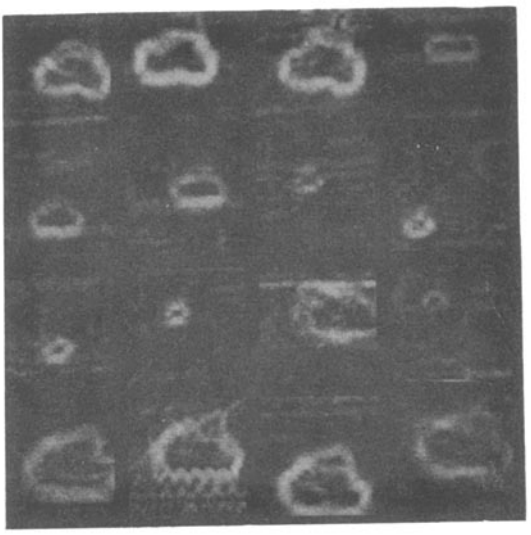


Thinned edge map thresholded to display only edge values > 2.

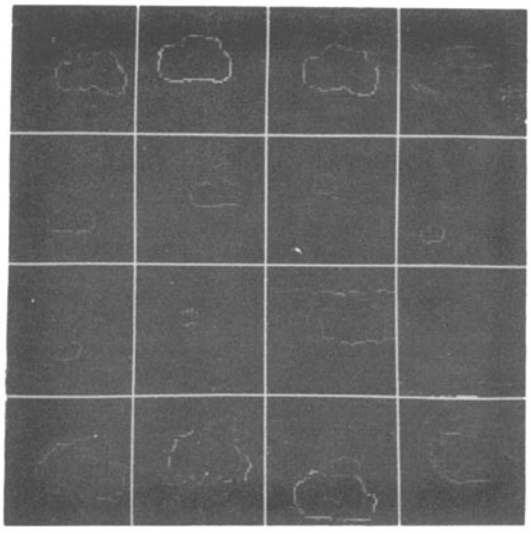
Figure 3.1. Results of edge detection and non-maximum suppression on 43 tank windows.



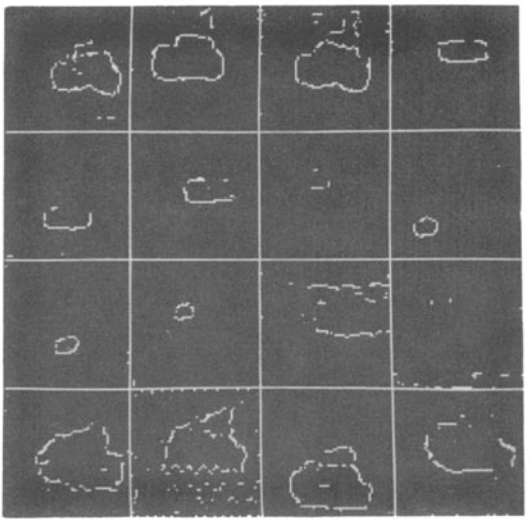
Originals.



Edge detector output.

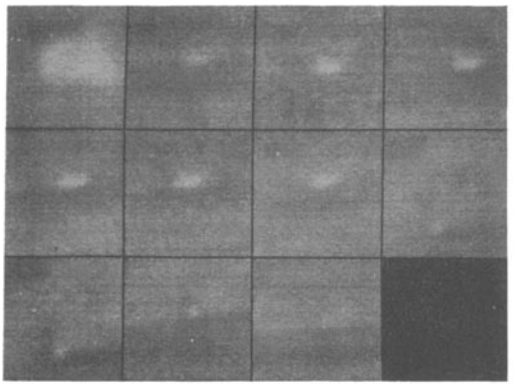


Thinned edge map.

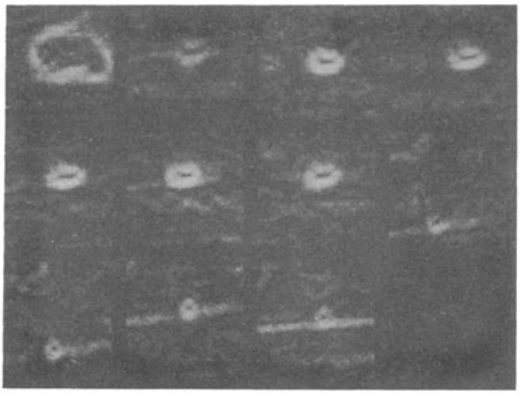


Thinned edge map thresholded to display only edge values > 2.

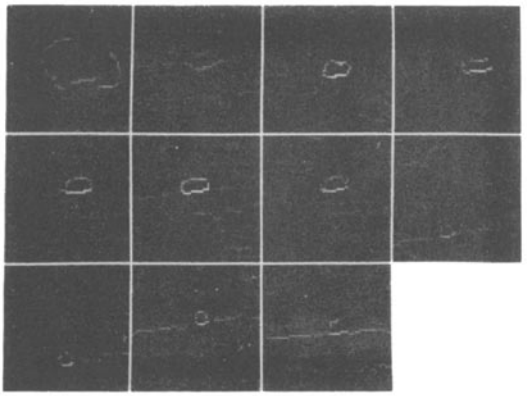
Figure 3.1. (continued)



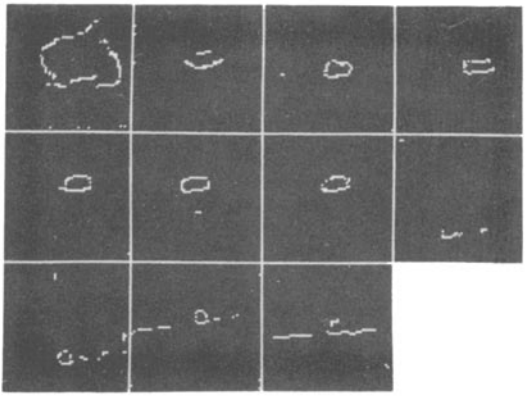
Originals.



Edge detector output.



Thinned edge map.



Thinned edge map thresholded to display only edge values > 2.

Figure 3.1. (continued)

There is a progression in these methods which corresponds on the one hand to the need for increased sensitivity in choosing the "right" threshold and, on the other hand, to the deemphasis of the commitment to that particular threshold. However, we still retain the notion that for each object region in the image there is a "best" threshold. In the worst case, every possible threshold yields a target region which is invisible (unextractable) at every other threshold. One must therefore be prepared to threshold at any gray level and within that thresholded image to discern the target regions and to ignore the noise regions. These last comments appear to call for the selection of every gray level as a value at which to threshold. Indeed, given sufficient parallel hardware and powerful target/noise discrimination criteria, this brute force approach could lead to a reliable and sensitive target cuer. A further discussion of this option is in Section 6 and some relevant experiments are to be found in Section 8. The remainder of this section describes techniques for finding appropriate thresholds when hardware and throughput considerations allow only a few thresholds to be utilized per frame.

4.1 Threshold selection based on edge values

In Section 1.1 a model was proposed for images consisting of objects and background, each with characteristic gray level distributions. If the gray level histogram of the image is markedly bimodal, one may choose the threshold at the valley between the two peaks (possibly shifted towards the smaller peak when using a maximum likelihood estimate). However, the smaller the object, the less likely the histogram is to exhibit strong bimodality. The background distribution engulfs the object's gray level range and tests for bimodality are inconclusive.

One approach [15] to solving this problem has been to select from the original image a set of points that are as likely to fall within the object as within the background. If one examines the output of operators which respond to edges, then high values should correspond to points falling at or near object edges. The mathematical model has shown the gray level distribution to be unimodal with a peak at the mean. Thus, these points are as likely to lie on the object as on the background and their mean value should correspond to the desired threshold.

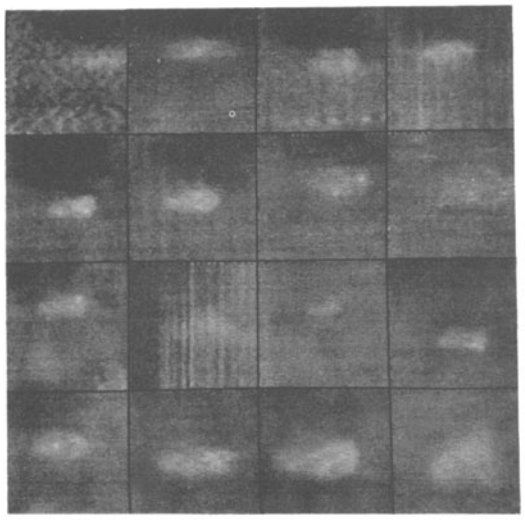
A brief description of the threshold selection method is as follows: Let e(i,j) be the edge value computed at (i,j) and let g(i,j) be its gray value. Then the chosen threshold is $\bar{g} = AVG\{g(i,j) \mid e(i,j) \geq t\}$, where t is lowest edge value considered significant. Computationally, two arrays are needed. One array, $TOTAL_0,\ldots,TOTAL_{63}$, accumulates the gray level g for each edge value e between 0 and 63; i.e., $TOTAL_e = TOTAL_e + g$. The second array, $N_0,\ldots,N_{63}$, tallies the number of points at each edge value. The desired average gray level $\bar{g} = (\sum_{i \geq t} TOTAL_i)/(\sum_{i \geq T} N_i)$.

Two parameters were treated in the experimental work: the choice of edge operator and the edge significance level t. Previous work with edge operators indicated that the 4x4 difference of averages operator was superior to the others as an edge detector in FLIR scenes. Experiments in threshold selection showed that thresholds chosen based on this operator were better overall [1].

The proper selection of a value for t is important because this parameter controls the size and quality of the sample points of high edge value used to compute the gray level threshold. Setting t too high decreases the statistical reliability of the sample; while a small t may admit too many noise values. The choice of t depends on the expected amount of object edge. Obviously, many assumptions are built into this notion, e.g., that the window contains only a single object of known size, shape, contrast, resolution, etc. In a tactical situation, one could make estimates of these parameters based on situation data. Based on estimates of target size, t was chosen as the edge value corresponding to the 95^{th} percentile. This estimate was shown by experiment to provide good thresholds for the windowed data set. This is illustrated in Figure 4.1.

The sensitivity of the chosen gray level threshold to different choices of t was tested and a graph of the threshold was plotted as a function of the gradient cutoff t; see Figure 4.2. There is a tendency for this graph to drift toward the mean gray value as t is decreased. The chosen threshold is stable for large objects. For small objects, the choice is quite sensitive to the bin size.

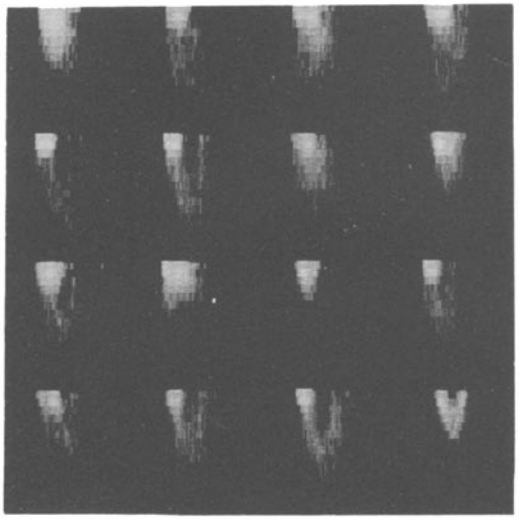
The approach above and several variations [5] can be viewed as methods of decision surface selection in (gray level, edge value)

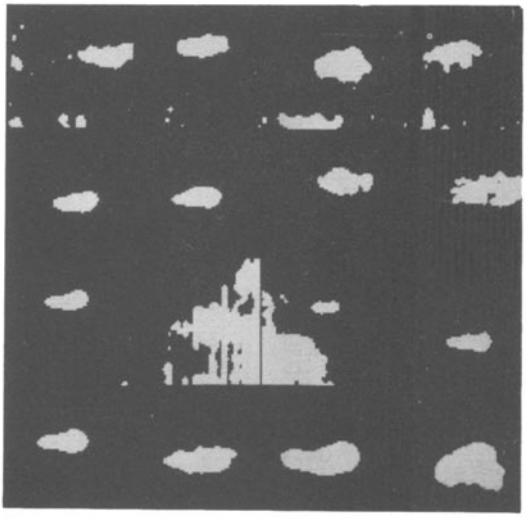


1T	2T	3T	4T
6T	8T	9T	10T
11T	12T	13T	14T
15T	16T	17T	21T

Originals.

Image reference numbers.

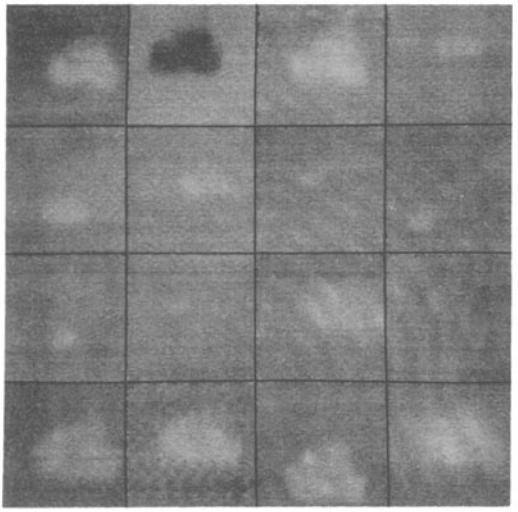




2-D Histograms.

Thresholded windows after shrink/expand (see Section 5.1).

Figure 4.1. Results of thresholding and post-processing 43 tank windows.

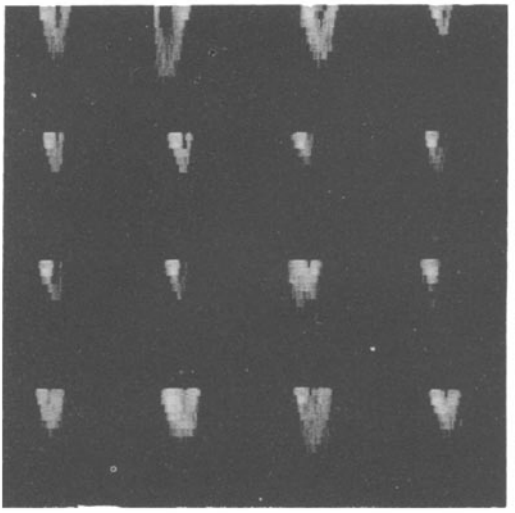


22T 23T 24T 26T

28T 31T 32T 33T

34T 35T 38T 40T

42T 43T 45T 46T



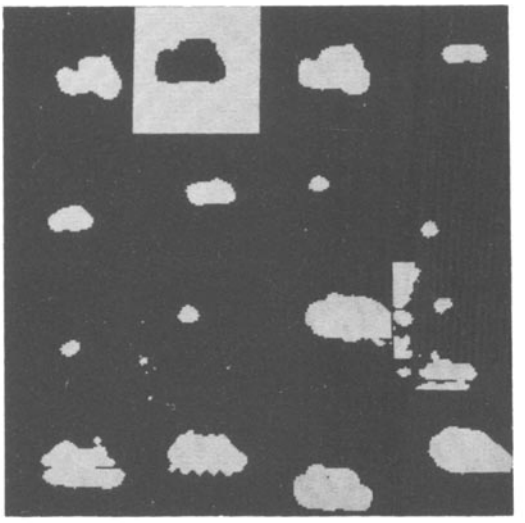
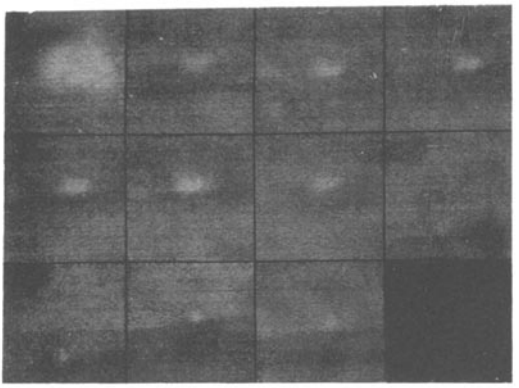


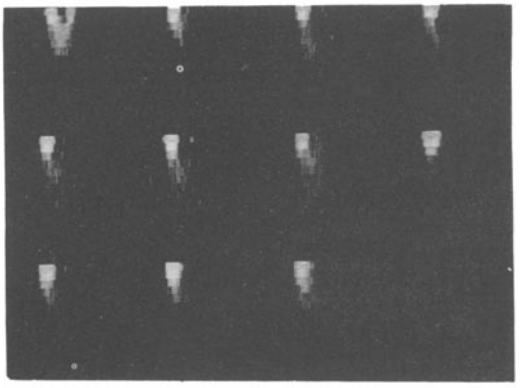
Figure 4.1. (continued)



48T 50T 51T 52T

53T 54T 55T 56T

57T 58T 59T



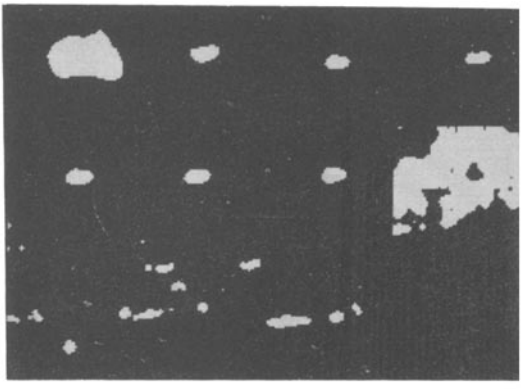


Figure 4.1. (continued)

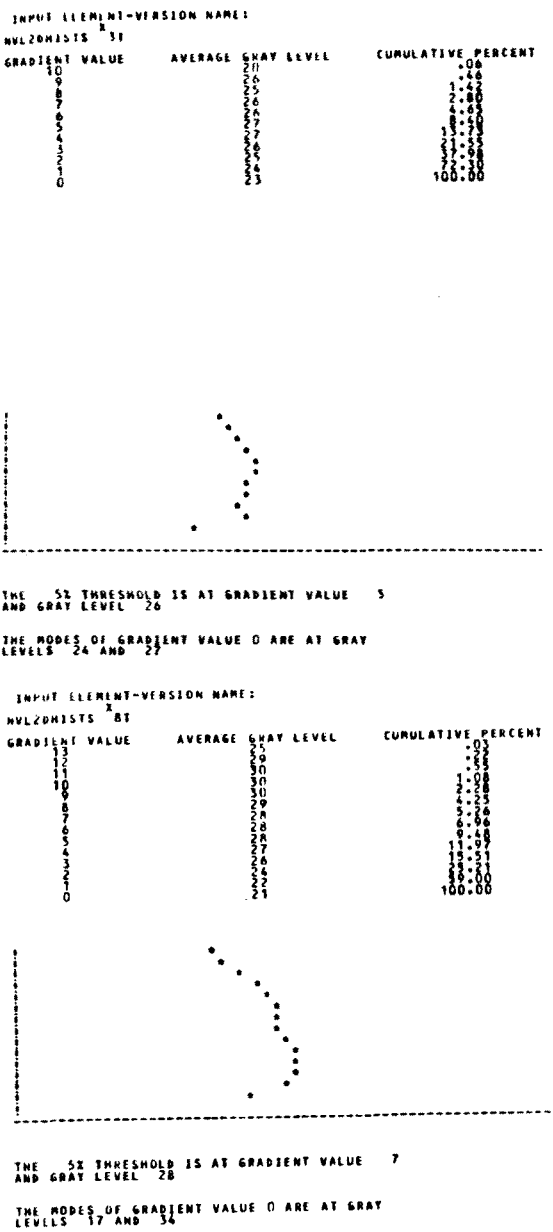
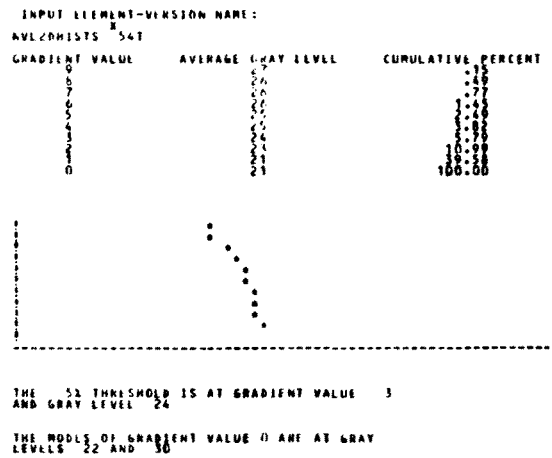
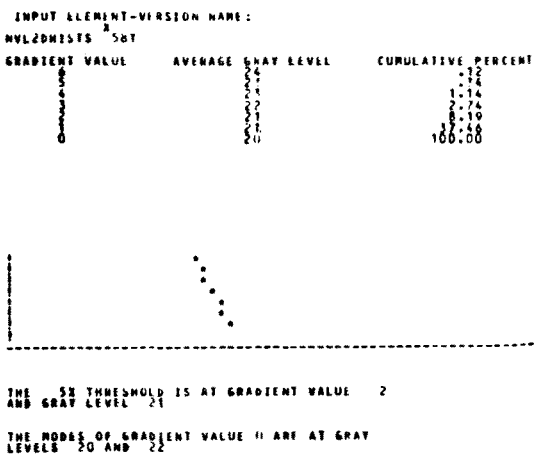
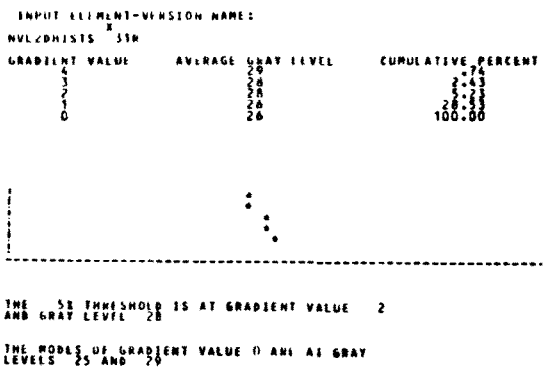


Figure 4.2. Graph of selected threshold as a function of percentage edge value cutoff. Abscissa: gray value increases to the right; ordinate: gradient decreases up the axis.







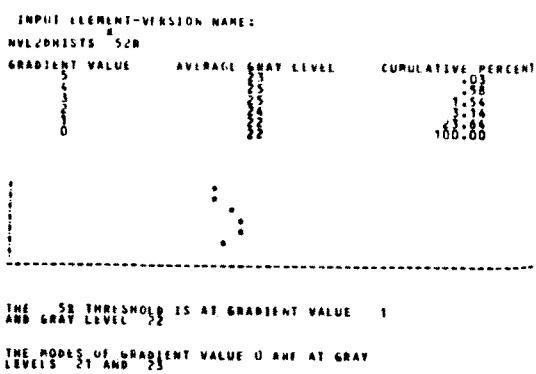
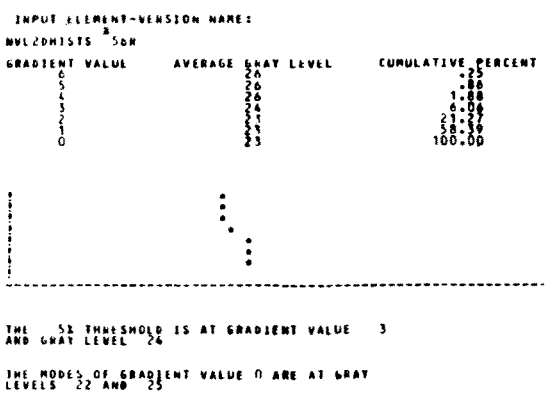
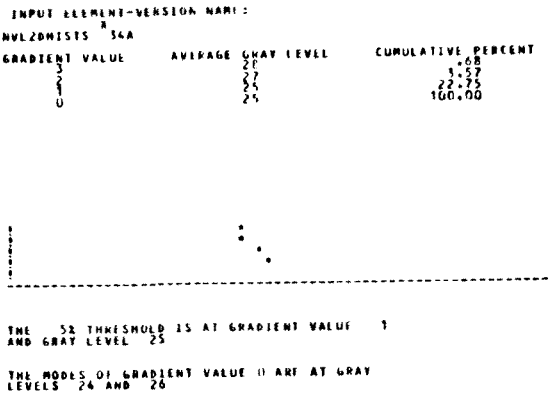
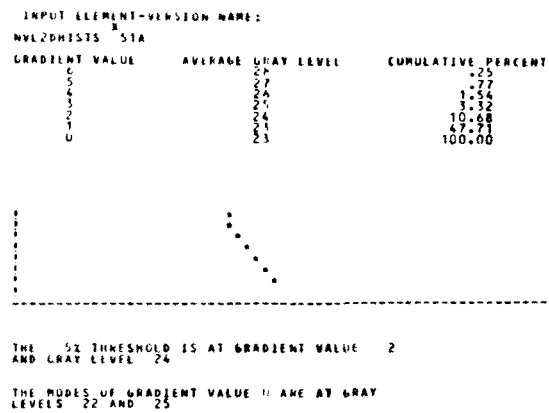


Figure 4.2. (continued)







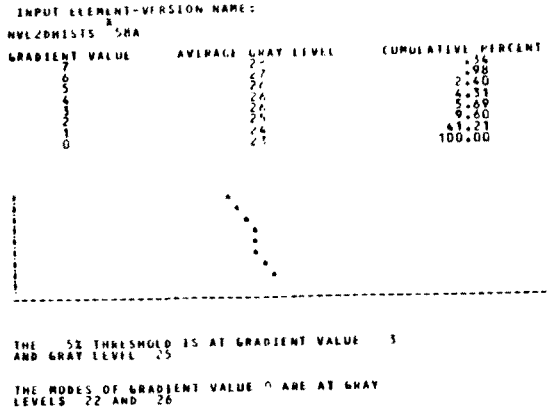
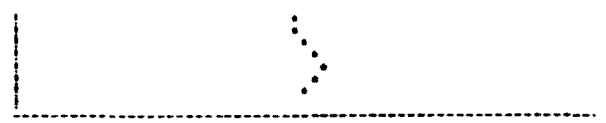


Figure 4.2. (continued)

```
 INPUT ELEMENT-VERSION NAME:
                X
NVL2DHISTS  2N

GRADIENT VALUE      AVERAGE GRAY LEVEL      CUMULATIVE PERCENT
      6                    29                      .03
      5                    30                      .18
      4                    31                      .94
      3                    30                     3.57
      2                    29                    15.05
      1                    28                    57.88
      0                    28                   100.00
```



```
THE   5% THRESHOLD IS AT GRADIENT VALUE    2
AND GRAY LEVEL   28

THE MODES OF GRADIENT VALUE 0 ARE AT GRAY
LEVELS   27 AND   30
 INPUT ELEMENT-VERSION NAME:
                X
NVL2DHISTS  2GN

GRADIENT VALUE      AVERAGE GRAY LEVEL      CUMULATIVE PERCENT
      3                    21                      .06
      2                    22                     4.56
      1                    23                    46.44
      0                    22                   100.00
```

```
THE   5% THRESHOLD IS AT GRADIENT VALUE    1
AND GRAY LEVEL   22

THE MODES OF GRADIENT VALUE 0 ARE AT GRAY
LEVELS   21 AND   24
 INPUT ELEMENT-VERSION NAME:
                X
NVL2DHISTS  3GN

GRADIENT VALUE      AVERAGE GRAY LEVEL      CUMULATIVE PERCENT
      2                    18                     2.25
      1                    19                    10.62
      0                    19                   100.00
```

```
THE   5% THRESHOLD IS AT GRADIENT VALUE    1
AND GRAY LEVEL   19

THE MODES OF GRADIENT VALUE 0 ARE AT GRAY
LEVELS   16 AND   21
```

Figure 4.2. (continued)

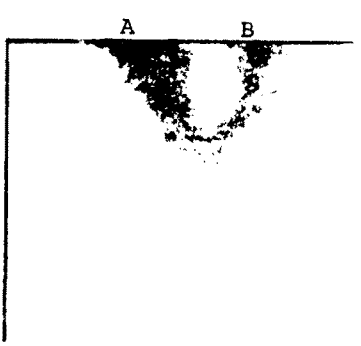


Figure 4.3. Ideal 2-D histogram of a scene containing an object and background with noise.

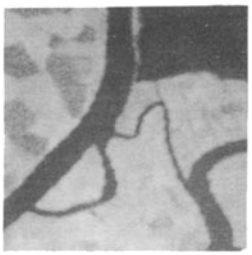
space. This space is visualized as a two dimensional histogram with gray level along one axis and edge value along the other. Figure 4.3 displays such a 2-D histogram for a hypothetical object on a background. Points at A represent background while object points (perhaps with some noise) cluster at B. The bottom part of the U-shaped region contains high-edge value points. As we have pointed out, the average gray level of these points is a good threshold. Figure 4.1 illustrated 2-D histograms for several target windows.

One may consider a threshold as a vertical decision surface separating object from background. Non-vertical partitions of the space have also been investigated [2] and were found to be capable of adding more points to the boundaries of object regions without substantially increasing the amount of noise. Several other partitioning schemes which were considered are discussed in [5].

4.2 Slice range selection

The methods discussed above predict a single threshold to be applied to an image. For images known to contain a single object class or for small windows, the use of a single threshold is appropriate. However, in general, no single threshold will separate all objects of interest from the background. It is therefore necessary to extend our threshold selection concept to allow the choice of multiple thresholds.

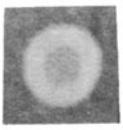
Our approach is to produce clusters of points corresponding to region borders and to associate the average gray level of each cluster with a threshold for the corresponding region. Edge detectors select at each point the maximum difference of averages of adjacent neighborhoods over several directions. By suppressing non-maximum responses normal to the selected direction (i.e., across the edge), thin contours result which appear to surround object regions (see Section 6.2). A by-product of this process are points with very low edge value, including values which truncate to zero. Such points correspond to the interiors of homogeneous regions. Figure 4.4 illustrates thinned detector responses with region interior maxima included. After thinning, each remaining point is plotted using edge value and average gray level in a two-dimensional histogram. Figure 4.5 shows examples of images together with their 2-D histograms based on thinned edges.



a.

b.

Figure 4.4a. LANDSAT window of Monterey, CA.
b. Thinned edge detector response (thresholded).

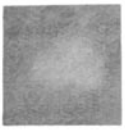


a.

b.

c.

d.



e.

f.

g.

h.

Figure 4.5a. Disk (gray level 30) within ring (gray level 40) within background (gray level 20).
b. 2-D histogram of (a) with gray level as x-axis (increasing left to right) and edge value as y-axis (stretched -- increasing from top to bottom). Interior of background is leftmost, topmost cluster.
c. Window containing house.
d. 2-D histogram of (c).
e. Window containing tank.
f. 2-D (stretched) histogram of (e).
g. LANDSAT window of Monterey.
h. 2-D histogram (thinned edge vs. average gray level).

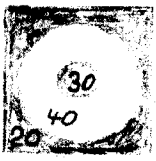
Two types of clusters are produced: interior clusters represent the interiors of regions, edge clusters represent boundaries between regions. The size of a cluster (i.e., the number of points in it) is closely related to properties of the region it describes. Thus interior clusters relate both to the area of the region and to the size of the neighborhood over which the local operations (edge detection, non-maximum suppression) are defined. For small object regions, there may be no points sufficiently far from the object boundary to resist suppression. Thus, interior clusters may be indistinguishable from noise, or may be nonexistent.

Clusters of points at higher edge values are more likely to be significant (based on our homogeneity assumptions). The size of an edge cluster is therefore related to the perimeter of the surrounded region in the image. Since perimeter increases (roughly, for digital images) as the square root of area, the edge clusters for objects of moderately different areas should, nonetheless, be of comparable size. A priori estimates of size are of use in discriminating true edge clusters from random noise.

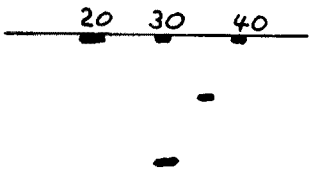
Each edge cluster corresponds (ideally) to these interior clusters whose locations can be determined from the location of the edge cluster. Thus a threshold derived from the edge cluster will separate the interior clusters. However, care must be taken not to split an interior cluster at a threshold since this introduces random noise regions. Figure 4.6 illustrates a compound decision surface in the 2-D histogram of a multi-object image. For further discussion see [16].

4.3 Variable thresholding

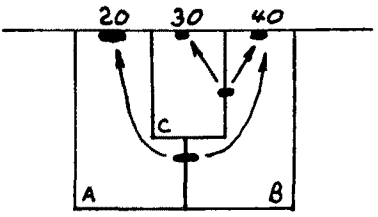
Previous approaches to thresholding apply the same threshold to all points of the image. In [17], Nakagawa adapts the work of Chow and Kaneko [18] to interpolate a best threshold for each point of the image. Briefly, the image is divided into small windows (say 32x32) and a test of gray level histogram bimodality is made for each window. A best threshold is chosen for each bimodal window and thresholds are interpolated to all image points. A binary image results when each threshold is applied to its corresponding pixel value. Figure 4.7 compares fixed thresholding and variable thresholding for several FLIR frames. Nakagawa



a.

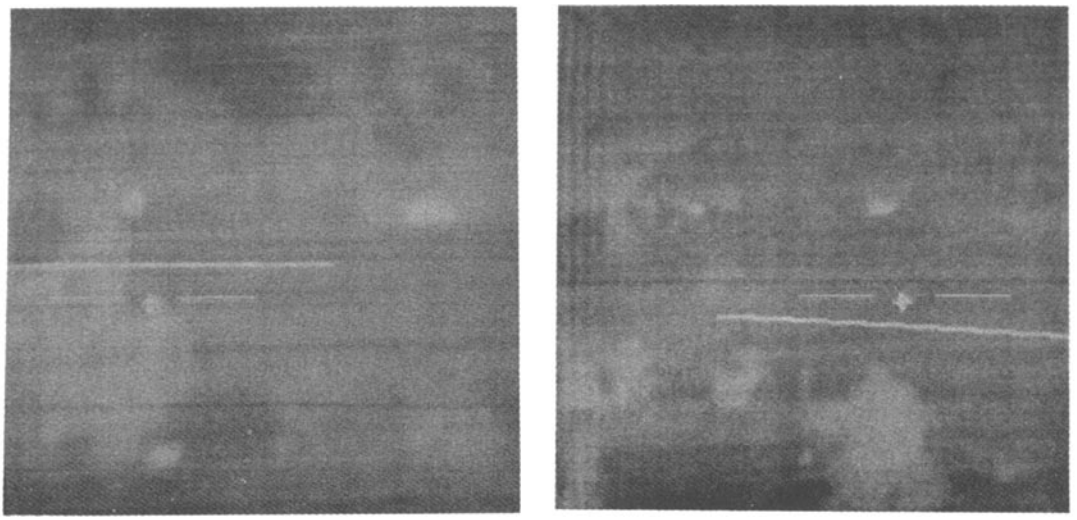


b.

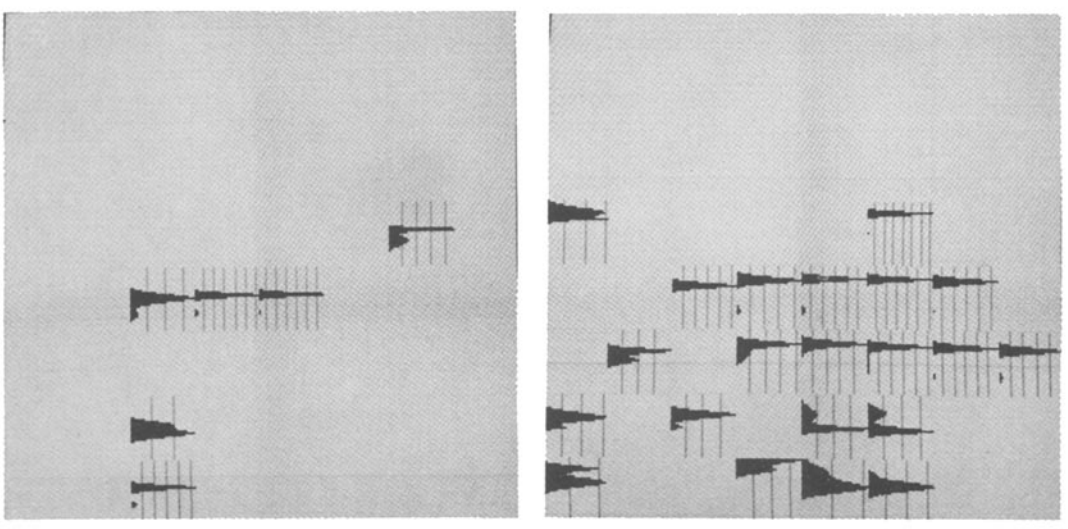


c.

Figure 4.6a. Adjacent object regions on background (same as Figure 4.5a).
b. 2-D histogram.
c. 2-D histogram partitioned into classification regions.



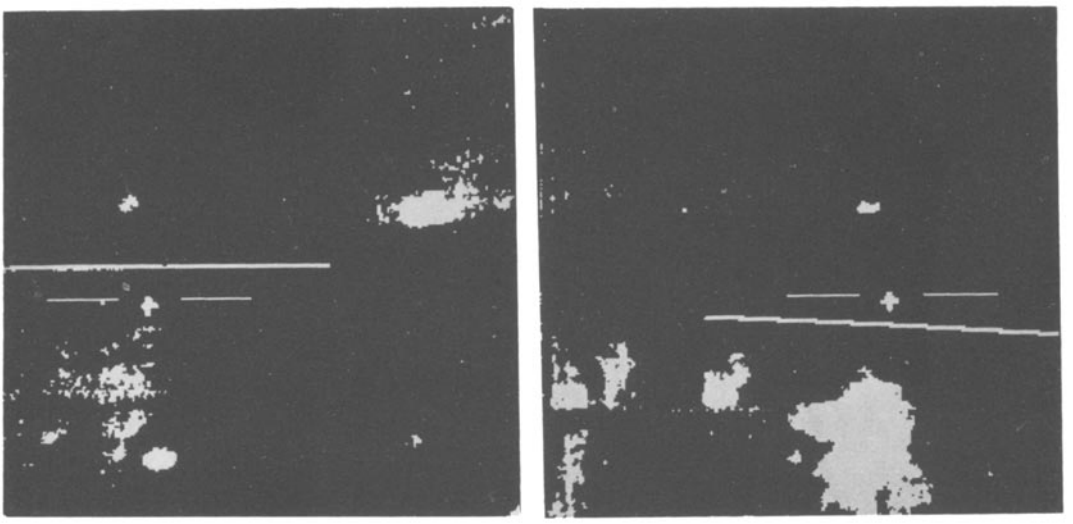
a.



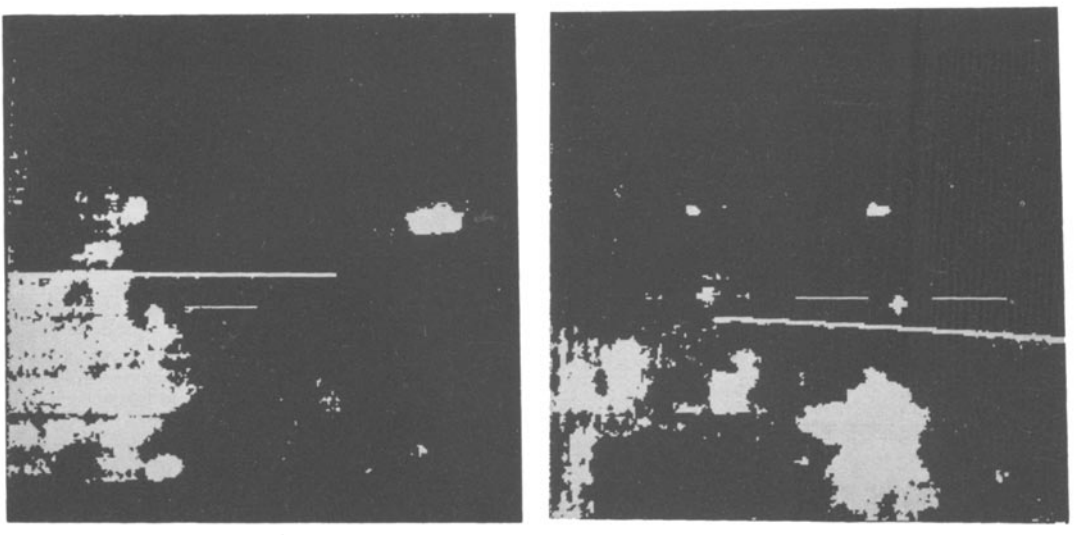
b.

Figure 4.7. Comparison of fixed and variable thresholding.

a. Two FLIR frames.
b. Two-Gaussian approximations to those 32x32 window histograms that were judged to be bimodal.



c.



d.

Figure 4.7 (continued)

c. Results of applying the interpolated point thresholds to (a).
d. Results of applying a fixed threshold to (a).

extended this method to allow multiple thresholds (for multi-object adjacencies). Figure 4.8 illustrates the results.

5. Noise cleaning and component labeling

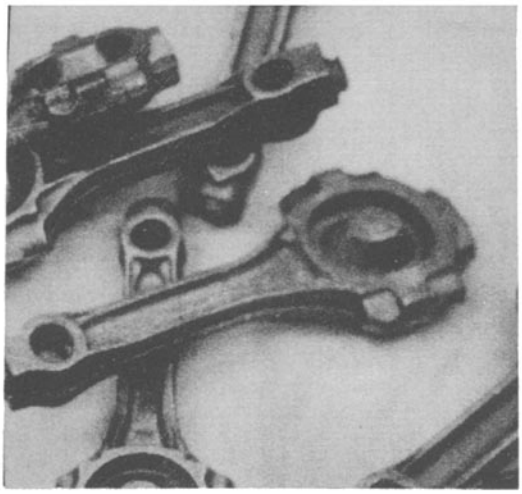
5.1 Shrink/expand and min/max

The result of thresholding is a binary valued image. It often contains isolated points and small noise regions which are artifacts of the thresholding and may not be readily visible in the original image. Smoothed images tend to have fewer (but larger) noise regions. One may delete noise regions by postprocessing the thresholded image.

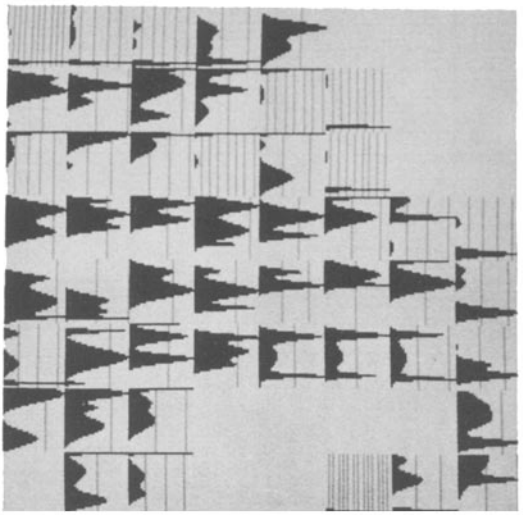
The method consists of multiple applications of two processes: "shrink" and "expand." The purpose of the sequence of shrinks is to shrink objects in a uniform manner so that small or insubstantial objects disappear entirely. The sequence of expands is meant to regrow the remaining shrunken objects to their original size. The result of the shrinks/expands is the elimination of tiny regions (presumed to be noise regions).

Each shrink or expand requires the simultaneous or "parallel" application of a local replacement rule at every point of the thresholded image. The form of the shrink rule is as follows: Eliminate all 1's adjacent to 0's. Zero values are unchanged. Such a rule decreases the number of 1's in the thresholded image; thus, the image "shrinks." Only 1's surrounded by 1's will survive a shrink. The number of successive shrinks determines the mimimum diameter of a surviving region.

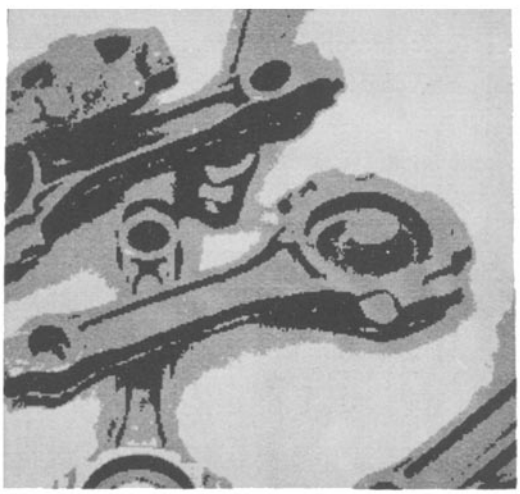
The expand rule is similar to the shrink rule: rewrite a 0 as a 1 if any of its neighbors are 1's, but leave 1's unchanged. Thus points adjacent to 1's become 1's, thereby increasing the number of 1's. If we wish to restore objects (that were not eliminated) to about their original sizes, t shrinks should be followed by t expands. Such a shrink/expand sequence produces an image whose 1's correspond to (a subset of the) 1's in the untransformed binary image. Thus, for example, isolated 1's are eliminated, and objects joined by narrow necks of 1's may become disconnected. Also, thin protusions from a region of 1's will disappear. Figure 5.1 illustrates the shrink/expand algorithm for both the 4 and 8 neighbor cases and t = 1,2,3 (the numbers of



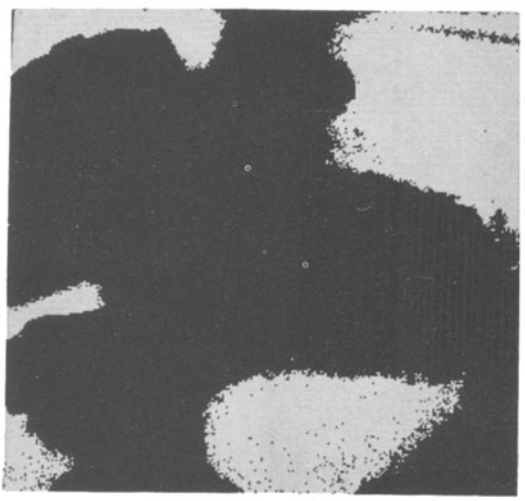
a.



b.



c.



d.

Figure 4.8. Comparison of fixed and variable thresholding.

a. Machine parts image.
b. Two- and three-Gaussian approximations to those window histograms that were judged to be bi- and tri-modal.
c. Three level pictures obtained after interpolating the multiple thresholds determined from (b) and applied to (a).
d. Results of applying a fixed threshold to (a).

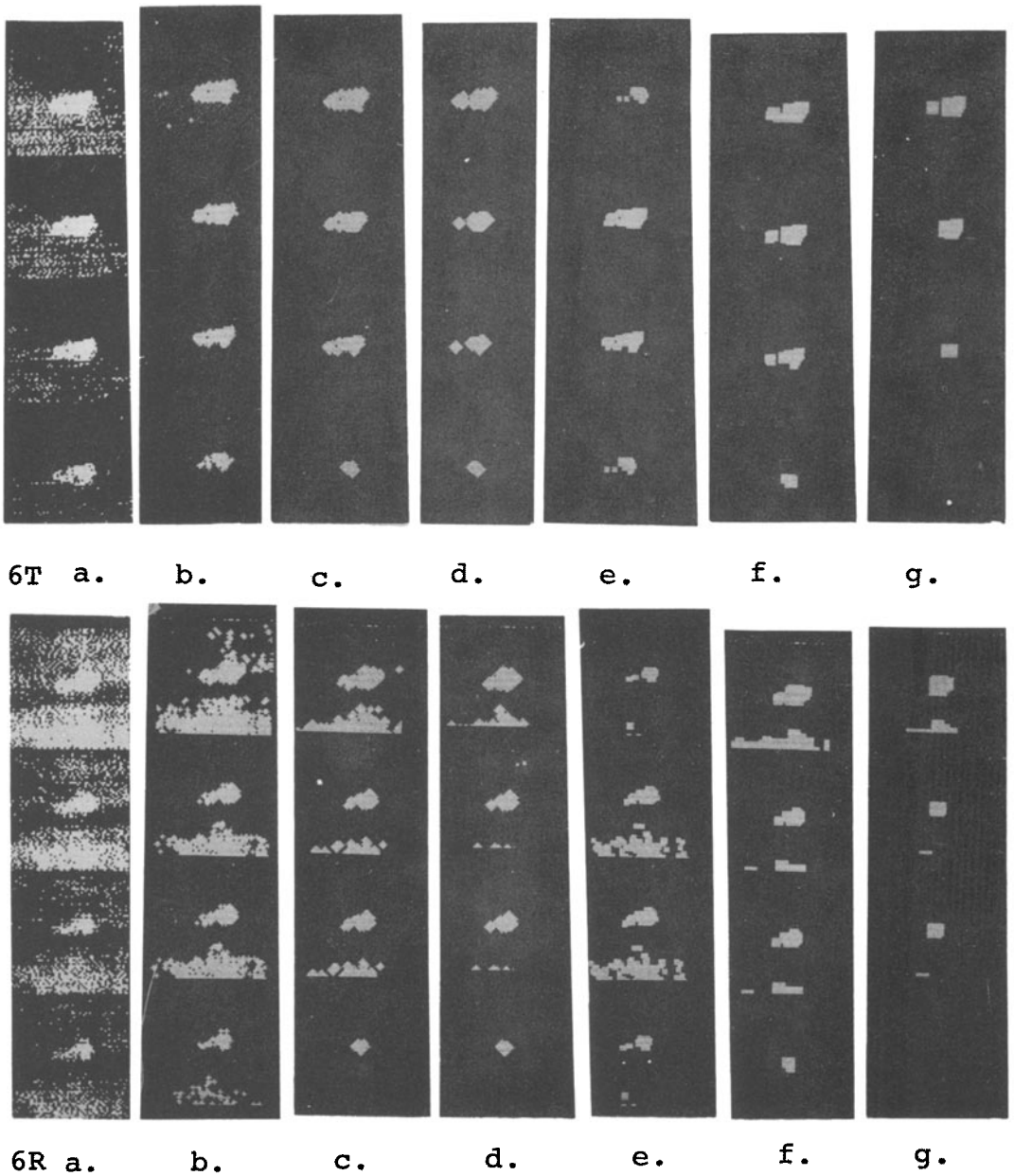
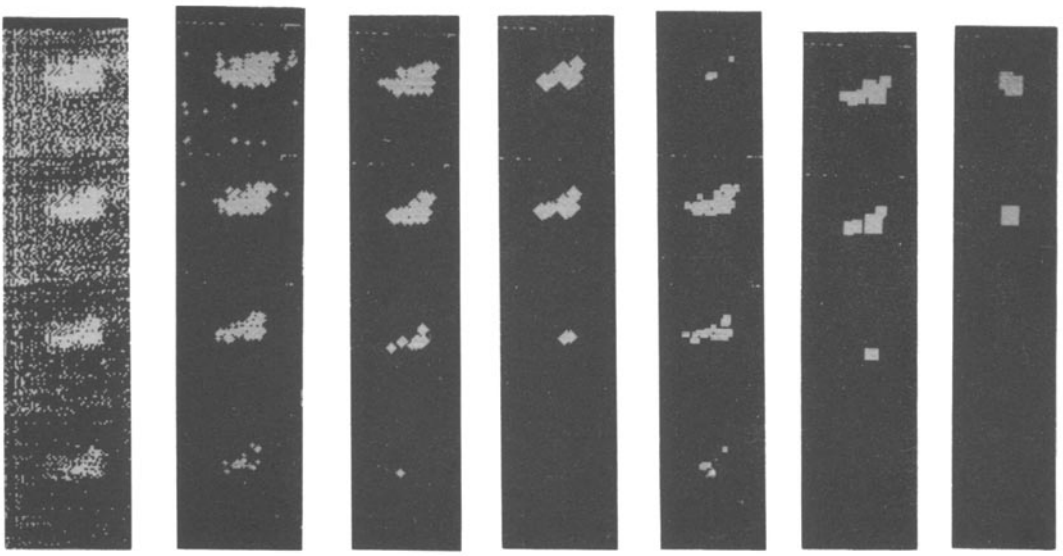
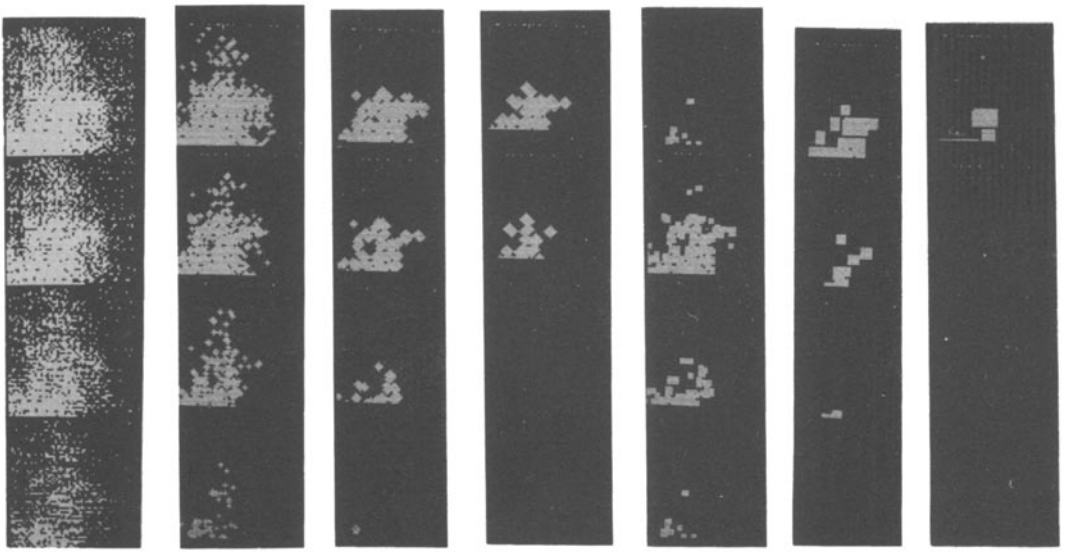


Figure 5.1. Effects of iterating SHRINK/EXPANDS (S/E's).
a. Original images - each column is a single image thresholded at four different values.
b. 4-neighbor rule - one S/E
c. 4-neighbor rule - two S/E's
d. 4-neighbor rule - three S/E's
e. 8-neighbor rule - one S/E
f. 8-neighbor rule - two S/E's
g. 8-neighbor rule - three S/E's



24A a. b. c. d. e. f. g.



8N a. b. c. d. e. f. g.

Figure 5.1. (continued)

shrinks and expands used).

A generalization of the shrink rule was formulated to fill pinholes and conserve small region shape as follows: delete a 1 if at least k of its neighbors are 0's (0's remain unchanged). The original shrink rule corresponds to k = 1. If k > 1, it takes more zero evidence to convert a 1 to 0. The generalized expand is analogously defined: Rewrite a 0 as 1 if it has at least k 1's as neighbors (1's remain unchanged).

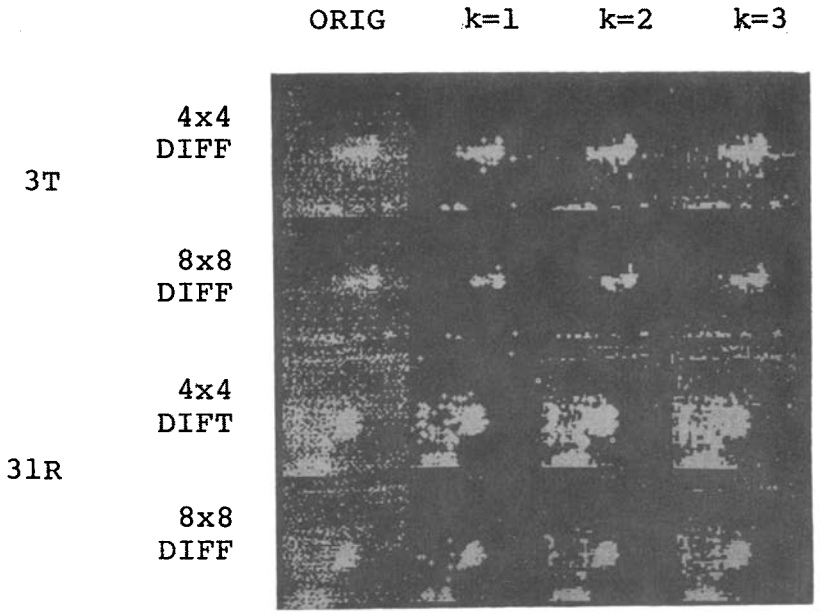
However, the generalized expand rule is not quite as generous in providing new 1 values, although it does fill pinholes in sufficently large regions. Figure 5.2 provides a comparison for t = 1, 2 and k = 1, 2, 3. The shrink/expand rule with t = 2 and k = 3 applied to each image point and its 8-neighbors provides efficient noise cleaning with most noise regions eliminated, pinholes filled, and only a modest amount of target shape distortion.

One may further generalize this process for application prior to thresholding. This technique, called "precleaning", involves a sequence of local MIN/MAX operations applied to the gray level image (analogous to shrink/expand applied to a binary image). The resulting precleaned image may now be thresholded as desired. The above threshold regions are as they would have appeared after shrink/expand processing. Figure 5.3 illustrates the process. This work is described in [19].

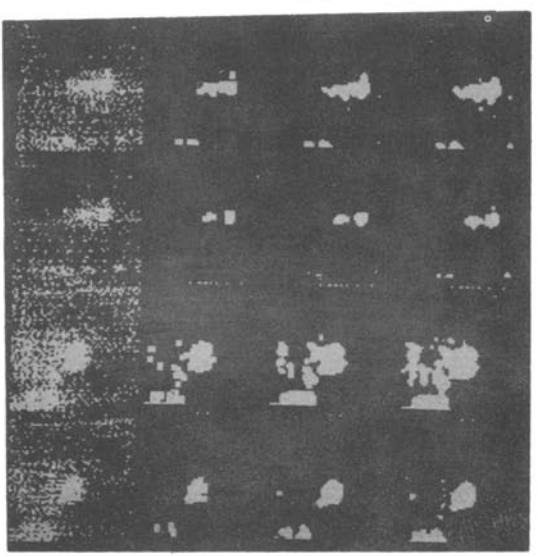
5.2 Connected component extraction

The result of thresholding is a binary image. After noise cleaning operations filter this image (as needed), it still remains to aggregate points into identified (labeled) regions. A process which labels the individual disjoint regions in the binary image, in a single raster scan, is well known in the literature [20]. It is described briefly here.

A set of 1's in a binary image is connected if any two points in it can be joined by a path (sequence) of pairwise adjacent points lying in the set. A maximal connected set is called a connected component. The algorithm to be described produces the (unique) decomposition into connected components, labels the individual components, and constructs for each connected component a descriptive feature vector. Although we do not specify the features, it is assumed that they are all extractable from a raster

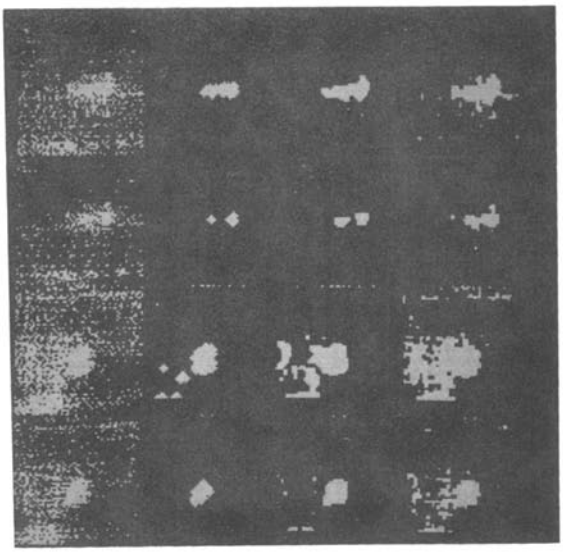


a.

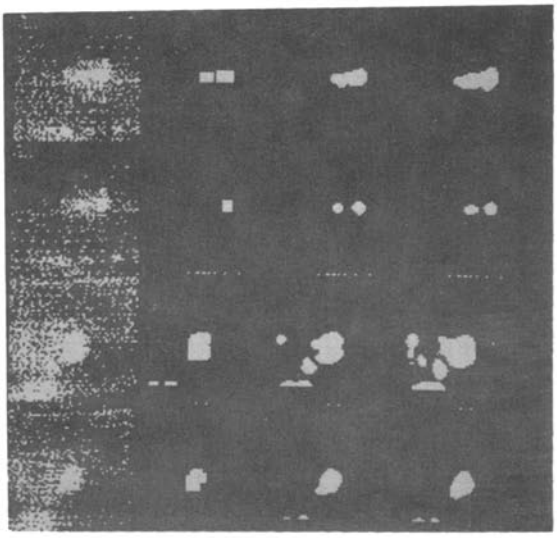


b.

Figure 5.2. Leniency in SHRINK/EXPAND definitions for windows thresholded by two methods.
a. 4-neighbor rule, one S/E, k=1,2,3
b. 8-neighbor rule, one S/E, k=1,2,3
c. 4-neighbor rule, two S/E's, k=1,2,3
d. 8-neighbor rule, two S/E's, k=1,2,3

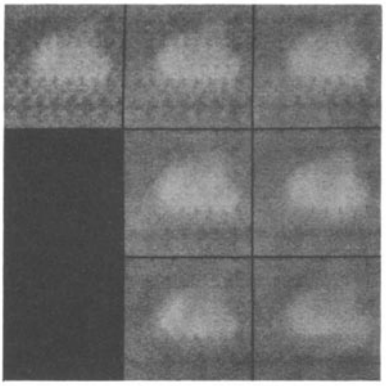


c.



d.

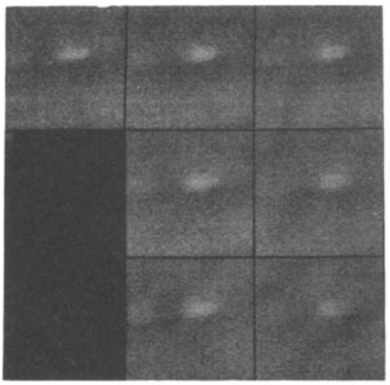
Figure 5.2. (continued)



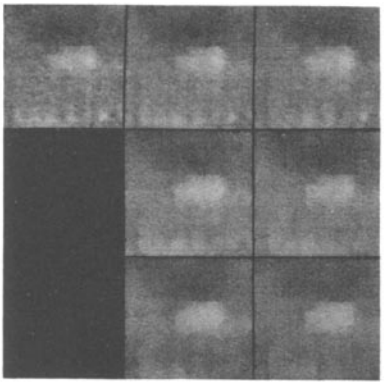
Key:

	4-nbr.	8-nbr.
Original	MIN·MAX	MIN·MAX
	$MIN^2 \cdot MAX^2$	$MIN^2 \cdot MAX^2$
	$MIN^3 \cdot MAX^3$	$MIN^3 \cdot MAX^3$

(a)



(b)



(c)

Figure 5.3.

Results of applying repeated local MIN and repeated local MAX to three FLIR images. In each part, the upper-left picture is the original; the second column uses 4-neighbor local MINs followed by 4-neighbor local MAXes (1, 2, and 3 repetitions, in the first, second, and third rows); and the third column is analogous, using 8-neighbor operations (i.e., including the diagonal neighbors).

scan using a 3x3 processing window. Additional storage is available to hold the feature values for the components. Section 7.2 describes the features.

When a new region is encountered during a raster scan, it is assigned a vector of registers to store its feature values. As the region is being tracked on the same row or continued on the next row, values continue to be accumulated into its feature vector. In order to specify the correspondence between a region and its register vector, a label is created and assigned to each point of the region which has already been visited. The label will identify the appropriate register vector, usually by some indexing scheme. Region points found to be adjacent to already labeled region points inherit that label and contribute their feature values to its register vector.

Often a region encountered for what is thought to be the first time may on a later row prove to be connected to a previously encountered region. Such regions are called subcomponents. Inasmuch as feature values were being maintained separately for each subcomponent, it becomes necessary to combine the feature values (eventually) and to create a flag that signifies that the two subcomponents belong to the same component. These flags reside in the label equivalence table. This table can be stored either as a bit matrix or as a list.

Since region labels propagate from point to point, we must also keep the labels of those points in the preceding row that are neighbors of unexamined points in the current row, with the labels of those examined points in the current row. The amount of storage necessary for labels of points is thus only a single row.

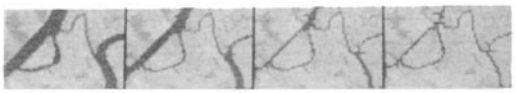
The label assigned to a component should designate whether the component is above or below threshold. If the background is not partitioned into regions (i.e., is ignored) by the algorithm then the data structure becomes simply a list of above-threshold regions. This is suitable for many applications, e.g., infrared target cueing. In general, though, the containment relation defines a tree structure. It is evident that if two components of a binary image are adjacent then one encloses the other. However, if more than one object-background transition has been detected, one cannot know which encloses which from strictly local information at the time of the initial label assignment. The

determining condition is "which region terminates first?" The region terminating first is enclosed by the adjacent region. Thus whenever a region terminates, the data structure is updated to reflect the containment relation. When a region is initiated it is entered onto an "active" list -- the list of unterminated regions. At the end of each row, the active list is compared with the list of component labels of the current label row. Any active component whose label does not appear in the current row is known to have terminated. Additionally, when overlapping regions are combined, the discarded label is deleted from the active list.

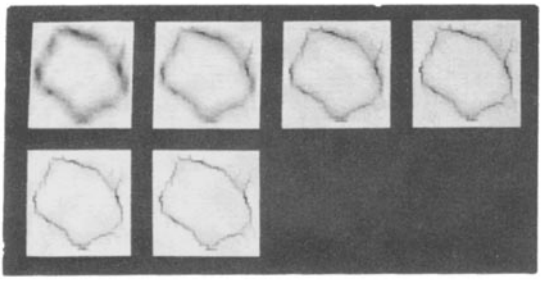
It is possible to modify the above to create a description of each connected component's boundaries. Such a description is called a "chain encoding" and is discussed in [20].

5.3 Fuzzy thinning

Objects which are everywhere elongated are often thinned down to a "medial line" for the purpose of extracting thickness-invariant topological features of the objects. The basic strategy for thinning is to iteratively delete border points (but not end points) of an object which do not locally disconnect it. For binary images, various parallel algorithms exist. The recent extension of the topological concept of connectedness to fuzzy subsets allows us to generalize thinning to gray level images [21]. Given thin dark objects on a light background, we define gray level thinning to be the successive replacement of points by the minimum gray level of their neighbors if those changes do not affect the local fuzzy connectedness for any pair of neighbors. The result of applying such an algorithm is a set of high gray level "curves" lying on the ridges and peaks of high gray level in the original picture. If the original picture is noisy there will be many local peaks; so while thinning is defined for unsegmented pictures, a local threshold is necessary to overlook these small noise peaks. Unlike binary thinning, however, we no longer need to distinguish between border and interior points since thinning a homogeneous region will not significantly change the gray level of any point; only a slight smoothing results. The results of experiments with this technique are described in a technical report [22]. See Figure 5.4 for examples of this process.



a.



b.

Figure 5.4a. Iterations 0-3 of fuzzy thinning on LANDSAT window of Monterey.
b. Iterations 0-5 of fuzzy thinning on the output of an edge detector.

6. Superslice

The object extraction task is somewhat simpler for FLIR imagery than for visible-light imagery since the objects of interest (military vehicales) are generally compact regions of (more or less) uniform thermal intensity. For this reason, thresholding has been chosen as an appropriate method of segmenting the scene. However, one can criticize threshold selection schemes on a number of grounds. First of all, if a window contains no object then thresholding it is dangerous, since above-threshold noise regions may often produce probable looking "objects." Secondly, if more than one object is present in the window then a single threshold will not suffice. Thirdly, if an object overlaps several windows then there may be no consistent representation of an object (i.e., no representation using a single threshold). Attempts to divide the scene up into overlapping windows, so that objects of maximal size are guaranteed to lie completely within a single window, answer this last objection at the cost of greatly increased overhead. In any case, the size of the smallest thresholdable region -- as well as the particular threshold chosen -- depends on the window size, the coarseness of the grid, and the type of statistical test used to determine if a region is thresholdable. One would prefer, however, to be able to extract a small region regardless of the clutter and noise beyond its borders.

Another objection to pure thresholding is the presence of noise regions in addition to object regions. Noise regions may be difficult to distinguish when based on size, shape, or gray level features. The broader and higher the valleys of the gray level histogram, the more likely that the noise regions will be extensive and numerous.

A final objection concerns the design of optimal thresholding techniques in which the optimality is based on a statistical model of the gray level population. In situations where an object contrasts strongly with the background, there may be a number of thresholds at which the object appears well defined. As the threshold decreases through this acceptable range, each object exemplar is contained within a slightly larger one. Thus although the exemplars may each look reasonable, the optimality criterion for the thresholding does not necessarily choose a "best" exemplar. This is because the optimality condition was based on the

whole window rather than on the component corresponding to the object.

For these reasons, a segmentation method which does not require a commitment to a single threshold in arbitrarily chosen regions of an image is preferable. Our method uses thresholding as a means of discovering candidate object regions. Candidates are then accepted or rejected based on the coincidence of an edge map with the region boundary. The surviving object regions are compared with the survivors of other thresholds, and those that best match the edge map are used to describe the actual objects in the image. Thus, while a number of thresholds are used, only the one defining the greatest coincidence of thresholded region border and (thinned) edge is deemed valid for a particular region. This method can be considered as defining a best exemplar for each object region.

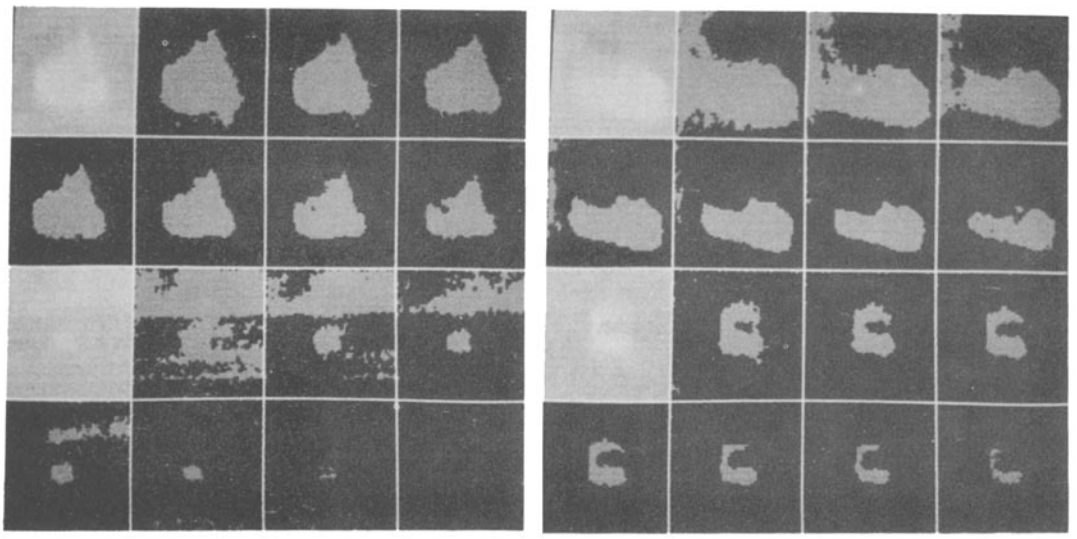
6.1 Algorithm

The algorithm consists of several steps as follows: median filtering; extraction of an edge mask by edge detection and thinning; thresholding; forming connected components; and object validity checking. For a given picture, smoothing and edge map extraction need to be done only once; whereas thresholding and the subsequent steps are to be performed over a range of thresholds sufficient to extract any objects in the picture.

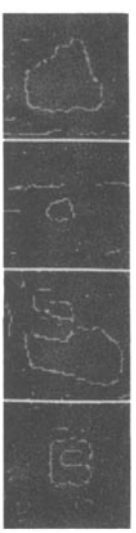
Figure 6.1 illustrates the basic concepts involved. Figure 6.1a shows several object windows along with a number of possible thresholds for each. Note that it is not at all obvious which threshold is best. However, when the edge map (Figure 6.1b) is overlaid on the thresholded picture (Figure 6.1c), we have much better guidance. Figure 6.1d shows the object region extracted from each window using the method to be described.

A number of steps of the Superslice algorithm have been discussed in previous sections: smoothing (Section 2.3), edge detection and thinning (Section 3.2), threshold selection (Section 4.1) and connected component extraction (Section 5.2). However, several problems associated with threshold selection deserve mention:

a) The omission of a threshold from consideration increases

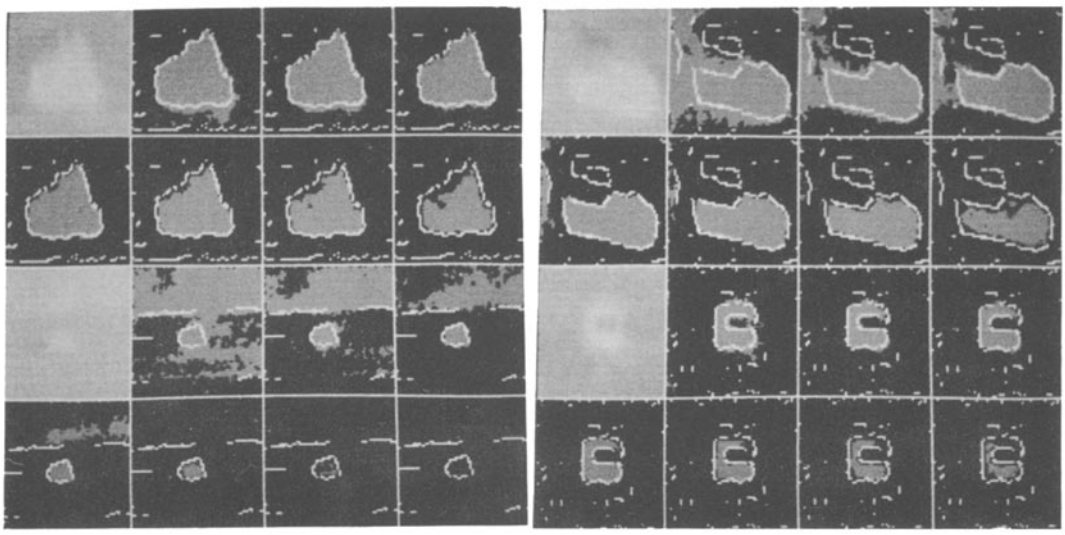


a.

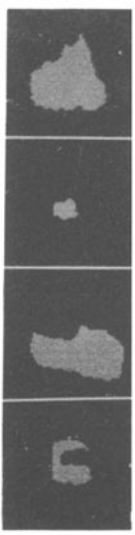


b.

Figure 6.1a. Four target windows (large tank, small tank, truck, APC) thresholded at seven different gray levels.
b. Edge maps (thresholded for visibility).



c.



d.

Figure 6.1c. Edge maps from (b) overlaid on (a).
d. Object regions extracted by the Superslice algorithm.

the probability of missing extractable regions.

b) The greater the number of thresholds considered, the greater the false alarm rate.

c) The speed of the algorithm is approximately linear in the number of thresholds used.

The probability of missing an object region due to the omission of a single threshold is the product of the probability that the scene contains an object region and the probability that the object region is discernible (by the algorithm) at exactly the omitted threshold. Although knowledge of the a priori probability is dependent on a model for the scene (which does not at present exist), experiments have demonstrated that an object region which is discernible at all by the algorithm can be extracted over a range of thresholds -- dependent, of course, on the steepness and homogeneity of the edge region bordering the object. Noise regions, on the other hand, do not tend to persist over a range of gray level thresholds. This tradeoff may therefore be posed as follows: By sampling at every kth gray level, we reduce the workload to a fraction (1/k) without appreciably increasing the false dismissal rate; however, we lose some redundancy in the extracted data which would help us discriminate object regions from false alarms.

The false alarm rate is a function of input window size, as well as a function of the number of thresholds and the positions of the thresholds in the overall gray level histogram. Certain thresholds are worse than others in producing false alarms -- specifically, those at or adjacent to peaks in the histogram.

After thresholding and connected component extraction, each component must be validated as to whether the extracted region really corresponds to an object in the scene. If one considers validity checking to be a classification process, then one can compute a large number of potential features and, using standard techniques, determine a discriminant function. We have established three heuristics to be of value. One is that objects should be "well-defined," i.e., have discernible borders. Note that not all real-world regions satisfy this constraint. For example, in LANDSAT scenes, forest, urban areas and clouds can blend into their surrounds with no discernible edge. The second

heuristic is that an object's interior should "contrast" with its surround. In this study, contrast is based on gray level difference. However, other local features including texture measures are worth considering as defining object interior. The third is that the region size lie within an acceptable range. The size test is applied first, eliminating any region with fewer than 20 or more than 1,000 points.

"Well definedness" of a region is measured by the percentage of border points which correspond spatially to (match) actual edge points in the edge map. "Contrast" is measured by the absolute difference of average gray level between the border region of the component and its interior. Figure 6.2 shows a scatter plot of these two features for the regions extracted from a set of windows. A reasonable discriminant based on these two features appears to be: match > .5 and contrast > .6 -- i.e., at least 50% of the border matches the edge map, and the contrast is at least .6 gray levels (out of 64). Note that neither feature is by itself reliable enough to discriminate noise regions from object regions. Optimal discriminants may be computed based on several models. Regardless of the particular model chosen, the discriminant value can be interpreted as a "score" for the component. Components with very low scores are discarded as pure noise. In practice, we have used the match measure as a score for objects which were above the pure noise threshold.

The score is important in comparing (nested) object regions corresponding to the same object. When an object is thresholdable at gray levels $t_1 > t_2 > \ldots > t_k$, this gives rise to k connected components, $C_{t_1} \subseteq C_{t_2} \subseteq \ldots \subseteq C_{t_k}$. Since each C_{t_i} represents the same object, we call each an "exemplar." In general, we wish to select a single exemplar as the best representative of an object. The score provides a criterion for selecting among exemplars. Thus, one could choose the exemplar C_{t_j} with the highest score. It is not always easy, however, to determine the nested sequence $\{C_{t_i}\}$. In particular, if one object thresholdable at gray level t is contained within another thresholdable at gray level $t' < t$, then regardless of the comparative difference between the two scores, we would want to retain C_t and $C_{t'}$. This situation can be handled by assuming that nested components whose areas are

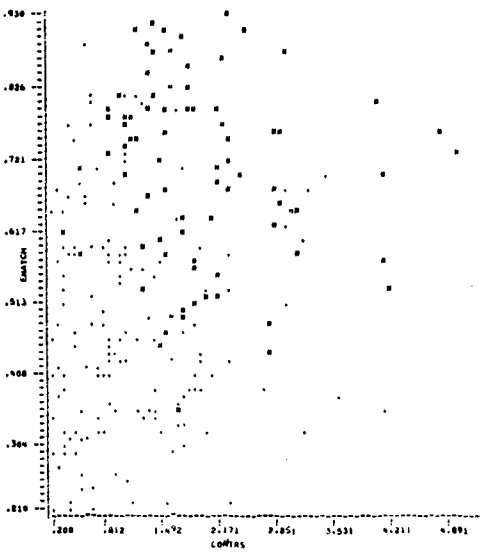


Figure 6.2. Scatter diagram plotting well-definedness against contrast for a set of noise regions (plotted as periods) and object regions (plotted as hash marks).

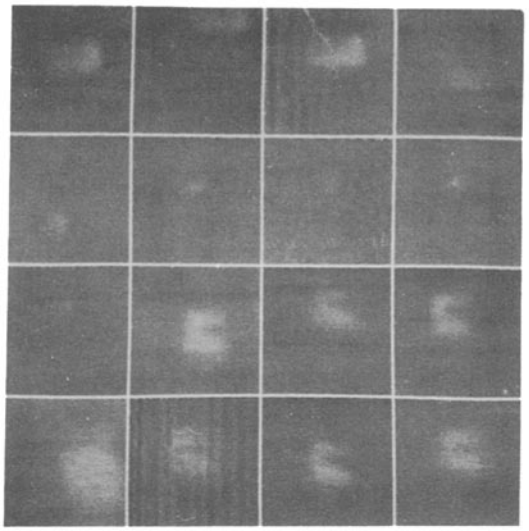
sufficiently different (say, 50% change in size) correspond to different (although nested) objects. In thermal images, this might correspond to a warm vehicle with a hot engine compartment, or to a vehicle on an asphalt road. The results of applying the algorithm to a set of 16 APC windows are illustrated in Figure 6.3. Note that in almost all cases (the negative image was not processed), the resulting labelled images contain the target regions (as well as other regions).

In summary, the algorithm for region extraction consists of the following steps:

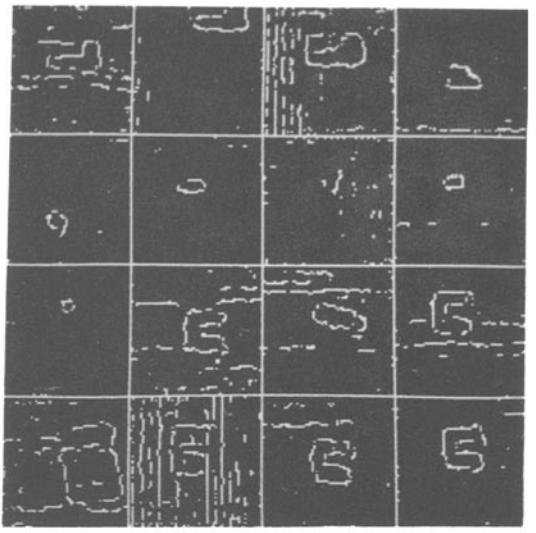
1. Smooth the image, if necessary (to promote clean thresholding).
2. Extract a thinned edge picture.
3. Determine a gray level range for thresholding.
4. For each gray level in the range:
   a. Threshold the smoothed image.
   b. Label all connected regions of above-threshold points.
   c. For each connected region:
      i. Compute the percentage of border points which coincide with significant thinned edge points.
      ii. Compute the contrast of the region with the background.
      iii. Classify the region as object/non-object based on the size, edge match and contrast.
5. Construct the canonical tree for the set of object regions based on containment.
6. Prune the containment tree by eliminating adjacent nodes which are too similar.

6.2 Conformity - a measure of region definedness

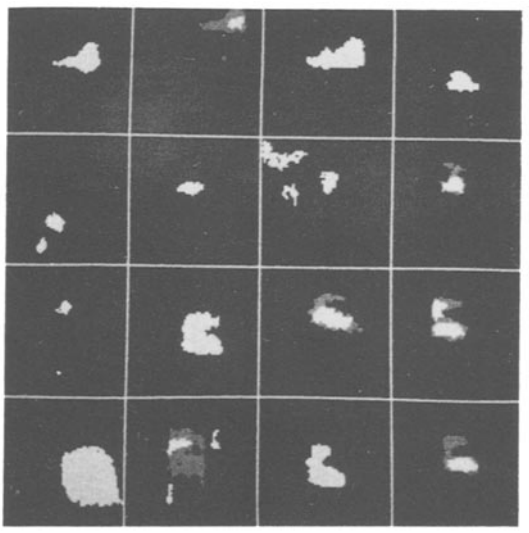
The Superslice algorithm relies on the heuristic that thresholded object regions are distinct from background because they contrast with their surround at a well-defined border. The coincidence of high contrast and high edge value at the border of a thresholded region is an example of the use of convergent evidence supporting the assertion of the object region. The definedness of the border may be evaluated as the percentage of the border points which coincided with the location of thinned edge (locally maximum edge response). Thus a match score of 50% means



a



b



c

Figure 6.3a. Sixteen APC windows.
b. Edge maps (thresholded for visibility).
c. Object regions extracted by the Superslice algorithm.

that half the border points are accounted for as being on the edge. However, it does not mean that the matched points adequately represent the object. Figure 6.4 illustrates two cases of 50% match. (Matched points are indicated by thick strokes.) Clearly, the second case is a better representation than the first.

The traversal of the border of a thresholded region induces an ordering on the matched points. Let $r_1,\ldots,r_n$ be the runs of matched points encountered during a border traversal. By connecting the proximal ends of runs along the traversal, one creates a polygonal approximation to the thresholded region. We define "conformity" as the measure of match of the polygonal approximation to the thresholded region. High conformity means that the region is well-represented by its approximation regardless of the actual percentage of matched border points. Figure 6.4a illustrates low conformity, while Figure 6.4b shows good conformity.

Conformity is evaluated as the ratio of the absolute difference in area (between the two polygonal representations) to the area of the threshold region. Experiments have indicated its utility as a feature for discriminating noise from objects. A quantitative study of its discrimination value is described in Section 8.4.2.

6.3 Hyperslice - An algorithm for recursive region extraction

The algorithm (Hyperslice) described here is an amalgam embodying the recursive control structure of Ohlander [23] and the object extraction techniques of Superslice. Hyperslice consists of the following steps [24]:

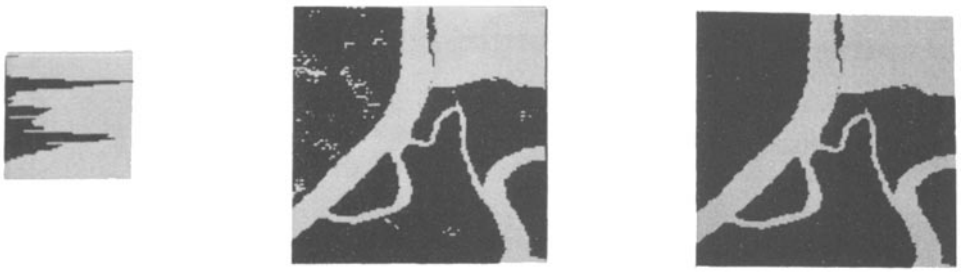
1. Preprocessing - image smoothing, thinned edge map extraction.
2. Initialize the extracted region mask (ERM) to the empty mask. Initialize the available points mask (APM) to the entire mask.
3. Compute histograms for all feature images based on the APM.
4. Determine a "best" slice range over all current histograms and slice the corresponding image.
5. Generate submasks for regions satsifying the Superslice criteria. Add them to the ERM; delete them from the APM.
6. Apply algorithm steps 3-5 recursively to the background

a. b.

Figure 6.4a. Contour whose matched edge points (thickened strokes) exhibit poor conformity.

b. Contour showing good conformity.

a. b.

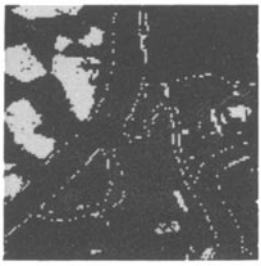


c. d. e.

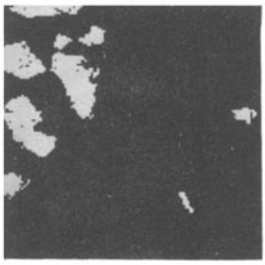
Figure 6.5. Recursivė region extraction on Monterey image.

a. LANDSAT window.
b. Edge map.
c. Histogram of (a), with selected slice range indicated.
d. Mask of slice range. Within range points are white.
e. Extracted regions mask.

f.

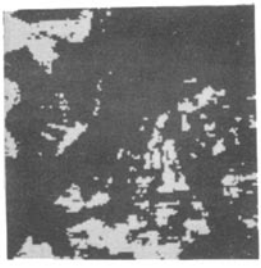


g.

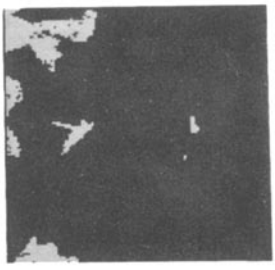


h.

i.

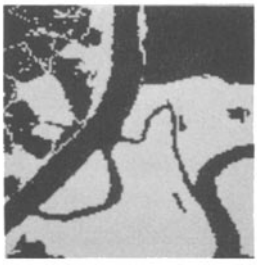


j.



k.

l.



m.

Figure 6.5 (continued)

f. Histogram of remaining points after deleting extracted regions of (e).
g. Slice range mask.
h. Extracted regions mask.
i. Histogram of remaining points.
j. Slice range mask.
k. Extracted regions mask.
l. Histogram of remaining points.
m. Mask of remaining points.

set (APM). The algorithm should also be applied recursively to each submask added to the ERM, since the extracted region may be a union of regions discriminable by some other feature.

Several comments are in order. First, the slice ranges chosen for Hyperslice should be rather liberal (i.e., extending beyond valley bottoms in the histogram), since points not corresponding to well-defined regions will be returned to the APM. The resulting histograms appear more natural (not "carved-out") for this reason. Secondly, the resulting decomposition is order-dependent, i.e., different results may be obtained if the order of selection of slice ranges is changed. If two adjacent regions in the image contribute adjacent peaks in the histogram, then points in the intersection of the overlapping slice ranges will generally belong to the shared edge region. Whichever region is sliced first will tend to accrete more of these points. Since these points lie at or near the true edge, they tend to increase the edge match criterion for that region. Once they are removed from the APM, they are not available to the adjacent region. Consequently, the edge match criterion of the adjacent region may suffer. This is most likely to occur for adjacent regions which lack a strong common border. The 2-dimensional histogram approach in [16] can detect adjacency along weak borders. In practice, the edge match criterion is relaxed somewhat from demanding actual coincidence to allowing proximity (e.g., a region border point adjacent to a thinned edge point is counted as a match).

The algorithm has been implemented as an interactive system of programs. Several examples illustrate its ability to segment images based on gray level alone (i.e., no other features were used to aid the segmentation). Figure 6.5 depicts a window of an ERTS frame of the Monterey area in California. The water area contrasts sharply with the land and very little noise is extracted and subsequently returned to the APM. The subsequent slices extract light and dark fields which contrast with the undifferentiated background region.

The second example is derived from Ohlander's house scene. The average of the three color bands provides the gray-scale. The resulting image has been smoothed by 3x3 median filtering. The first slice range extracts the sky regions and the bright crown

of a bush. Next the shadow regions appear along with the bushes. The somewhat darker grass is extracted in the third slice range. Finally, the brick is extracted. Figure 6.6 illustrates this sequence.

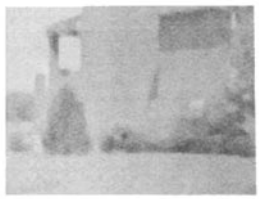
Images such as the Monterey and house images are difficult to analyze since regions need not be well defined due to the complexity of light reflections and shadows. Nonetheless, this algorithm provides a mechanism for retrieving those regions which are well-defined.

7. Feature extraction

7.1 Feature design

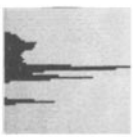
In this section, as in most work dealing with pattern classification, a "feature" is taken to be some numerical quantity which can be calculated for each object to be classified. ("Shape" is not a feature, since many features, such as height/width, measure characteristics of the shape.) To be consistent with a high processing rate throughout, all features used in this study are based on accumulatable quantities. That is, a number of crude features have been chosen (listed in Table 7.1a) which are defined at each pixel. The value of any of these features for a region is just the sum of the values over all the pixels of the region. These crude features can be accumulated as the image is being segmented, and are therefore immediately available for any region as soon as it has been completely extracted. The descriptive features actually used are simple functions of these accumulatable quantities, so that once any region has been extracted, brief calculations produce all the information required for classification of that region, with no further reference to the original image. One additional feature, "conformity," has been obtained for many of the images. This feature requires rather more postprocessing after region extraction, and is included as a nearly optimum measure of one region characteristic which should be of importance in target detection: cooccurrence of the region perimeter and points of high brightness gradient. This gives a useful standard for measuring the adequacy of the rapidly calculated feature (E&P, in Table 7.1c) which is used as a measure of the same property.

A decision rule is effectively a mapping from the feature space



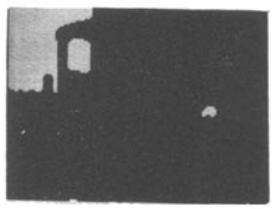
a.

b.

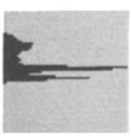


c.

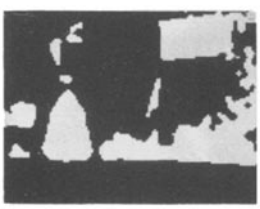
d.



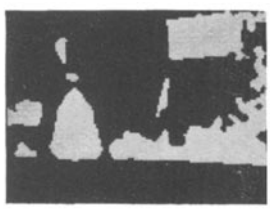
e.



f.



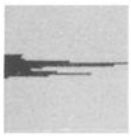
g.



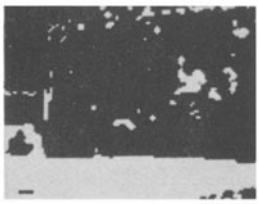
h.

Figure 6.6. Recursive region extraction on house image.

a. House window.
b. Edge map.
c,f,i,l,o. Histograms after successive deletion of extracted regions. New slice ranges are indicated.
d,g,j,m. Slice range masks.
e,h,k,n. Extracted region masks.
p. Mask of remaining points.

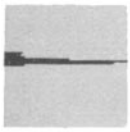


i.



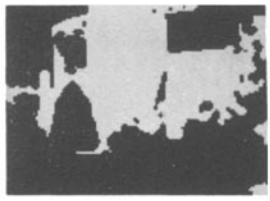
j.

k.



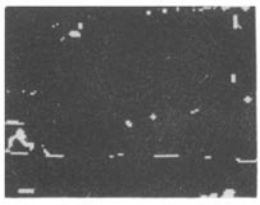
l.

m.



n.

o.



p.

Figure 6.6 (continued)

a. Accumulatable features per connected component

	Symbol	Meaning
1.	N	Area
2-3.	SX,SY	$\Sigma X, \Sigma Y$ - first moments
4-6.	SX^2, SY^2, SXY	$\Sigma X^2, \Sigma Y^2, \Sigma XY$ - second moments
7.	P	Perimeter point count
8.	E	High edge point count
9.	SPE	Total edge value on the perimeter
10.	SIG	Total interior gray value
11.	SPG	Total perimeter gray value
12-13.	SG, SG^2	Total gray level, total squared gray level

b. Intermediate quantities

1.	X_{AVE}	$4 * \sqrt{SX^2}$
2.	Y_{AVE}	$4 * \sqrt{SY^2}$
3.	R^2	$SX^2 + SY^2$
4.	V	$SG^2/N - (SG)^2/N^2$

Table 7.1. Features.

c. Recognition features

1.	h/w	Y_{AVE}/X_{AVE}	shape
2.	(h/w)'	$\lvert X_{AVE}-.8*Y_{AVE}\rvert/\sqrt{X_{AVE}*Y_{AVE}}$	
3.	(h*w)/A	$X_{AVE}*Y_{AVE}/N$	
4.	(h+w)/P	$(X_{AVE}+Y_{AVE}-4)/P$	
5.	diff	$(SX^2-SY^2)/R^2$	
6.	skewness	$\lvert SXY\rvert/R^2$	
7.	asymmetry	$((SXY)^2-SX^2SY^2)/R^4$	
8.	SDEV	$\sqrt{V}$	brightness
9.	Gray level difference	SIG/(N-P) - SPG/P	
10.	E & P	(Number of perimeter points at high edge local maxima)/P	
11.	E_p	SPE/P	

d. Special features

1.	conformity	(See Section 6.2)

Table 7.1, continued

to a lower-dimensional space (the decision space) in which each point is associated with a fixed class. While this structure is very general, commonly used decision rules are very severe specializations of this general scheme. Usually the initial mapping is produced by a set of polynomial functions on the features, one function for each dimension of the decision space. Within this space, the class regions are usually separated by planar boundaries. Thus, the Fisher method utilizes a single linear mapping onto the line, which is bisected by a point (at the Fisher "threshold") to establish the two class domains. Specialization of decision rules places sharp restrictions on what constitutes an appropriate feature.

To discriminate tanks from trucks, a naive observer might point out that one need only examine the shapes. One more familiar with computational measures would recognize that the shape of an object involves a great many features, but might suggest that the height-to-width ratio would be one useful feature. However, height-to-width, width-to-height, log(height-to-width), etc. are all quite distinct features, one of which may be highly effective in the desired decision while others may be totally useless. Useful features must thus satisfy a number of conditions, some of which are general, the others being imposed when particular simple decision rules are to be applied. The present classification study has considered linear and quadratic classifiers, a decision space with no more dimensions than the number of classes, and simple boundaries for each class within the decision space. Several levels of restriction on the features to be used with such a classifier can be stated:

1. Each feature must exhibit a different distribution for each of at least two classes.
2. The classes should tend to fall in different value ranges for each feature, since class assignments in the decision space will be to connected regions.
3. When the classifier utilizes sample means and variances to estimate parameters for the mapping (as those used here do), the true feature distributions of each class should be unimodal, approximately symmetric about the mode, and with a minority of points contained in the wings of the distribution.

4. For use with linear classifiers, each feature should have a distinctly different mean for at least two classes. For use with quadratic classifiers, it is only necessary that some range of values tend to characterize one class, while the other class predominates on the complement.

Despite these "rules" for good features, it should be noted that for a multi-feature decision scheme, none of these rules is essential. However, only when some of the features are very strongly correlated can the above principles be violated without destroying the classification, and while this situation is not necessarily to be avoided, it makes interpretation of decision rules much more difficult. Moreover, as a practical matter, features which fail to have the above properties normally turn out to be ineffective (or worse, countereffective) when employed in automatic classification. Since one is not really restricted in the particular form of the features to be used (but only in the underlying characteristic being represented) one may as well assure that the features being considered are, as far as possible, individually effective means of class discrimination.

Finally, one more restriction should be stated.

5. The features should not reflect characteristics which effectively delineate the sample classes, rather than the true classes.

This, of course, is the familiar failing of "small" samples, but may appear even in apparently large enough samples. In our data base (Section 8.1), several such "extraneous differentiations" did arise. In cases where a large number of features are employed in a classifier, there must always be doubt about whether condition 5 will hold. It is this condition, more than any other, which restrains the number of features which can usefully be included in a classifier. If an arbitrarily large number of features are measured for a particular set of classified samples, it is virtually certain that spurious characteristics will allow them to be well separated by a decision function based on those features, but there is no reason to expect anything other than random classification of new samples. The problem is sufficiently pervasive that a simple means of dealing with it could almost be elevated to a principle:

5'. Features should be included in a classifier only if they

identify true differences between the classes more than they do spurious differences between the samples.

While the above rule may seem obvious, it is important to realize that including additional features that do not discriminate between classes makes the classifier worse, as the features may very well distinguish the class samples, even though they do not distinguish the classes. (Self-classification of the training set improves, while classification of independent test sets degrades.) Class differences must be effectively reflected in the feature to make it safe to use. "Height-to-wdith" ratio is a dangerous feature to include in a linear classifier for target vs. non-target since its mean values for target and non-target classes may not be greatly different (though the distributions may differ greatly), so that small spurious differences in sample means may produce most of the "strength" of the feature. In a quadratic classifier, however, the problem would be much less severe, since the discrimination provided by the feature more nearly matches the requirements of the decision function employed.

7.2 Computation

The principal attributes of image regions which can be used to identify them are shape and relative brightness. Corresponding locally accumulatable properties are pixel coordinates, and functions of them, and gray level, and functions of it. Additional information can be obtained from the contrast between the region and its surround at the region boundary. One can know as one examines each image point whether it is in the interior of a region, on the region boundary, or in the background. Statistics of interest can therefore be accumulated separately for these classes. Finally, the pre-computed edge value (gray-level gradient) is associated with each point, and these values may be accumulated or may be used to index subsets of points (e.g., "high edge" points) for which other quantities may be accumulated separately. The accumulated features actually used are all of one or the other of the above types, and were listed in Table 7.1a.

The features calculated for use in classification studies are given in Table 7.1c-d. They are further divided into two groups -- those that are purely shape measures, and those that depend in some way on the brightness of the region (or some part

of it). Many of the functions appear to be straightforward measures of significant characteristics, but others seem less straightforward. The criteria for choosing the specific functional forms used are discussed in Section 8.4. A discussion of the relative utility of the features appears in that same section.

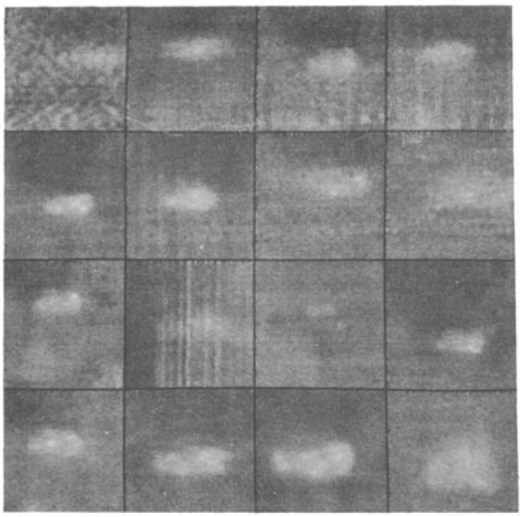
8. Region Classification and Experimental Results

8.1 Data base description

For a description of the complete "NVL" data base and its ground truth see [1]. From it a set of 174 128x128 windows were selected, extracted, requantized, median filtered and sampled 2 to 1. The set consists of 164 target windows (75 tanks, 34 trucks, 55 APC's) and 10 non-target (noise) windows. Figure 8.1 displays this set of windows and their identifiers.

8.2 Overview of classification

There are two general approaches to classification of objects into a preassigned set of mutually exclusive categories. The first might be called "semantic" classification. Each category is examined for particular characteristics which distinguish its members from those of every other category being considered. These characteristics are used to identify each object submitted for classification. (Difficulties, of course, occur if an object has none of the "key" characteristics, or has "key" characteristics suggesting more than one classification. Such an occurrence indicates that the classes suggested simply do not include everything within the domain of interest, or are not truly mutually exclusive -- at least as defined by the set of "key" features.) This is a form of classification which is ubiquitous in human experience. Unfortunately, in many cases of practical importance, the objects to be classified cannot be characterized by properties which will always be observed within one class, and never in any other class. If the classes really are well-defined, this difficulty may arise because of the need to classify using noisy or poorly resolved data. It may also occur because characteristics quite plain to human observers may defy expression as calculatable quantities (one vehicle may be "sleek and speedy looking", another "squat and out-of-date"). For whatever reason, when such incom-

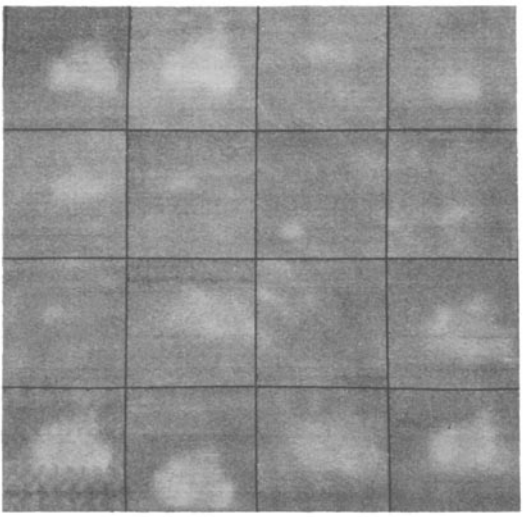


1T 2T 3T 4T

6T 8T 9T 10T

11T 12T 13T 14T

15T 16T 17T 21T

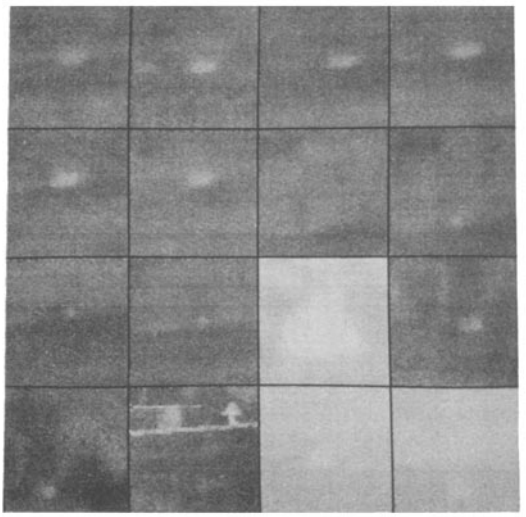


22T 24T 26T 28T

31T 32T 33T 34T

35T 38T 40T 42T

43T 45T 46T 48T



50T 51T 52T 53T

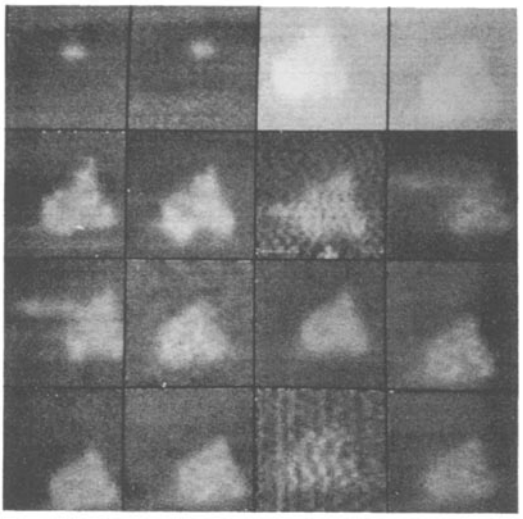
54T 55T 56T 57T

58T 59T 61T 62T

63T 64T 65T 66T

Figure 8.1. NVL data base consisting of 164 target windows and 10 non-target windows.

a. 75 tanks.

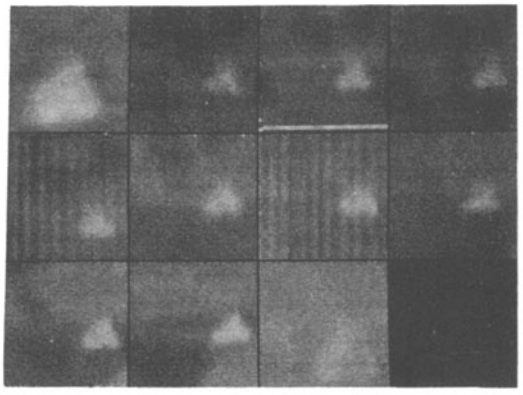


68T 69T 73T 74T

75T 76T 78T 79T

80T 89T 92T 95T

99T 105T 109T 110T

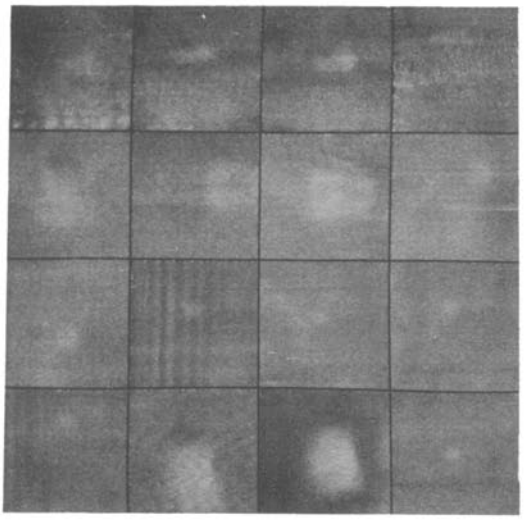


114T 122T 123T 124T

125T 126T 127T 128T

129T 130T 131T

Figure 8.1 (continued)

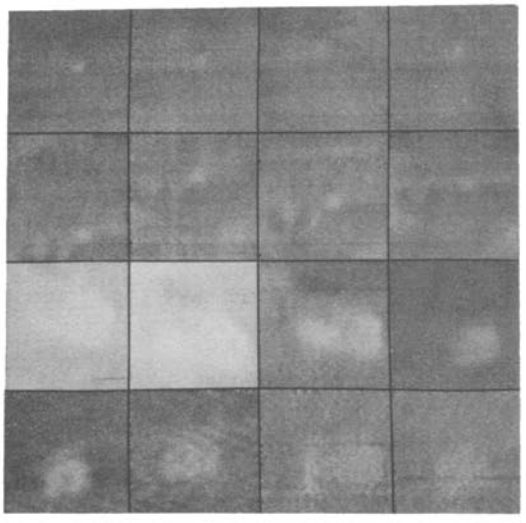


3R 4R 6R 9R

18R 22R 24R 26R

31R 32R 33R 34R

35R 41R 47R 51R

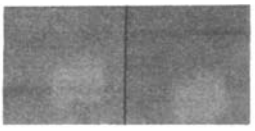


52R 53R 54R 55R

56R 57R 58R 59R

71R 72R 77R 100R

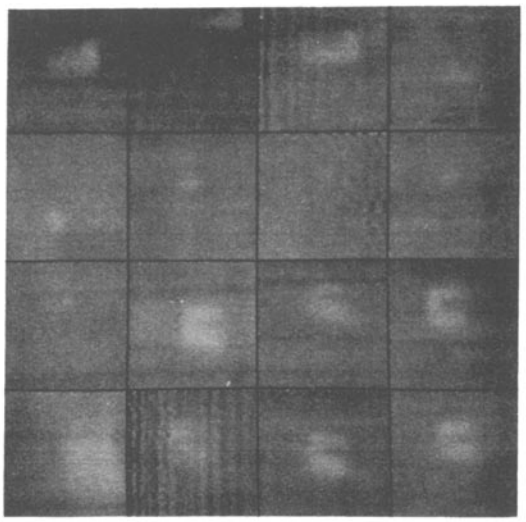
104R 109R 132R 133R

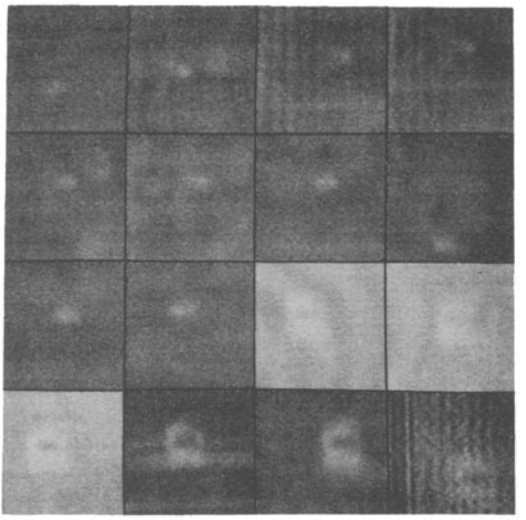


134R 135R

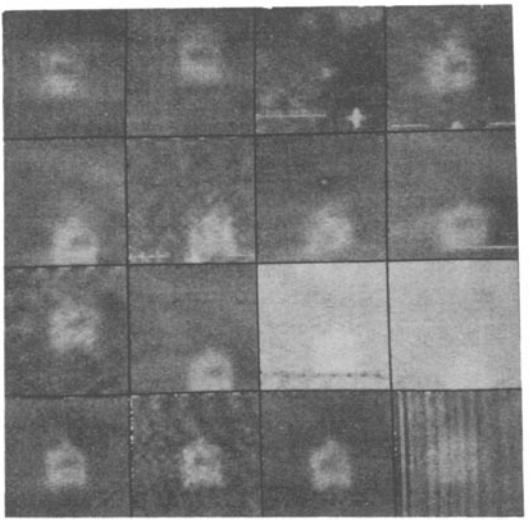
Figure 8.1 (continued)

b. 34 trucks.



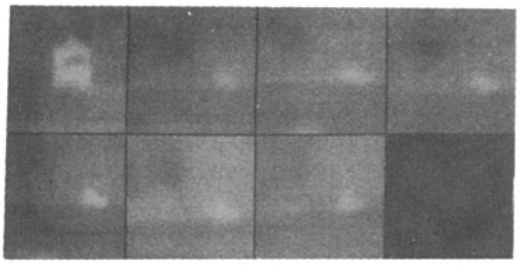


21A	22A	24A	27A	50A	51A	52A	53A
28A	32A	33A	34A	54A	55A	56A	57A
35A	37A	38A	42A	58A	59A	61A	73A
44A	45A	46A	48A	74A	75A	76A	78A



79A	80A	86A	90A
91A	93A	94A	96A
97A	98A	101A	102A
111A	112A	113A	114A

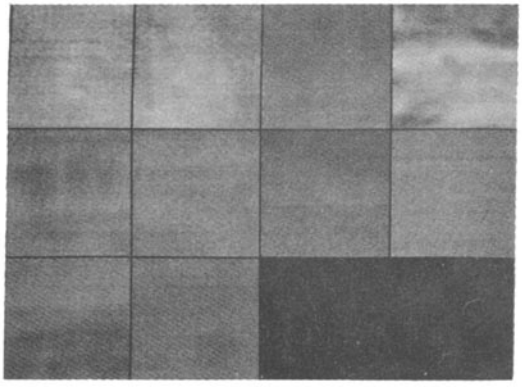
Figure 8.1 (continued)
c. 55 APC's.



115A 122A 123A 125A

127A 129A 130A

c. APC's (continued).



2N 8N 14N 20N

26N 32N 38N 44N

50N 56N

d. 10 non-target windows.

Figure 8.1 (continued)

pletely characterized problems arise, a method is required which provides a computable "best guess" classification. All such methods accept a number of (usually numerical) features which are assumed to be relevant to the classification intended. The distribution of these features for a large number of objects whose identity is already known is then used to provide a rule which assigns a class to an object given the n-tuple of features measured for that object. Typical rules of this sort are simple polynomials over the features, whose values are used to determine the class assignments.

"Statistical" classification finds the best rules for a fixed class under some (usually very restrictive) assumptions about the way the features ought to be distributed. Since the data available in this study appear not to provide enough resolution to produce a semantic classification, we have utilized a procedure which includes a statistical classification component. A completely statistical classifier was not used, however. The full procedure consists of a semantic pre-classification of regions which could not represent targets, followed by a statistical classification of the "reasonable" regions. This approach was chosen primarily to ensure greater robustness in the resulting classification scheme, as will be discussed more fully below.

Finally, it is important to analyze the types of errors made by a classifier. For example, a well-behaved classifier should be wrong more often on distorted images than on undistorted ones. This type of performance may be tested by training a classifier of the same type on a "training set" of half the samples, distributed evenly through the classes. The resultant classifier can then be used to reclassify the whole data set. If the "training" and "test" results are similar, then the classifier is judged fairly stable. If the results are good, then the classifier can be considered fairly powerful.

It is important to distinguish between human interaction in classifier design and human interaction in the operation of the classifier. The former is permissible since the classifier is fixed once it has been effectively designed and trained. No further human assistance is allowed and the classifier is applied in an automatic fashion to the test set.

8.3 Detailed classification description

The objects to be classified in this study are connected regions of an input picture, extracted by thresholding the image. More than one threshold may have been used on any given picture, so the regions need not be disjoint; rather, one may be entirely contained in another. For each region, a feature vector containing information about shape and brightness (as described in Section 7) is used as the sole source of information about the region for classification. The extraction procedure has somewhat preselected these regions, so that every region examined has at least minimal (20%) correspondence between its perimeter and the high-edge points, has at least minimal contrast (.2 gray level) and is of roughly appropriate size (between 20 and 1000 pixels).

8.3.1 Stage 1: pre-classification

If the classification is thought of as a two-stage process (shown schematically as Figure 8.2), the first stage is a crude "semantic" classifier which identifies some regions as having properties which indicate that they are not targets. Thus, all targets have similar height and width, seen at any aspect angle. Any region with h/w greater than 3 or less than 1/3, then, may be confidently rejected from further consideration. Similarly, targets "should" show some minimal contrast at their perimeters, a good edge-perimeter overlap, and small targets should be of nearly uniform brightness. All these criteria are set by establishing numerical thresholds such that at least 95% of the sample targets satisfy the criteria.

This is called "semantic" classification, rather than a very crude statistical classification, because the particular criteria used have been chosen to distinguish the targest on the basis of physical characteristics of true target images. A statistical classifier, even if it arrived at the same scheme, would be assessing discriminatory ability on the sample of classified regions provided for training, and could reflect any peculiarities which happened to distinguish the categories in that sample. (In the NVL data, APC's often exhibit an asymmetry which is due to the fact that most of those in the sample appear in only a single aspect. An apparently good statistical classifier could be formed which would unhesitatingly identify any APC in some other aspect

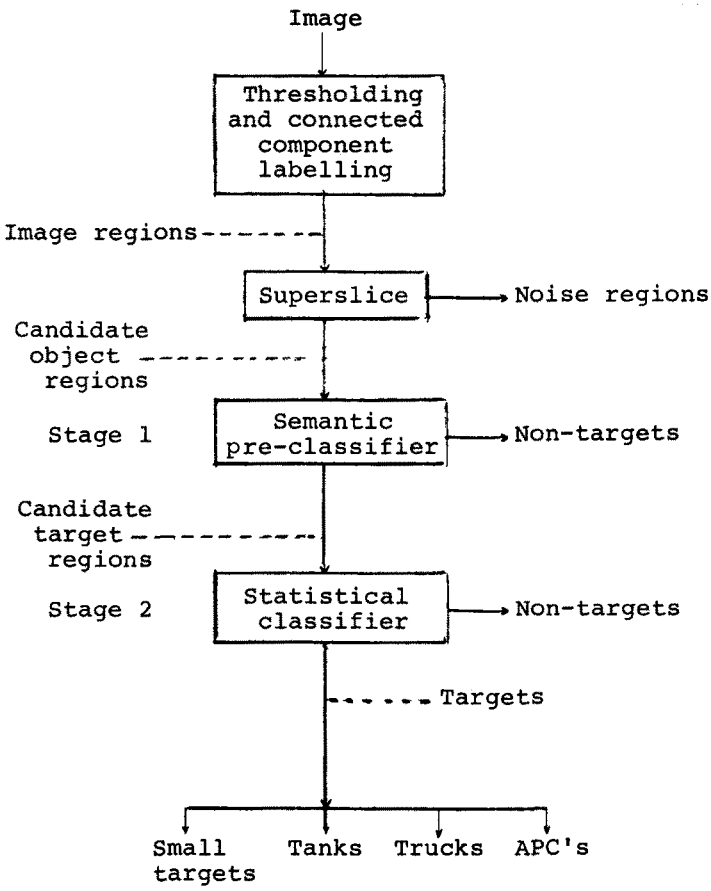


Figure 8.2a. The classification process.

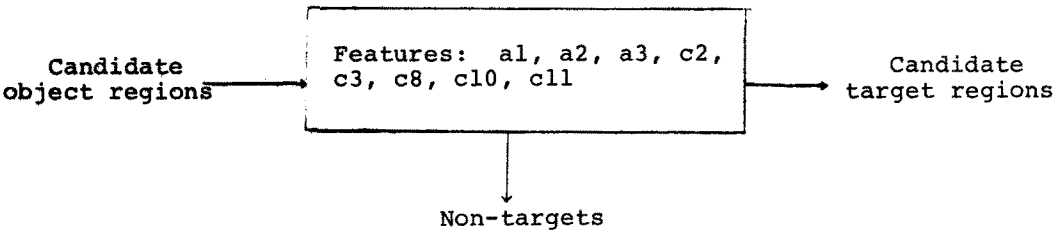


Figure 8.2b. Stage 1 - the pre-classifier (for feature list, see Table 7.1).

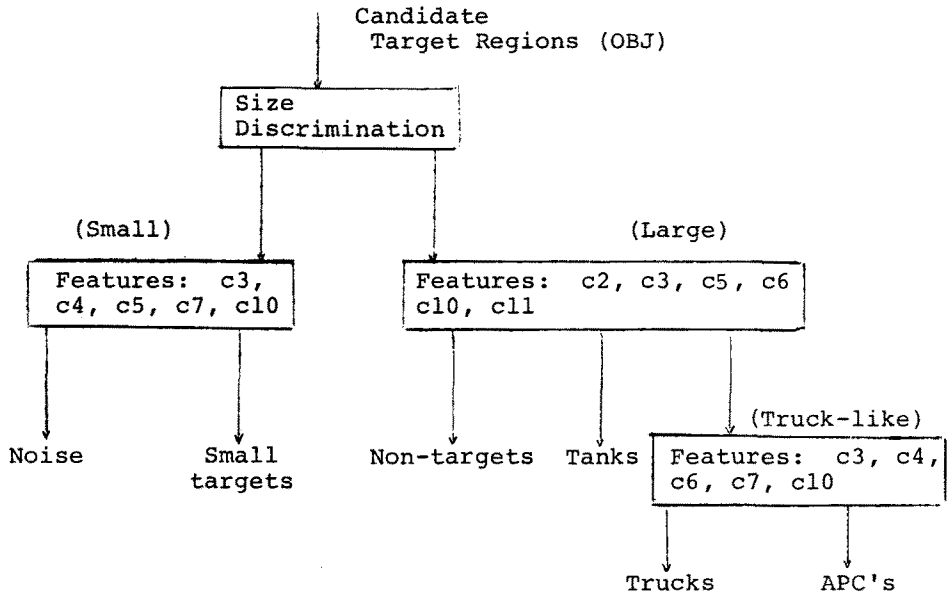


Figure 8.2c. Stage 2 - the classifier (for feature list, see Table 7.1).

as a tank.)

This pre-classification examines individual features to determine whether they could be reasonably associated with true targets, and discards "ridiculous" cases. A side-effect of this sorting is to assure that feature values seen by the subsequent statistical classifier are never very far from their characteristic values. This makes the classifier much better-behaved than one which accepts non-normally distributed features (as most do) that have not been "critiqued."

8.3.2 Stage 2: statistical classification

Once the set of extracted regions has been reduced to a set of bright, compact, reasonably uniform regions, statistical classification is used to assign a class to each particular combination of features (or rather, to its associated region). A great many kinds of statistical decision rules exist. Access to the MIPACS [25] interactive system allowed us to design a decision tree (each node of which is a standard classifier) for efficient classification. The system allows individual decision functions to be either linear (e.g., Fisher), quadratic, or maximum likelihood, and provided a convenient mechanism for selecting which decisions to make, and just which features to use at each decision point.

The basic structure selected was shown in Figure 8.2c. The first node actually represents a non-statistical selection. Because of the wide range of apparent sizes of the target images (from 25 to 1000 pixels) and the consequent wide range in visible complexity of detail, it was quickly determined that statistical classifiers would not provide good discrimination over the entire size range. (Almost every feature measured showed substantial correlation with apparent size, and since the various sample classes happened to have rather different image size distributions, our earliest classifiers used that factor as a main classification indicator.) Therefore, the first step in the classification is a simple split on image area -- with all regions of less than 95 pixels going to the "small" subtree, and the remainder passing into the "large" subtrees. For several reasons, principally a presumed lesser urgency for detailed identification of small or distant objects and the fact that in the smallest images

no significant differences between the various target classes are apparent, the small regions are simply sent to a node which classifies them as (small) "target" or "non-target" -- the specific type of target is left unspecified. For the large regions, a two-stage process followed. As neither APC's nor trucks are particularly well characterized by the features used and their distributions are very similar, they were merged into a composite "truck-like" class. Any region found to be in this class is then assigned as APC or truck by a Fisher discriminant. (A major reason for this breakdown is that it permits fairly large samples to be used at an important decision point and relegates use of the sparsely sampled truck class to a relatively inconsequential discrimination.) The principal decision was therefore between the "tank" and "truck-like" classes and the "non-target" class. Two different approaches were tried for making this decision, both based on a quadratic maximum-likelihood discriminant. These are described more fully in Section 8.4. One approach ("fixed classes") applied the maximum likelihood criteria directly to the tank, truck-like, and non-target classes. The second approach included two "reject" possibilities as well -- non-target, and unclassified target. (Notice that the non-target label is applied either if a region looks sufficiently like a "typical" non-target or if the best label implies too unlikely a value for the features measured.) The latter approach was included to further minimize reliance on characterizing non-targets in detail.

Given the tree structure for the classification, the kind of classifier and the set of features at each node were determined. The number of features which can reliably be used depends on the size of the sample set used for training. Assuming that the features are chosen so as to avoid apparent vagaries in the set of exemplars, one can confidently use an additional feature for each ten samples in the smallest group, and sometimes may use up to one-third the sample number (for a linear classifier). As quadratic classifiers utilize more detail of the presumed distribution one is restricted to the conservative end of that range. These rules of thumb, while not universally valid, are nonetheless useful guides.

By merging the truck and APC classes, we allow comfortable use of a quadratic classifier on five or six features at the main

decision node, while the smaller samples make a linear classifier or a three or four feature quadratic classifier more reasonable at the lower node. The "small" node could utilize five or six features -- but one is hard-pressed to find even that many which provide any discriminatory power at all. (However, one feature, E&P, is very powerful indeed.)

8.4 Experimental results

8.4.1 Feature selection

As in any classification problem, much of the initial feature selection for the vehicle recognition task was carried out informally. This phase is largely introspective, determining characteristics of the images that seem helpful for human judgement, then identifying some features that suitably reflect these characteristics. This initial feature set (conveying "shape" and "relative brightness") is listed in Table 7.1, Section 7. All of these features seem appropriate for use with linear or quadratic classifiers.

The features were examined in several ways. First, histograms for each feature were produced for every sample class. These histograms were examined to see whether the sample distributions satisfied the criteria noted in the last section. The differentiation that appeared was interpreted as to whether it was a true difference between classes, or simply a sampling anomaly. (At this stage too, particular features might be replaced by similar features of slightly different functional form, to better satisfy the requirements of automatic classification.) Second, those features that seemed to have some merit were ranked for classification power at each node of the decision tree. The "Automask" method, available within MIPACS, was used ([25]). Briefly, Automask finds, for each feature, its "share" of the total dispersion both between and within sets, and finds the single feature which produced the greatest comparative variance between sets. This feature is then deleted from consideration, and the other features reexamined to find the next best feature, and so on. The relative merits of the features for each node are shown below.

Node	Good features	Usable features
Small	E&P	(h/w)', (h*w)/A, (h+w)/P, diff, skewness, asymmetry
Large	E&P, diff	(h/w)', (h*w)/A, skewness, asymmetry, E_p
Trucklike	E_p, asymmetry	(h/w)', (h+w)/P, skewness, E&P

Shape features:

In the first stage, the (h/w)' height-to-width feature was useful in identifying small bright streaks as non-targets. In the statistical classifier for small targets, shape features were individually very weak in distinguishing targets from non-targets. For large targets, diff was the best shape feature at node LARGE; all the others but asymmetry were also of some use. At node TRUCK-LIKE, on the other hand, asymmetry was the best shape feature, with the remainder of no value.

Brightness-related features:

Edge-border coincidence (E&P) was by far the strongest single feature for both nodes involving target/non-target discrimination (OBJ and LARGE). For small targets, it provides nearly all the discrimination in the second stage. For large targets, it provides evidence which is well complemented by shape information--both must be included for adequate performance. Also very useful, particularly at stage 1, is E_p, which provides substantially different information from E&P. Gray level variance is used to some effect in the first classifier stage, but is not effective in the second stage. Perimeter contrast information appears to be much more effectively conveyed through E_p than dgl.

These rankings, while not dependable when taken alone, have been very helpful in suggesting which features could usefully be included in decisions at each node and which should be omitted. This was especially helpful in the case of the shape features, for which estimates of relative merit were not obtainable.

The final stage of feature testing was experimental. Features suggested either by Automask or by the problem definition were included in decision functions, and self-classification attempted. In many cases, the results were not satisfactory and one or more features were added or deleted until "good" results were obtained. If too many features were present in this classifier, features were removed until the best classification obtained with an

acceptable number of features was found.

8.4.2 Classification

The NVL data base as windowed for classification purposes consists of:

75 Tanks
34 Trucks
55 APC's
164 Target windows
10 Non-target windows
174 Total windows

Associated with each window was a liberal threshold range extending from the shoulder of the background peak gray level to the highest gray level at which there was significant sensor response. Although these ranges were manually selected, this is not a significant intereference with the automatic nature of the algorithm since the gray level ranges can be chosen by a simple scheme which identifies the background peak and proposes every threshold above the peak. (If a coarse temperature calibration is available, this task is even simpler.) See Section 8.4.3 for further discussion.

The Superslice algorithm was run on these windows using the selected gray level ranges. Connected components whose contrast, edge-perimeter match score and size were within tolerance were retained. The resulting sets of regions are described by the containment forests in Table 8.1. Within each containment tree, Superslice selects the best exemplar(s) for the candidate object region based on edge match. Thus, every tree has one or more best exemplars associated with it. All other (non-exemplar) regions are suppressed since the algorithm has proposed better representatives for classification.

Each containment tree is manually labelled as either "target-related" (containing regions associated with the target) or noise (spatially apart from a target region) so that false dismissals can be determined.

Of the 164 target windows, two windows (64T, 86A) had containment forests with no target-related regions present. At this stage, the false dismissal rate is 2/164 ~ 1% for Superslice. Determination of a false alarm rate is inappropriate since the discrimination performed by Superslice is "object vs. non-object,"

Window Reference Number	Lowest Threshold	Containment Forests
1T	23	X(N,TTT(PPPPPP,PP),NNN,N(N,NN),NN);NN
2T	23	TTTTTTTT
3T	25	TTTTTTT(PP,P);NN;NN
4T	30	TTTTTT
6T	25	TTTTTT
8T	26	TTTTT
9T	24	TTTTTT(P,P)
10T	25	TTTTTT
11T	25	TTTTTTTT;NN
12T	22	X(PPPPP(P,P(P,P)),N)
13T	20	XX(N,TTTT);N
14T	22	TTTTTTT
15T	30	TTTTT
16T	24	TTTTTTT
17T	26	TTTTT
21T	26	TTTTT
22T	25	TTTTTT
24T	29	TTTTT
26T	26	TTTTT
28T	27	TTT
31T	27	TTT
32T	21	X(TTTT,N,N)
33T	23	VTTTT;N
34T	26	TTT
35T	24	TTTT;N
38T	24	TTTTTTT
40T	23	TT;NN;N
42T	24	TTTTT(P,P(PP,PP))

Table 8.1. Containment forests of regions extracted by Superslice(Tanks). "AB" means that region A contains region B. "A(B,C)" means that region A contains the disjoint regions B and C. "A;B" means that A and B are disjoint regions in the window. Underlined letters denote "best" exemplars of the target region. Target trees begin at lowest threshold.

Legend:
- T target
- P partial target
- X target with additional noise
- O target invisible in noise
- N noise region
- F fiducial mark
- V target region not present at this threshold

43T	26	TTTTT
45T	25	TTTTTT
46T	26	TTTTTT
48T	24	TTTTTTTTT
50T	22	OTTTT
51T	24	TTTTT
52T	23	TTTTT
53T	23	TTTTT
54T	23	TTTTT
55T	23	TTTTT
56T	22	T;N;N
57T	22	TTT;NN;N
58T	20	X(NN,NN,TT)
59T	21	X(TT,NN);N;N
61T	43	TTTTTT
62T	24	TTTT;N
63T	24	TTT;N
64T	28	FFFFF;N;N (no target region found)
65T	46	T
66T	47	TT
68T	26	TTTT;N
69T	26	TTT;NN
73T	43	TTTTTTT
74T	45	TTTTT
75T	22	TTTTTT(P(P,P),P)
76T	23	TTTTTTTTTTT
78T	27	TTTTTTT(P,P)
79T	24	TTT(PPPP,PPP)
80T	22	TTTT(P,PPPPP(P(P,P),PP))
89T	23	TTTTTTTTTT
92T	22	TTTTTTTTTTTT
95T	24	TTTTTTTTTT
99T	21	TTTTTTTTTTTT
105T	24	TTTTTTTTTT(P,P,P)
109T	28	TT(PPPP,PPPPP,P(PPPP,PP))
110T	24	TTTTTT
114T	25	TTTTTTTTT
122T	20	TTTTT
123T	22	TTTTTTT
124T	21	TTTTTTT
125T	24	TTTTT
126T	23	TTTT
127T	24	TTTTTTTT
128T	23	TTTTTTT
129T	24	TTTTTTTT
130T	25	TTTTTT
131T	26	TTTTT

Table 8.1. Continued.

Window Reference Number	Lowest Threshold	Containment Forests
3R	23	X(TTT,NNN(NNNN,NNNNN));N
4R	22	TTTTTT;N;N;NNN
6R	23	OTTTTTT;NN
9R	23	X(TTTT,N(NN,N),NN,N,N)
18R	26	VTTT;N
22R	24	TTTTTTT
24R	28	X(X(TTT(PP,P),N))
26R	27	TTT;N
31R	26	OTTTT
32R	21	X(PP,N,N,N);NN
33R	23	X(X(TTT,N),N
34R	24	VVTTT;N;N;N
35R	23	TTT;N;N;N
41R	25	TTTTTTTTTTT
47R	25	TTTTTTTTTTT
51R	25	TTT;N;N
52R	23	TTT
53R	24	TT
54R	23	TT;N
55R	23	VTTT;N;N
56R	24	TTTTTT;NNN
57R	24	TTTT;NNN;N;NN
58R	24	TTTT;NN
59R	23	TTTT;NN;N;N
71R	44	TTTTT
72R	46	TTT;NN;N
77R	27	TTTTTT(P,P)
100R	23	TTTTTTT
104R	27	TTTTTTT
109R	27	TTT(P,PPP)
123R	27	X(TT(P,P),N)
133R	27	TTTT
134R	27	XTTT(P,P)
135R	26	TTTT

Table 8.1. Continued. (Truck windows)

Window Reference Number	Lowest Threshold	Containment Forests
21A	26	TTTTTT
22A	22	TTTTTTT
24A	28	TTTTTT
27A	27	VTTT;N
32A	25	TT
33A	25	T;N;N
34A	26	TTT
35A	25	TT
37A	27	TTTTTTTT
38A	23	TTTTT
42A	24	TTTT(PP,PP)
44A	28	TTTTTTTTT
45A	26	TTTT;N;N
46A	26	TTTTTT
48A	26	TTTTTTT
50A	24	TT
51A	25	TTTT;N;N
52A	25	TT;N;N
53A	24	TTT;N;N;NN
54A	25	TTT
55A	26	T
56A	25	TTTT
57A	24	TTTT
58A	25	TTTTT
59A	24	TTTTTTT
61A	41	TTTTTTT
73A	43	TTTTTT
74A	43	TTTTTTT
75A	25	TTTTTTT;N
76A	26	TTTTTTTTT
78A	31	P(PP,P)
79A	25	TTT(PPPP,PPP)
80A	24	TTTTTTT
86A	24	FFFFF;NN;N;N (no target related region found)
90A	25	TTTTTTTTTT
91A	26	TTTTTTTTT
93A	26	TTTTTTTTT(P,P)
94A	26	TTTTTTTT
96A	27	TTTTTT
97A	24	TTTTTTTT;N
98A	24	TTTTT
101A	44	TTTTT
102A	44	TTTTT;N
111A	24	TTTTTTTTT
112A	24	TTTTTTTTT
113A	23	TTTTTTTT
114A	29	X(TT(P,P),NN)
115A	24	TTTTTTTTT(P,P)
122A	23	TTT
123A	24	TTTTTT
125A	24	TTTT
127A	24	TTTT
129A	26	TTTT
130A	23	TTTTTT

Table 8.1. Continued. (APC windows)

not "target vs. non-target," and there is no ground truth for the number of objects (including targets, hot rocks, trees, etc.) in the frames.

The next stage - preclassification - performs possible-target vs. non-target screening. [For the purpose of building the screening criteria and subsequent classifier, a single exemplar per target was hand-chosen. No other target-related regions were considered; all noise regions, however, were retained.] Of the 162 target windows, the preclassifier retained 161 for a false dismissal rate of 1%. In addition, 44 noise exemplars also survived as possible targets. The false dismissal was 66T (small, very faint).

After preclassification, 150 selected target exemplars and all 44 noise exemplars were split into a training set (74 targets and 22 noise regions) and a test set (76 targets and 22 noise regions). The training set was used to design the optimum decision rule. It was felt that similar results in classifying both sets would then indicate that the classifier had utilized robust characteristics of the target class and thus could be expected to give similar results on further data of the same type.

A linear discriminant was used at the trucklike node while a maximum liklihood discriminant was used at the small target/non-target node. Five features were used at both nodes, of which four were the same: (h*w)/A, (h+w)/P, asymmetry, E&P. The fifth feature was diff for the small target discriminant and skewness for the truck/APC discriminant. The large targets are divided into three classes (tank, truck/APC, other) by a quadratic maximum likelihood discriminant using six features: (h/w)', (h*w)/A, diff, skewness, E&P and E_p. Two different procedures for classifying large regions (> 94 pixels) were tested. One procedure attempted to discriminate between four fixed classes (tank, APC, truck, other); the other procedure used three classes (tank, APC, truck) and two "reject" categories (non-target, unidentified target). Both used identical polynomial maps into decision space. In the latter classifier, however, the maximum likelihood class assignment of a region had to be significantly better than for random noise regions (otherwise, the non-target class was assigned) and significantly better than the next best target class assignment (otherwise, it was called an unidentified target).

The detection results using the fixed class classifier on the

150 selected target exemplars are summarized by:

	Train	Test	Total
Large	53/53	53/55	106/108
Small	20/21	20/21	40/42
Total	73/74	73/76	146/150

where "M/N" means "M successes out of N tries." This classifier thus appeared to be robust.

Table 8.2 displays the results of this classifier for all extracted regions, including all target and noise exemplars. A false dismissal for a window containing a target occurs when no target exemplar (at any of the thresholds) is classified as a target (i.e., classified as tank, truck, or APC). Similarly, a false alarm is any noise exemplar (i.e., not associated spatially with a target region) classified as a target. However, multiple exemplars for the same noise region are counted only once. In effect, we are counting the image regions (as opposed to exemplars) which are classified as target regions by at least one exemplar. If a region is, in fact, a target region and some exemplar of it is called a target, that is a success. If no exemplar is so called, then a false dismissal has occurred. Finally, if the so-called target region does not, in fact, contain a target, then a false alarm has occurred.

The classifier results consist of 6 false alarms and 3 false dismissals from the 162 target windows and 2 more false alarms from 10 non-target windows. No window contained more than one false alarm cue. Details are as follows:

False Dismissals	False Alarms
32R	3T
35R	11T
33A	3R
	56R
	59R
	86A
	2N
	8N

	Thresholds																													
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
1T				[0	0	+	+	+	+	+	0	0	0]																	
2T				[+	+	+	+	+	+	+	+]																			
3T						[0	0	0	0	±	±	+	+	+]																
4T											[+	+	+	+	0	0]														
6T						[+	+	+	+	+	+]																			
8T							[+	+	+	+	+]																			
9T					[0	0	0	0	+	+	0]																			
10T						[0	0	0	0	+	+]																			
11T						[+	±	+	+	+	+	+	+]																	
12T			[0	+	0	+	0	0	0	0]																				
13T	[0	0	+	+	+	0]																								
14T			[+	+	+	+	+	+	+]																					
15T											[+	+	+	+	+]															
16T					[+	+	+	+	+	+	+]																			
17T							[+	+	+	+	+]																			
21T							[+	+	+	+	+]																			
22T						[+	+	+	+	0	+]																			

Table 8.2 Region classification (tank windows).

Each entry represents the outcome of the classifier for the purpose of target detection. Brackets indicate the range of thresholds considered for each window. "+" means that the target was detected at that threshold. "0" means that the target was dismissed. "-" indicates a false alarm for that threshold. No window had two or more distinct false alarm regions.

	Thresholds																														
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
24T										[+	+	+	+	+]															
26T							[0	+	+	+	0]																		
28T							[	+	+	+	]																		
31T							[	+	+	+	]																		
32T	[		0	0	+	+	0	]																					
33T				[	+	+	+	+	]																				
34T						[	+	+	+]																				
35T					[0	+	+	+]																					
38T					[0	0	0	+	+	0	0	]																	
40T			[	+	+			]																					
42T					[+	+	0	0	+	+	0	0	]																
43T							[+	0	+	+	+]																		
45T						[+	0	+	+	+	+]																		
46T							[0	0	+	+	+]																		
48T					[+	+	+	0	+	+	0	+	0	]															
50T			[+	+	+	+	0]																						
51T					[+	+	+	+	+]																				
52T				[+	+	+	+	0]																					
53T				[+	+	+	+	+]																					
54T				[+	+	+	+	+]																					
55T				[+	+	+	+	+]																					
56T			[+		]																								
57T			[+	+	+		]																						

Table 8.2. Continued.

	Thresholds																												
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
58T	[0	+	+			]																							
59T		[0	+	+		]																	[+	+	+	+	+	+	
61T																													
62T					[+	0	+	+]																					
63T					[+	+	+]																						
64T									[	0	0	+	0	+	+]														
65T																											[+]		
66T																												[0	0]
68T							[+	+	+	0]																			
69T							[+	+	+	0]																			
73T																								[+	+	+	+	+	+]
74T																										[+	+	+	+]
75T			[+	+	+	+	+	+	+	+]																			
76T				[0	+	+	+	+	+	+	+	+	+	+	0]														
78T								[+	+	+	+	+	0	+	0]														
79T					[+	+	+	+	+	0	+]																		
80T			[+	+	+	+	+	+	+	+	+	+	+]																
89T				[+	+	+	+	+	+	+	+	+	+]																
92T			[+	+	+	+	+	+	+	+	+	+	+	0]															
95T					[+	+	+	+	+	+	+	+	+	+]															
99T		[0	0	+	+	+	+	+	+	+	+	+	0]																
105T					[+	+	+	+	+	+	+	+	+	0	0]														
109T									[+	+	0	0	0	0	0]														

Table 8.2. Continued.

Frame	Thresholds 20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
110T					[0	+	+	+	+	+	+]																		
114T						[+	+	+	+	+	+	+	+	+	0]														
122T	[+	0	+	+	0]																								
123T			[+	+	+	+	+	+	0]																				
124T		[0	+	+	+	+	+	0	0]																				
125T					[+	+	+	+	0]																				
126T				[0	+	+	+	+]																					
127T					[+	+	+	+	+	+	+	0]																	
128T				[	+	+	+	+	0	0]																			
129T				[	+	+	+	+	+	+	+	0]																	
130T						[+	+	+	+	+	0]																		
131T							[0	+	+	+	0]																		

Table 8.2. Continued.

Frame	Thresholds 20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
3R				[+	+	±	0				–	–			]														
4R			[+	+	+	0	0	0		]																			
6R				[+	+	+	+	0	+	+]																			
9R				[0	+	+	0	+		]																			
18R							[	+	+	0	]																		
22R					[0	0	+	0	+	0]																			
24R									[+	+	+	0	+	0	0]														
26R								[	0	+	0	]																	
31R							[	+	+	+	0]																		
32R		[0	0	0		]																							
33R				[	0	0	+	+]																					
34R					[		0	+	+]																				
35R				[0	0	0]																							
41R						[0	+	0	+	+	+	0	0	0	0	0	]												
47R						[+	+	+	+	+	+	+	+	+	0	0	]												
51R						[0	+	0		]																			
52R				[0	0	+		]																					
53R					[0	+	]																						
54R				[0	+	]																							
55R				[	0	+	0]																						
56R					[+	+	0	0	0	+]																			

Table 8.2 (continued): Truck windows.

	Thresholds																													
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	
57R					[+	0	+	+		]																				
58R					[+	+	+	+		]																				
59R				[0	+	+	0		]																					
71R																									[+	+	+	+	0]	
72R																											[+	+	0]	
77R								[+	+	+	+	+	+	0]																
100R				[0	+	+	+	+	0	+]																				
104R								[+	+	+	+	+	+	+]																
109R								[0	+	+	+	+	0]																	
132R								[+	+	+	0]																			
133R								[+	+	+	0]																			
134R								[+	+	+	0]																			
135R							[+	+	+	+]																				

Table 8.2. Continued.

Frame	Thresholds 20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
21A							[+	+	+	+	0	0]																	
22A			[+	+	+	+	0	0]																					
24A									[+	+	+	+	+		]														
27A							[0	+	+	+	]																		
28A								[	+	+	0]																		
32A						[+	+		]																				
33A						[0			]																				
34A							[0	+	+		]																		
35A						[0	+			]																			
37A								[+	+	+	+	+	+	+	+	]													
38A				[+	+	+	+	0	]																				
42A					[+	+	+	+	+	0	]																		
44A									[+	0	+	+	+	0	+	0		]											
45A							[+	+	+	+	0		]																
46A							[0	+	+	+	+	0	]																
48A							[+	+	+	+	+	+	0]																
50A					[+	0		]																					
51A						[0	+	+	0	]																			
52A						[0	+			]																			
53A					[+	+	+]																						
54A						[+	0	0	]																				
55A							[+			]																			
56A						[+	0	+	0	]																			

Table 8.2 (continued): APC windows.

Frame	Thresholds 20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
57A					[+	0	0	0]																					
58A						[+	0	+	+	+]																			
59A					[0	+	+	+	+	+	0]																		
61A																						[+	+	+	+	+	+	+]	
73A																								[+	+	+	+	+	+]
74A																								[+	+	+	+	+	+]
75A						[+	+	+	+	0	0	0]																	
76A							[+	+	+	+	+	+	+	+	+]														
78A												[+	0	0]															
79A						[0	+	+	+	+	0	+]																	
80A					[+	+	+	+	+	+	+]																		
86A					[_	_				]																			
90A						[+	+	+	+	+	+	+	+	0	0]														
91A							[+	+	+	+	+	+	0	+	+]														
93A							[+	+	+	+	+	+	+	+	0]														
94A							[0	+	+	+	+	+	+	+]															
96A								[+	+	+	+	+	+]																
97A					[+	+	+	+	+	+	+	+]																	
98A					[+	+	+	+	+]																				
101A																									[+	+	+	+	+]
102A																									[0	+	+	+	+]
111A				[+	+	+	+	+	+	+	+	0]																	
112A					[+	+	+	+	+	+	+	0	0]																
113A				[+	+	+	+	+	+	+	0]																		
114A										[+	+	+	+]																
115A					[+	+	+	+	+	+	+	+	+	0]															

Table 8.2. Continued.

	Thresholds																													
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
122A				[+	+	0]																							
123A					[+	+	+	0	+	0]																			
125A					[+	+	+	+]																					
127A					[+	+	+	+	]																				
129A							[+	+	+	0]																			
130A				[+	+	+]																							

Table 8.2. Continued.

	Thresholds																												
Frame	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
2N								[	_			_	_	]															
8N								[				_			]														
14N	[			]																									
20N							[								]														
26N		[				]																							
32N			[			]																							
38N	[		]																										
44N			[				]																						
50N			[			]																							
56N	[				]																								

Table 8.2 (continued): Noise windows.

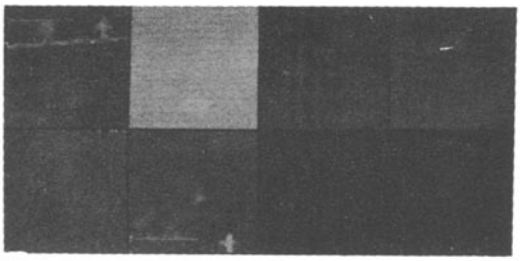
Figure 8.3a displays the 6 (total) false dismissals. Masks of the 8 false alarms along with their gray level windows are shown in Figure 8.3b.

The question of how target identifications can be made in this environment of multiple exemplars, while secondary to the task of detection, is an interesting one. Since each exemplar in a containment tree can be classified independently, there are many ways of arriving at a final region label. Section 8.5 discusses the use of context and considers the identification of object regions from the classifications in their containment trees as an example of context. We discuss the issue here simply from the point of view of critiquing the classifier performance. For each containment tree containing at least one exemplar as a target, we chose the target type of the exemplar with the best edge-match (E&P) score in the tree and used that target type to designate the region. In the event that the "best" exemplar was not described as a target, we labelled the object region "unknown target". Only large targets were considered, since small targets while detectable were not considered identifiable.

In a test which classified all best exemplars of large targets (55 tanks, 21 trucks, 36 APC's) the between-types confusion matrix was:

		classified as			
		T	Tr	A	UT
A priori	T	40	5	6	4
	Tr	6	8	7	0
	A	9	5	20	2

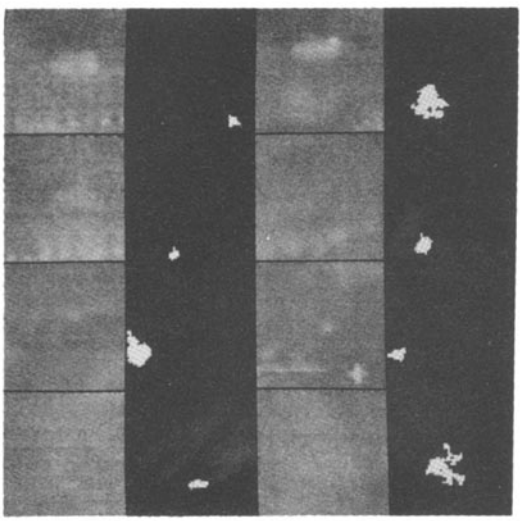
where "UT" is the "unknown-target" type. The 8 false alarms were classified as 1 truck, 2 APC's, and 5 small targets. Between-class confusion is high, with tanks being the most successful class. Trucks and APC's were often confused with tanks. A number of reasons can be advanced for this performance. First, tanks were the most numerous target and therefore could be identified most confidently. Second, large APC's appeared with the wooden wave deflection board in view, producing a characteristic "c" shape. No attempt was made to utilize this special knowledge.



64T 66T 32R 35R

36A 86A

a.



3T 11T

3R 56R

59R 86A

2N 8N

b.

Figure 8.3. Classification results for NVL data base.

a. Six false dismissals.
b. Eight false alarm region masks with their gray level windows.

Third, the large targets appeared in only a single aspect and no generalized shape descriptors separating the different types could be extracted reliably. It seems most sensible to model the target types as three-dimensional objects and to derive discriminators from their inherent shape and size differences from all aspects.

The second classifier (which applied a threshold to reduce the false-alarm rate) did not improve classification as might have been expected. Any threshold which would have reduced the number of false alarms also caused a number of false dismissals. Thus while the method might be of use, its utility could not be judged on the limited data set available especially since there is no model relating the false alarm rate to the false dismissal rate.

We may summarize the principal classification results as follows: the false dismissal rate of the system is less than 4%, giving a system detection rate of 96%. The false alarm rate, based on the number of false alarm regions per unit area, is 8 false alarms in 174 (128x128) windows. Assuming there are 500x800 pixels per frame and that a target occupies about 1/10 of a window, we conclude that the total processed area corresponds to about 6 frames. Thus the false alarm rate is 8/6 or 1.3 per frame. A separate test of the false alarm rate was made using a set of four 512x512 pixel frames (Figure 8.4). All available targets were detected. In addition, 4 large false alarms and 8 small false alarms were detected (see Figure 8.5). However, 5 of the 8 small false alarms corresponded to fiducial marks. Moreover, one large false alarm (in F1) appears to be a target. In any case, 7 false alarms in 4 frames agrees well with the previous estimate of the false alarm rate.

8.4.3 Threshold selection evaluation

Our method of threshold range selection was described previously. However, it bears repetition in this section. Using the histogram of gray levels (perhaps of the previous image), choose as a range the sequence of gray levels from the mode to the highest gray level with appreciable response (e.g., more than 5 points). The previous subsection demonstrated that this brute force approach gave excellent system detection efficiency. Naturally, the liberal range of thresholds has important effects on system architecture.

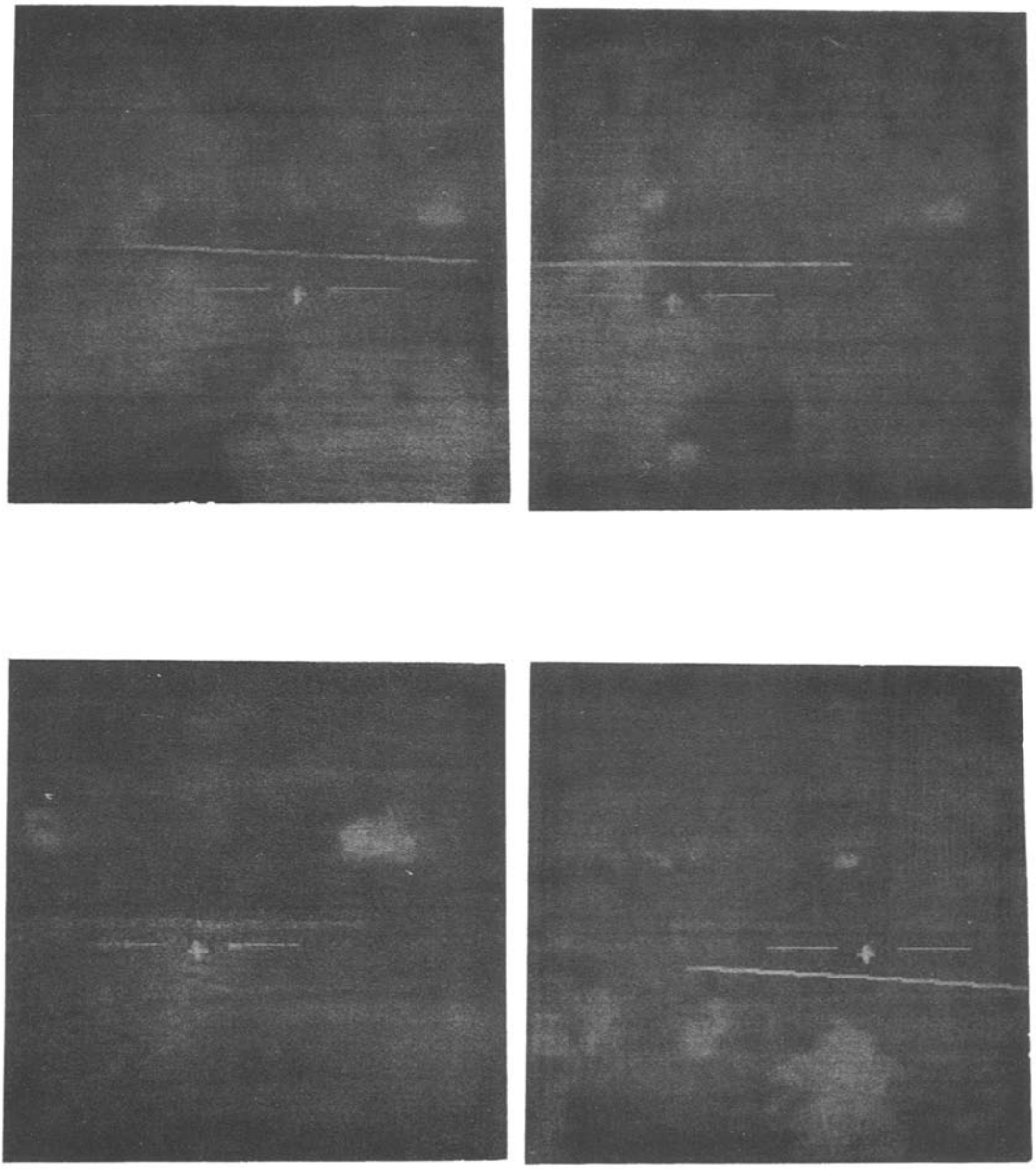


Figure 8.4. Four 256x256 frames (after median filtering and sampling).

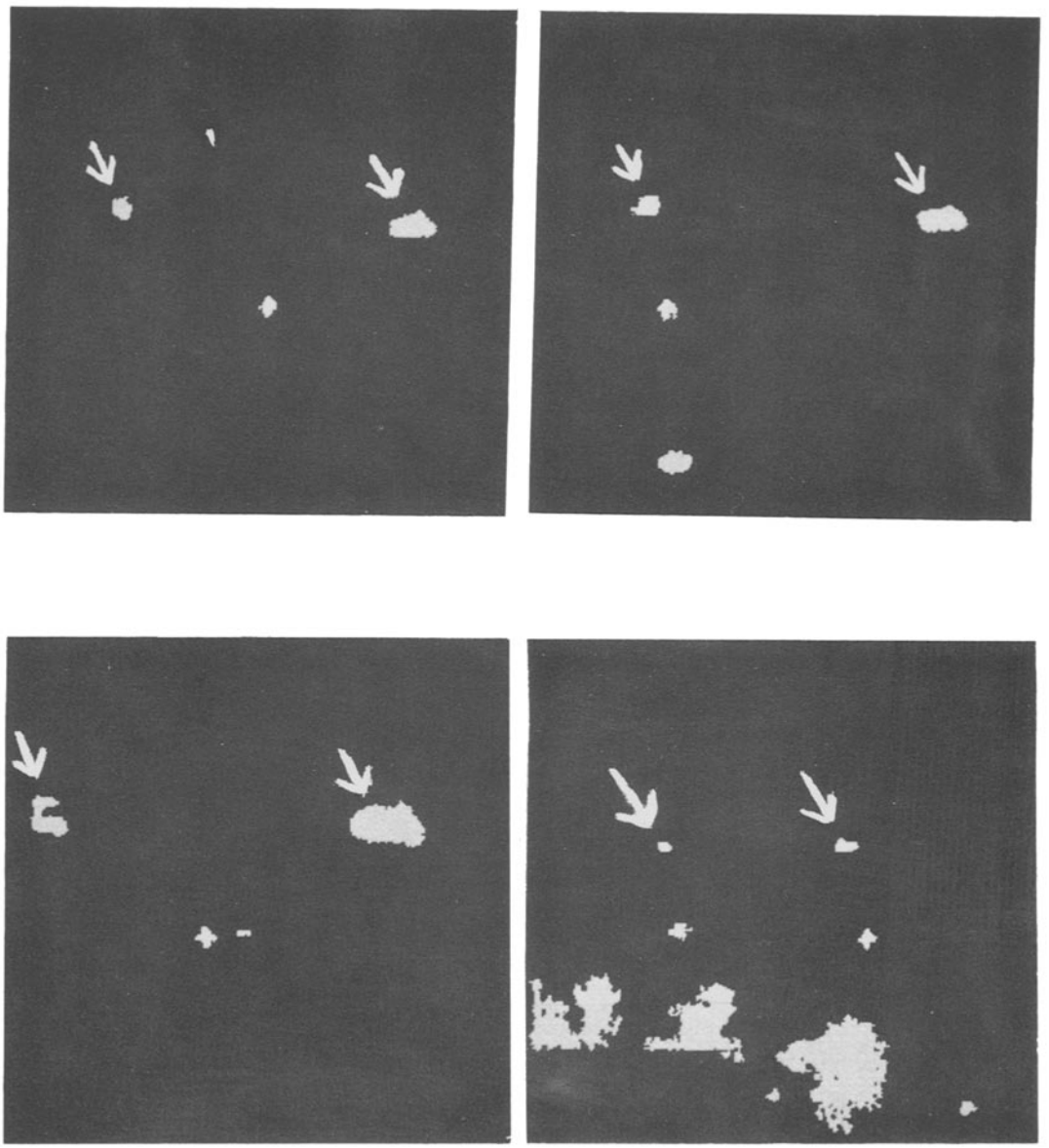


Figure 8.5. Cued regions in the four frames of Figure 8.4. All targets were detected (masks indicated with arrows), along with 12 false alarms (5 corresponding to fiducial marks).

Since the number of thresholds used determines the time cost (in a sequential implementation) or the hardware replication cost (in a parallel implementation), it is appropriate to consider methods which can accommodate a limited number of thresholds. "Intelligent" methods of threshold selection are discussed in Sections 4 and 9. We wish to consider "brute force" methods which select thresholds at every gray level, at every third gray level, etc.

As may be seen from Table 8.2, correct target detections for single windows tend to occur in extended runs. Table 8.3 provides a histogram of run lengths. In general, large targets had better constrast and their detections were stable over long runs. Small targets were fainter and were detectable over only a few thresholds at most. Table 8.3 shows what percentage of the targets were detected within runs of I or longer for I = 1,2,... . Thus the false dismissal rate would be 11% if every other threshold in the range were omitted. Since there were so few false alarms, it is not possible to give comparable statistics of any reliability, but any scheme which considers fewer exemplars is bound to detect fewer false alarms.

From a slightly different point of view, we might consider how to allocate a fixed number of thresholds within a given gray level range. Suppose that five thresholds are implemented in parallel hardware. Thus, for a gray level range of 10, thresholds would occur at every other gray level; for a range of 20, thresholds would occur at every fourth gray level. If we use the gray level ranges indicated by brackets in Table 8.2 and distribute N (=1,2,3,...) thresholds equally spaced (where feasible) throughout the range, we compute the following results:

N	# False Dismissals	# False Alarms
1	25	1
2	14	3
3	7	7
4 and above	5	8

Thus, for four or more thresholds equally spaced throughout the available gray level range of each window, no additional false dismissals occurred beyond those already dismissed using the whole

Run length	# of windows	Cumulative count	% of 164 windows
0	5	164	100
1	17	159	97
2	25	142	87
3	29	117	71
4	27	88	54
5	19	61	37
6	12	42	26
7	10	30	18
8	8	20	12
9	7	12	7
10	3	5	3
11	2	2	1

Table 8.3. Statistics of longest runs of correct target detections in 164 target windows.

range. Interestingly, for small N the increase in false dismissals is just about compensated by the decrease in false alarms. One is doubled as the other is halved.

Naturally, the threshold ranges depend both on window size and on window content. It is therefore not likely that three thresholds will be sufficient in practice. The best choice of N, the number of thresholds, will result from estimating the probability/cost tradeoff for faint targets. Given a range of x gray levels for target regions, N should be about x/2 or x/3, which for the current data base suggests that N should lie between 5 and 10. For an extension to image sequences, see Section 9.1.

8.4.4 Classifier extension

An attempt was made to apply the classifier derived from the NVL data base to a different set of thermal images. The Alabama data base is a set of imagery taken with a thermoscope. The actual sensor data are classified; radiometric noise was added to mask the source. Figure 8.6 exemplifies the type of imagery involved. The gray level histograms are not smooth and in some cases runs of gray level bins contain no points. Median filtering (using odd sizes) cannot be used to smooth such images since it preserves false contours. Median filtering using even sizes provides a small degree of smoothing. We elected to smooth by locally averaging over a 2x2 neighborhood just to introduce sufficient gray level variation so that 5x5 median filtering would be effective.

The resultant images were windowed and threshold ranges were selected. The Superslice algorithm was then applied in order to extract candidate object regions. It was necessary to increase the contrast threshold since the inherent contrast (including false contours was higher than in the NVL data base. With this adjustment, the Superslice algorithm extracted regions corresponding to 64 out of 65 targets. After classification, 60 out of 65 were detected. In addition, there were 3 false alarms in the 48 64x64 windows considered (although one of the false alarms appears to be a target missing from the ground truth).

8.5 Classification and context

Our approach to the target cueing problem has been to extract

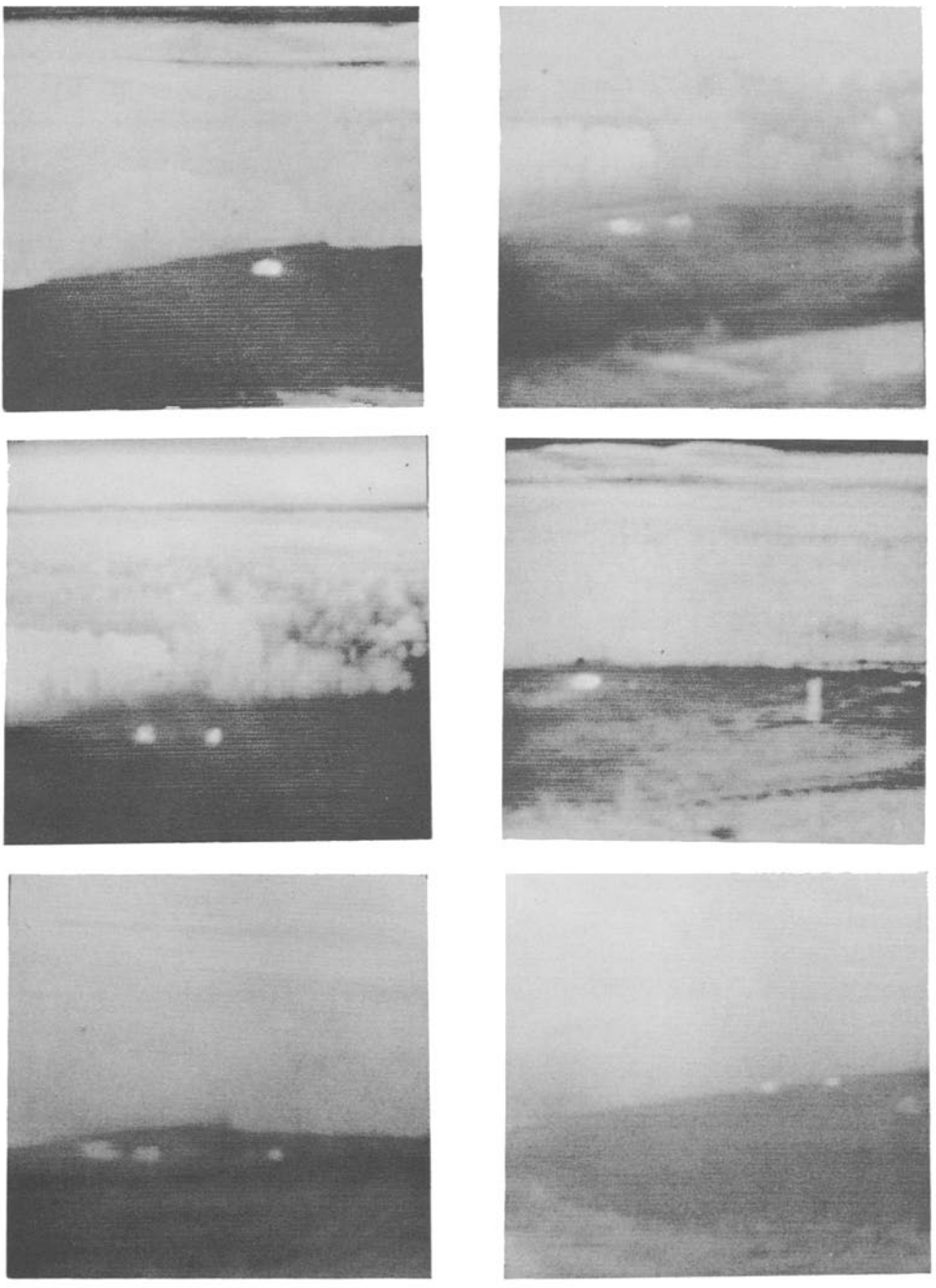


Figure 8.6. Alabama data base (selected frames).

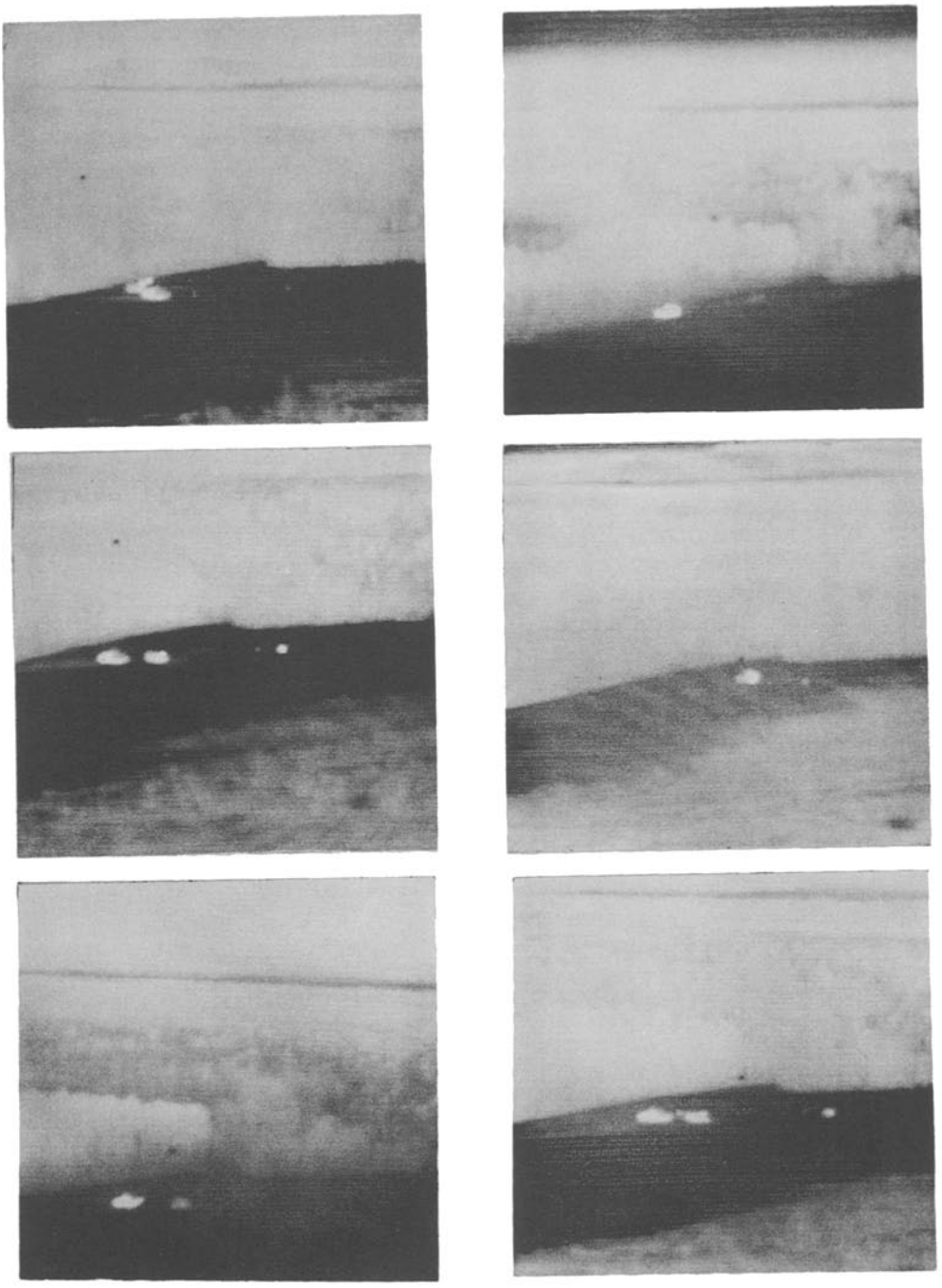


Figure 8.6 (continued)

and classify object regions independently of one another. That is, segmentation is based on the assumption that the object regions are individually thresholdable, though not necessarily by the same threshold. Classification is based on information derived from measurements on the individual components but does not take into account the intra- and inter-frame context of a region.

The Gestalt laws of grouping (see [26]) are of interest in this respect since they refer to factors that cause some parts to be seen as belonging more closely together than others. These rules are applications of the basic principles of similarity which assert that region association is partly defined by region resemblance.

There are several types of similarity which could be used with FLIR imagery, e.g. similarity of appearance (size, shape, brightness, etc.), similarity of location or proximity, similarity of spatial arrangement, and temporal similarity (multiple views of the same object in different frames).

Whenever one can confidently group a set of N objects as being similar (based on one or more of the types of context discussed above), it may be advantageous to classify them collectively (The Compound Decision problem) rather than independently (The Simple Decision problem).

The compound decision problem can be stated briefly as follows:

There are a set of states of nature $\Omega = \{1,2,\ldots\gamma\}$ and a set of actions $A = \{1,2,\ldots s\}$, associated with an rxs loss matrix L_{ij} being defined for every $i \in \Omega$ and $j \in A$. When the same decision problem is confronted N times, there exists a vector $\vec{\theta}_N = \{\theta_1,\theta_2,\ldots\theta_N\}$ of states of nature where $\theta \in \Omega^N$ and a corresponding vector $X_N = \{x_1,x_2,\ldots x_N\}$ of random variables. θ_k denotes the state in the kth problem and the distribution of x_k is $P(x_k|\theta_k)$. For a given θ_k, x_k is independent of other x's and θ's. In other words

$$P(\vec{x}_N|\vec{\theta}_N) = \prod_{j=1}^{N} P(x_j|\theta_j)$$

We do not assume that the θ's are necessarily independent.

The loss in the compound decision problem is taken to be the average of the losses incurred at each of the N decisions and the compound risk is defined correspondingly.

If all the observations $\vec{X}_N$ are at hand before the individual decisions must be made, one can use a compound decision rule $\vec{t}_N = \{t_1, t_2, \ldots t_N\}$ where $t_k = t_k(j|\vec{X}_N)$ for each $\vec{X}_N$ is a distribution over A, according to which the kth action is chosen. Also one can define a sequential compound decision rule if only the observations $\vec{x}_k$ are at hand before the kth decision is made.

It is possible to work out a decision procedure which is compound Bayes against the distribution $G(\vec{\theta}_N)$ where $\vec{\theta}_N \in \Omega^N$ (for the details see Abend [27]).

It would be desirable, in principle, to make effective use of context in general and of the compound decision rule in particular as a way of combining related observations. Naturally, this would require a data base which is sufficiently structured to provide the necessary context. However, a recasting of the problem makes another type of context available.

Consider a set of nested regions (exemplars) produced by the Superslice algorithm. We wish to investigate how these regions can be treated in ensemble as defining (perhaps) a target region. This suggests the following experiment: Given a set of object regions generated by Superslice, classify them independently. Choose a nested region of significance: namely, a subtree in the containment forest (corresponding to a given window or frame) all of whose paths from the root to the terminal nodes are of length $\geq nt$ where $0 \leq t \leq 1$ and n is the number of thresholds used by Superslice. This insures that for a proper choice of t we only consider nested regions which keep appearing for a large fraction of the total number of thresholds.

For each such nested region (subtree), suppose that there is a class, say, w (tank, APC, truck, or noise) such that M of the N objects in the subtree have been assigned to w and $M \geq t\ell$ where ℓ is the length of the longest path in the subtree. (This rule insures that for a proper choice of t the chosen class w really dominates the subtree.) We then assign class w to all N objects in the subtree. Otherwise, we leave the classifications unaltered.

In an experiment using the NVL data base, 315 objects generated by Superslice from 52 windows were considered. The objects were hand picked to belong to the a priori classes tank, APC, truck and noise, and were then classified into five classes, viz. Tank, APC, Truck, Small target, and Noise. The corresponding

confusion matrix is shown in Table 8.4a.

We then applied the majority logic context rule on all the containment forests (52 of them) for t = .5; the resulting confusion matrix is shown in Table 8.4b.

A comparison between the two matrices shows an improvement in the false dismissal rate. The false alarm rate is left unchanged, since no significant nested regions (for t = .5) could be found where the noise class dominated the target class. Within the target classes we find a marked improvement in the self-classification of tanks and APC's. However, more trucks in the second case have been misclassified into APC's. This is presumably not due to an error in the majority logic rule, but rather due to the inability of the classifier to discriminate trucks from APC's.

The majority logic context rule is not necessarily a superior classification procedure, since Superslice considers only the best exemplars and may therefore produce a better classification. However, the present study does support the relevance of low-level context for classification validation.

9. The Dynamic Environment

The work described heretofore has considered the analysis of single frames. However, inasmuch as the sensor is capable of generating 30 frames per second and the hardware is capable of analyzing about 3 frames per second, it is worthwhile to investigate how information culled from sequences of frames can improve the performance of the system. There are two ways in which sequence data can be helpful. First, a high scanning rate allows a succession of views of the same scene with only a small amount of change (dependent on platform motion). Thus, image statistics should be relatively stable and multiple measurements may allow a reduction of the standard deviation of feature values. Second, the use of motion information can provide a better description of the object regions in a scene. For this project only a small data base of ten sequential frames was available (Figure 9.1). The image content and quality are similar to those of the NVL data base. The sequence corresponds to every other frame from the FLIR sensor over a span of 2/3 of a second. The images show a tank against a background of trees, and fade away more with each frame. While this data base was not large enough to permit meaningful tests, it

A Priori		Classified as				
		Tank	APC	Truck	Small Target	Noise
	Tank	28	1	2	4	19
	APC	10	26	15	35	22
	Truck	6	10	10	27	23
	Noise	6	1	1	7	62

Table 8.4a. Independent classification confusion matrix

A Priori		Classified as				
		Tank	APC	Truck	Small Target	Noise
	Tank	40	1	0	0	13
	APC	13	38	11	30	16
	Truck	6	15	6	27	22
	Noise	6	1	1	7	62

Table 8.4b. Majority logic classification confusion matrix

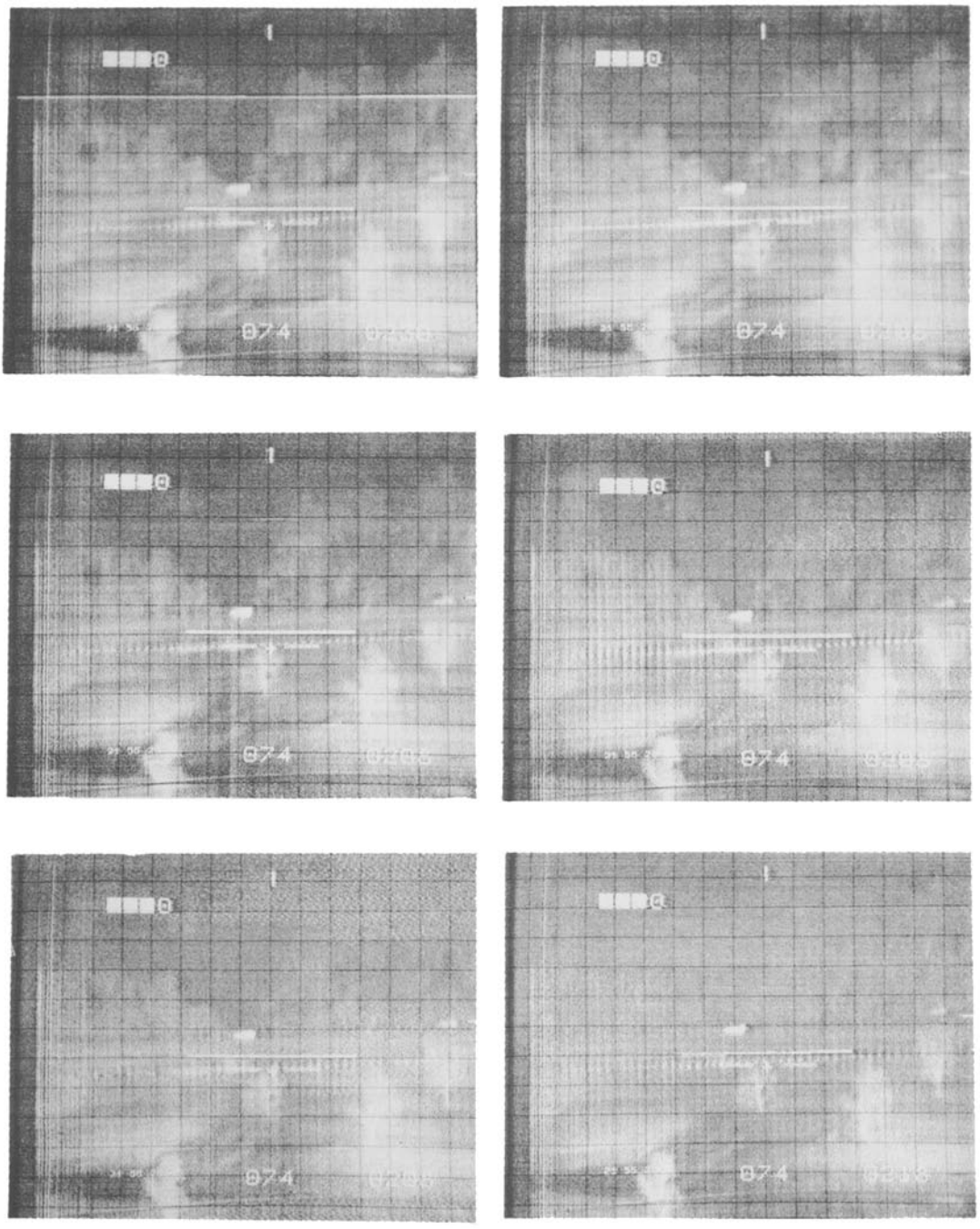


Figure 9.1. Ten sequential frames.

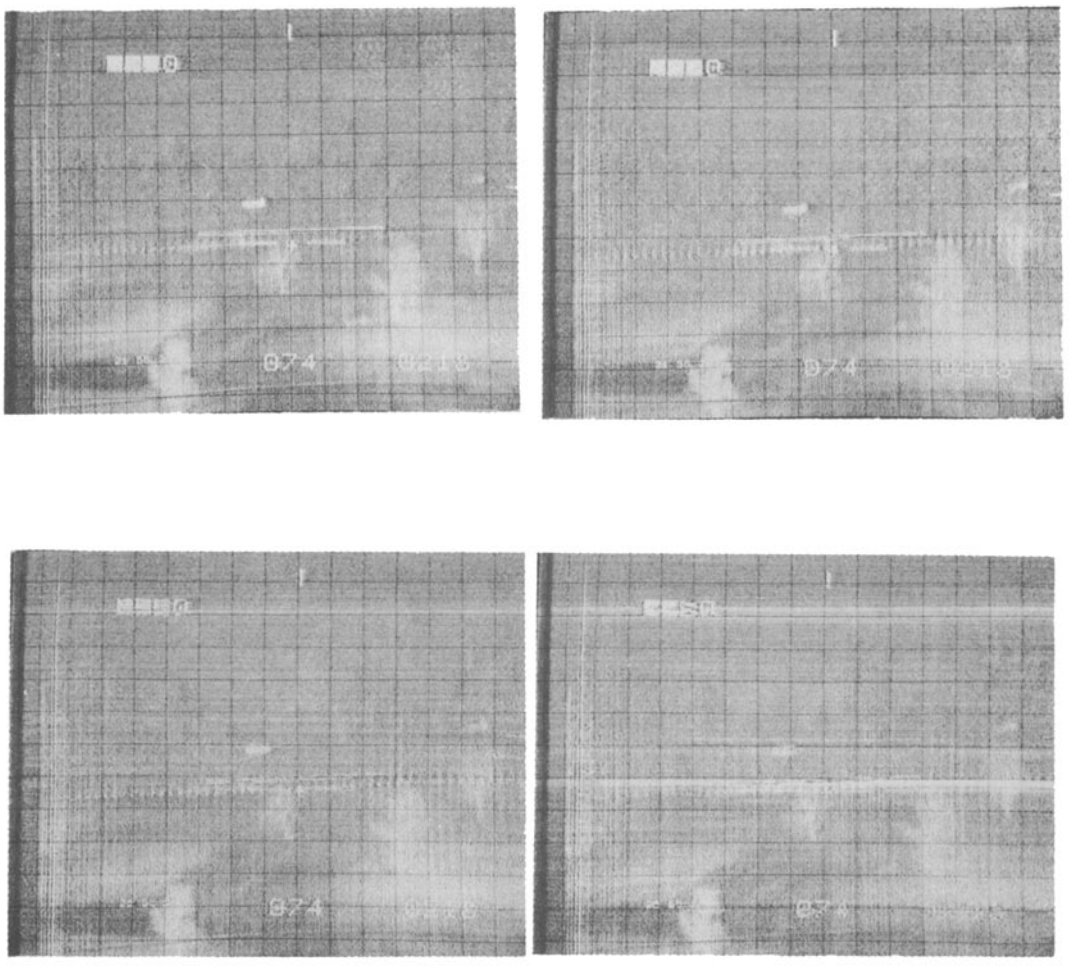


Figure 9.1.(continued)

did allow some exploratory work.

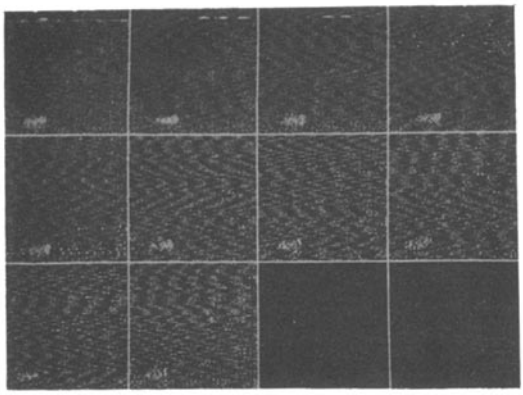
9.1 Threshold selection

One does not expect (time-) adjacent frames to differ radically and therefore it should be possible to use good thresholds from the previous frame to segment the current frame or at least to guide the selection of thresholds in the current frame. A sequence of 10 windows was extracted and smoothed (Figure 9.2) and a best threshold was chosen for each. Figure 9.3 shows the effect of choosing a lower threshold or a higher threshold. As may be noted, the adjacent thresholds have a fairly negligible effect on the target region although there is a sizable change in the amount of noise (which can be eliminated by shrink/expand noise cleaning.) However, if one considers the sequence of best thresholds as determined by the border/edge match score (Table 9.1), there is a large shift (from gray level 27 to 17) even in this short sequence of frames. Thus no single threshold is appropriate for the whole sequence. Nonetheless, the previous threshold when used on the current frame is a fairly good choice. This suggests the following approach: In a single pass over the frame, segment the current frame using the best threshold(s) from the previous frame and simultaneously compute the best threshold(s) for this frame (to be applied to the next frame in sequence). The advantage of this scheme is that the frame is not stored, thereby realizing a considerable saving in chip size and complexity.

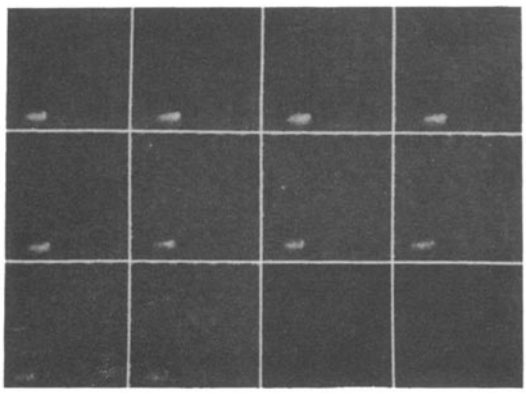
A somewhat different approach attempts to distribute N thresholds across the threshold range dynamically. Suppose the threshold range is X gray levels. It would take X/N frames to investigate each threshold in the range. However, as mentioned earlier, X/N is likely to be ≤ 3. Thus the entire gray level range capable of harboring targets can be sampled every 3 frames. At a projected processing rate of 3 frames per second, the range would be sampled once per second. A hybrid approach is also appropriate, devoting K of N thresholds to the most likely gray levels and letting N-K thresholds "rove" over the rest of the applicable gray scale.

9.2 Region tracking

The Superslice algorithm builds a forest-like structure of



a.



b.

Figure 9.2a. Ten 64x64 windows from the sequential data base.

b. 5x5 median filtered windows of 128x128 originals, then sampled 2 to 1.

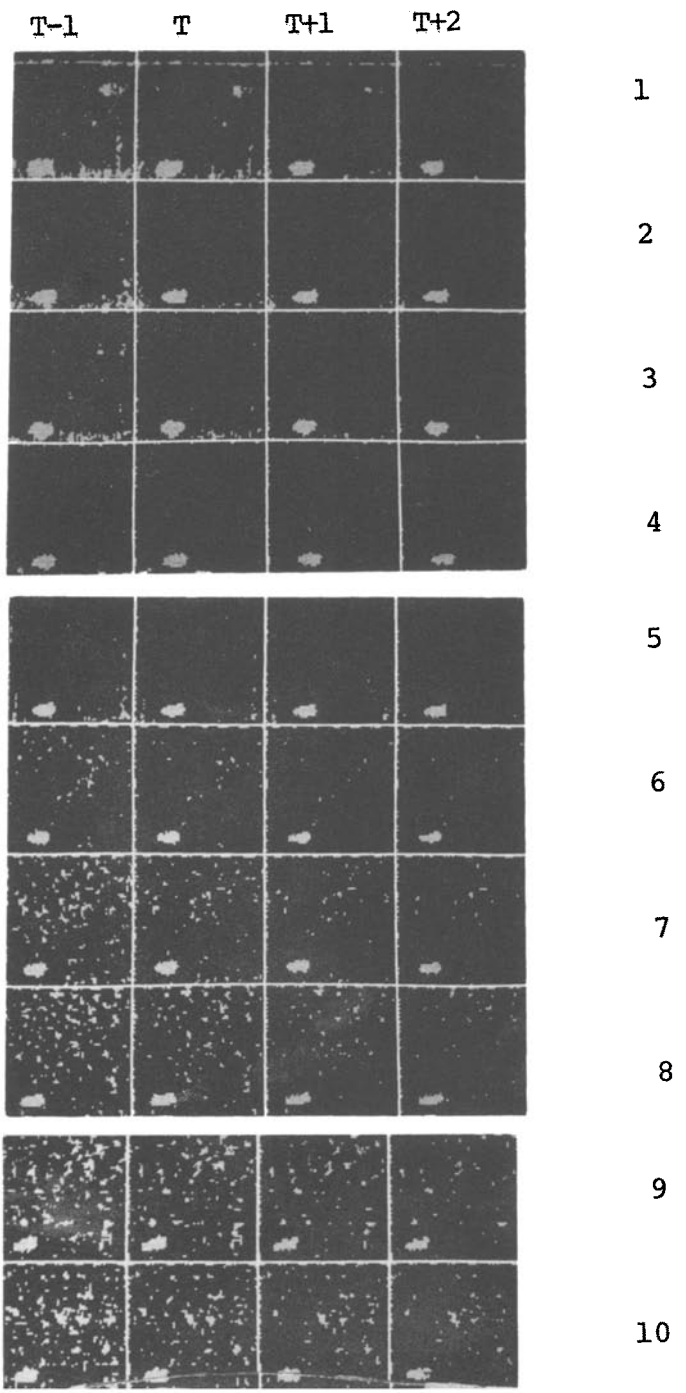


Figure 9.3. Effect of choosing lower or higher thresholds. The column labeled T shows the result of applying the chosen threshold to each window in the sequence. Columns T-1, T+1, T+2 show the results of using thresholds 1 lower, 1 higher, and 2 higher, respectively.

Threshold	Sequential Window #									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
14	26	29	34	47	35	49	54	74	68	69
15	34	29	42	35	43	48	54	88	82	79
16	52	35	48	49	43	64	72	88	83	72
17	60	43	52	51	57	74	81	84	82	81*
18	53	51	72	58	72	72	90	90*	84*	69
19	54	53	76	67	70	76*	93*	87	71	65
20	59	60	85	75	72	59	89	66	75	65
21	67	67	87*	85	72	59	88	66		62
22	67	66	87*	100*	75*	50	80	63		50
23	67	70	87*	100*	68	42	80	63		
24	67	72	87*	97	68	38	80	63		
25	71	70	79	97	73	33	83	61		
26	76	76	81	85	69					
27	79*	79*	62	58	68					
28	76	77	64	54	68					
29	75	75	62	43	52					

Table 9.1. Percentage border/edge match as a function of threshold for the sequence data (maxima indicated with "*").

regions from each frame. Within each structure, a sequence of nested regions which are roughly similar in size (but arising from different thresholds) constitutes a set of exemplars of a possible object. In addition, a certain number of accidents tend to be present. Regions of either type are called "candidate object regions". The frame to frame tracking process attempts to discover consistent temporal sequences of candidate object regions by selecting one exemplar per candidate object per frame, according to a dynamic programming model (see [28]).

Two evaluation functions, S and D, are used. The static evaluation function S(c) defines a figure of merit for each candidate object region c. The Superslice algorithm provides such a figure of merit based on contrast and well-definedness. The assumption is that the best exemplar for an object region is identified by having the greatest figure of merit. The dynamic evaluation function D(c,c') defines the similarity of one candidate object region (c) to another (c'). This is evaluated by considering the scaled differences between the feature vector of c and that of c'. If c is a perfect exemplar then S(c) = 0 and D(c,c) = 0.

Let $\{c_{ij};\ j = 1,\ldots,N_i\}$ be the set of candidate regions in the ith frame, i = 1,...,M. We define the dynamic programming problem as: find $\{c_{i\pi_i};\ i = 1,M\}$ such that $T(C_{M\pi_M})$ is minimum over all selection functions, π. The solution is achieved by the following:

Basis step: $T(c_{1j}) = S(c_{1j});\ j = 1,\ldots,N_i$

Iterative step: $T(c_{i+1,j}) =$

$$S(c_{i+1,j}) + \min_{K=1}^{N_i} \{T(c_{ik}) + D(c_{ik},c_{i+1,j})\}$$

$$\text{for } j = 1,\ldots,N_{i+1}$$

The above procedure finds the minimum cost sequence of candidate object regions. Candidate regions which are accidental are unlikely to persist from frame to frame; thus their D terms are likely to be large, thereby increasing the total cost of any sequence which includes them. Note that there will be many sequences which are only slightly more costly than the minimum. These suboptimal sequences will be based on other exemplars for the same object. The optimal sequence is thus optimal for the

particular formulations of S and D. Giving more weight to S and less to D will tend to select best exemplars; while the reverse weighting will tend to favor frame to frame consistency. A semantic model can provide guidance.

In general, the image sequence may contain more than one object. The scheme described above identifies the "best" object region sequence. In order to extract region sequences corresponding to other objects in the image sequence, we must delete all candidate object regions accounted for by the optimal sequence. The inherent data structure specifies which regions are exemplars for each object. By deleting all candidate object regions in each frame which are similar to the selected region of the optimal sequence (i.e., contain it or are contained in it), we can set the stage for another application of dynamic programming. This process is repeated until only very poor (high cost) sequences are obtained. Presumably at this point all objects have been accounted for.

Occasionally, a deletion step may leave a particular frame empty of candidate object regions. This may occur for two reasons: All objects were accounted for by the last dynamic programming step, or the candidate region proposer failed to elicit an exemplar for an actual object. In the former case, the process will have terminated. The latter case can be handled by associating a fixed "empty frame" cost which is the price paid for skipping a frame. Of course, one can't know which case applied. The conservative approach is always to assume the second case and apply the empty frame cost. The termination criterion will then be based on a threshold for the total cost.

The problem of an object leaving the field of view can be handled in a different manner by flagging candidate object regions which lie on the border of the image. A partial sequence whose last element is flagged but which overall has low cost can be accepted as depicting an object which has moved off the image.

The dynamic programming algorithm described above has been implemented and tested on a sequence of ten windows of FLIR data containing a tank (Figure 9.4). These windows were already smoothed by a 3x3 median filter to provide better response to thresholding. The Superslice algorithm extracted a modest number of candidate object regions. Figure 9.5 displays these regions

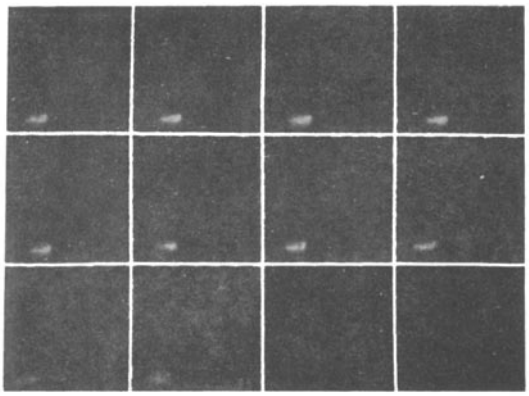


Figure 9.4. A sequence of 10 median filtered windows of a tank.

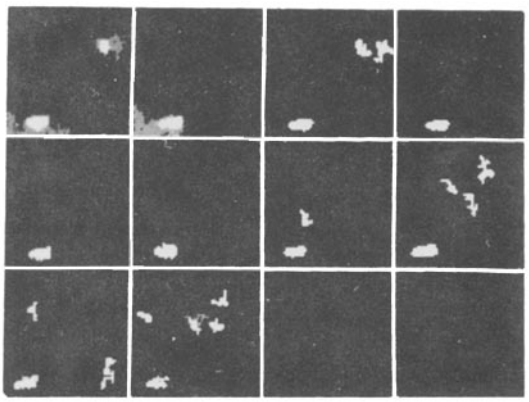


Figure 9.5. Output of the Superslice algorithm.

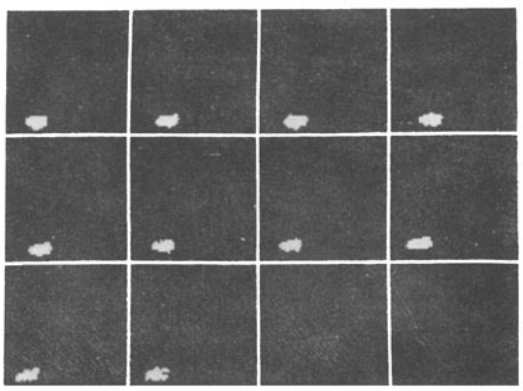


Figure 9.6. Optimal sequenced regions using dynamic programming.

(although for nested sequences only the best static exemplar is displayed). The solution to the dynamic programming problem was computed and the exemplars which correspond to the solution are shown in Figure 9.6. There are of course many suboptimal solutions which are quite similar to this one. Their cost is not significantly greater than the minimal cost. When the indicated regions were deleted along with all other similar candidates, the only remaining regions corresponded to noise and any minimal cost path attempting to span several frames was substantially more costly than the optimal path or any of its similar suboptimal paths. It seems reasonable therefore to establish thresholds for static and dynamic cost in order to prune the search space.

10. Concluding remarks

The work described in this paper resulted from the consideration of a specific problem environment, that of object detection in FLIR imagery. Nonetheless, our intent was not to produce a "special purpose" solution having limited generality. Rather, it has been our goal to develop concepts and approaches which would be of use in a wide variety of applications and would contribute to more successful image understanding.

References

1. Algorithms and Hardware Technology for Image Recognition, First Quarterly Report, Computer Science Center, Univ. of Maryland, College Park, MD, July 1976.

2. Algorithms and Hardware Technology for Image Recognition, First Semiannual Report, Computer Science Center, Univ. of Maryland, College Park, MD, October 1976.

3. Algorithms and Hardware Technology for Image Recognition, Second Semiannual Report, Computer Science Center, Univ. of Maryland, College Park, MD, April 1977.

4. Algorithms and Hardware Technology for Image Recognition, Third Semiannual Report, Computer Science Center, Univ. of Maryland, College Park, MD, October 1977.

5. Panda, D. P., "Segmentation of FLIR Images by Pixel Classification", University of Maryland, Computer Science TR-508, Feb. 1977.

6. Panda, D. P., "Statistical Properties of Thresholded images", University of Maryland, Computer Science TR-559, July 1977.

7. Panda, D. P., "Statistical Analysis of Some Edge Operators", University of Maryland, Computer Science TR-558, July 1977.

8. Hueckel, M., "A Local Visual Operator Which Recognizes Edges and Lines", JACM, Vol. 20, 1973, pp. 634-647. [Erratum: JACM, Vol. 21, 1974, p. 350.]

9. Hueckel, M, "An Operator Which Locates Edges in Digitized Pictures", JACM, Vol. 18, 1971, pp. 113-125.

10. Hummel, R. A., "Edge Detection Using Basis Functions", University of Maryland, Computer Science TR-569, August 1977.

11. Mero, L., Vassy, Z., "A Simplified and Fast Version of the Hueckel Operator for Finding Optimal Edges in Pictures", Proc. 4th Intl. Conf. on Artif. Intelligence, Tbilisi, USSR, Sept. 1975, pp. 650-655.

12. Shaw, G. B., "Local and Regional Edge Detectors: Some Comparisons", University of Maryland, Computer Science TR-614, December 1977.

13. Peleg, S., "Iterative Histogram Modification, 2", University of Maryland, Computer Science TR-606, November 1977.

14. Davis, L. S., "A Survey of Edge Detection Techniques", Computer Graphics and Image Processing, Vol. 4, 1975, pp. 248-270.

15. Weszka, J. S., Rosenfeld, A., "Threshold Selection Using Weighted Histograms", University of Maryland, Computer Science TR-567, August 1977.

16. Milgram, D. L., Herman, M., "Clustering Edge Values for Threshold Selection", University of Maryland, Computer Science TR-617, December 1977.

17. Nakagawa, Y., Rosenfeld, A., "Some Experiments in Variable Thresholding", University of Maryland, Computer Science TR-626, January 1978.

18. Chow, C. K., Kaneko, T., "Automatic Boundary Detection of the Left Ventricle From Cineangiograms", Comput. Biomed. Res. 5, 1972, pp. 388-410.

19. Nakagawa, Y., Rosenfeld, A., "A Note on the Use of Local MIN and MAX Operations in Digital Picture Processing", University of Maryland, Computer Science TR-590, October 1977.

20. Milgram, D. L., "Constructing Trees for Region Description", University of Maryland, Computer Science TR-541, May 1977.

21. Rosenfeld, A., "Fuzzy Digital Topology", University of Maryland, Computer Science TR-573, September 1977.

22. Dyer, C. R., Rosenfeld, A., "Thinning Algorithms for Grayscale Pictures", University of Maryland, Computer Science TR-610, November 1977.

23. Ohlander, R., "Analysis of Natural Scenes", Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA, December 1976.

24. Milgram, D. L., Kahl, D. J., "Recursive Region Extraction", University of Maryland, Computer Science TR-620, December 1977.

25. Stockman, G. C., "Maryland Interactive Pattern Analysis and Classification System, Part I: Concepts", Dept. of Computer Science, University of Maryland TR-408, College Park, MD, September 1975.

26. Wertheimer, M.,"Principles of Perceptual Organization", in Readings in Perception, D. C. Beardlee and M. Wertheimer (eds.), p. 122, Van Nostrand-Reinhold, Princeton, NJ, 1958.

27. Abend, K., "Compound Decision Procedures for Unknown Distributions and for Dependent States of Nature", Pattern Recognition, L. Kanal, Ed., Washington, DC, 1968, pp. 207-249.

28. Milgram, D. L., "Region Tracking Using Dynamic Programming", University of Maryland, Computer Science TR-539, May 1977.