Application of Pattern Recognition and Image Processing Techniques to Lock-on-After-Launch Missile Technology

Lewis G. Minor

US Army Missile Laboratory, US Army Missile Command
Redstone Arsenal, Alabama 35898

Abstract

This paper presents image processing techniques used for automatic target recognition where targets are military such as tanks. A missile concept is briefly developed to provide a context for algorithms presented in the paper. Algorithms presented include a technique for locating potential targets in an image with a procedure similar to Hough transforms. A new technique for segmenting targets from the image is also presented. Test results on a data base of 43 images are presented to illustrate results.

1. Introduction

The first adaptive gate TV imaging tracker for homing missile applications was first developed in 1961 by the US Army Missile Command. The technique permits the tracking of any selected object which contrasts with its surround in the sensed TV video. Since the first imaging trackers were developed, many improvements have been made. The newest imaging trackers utilize an adaptive gate centroid tracker used in conjunction with a moving target tracker and a correlation tracker. The correlation tracker can either track scene background or the target itself. These trackers use distributed architecture microprocessor technology and can make "decisions" as to which track mode is the most confident.

All of the imaging trackers developed in the past 20 years can track objects which contrast with their surround, or track objects which have appropriate texture for correlation tracking. However, in each case the object to be tracked must be selected by a human interpreter. Since the human interpreter must first recognize a given object in the TV video as a target and then manually select the object to be tracked, the net effect is that local processes may now be used at each of the candidate areas. The technique is practical only if the number of candidate areas does not become large. The number of pixels in these local areas should be far less than the total in the image for a successful global process. It is expected that more sophisticated algorithms can be used in the local process than in the global process since there are fewer pixels involved in the computations. The preprocessing by locking on to the target after launch. Such systems are generally termed lock-on-after-launch missiles or LOAL missiles. These missiles are not very mature and a great deal of technology development must take place before practical systems can be built. This paper describes recent work in the area of algorithm development. The work should be considered to be only a small part of a larger problem.

2. Hypothetical System Concept

Figure 1 shows a hypothetical system concept. The missile in this concept has been targeted into an area known to contain targets. The sensor shown is a TV imager operating in the infrared spectrum. As target-like objects pass through the field-of-view, they are detected, segmented, and classified. Objects which are classified as high priority targets are selected for tracking in a manner analogous to the human interpreter. Targets determined to be of sufficient priority are engaged. A proposed set of algorithms for detecting, segmenting, and classifying targets is described in the following Section.

3. Overview of Algorithms for LOAL

Figures 2 and 3 show a block diagram of a hierarchical automatic target acquisition process. A hierarchical process is used to minimize computational requirements. The process shown has been divided into a global process and a local process. The global process includes preprocessing, feature extraction, and a classification stage leading to the selection of candidate areas in the image. These candidate areas are areas in the image with a higher probability of being a target location. The candidate areas will typically contain many false targets along with any true targets. At this level of the hierarchy, only low cost operations have been used, yet many unlikely candidate areas have been rejected. The net effect is that local processes may now be used at each of the candidate areas. The technique is practical only if the number of candidate areas does not become large. The number of pixels in these local areas should be far less than the total in the image for a successful global process. It is expected that more sophisticated algorithms can be used in the local process than in the global process since there are fewer pixels involved in the computations. The preprocessing
algorithms shown in Figure 2 are described in a paper by Minor and Sklansky [2] and elsewhere [3]. The preprocessing consists of intensity normalization, dc notch filtering [4], and an edge operation. Figure 2 shows a block for motion detection and range estimation. These features were listed because of their importance but are not used in the following section which was limited in scope. Since the velocity of the missile can be estimated from a pilot tube range can be accurately determined by measuring the line of sight rate and depression angle to a given point where a range estimate is needed.

4. Edge Operation

The edge operation consists of a gradient operator, followed by a 90 degree rotation of each computed gradient element. Our gradient operator is the Sobel operator. The direction of the gradient is estimated from the arctangent of the ratio of the vertical to horizontal components of the gradient. This direction is quantized into eight directions: 0°, ±45°, ±90°, ±135°, and 180°. The application of this gradient operator yields the vector image $G(x)$. (We define a vector image as a two dimensional array of vectors.) Each element of $G(x)$ with a modulus less than the noise threshold $T$ is set to zero. In our experiments, $T$ was chosen so that 10% of the gradient elements had moduli greater than or equal to $T$. In our data base, these moduli had a distribution similar to a Rayleigh distribution, with most of the moduli tending to be zero or very small. In general, $T$ should be chosen so that there is a low probability of missing the blobs and a low probability of detecting spurious edges.

Each element of $G(x)$ is rotated 90 degrees counterclockwise yielding the vector edge image $E(x)$. Each element of $E(x)$ is an edge element, pointing in a direction so that the intensity to the right of the edge element is greater than that to the left. Thus, the edge elements associated with a blob will tend to align themselves with tangents to the blob's boundary. A positive contrast blob (brighter than background) will be surrounded by edge elements flowing in a clockwise direction; a negative contrast blob will be surrounded by edge elements flowing in a counterclockwise direction.

5. The Spoke Filter

The spoke filter is a digital processor that detects digital blobs of widely varying shapes and sizes in the presence of noise. The filter is an extension and elaboration of the Hough circle detectors, which has been used for the detection of approximately circular blobs. The spoke filter consists of three stages in cascade.

The first stage of the spoke filter generates the spoke register image $R(x)$ from the edge image $E(x)$. The domain of $R(x)$ at point $x$ represents the contents of an 8-bit spoke register at point $x$. The ith bit of this register (+1 runs from 0 to 7) is associated with the chain code of an edge element displaced from $x$ by a portion of a "spoke." Each preprocessing edge element of $E(x)$ generates a digital straight line segment (a "spoke") of length $L$ pixels at a distance of $S$ pixels from the edge element. If the edge element's direction is 1, every digital point $x$ in this line segment sets bit position $i$ of $R(x)$ to 1. This process is illustrated in Figure 4 for edge elements in $E(x)$ labeled "3" and "4."

The relative directions of the spokes shown in Figure 4 with respect to the edge elements results in the detection of blobs that are brighter than the background. If the detection of blobs that are dimmer than the background is required, each spoke must be directed 180 degrees from those shown in Figure 4.

The length of the spoke and its distance from the edge element, measured in pixel-widths (assuming the pixels are squares) are denoted by $L$ and $S$, respectively, where $L$ and $S$ are integers. If the spoke is oriented along an intermediate direction, such as northwest, the length of the spoke and its distance from the edge elements, measured in pixel-diagonals, are denoted by $L', S'$, respectively. The parameters $L, C$ and $L', S'$ are chosen for each spoke to detect blobs in a prescribed range of sizes and shapes.

In Figure 4, the horizontal spoke is specified by $L=6, S=1$. The diagonally oriented spoke is specified by $L'=4, S'=1$. As illustrated in this Figure, spokes tend to intersect when they are generated by edge elements of an arc or curved path. We refer to such an intersection as a "crossing event."

The number of detected directions of the spokes passing through each integer point of the $x$-domain is equal to the number of bit positions set to 1 in the corresponding spoke register. In Figure 4, the numbers in each pixel of a spoke are hexadecimal representations of the eight-bit number in each spoke register. Thus, 18 denotes the binary number 00011000 in the spoke register of the crossing event.

The second stage of the spoke filter carries out an OR operation on the spoke registers in each 3x3 neighborhood, producing $R*(x)$. Let $N(x)$ denote the nine integer points in the 3x3 neighborhood of $x$. Then $R*(x)$ is related to $R(x)$ by

$$R*(x) = \bigcup_{u \in N(x)} R(u)$$

We refer to each register in $R*(x)$ as an ORed spoke register. The number of distinct directions of the spokes passing through each 3x3 neighborhood $N(x)$ is equal to the number of bits in the corresponding ORed spoke register.

Every ORed spoke register in $R*(x)$ contains one of 256 possible permutations of 8-bits. The third stage of the spoke filter maps each such
permutation to either 0 or 1, where the 1 denotes a detection of a blob. \( F(x) \) denotes the binary-valued image obtained by this mapping from the contents of the array of ORed spoke registers, \( R^*(x) \).

The mapping from \( R^*(x) \) to \( F(x) \) can be chosen to respond, for example, to the number of distinct directions of spoke elements passing through \( N(x) \) or to the maximum number of adjacent distinct directions of these spoke elements.

In our implementation of \( F(x) \) we resolved the mapping of the ORed spoke registers into four colors:

- Blue (dark gray) for six adjacent 1's
- Green (light gray) for seven 1's (necessarily adjacent)
- Red (white) for eight 1's
- Black (black) for all other permutations of 0's and 1's.

In our implementation of the classifier, described subsequently, we evaluated two \( F(x) \)'s. One \( F(x) \) was obtained by assigning green and red to 1, and blue and black to 0. This corresponds to a mapping of \( R^*(x) \) to 1 for \( N>7 \), and 0 otherwise, where \( N \) denotes the number of 1's in each ORed spoke register. The second \( F(x) \) is 1 for \( N>8 \), and 0 otherwise.

We refer to these two \( F(x) \)'s as "\( F(x)=1 \) for \( N>7 \)" and "\( F(x)=1 \) for \( N>8 \)" respectively.

6. The Gradient-Directed Segmenter

A segmenter is a computer process that finds the boundaries of meaningful objects in an image. We describe an effective segmenter of blobs that exploits the manner in which blobs are formed in infrared images. In many cases, an image of a vehicle can be segmented form the scene by a single threshold. However, different thresholds may be needed for different vehicles in the same scene.

Our segmenter finds an optimum threshold for each blob based on the degree of agreement of the edge elements in \( E(x) \). Milgram and others have developed segmentation techniques based on the coincidence of high edge modulus with the blob's boundary [3]. However, in those techniques, the edge directions were not exploited. Our segmenter extracts low contrast blobs even when parts of the boundaries are poorly defined and edge values are small (for edges with modulus greater than 1).

The segmentation procedure is applied to the output image \( F(x) \) of the spoke filter. Since \( F(x) \) has several connected components, we let \( C_i \) be the label for the \( i \)th component. The components represent interior regions of "interesting" objects.

We find the blob segmentation associated with the component \( C_i \) as follows. Let \( a_i \) denote an integer point in component \( C_i \). Let \( A=D(a_i) \), where \( D(a) \) is the d-c notch image. We compute

\[
M(x) = \begin{cases} 
1 & \text{if } D(x) > A \\
0 & \text{otherwise} 
\end{cases}
\]

Since \( a_i \) is known to be an above-threshold point in \( D(x) \), a procedure similar to region growing can be used to extract the blob associated with \( C_i \). The outermost perimeter of the connected component which contains \( a_i \) is found by scanning from \( a_i \) in \( M(x) \), and following the boundary until closure occurs. This process defines the boundary \( B_{k_i} \) associated with the connected component \( C_i \) in \( F(x) \). The threshold value \( A \) is lowered by unit and the process above is repeated until too many perimeter points are generated or the border of the image is encountered. Above-threshold points in \( M(x) \) that are not connected to \( a_i \) are not considered in any stage of the process.

For each threshold value \( A \), the boundary \( B_{k_i} \) is found. For each \( B_{k_i} \), a digital closed curve \( B_{i_1} \) immediately inside \( B_{k_i} \) is found, and a digital closed curve \( B_{i_2} \) immediately outside \( B_{k_i} \) is found. The direction of each of the edge elements of the segmented regions inclosed by and including \( B_{i_k} \) is within \( +45 \) of the corresponding edge element in \( E(x) \). If this direction is within \( +45 \), the count is added to a counter \( N_{B_{i_k}} \). Then, \( \mu_k(A) = N_{B_{i_k}}/NE \), where \( NE \) is the total number of edge elements in \( B_{i_k} \) (\( k=1,2,3 \)). The quantity \( \mu_k(A) \) is an index of directional agreement for \( B_{i_k} \). For each \( B_{i_k} \) with the edge elements in \( E(x) \), a search for a peak \( \mu_k(A^*) \) in the curve of \( \mu_k(A) \) versus \( A \) is performed. For each connected component \( C_i \) in \( F(x) \), the process is repeated for all \( C_i \), resulting in an optimal threshold value \( A^* \) and a boundary \( B^{opt} \) which defines the border of the optimal segmentation for each blob.

7. Feature Extraction and Object Classification

Once objects have been segmented from the background, the next process extracts features or attributes from each object. Only two class assignments were implemented in the work reported here: target and nontarget. The following two features proved to be effective for discriminating between these classes:

- \( c = \text{standard deviation of the gray value interior to the segmented object} \)
- \( e = \text{fraction of points in } B^{opt} \text{ where } |E(x)| \text{ is in the upper } 5\% \text{ of the distribution of } |E(x)|. \)

The last process is the assignment of a class (i.e., target or nontarget) to each pair \((c,e)\).
This process was carried out by a three nearest neighbor classifier [5].

A typical curve of $u_i(A)$ is shown in Figure 5 for a digitized 3/4 rearview image of an M48 US Army tank. The image of this tank is shown in Figure 6a. The peak of the curve of $u_i(A)$ for this image indicates a degree of directional agreement of about 99%. Figures 6b and 6c are the images obtained by operating on Figure 6a, gradient operator, and spoke filter in succession. When the threshold producing this optimal segmentation of this tank is applied to Figure 6a, a segmentation corresponding to the dashed perimeter points shown in Figure 6b is obtained.

To design and test our classifier, we divided our database of 41 images in approximately equal parts: a design set of 21 images containing 34 targets and a test set of 20 images containing 47 targets. A 3 nearest neighbor classifier for feature vectors $(x, e)$ extracted from $F(x)$ for N>7 was constructed from the design set. The results of applying this classifier to the test set are summarized in Figure 8. This classifier removed all false alarms, and missed only two targets out of 47.

8. Experimental Results

We tested our procedures on a data base of 42 infrared images containing 81 military vehicles with blob-like images. These vehicles fall into three categories: tank, jeep, APC (armored personnel carrier).

Each of the connected components of $F(x)$, the output of the spoke filter, indicates a likely location of a military vehicle or blob. Thus $F(x)$ acts as a detector of "suspicious regions" in the image. The performance for N>7 and N=8 is summarized in Figure 8. Single-point components of $F(x)$ were not included in the table. If single-point components for N=8 are included, only three targets are missed and the number of false alarms increase to 39.

At this stage of the process, the number of false negatives (missed targets) is far more important than the number of false positives (false alarms) which are removed by subsequent processing. The amount of image data processed in subsequent stages has been dramatically reduced by the spoke filter without restricting the detected blobs to a specific shape. Prior to processing by the spoke filter there were 638,128 possible locations for targets (42 images x 128 rows x 128 columns). After filtering, the number of potential locations was $25 + 77 = 102$, with only 5% of the targets missed (for the case N=6). For N>7 the number of potential target locations was $81 + 101 = 182$, with no targets missed. Thus, the technique permits an effective use of local processing for subsequent stages of processing.

The gradient-directed segmenter was applied to this data base. Human observers' estimates of the blob boundaries corresponded to those produced by the segmentation algorithm, except for one scene. That scene contains several overlapping blobs which were grouped too closely together to be resolved as more than one vehicle. A segmentation using a manually adjusted threshold was carried out on all scenes. This experiment yielded segmentations nearly identical to those achieved by the gradient-directed (automatic) segmenter. As in the gradient-directed segmenter, the manual segmentation failed on the scene containing overlapping blobs.
Figure 5. Degree of match $M_i(A)$ versus threshold $A$.

Figure 6. Test images for spoke filter and gradient guided segmenter.

<table>
<thead>
<tr>
<th>Version of $F(x)$</th>
<th>Number of Targets Detected</th>
<th>Number of Targets Missed</th>
<th>Number of False Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N=7$</td>
<td>81</td>
<td>0</td>
<td>101</td>
</tr>
<tr>
<td>$N=8$</td>
<td>77</td>
<td>4</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 7. Performance of the Spoke Filter, with single-point components excluded.

<table>
<thead>
<tr>
<th>Target</th>
<th>Clutter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Priori Target</td>
<td>45</td>
</tr>
<tr>
<td>A Priori Clutter</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 8. Confusion matrix for test of classifier.

REFERENCES


