Chunk-based matching of images for ATR

Raj Bhatnagar[a], Brent Seibert[a], Lynne Vettel[a], Eric Freudenthal[b], David Morgenthaler[c]

[a]ECECS Department, University of Cincinnati, Cincinnati, OH 45221 USA
[b]New York University, New York, NY 10003 USA
[c]Lockheed Martin, Denver, CO

ABSTRACT

This paper presents the results of an enhancement to the image matching component of an automatic target recognition (ATR) system that improves the ability to handle variations and articulations within a given class of targets. This method is based on chunking of an image and can be applied to any image matching system that uses templates to match against a given input image. Using information theoretical measures, templates are divided into sub-parts, called chunks. These chunks are scored individually against corresponding parts of an input image. Sub-part scoring adds the ability to distinguish poorly matching areas of the target from those that match well. If a very small set of chunks score significantly worse than the other chunks then the poor-scoring chunks may be discarded. This increases the scores of an input image that is of the same class but there is little or no effect on the score of an input image that is of another class.

Keywords: target recognition, SAR, chunk-based matching

1. INTRODUCTION

A template-based automatic target recognition (ATR) system matches an input image, (a chip), against a set of templates. Our experiment uses 72 templates per class and each template is a composition of some images for the class from within a five degree azimuth range, and is used to represent input images in that five degree range. As an input images enters the system, it is scored against the templates to find the best match. The template with the best match determines the class of the image chip. If no template matches the input well, the target is deemed clutter.

The matching process can be cumbersome, depending on the desired number of target classes. Consider a system in which there are 72 templates per class (one for each five-degree azimuth interval). Each class that is added to the identifiable targets adds another 72 match cycles to the matching process per input image. Hence, a trade-off exists between identification robustness and speed.

2. PROBLEM STATEMENT

This paper proposes a method that has the potential to reduce the computational cost of matching images to the templates from a class having significant intra-class variability. The method, called chunking, has proven to be an aid to this problem, and has many other potential benefits which have been discussed later.

Suppose there exists a certain group of target vehicles that are alike in most respects, but are distinct enough to be considered different vehicles. It would be beneficial to use one template to represent the entire class. One possible way to do so would be to include test images from all of the group's vehicles to form templates. The drawback is that the distinguishing characteristics of each vehicle's radar signature will be muddled and distorted by the characteristics of each other vehicle. If each vehicle's distinct identifiers appear on different locations of the radar signature than all the other vehicles in the group, the template formed by combining the vehicles may not match any vehicle well and can become meaningless. Chunking allows for the use of one template for the group without depending on the location of the differences in radar signatures.

Chunking involves dividing a template into sub-parts called chunks, then scoring each chunk separately. This process localizes the poorly scoring areas of an input image. Provided that the low-score areas are few and the high scoring areas are many, the low scoring chunks may be discarded by some heuristic measure. When only a few chunks score poorly, while the rest score well, this results in a good chip-to-template match. Thus, a group of vehicles that are similar in more respects than they differ can be represented by the chunks of a template of one vehicle from the group. The chunks of a template will score very well for the many parts of the vehicles that are similar, and will score poorly for those few parts which differ. After the poor scoring chunks are discarded, the result is a high overall output score from the matching component of the system. Thus, the number of templates in the system can be significantly reduced without a loss of identification performance. The only drawback is that every vehicle in the group of alike vehicles will be identified as a member of the group rather than its exact vehicle type.

3. RELATED WORK

The problem to be solved by automatic target recognition can be divided into two parts. The first part involves detecting a target from the surrounding clutter. Once a target is detected, the second part, classification, must be addressed.[3] This paper proposes an enhancement to the classification portion of the ATR problem. A method of image segmentation is proposed by Su Chang and Rae-Hong Park[4] that segments an input image into sub-sections representing three-dimensional surfaces and characteristics. It requires the use of an intensity matrix, also used in the method described in this paper, and a phase matrix to segment the images. This method is not directly applied to ATR, but potentially could be used in combination with the chunk scorer discussed in this paper. The chunk scoring method discussed in this paper segments the template images using edge-detection and information theoretical measures and matches each segment separately, allowing for the localization of poor scoring areas and the reduction of the number of templates required for successful matching.

4. CHUNKING PROCESS

Chunking is performed before the actual matching takes place. Once the chunks have been determined for a particular template, those chunks can be stored for loading when that template is accessed in the future. Alternatively, chunks can be generated "on the fly" as templates are loaded for matching. The system's speed in loading images from files determines which method is more efficient. In the optimal case, the matching system would have ample resources to load all templates and chunks into RAM upon start-up, producing the fastest access time.

When a template is loaded from a file, a mask image of the same dimensions is also loaded. The mask image isolates the target area of template by labeling the pixels in the template that are included in the region of interest (ROI). The mask consists of a two-dimensional matrix of the same size as the template, in which ones mark the the pixels that are considered to be part of the target, and zeros mark the pixels that are not of interest. In the mask image shown in Figure 1, the white areas are considered part of the ROI and the black area is not. All areas marked with a one in the mask are considered as part of the template for chunking and scoring.

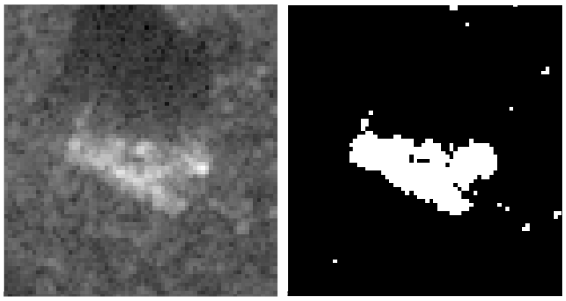

Figure 1. Template for vehicle at 60 degrees azimuth and its corresponding mask.

The chunking process involves four stages: finding features, planting chunk seeds, growing chunks, and filtering chunks.

4.1. Finding features

In the context of the chunking process, the term *features* denotes the characteristics of each pixel in the template image in relation to its *neighboring* pixels. The features are expressed as a two-dimensional matrix of the same size as the template, where each pixel consists of a value from zero to eight that represents the feature of the corresponding template pixel. The features determined in this step provide important information for the next step of the chunking process, planting chunk seeds.

To compute the feature values, first the pixel intensities of the gray-scale template are scaled from 0 to 255 for standardization of functional parameters. The basis for computing features of a template is edge-detection. Edge-detection provides information about a pixel's relationship, in intensity, to the pixels adjacent to it.

1. Horizontal scan, left-to-right edge detection

   First, horizontal edge detection is performed. The gray-scale template is scanned from left to right. If the intensity of the current pixel is an amount δ greater than or less than the pixel to its left, then the current pixel is marked with a 1, or a -1 respectively. If the intensity of the current pixel is less than δ different from that of its left neighbor, then the current pixel is marked with a zero. The result is a matrix of the same size as the template, in which each pixel is marked with a 0, 1, or -1 representing the presence and characteristic of an edge encountered through a left-to-right scan.

2. Vertical scan, top-to-bottom edge detection

   Vertical edge detection is performed in a similar manner as horizontal edge detection. Rather than left-to-right, the template is scanned top-to-bottom. Each pixel is compared to the neighboring pixel above it. If the intensity of the current pixel is δ greater or less than its upper neighbor, then the current pixel is marked with either 1 or -1. If the difference is less than δ then the current pixel is marked with 0. The resulting matrix is the same size as the template, with each pixel marked with 0, 1, or -1 indicating the presence and characteristic of an edge encountered through a top-to-bottom scan.

3. Combining edges from horizontal and vertical scans

   After performing horizontal and vertical edge detection, there are two matrices of equal dimensions representing pixel relationships in the template. By combining the information into a single matrix, the characteristics of each pixel as determined from both edge-detection phases can be easily examined. A new matrix is produced by "overlaying" the vertical-scan matrix on the horizontal-scan matrix, and computing pixel values based on the combination of values from each matrix.

   The value of each pixel in the combined matrix is determined as shown in Table 1.

Table 1. Feature value resulting from each combination of left-to-right and top-to-bottom edge values.

L-to-R Edge	T-to-B Edge	Feature Value
-1	-1	0
0	-1	1
1	-1	2
-1	0	3
0	0	4
1	0	5
-1	1	6
0	1	7
1	1	8

The result is a 0 to 8 encoding of the *features* of a template, as shown in Figure 2.

These features are actually a representation of the direction of the *flow* of pixel intensities. Nine feature values allow for the representation of eight *flow* directions as well as the lack of *flow*.

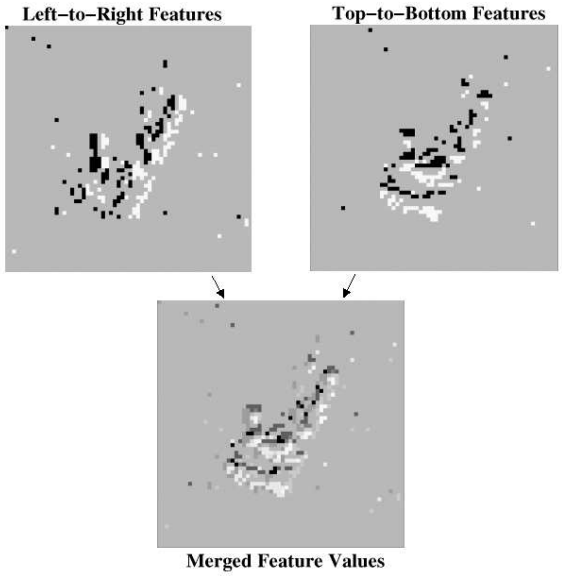

Figure 2. Left-to-right and top-to-bottom edges are combined to produce 'features'.

In practice, the steps of this problem can be combined and performed in one scan of the template, using an exhaustive left-to-right, top-to-bottom scan locating every pixel in the template matrix. At each pixel, the horizontal-scan value, the vertical-scan value, and the combined feature value can be determined for each pixel in the template according to

$$f = (lr + 1) + 3 * (tb + 1), \tag{1}$$

where f is the feature value of the pixel, $\{lr : -1, 0, 1\}$ is the left-to-right edge value of the pixel, and $\{tb : -1, 0, 1\}$ is the top-to-bottom edge value of the pixel. The cost of feature computation is decreased three-fold as a result of combining these steps.

4.2. Planting chunk seeds

Using the information present in the two-dimensional matrix of features, *seeds* that serve as starting points for chunks can be calculated. To determine the best positions for chunk seeds, it is important to consider what information should be captured in a chunk. The idea behind chunk scoring is that each chunk should be representative of a distinguishable area of a target. For example, it would be preferable for a chunk seed to be placed at the apex of the gun turret of a tank and grow outwards to cover all or part of the turret. A poor score on such a chunk may indicate different turret styles within a certain class of targets, or may account for a different configuration of that turret, such as a different angle of rotation for the gun barrel. The ability to capture such information is key to the entire sub-part scoring concept. Therefore, an effective chunk seed should be placed on an apex, edge, valley, or some other defining point indicating a noticeable property of some distinguishable area of the target. The features matrix, as derived in the previous step, provides ample information to determine such points on a target. Placing the seeds in these particular areas of a target allow these parts to be separate chunks, and therefore scored separately.

The features matrix provides values indicating the slope of pixel intensities for each pixel in the template image. Because it is necessary to place seeds in areas of noticeable variation, it is preferable to keep seeds away from areas of little or no variation. Such areas of a template are identifiable by having the same type of features for a given area of the feature matrix. For example, if an area of the feature matrix has all of the same or mostly the same pixel intensity flow, its corresponding area of the template will be a flat, featureless area of the vehicle.

Rather than placing seeds in these flat areas, the preference is to place seeds in areas of high variation. This ensures that useful information is captured by each chunk. The direct result is that each chunk is necessarily centered at an area of high variation, encompassing some kind of distinguishable area of the vehicle, such as a hard edge, a corner of a component, an apex, a valley, or any other high variance point.

To determine the locations for chunk seeds, each pixel of the features matrix is examined. At each pixel, the three-by-three group of pixels with the given pixel as its center are examined. If no more than three of the nine pixels have the same feature value, then the center pixel of the group is considered as a *possible* chunk seed. Since it is necessary to keep chunk seeds a certain distance apart from one another, so that the chunks have room to grow outward, each chunk is given at least a five-by-five block of pixels to grow. Therefore each seed must be at least a distance of five pixels from any other seed. To ensure this, the seven-by-seven block of pixels surrounding the given pixel is checked for other seeds. If no other seed is found in the block, the given pixel becomes a chunk seed. A limit may be placed on the number of seeds allowed, but the logic behind the feature and seed generators makes this usually unnecessary. In this experiment, the number of seeds generated is generally between six and twelve, whereas the maximum number of seeds allowed is thirty. The number of resulting seeds is dominated by the denseness and variation of extracted features in the feature-finding step. Figure 3 shows the seeds derived from the features for Vehicle Variant 2.

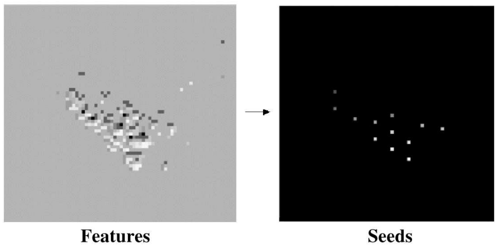

Figure 3. Formation of seeds for Vehicle Variant 2 at 45 degrees.

4.3. Growing chunks

Once seeds have been planted, chunks can be grown to cover the target area. As previously discussed, each seed is necessarily at the center of an area of relatively high variation. Each chunk should grow in such a way that it encapsulates this variation and expands to include enough of the surrounding area to form a contrast between the variation and its surroundings. According to information theory, a higher amount of information is contained in a set of data in which there exists a noticeable contrast, rather than random noise.[5] For example, a data set containing the values {0,0,0,0,0,10,10} contains more information than one containing {0,6,1,3,5,2,4}. The latter data set contains no useful or noticeable contrast, boundaries, or patterns, whereas the first contains potentially useful information. In order to capture the maximum amount of information while growing chunks, a measure of the entropy of pixel values within the chunk is considered when adding a pixel to a chunk. Entropy is negatively related to information[5] and is indicative of chaos in a data set. Therefore, to maximize the information in each chunk, we decrease the entropy of the chunk pixel values with each pixel we add. The process of growing chunks is as follows.

For each chunk seed:

1. Include 5x5 block of pixels surrounding and including seed pixel to chunk.

   In the seed-planting step, seeds were chosen based upon the 3x3 block of feature values with the seed at the center. This 3x3 block of features is the result of computations involving the 5x5 block of pixels with the seed at the center. SO including the 5x5 block of pixels ensures that the area of interest that caused a seed to be planted in that spot will be included in this chunk.

2. Compile a vector (list) of pixels adjacent to the chunk.

   This step involves creating a vector of the coordinates of all pixels adjacent to, but not included in the current chunk. Only pixels included in the target mask can be adjacent pixels. This vector contains the exclusive set of pixels that will be considered for addition to the chunk during this cycle of the process.

3. Examine each pixel in the adjacency vector.

   For each pixel, if adding the pixel to the chunk decreases the overall entropy of the chunk pixel values, then the pixel is added to the chunk. Otherwise, the pixel is not added. The entropy of a chunk is calculated using

$$entropy = -\sum_{i=1}^{N} p_i log p_i \tag{2}$$

   where N is the number of pixels in the chunk. Thus the algorithm for growing chunks is as follows.

```
changed = true;

While (changed)
    changed = false;

    For each chunk, C:
        Create adjacency vector, V, for C
        (list of all pixels adjacent to pixels in C);

        For each pixel, p, in V
            If (entropy(C) > entropy (C + p))
                C = C + p  (Pixel p is added to chunk C);
                changed = true;
```

   This step ensures that each pixel added to the chunk decreases the entropy, thus increasing the information present in the chunk. This step iteratively checks each pixel in the adjacency list. If a pixel has the proper criteria to be added to the chunk, it is added immediately, rather than at the end of the entire step, and therefore can influence whether the remaining pixels will be added.

4. If any pixels have been added during this iteration, loop back to Step 2.

   If one or more pixels are added to the chunk in this iteration of the process, the adjacency vector is recreated for the new chunk and each adjacent pixel is checked again. This process repeats until no pixels are added to the chunk during an iteration, or there are no more adjacent pixels.

4.4. Filtering chunks

Once chunks have been grown, each chunk will include the pixels that maximize the information contained in the chunk. However, several pixels remain that are part of the target mask, but do not yet belong to any chunk. Since all the pixels in the target mask are scored in the matching step, it is necessary to make sure each belongs to a chunk. The filtering process ensures that each pixel in the target mask is part of a chunk.

To filter the chunks, the unassigned pixels that neighbor each chunk are examined to determine whether they are in the target mask and therefore should be included in the chunk. The algorithm is as follows.

```
While (mask area not fully chunked)
    For each chunk, C:
        Create adjacency vector, V, for C
```

```
(list of all pixels adjacent to pixels in C);

For each pixel, P, in V
    If P is contained in the target mask
        C = C + P  (Pixel P is added to chunk C)
```

The chunks are not drastically affected by the filtering process, because the majority of pixels within the mask area are already contained in a chunk before filtering takes place. This step of the chunking process is necessary to ensure that a valid match takes place. If any pixel within the target mask is not assigned to a chunk, then the missing pixels cannot be scored, leading to an inaccurate overall match. Once filtering is completed, the template is prepared for matching against chips.

5. IMPLEMENTATION OF THE CHUNK SCORER

Each of the following steps are performed in the matching program. Step 1 is done for each template image, and can be done before any matching occurs or on-the-fly during each match. The latter method results in repeated processing of the same templates which requires greater calculation time. However, the time required to save each template's chunks to a binary file, then reload it when needed can be more than simply generating the chunks on-the-fly when needed. The chips used in the implementation of this program are shown in Figure 4. The chips consist of eight variants in the same vehicle class and two confuser classes. All chips are exhaustively matched to all templates using the regular and chunk-based methods.

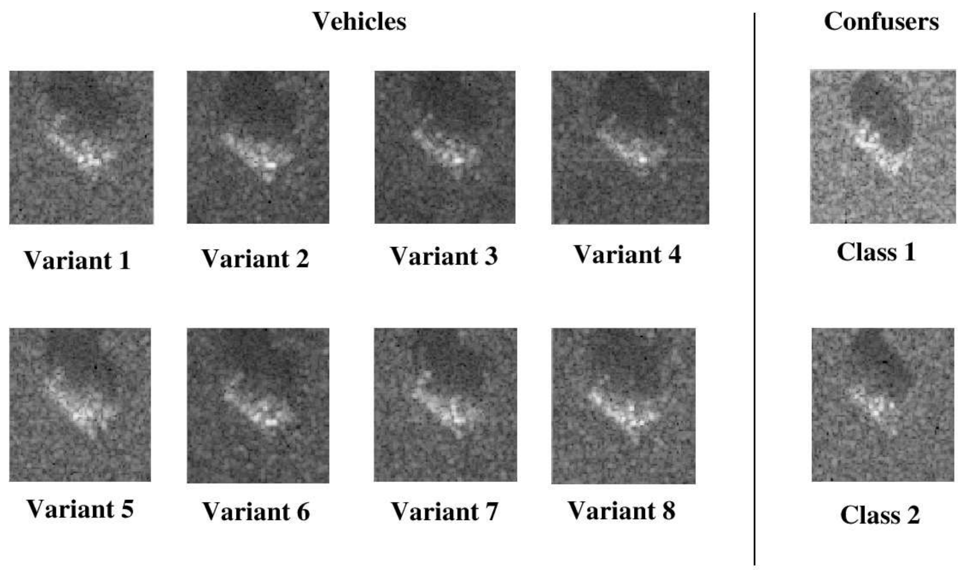

Figure 4. Chips used in the matching process – eight vehicle variants, all of same class, and two confusers.

1. Chunking
   The chunk processing described previously is performed for each template. The result is a two-dimensional matrix of the same size as the template with each entry consisting of the number of the chunk to which the pixel at the corresponding position in the template belongs, or zero if the corresponding pixel does not belong to a chunk.
2. Registration
   To match two images pixel by pixel, the images must be properly "aligned" so that each pixel of the chip

image is in the same position as its corresponding pixel in the template image. To accomplish this, the mask matrix produced from the template image is used. Another mask matrix is produced using the chip image. The template mask is aligned with the chip mask in such a way that the maximum number of one-valued template mask pixels with the value of one reside in the same coordinates as the one-valued mask pixels of the chip mask. This can be visualized in the following way: the template mask image is placed over the chip mask image. The template mask image is then shifted left to right, top to bottom, until the maximum number of mask pixels match up.

3. Scaling
   In order to properly match the two images, they must be scaled appropriately so that each exhibits the same level of overall intensity within the mask region. Because only the pixels that reside in the template mask are used in scoring, the scaling factor is determined by examining only the pixels in the template mask. The scaling factor is computed by computing the mean pixel value of the pixels residing in the template mask for the chip and for the template. The factor is the template average divided by the chip average. This factor is multiplied by the value of each chip pixel when matching against the template pixels.

4. Pixel comparison
   All pixels contained in the template mask of the chip are exhaustively compared to the pixel of the same coordinates in the scaled and registered chip image. Because the mask region of the template is fully chunked, each pixel is necessarily a member of a chunk. Each pixel score is the square of the difference (error) between the chip pixel and the template pixel. The pixel scores are maintained in separate vectors for each chunk and the overall number of pixels for the image and each individual chunk is also maintained.

5. Score computation
   Scores are computed for each chunk and for the entire image in this step. A chunk score is simply the mean squared error of all the pixel scores contained in the chunk, and can be computed for each chunk as

   $$chunkscore = \sum_{i=1}^{N} (tp_i - cp_i)^2, \tag{3}$$

   where N is the number of pixels in the chunk, cp_i is the pixel value for pixel i in the chip, and tp_i is the pixel value for pixel i in the template.

   Chunk scores are indicative of the overall match quality of the region represented by that chunk. These are the scores that are the significant factor in chunk scoring. One other score can be computed at this point. The overall score of the entire image match is the average of the chunk scores, where each score is weighted by the number of pixels contained in the corresponding chunk.

6. Discarding chunks
   Discarding the scores of particular chunks is the heart of the chunk-based scoring process. Consider a set of ten chunk scores resulting from a template to chip match. The scores are actually error measurements, so the lower the score, the better the match. Eight of these scores fall in the range of 0.005 to 0.035, considered good scores, and the other two are 0.06 and 0.07, considered poor scores. It is obvious that the majority of the surface of the vehicle represented by the chip matches the template, while two small regions have distinct differences. Without chunk scoring, and assuming that each chunk has the same number of pixels, the overall score is 0.029, which is a relatively average score. We want to eliminate the negative effects of a few poorly scoring chunks while maintaining the well-scoring majority. The quality of each chunk score can be identified using a measure of the standard deviation of all the chunk scores. The mean and standard deviation of the scores in this example case is 0.029 and 0.0143 respectively. The deviation of the good scores from the mean is between -0.019 and 0.001 and the deviations of the bad scores are 0.031 and 0.041. The standard deviation multiplied by a factor of 2 is 0.0286 and multiplied by a factor of 1.5 is 0.02145. If we use 1.5 times the standard deviation above the mean as a threshold for classifying poor scores, we completely eliminate the bad scores from the overall score, resulting in a score of 0.2, which is a significantly better score than 0.029. Experimentally, it has been determined that using 1.5 times the

standard deviation greater than the mean score as the threshold for eliminating poorly scoring chunks is an effective way to significantly improve the match scores of vehicles when matched against another vehicle of the same class. The results show that this method does not significantly improve the scores of vehicles of a different class. This method greatly improves the ability to use the template of one vehicle of a particular class to represent all vehicles of that class.

The chunk-discarding algorithm operates in the following way:

```
Compute scores for all chunks;

mean = weighted average of chunk scores
       (weighted by # of pixels)
std_dev = standard deviation of chunk scores
t = standard deviation threshold for rejection

For each chunk, C:

    If ((score(C)-mean) > t * std_dev;
        Mark C for removal;

Remove scores of marked chunks;
Recalculate mean to get final score;
```

6. RESULTS AND EVALUATION

6.1. Chunks

The chunking subsystem performs very consistently throughout the chunking of all vehicle templates. Generally, the chunker produces between six and twelve chunks per template. As noted earlier, these chunks are created to cover a specific area of interest on the template. Each is designed to be centered at a point of high variance (peak, valley, ridge, etc.), and to expand in such a way that allows for the maximum amount of information in that chunk. Visual examination of these chunks reveals that they seem to be dispersed appropriately within the template mask. Figure 5 shows the chunks resulting from a Vehicle Variant 4 template at 30 degrees. The chunks are color-coded according to their scores when matching against a Vehicle Variant 2 chip at 30 degrees. The scores, are error measurements, so a high score represents a bad match. The lighter-colored chunks are those that score poorly. The chunks of the poor-scoring area are discarded in each case resulting in an improved score for the match. Thus, the chunks are able to achieve the goal for which they are created.

6.2. Scoring

To illustrate the scoring portion of the chunk-based scoring system, the results of the match in Figure 5 will be used. In this case, a Vehicle Variant 2 chip is being matched against a Vehicle Variant 4 template. A good score is desirable if we wish to use the Variant 4 template to represent all vehicles.

The chunk scores are 0.010, 0.014, 0.016, 0.020, 0.046, 0.020, 0.013, 0.026, 0.061, *0.127*, 0.044, and 0.026.

The score resulting from standard MSE scoring is 0.036. In the context of this experiment, this score is not particularly good or bad. The tenth chunk in the list has a very poor score of 0.127. This poor score is likely the result of a particular vehicle variation that is not shared between Variant 2 and Variant 4. With chunk-based scoring, this score falls outside of the standard deviation threshold of 1.5 and is discarded. The result is an overall score of 0.026, a substantially better score than the original. Matches against targets of other classes (confusers) result in all chunk scores being poor. In this case, it is less likely for chunk scores to be discarded. However, if a poor score is discarded from a group of poor scores, the end result remains poor. Therefore, out-of-class targets will score poorly against the chunked vehicle templates.

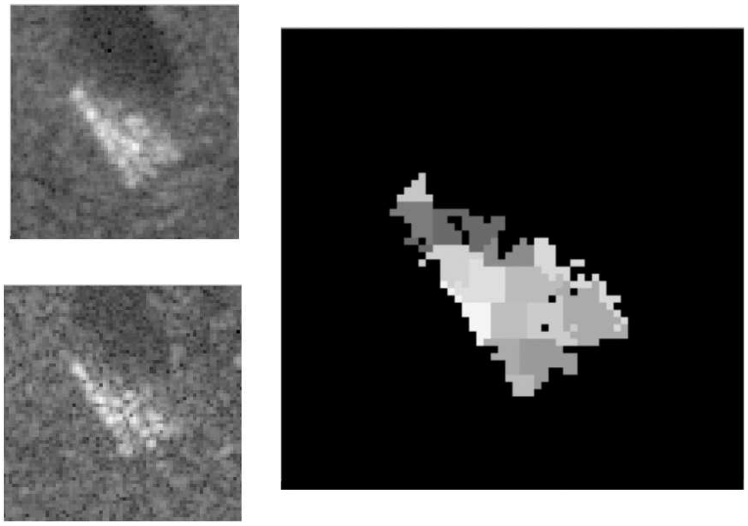

Figure 5. Variant 2 chip (top), Variant 4 template (bottom), and the resulting chunk scores (right) from matching the chip against the template (darker = lesser error, lighter = greater error).

6.3. Results

An example ROC curve is given in Figure 6. The x-axis represents the probability of false detection (Pfd) and the y-axis represents the probability of detection(Pd). The plot in Figure 6 represents the ability of the Vehicle Variant 2 template to distinguish Vehicle Variant 1 from Confuser Class 1. Chunk-based scoring significantly outperforms standard scoring. The left histogram in Figure 7 compares the score distributions of Variant 1 and Confuser Class 1 when matched against the Vehicle Variant 2 template using standard scoring. Notice that there is a significant amount of overlap in the distributions. The right histogram in Figure 7 compares the score distributions for the same matching, but using chunk-based scoring instead. The amount of overlap is reduced when using chunk-based scoring, thus resulting in a better ROC curve. The closer a ROC curve can get to (0,1) the better the system's ability to detect a target.

6.3.1. Standard MSE scoring vs. chunk-based scoring

Figure 8 shows the ROC curves resulting from using Vehicle Variants 2 and 5 to represent the entire class of vehicles in the matching system. The results are similar for all other Variants. In every case, the use of chunk-based scoring improves the ability to represent the whole class of vehicles using one vehicle variant as a template.

6.3.2. Comparing performance of each vehicle as template

Figure 9 compares the ROC curves of each vehicle variant representative to show each variant's ability to represent the entire class of vehicles. Variant 2 is clearly the best choice as a chunked template to represent all vehicles in the class. It is also the intuitive choice to the human eye. It appears to have the most similarities with the other vehicle variants. The ROC curves shows that Variant 2 can achieve approximately 90% probability of detection with a 10% probability of false detection. While these numbers are not optimal, they are a significant improvement over the non-chunk-based method.

7. CONCLUSIONS

This paper has proposed a method for reducing the computational cost of a template-based automatic target recognition system. By representing a group of similar vehicles with one chunked template, the speed of an ATR system can be greatly improved without suffering a great deal of matching performance. In the experiment given here, an eight-fold speed-up was achieved, while maintaining a 90% probability of detection and a 10%

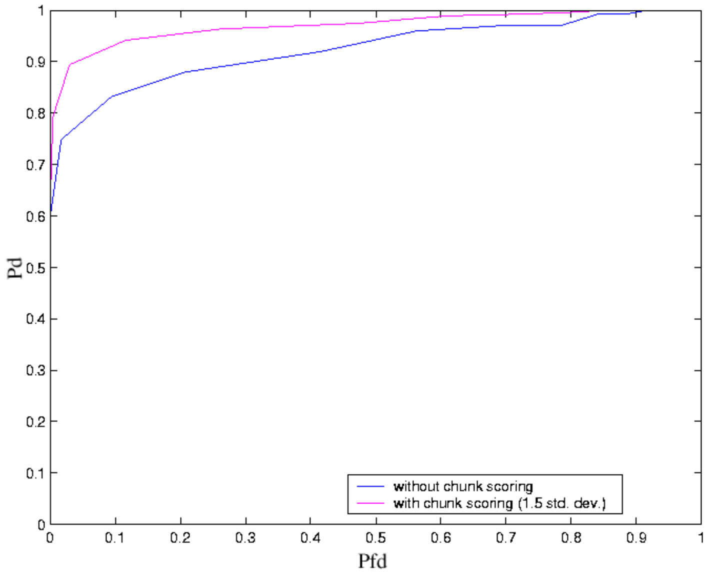

Figure 6. ROC curves show effectiveness of a Variant 2 template to distinguish Variant 1 from Confuser Class 1.

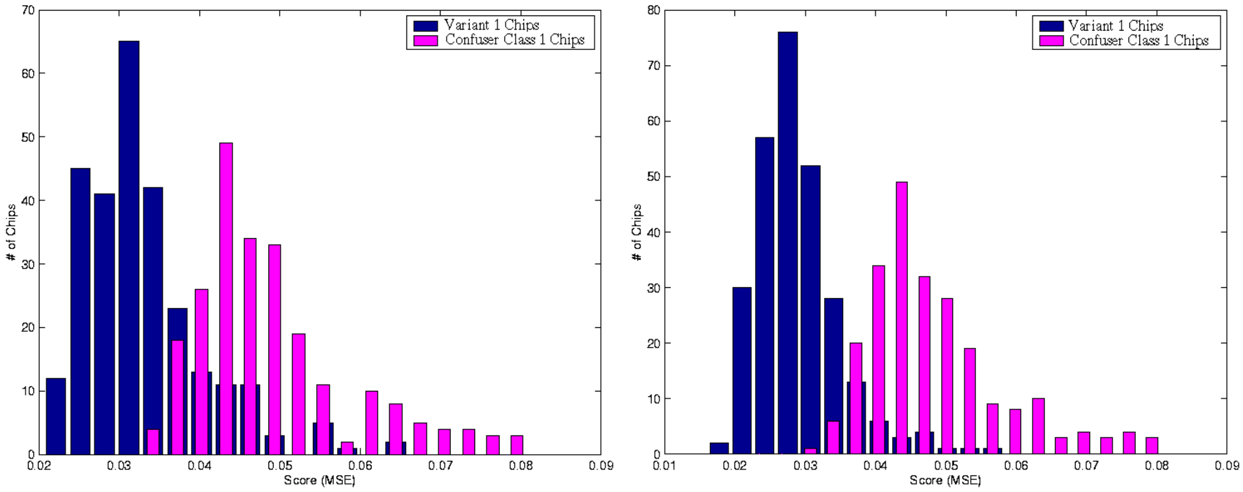

Figure 7. Score distributions as a result of matching all Confuser Class 1 chips and all Vehicle Variant 1 chips against a Vehicle Variant 2 template using the standard scoring method (left) and chunk-based scoring (right).

probability of false detection. ROC curves can be used to test the ability of a chunked template to represent a particular group of vehicles. Any template-based ATR system can implement the method given here to achieve a performance boost, given that the system has groups of similar target vehicles.

REFERENCES

1. S. A. Stanhope, E. Keydel, W. Williams, V. Rajlich, and R. Sieron, "The use of the mean square error matching metric in a model-based automatic recognition system," in *Algorithms for Synthetic Aperture Radar Imagery V*, E. G. Zelnio, ed., pp. 360–367, 1998.
2. S. Myers, "A study of intra-class variability and class models," Master's thesis, University of Cincinnati, 1999.

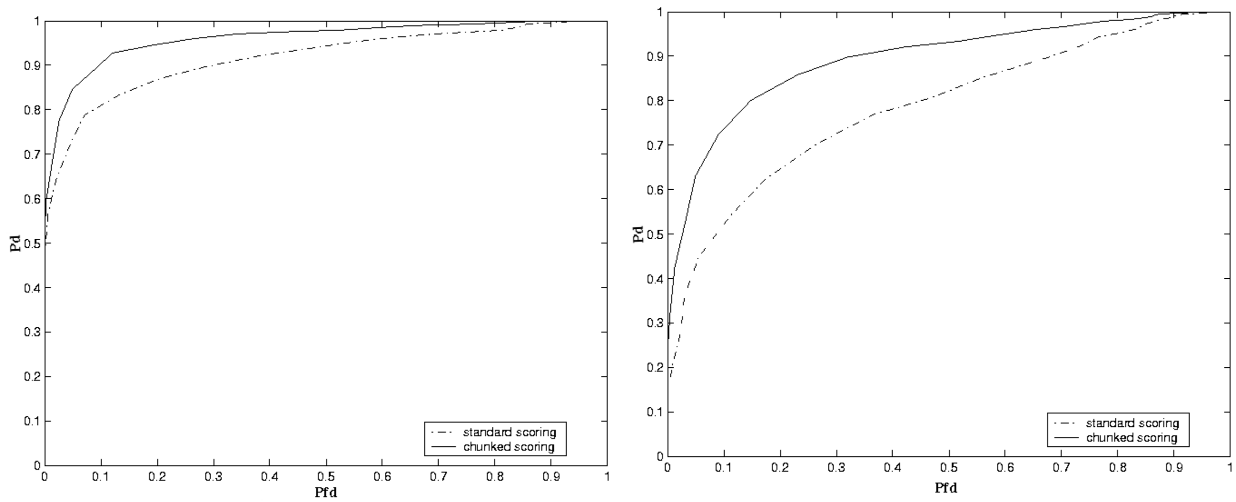

Figure 8. ROC comparison of chunk-based scoring vs. standard scoring using Vehicle Variant 2 (left) and Vehicle Variant 5 (right) as a template.

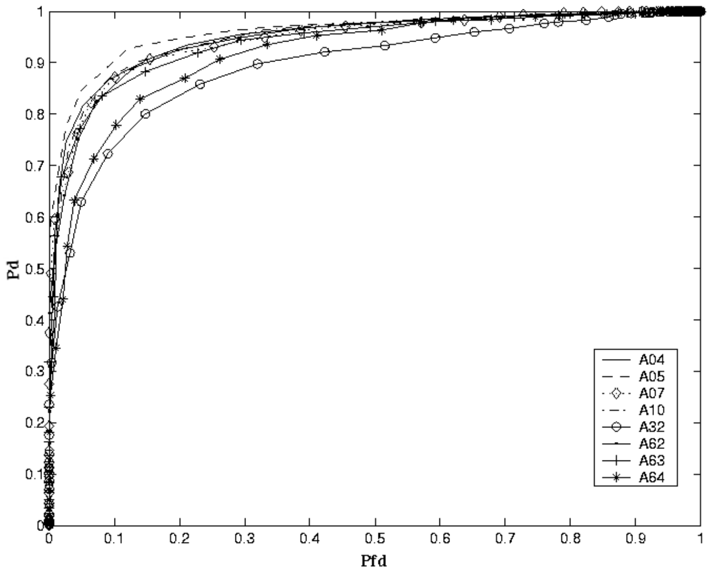

Figure 9. ROC comparison of chunk-based scoring using each vehicle variant as a template.

3. D. E. Dudgeon, "Atr performance modeling and estimation," *Digital Signal Processing* , 2000.
4. S. Chang and R.-H. Park, "Segmentation based on fusion of range and intensity images using robust trimmed methods," *Pattern Recognition* **34**(10), pp. 1951–1962, 2001.
5. K. Mayumi, "Information, pseudo measures and entropy: An elaboration on nicholas georgescu-roegen's critique," *Ecological Economics* **22**, pp. 249–259, September 1997.