Efficient Indexing for Object Recognition Using Large Networks *

Mark R. Stevens Charles W. Anderson J. Ross Beveridge
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
{stevensm,anderson,ross}@cs.colostate.edu

Abstract

Template matching is an effective means of locating vehicles in outdoor scenes, but it tends to be a computationally expensive. To reduce processing time, we use large neural networks to predict, or index, a small subset of templates that are likely to match each window in an image. Results on actual LADAR range images show that limiting the templates to those selected by the neural networks reduces the computation time by a factor of 5 without sacrificing the accuracy of the results.

1 Introduction

Neural networks are often used to extract complex, nonlinear relationships among the variables of a set of data. However, in this article we use the nonlinear mapping capability of neural networks to reduce the computation associated with the essentially linear procedure of template matching for automatic target recognition (ATR). Template matching for vehicle identification requires the application of numerous templates to rectangular windows of an image [12, 8, 5, 6]. Each template corresponds to a particular type of vehicle at a particular orientation. The windows are small, rectangular subsets of pixels just large enough to contain each vehicle and be within the image bounds. After all templates are applied to an image, the vehicle type and orientation corresponding to the best matched templates are returned as the most likely to be present in the image [3].

In order to detect a wide range of vehicle types and orientations, a large number of templates must be applied. For template matching to be feasible, processing time must be fairly quick. Computation time for template application is usually $O(n)$ where n is the number of templates. Two methods are typically used for reducing this processing time: parallel hardware is added to increase computation speed [4, 7], and focus-of-attention mechanisms are used to predict the vehicle location [1]. Neither method reduces the number of templates to apply.

A different approach is to index, or select, a subset of the available templates to apply to each window. Here we describe how neural networks can be used to predict the utility of applying each template to a given window. Only those templates predicted to match well are applied. We show that our neural-network indexing approach reduces the computation time by a factor of 5 without sacrificing vehicle and orientation identification on a set of real images.

The remainder of this article is organized as follows. In Section 2, we describe the methods with which we generate and use templates, collect the training and testing data, and train and apply the neural networks. In Section 3, we present the results of our experiments and our conclusions are stated in Section 4.

2 Methodology

In this section, we first define the standard, brute-force approach of applying all templates. Then, we describe how we train and use neural networks to select a subset of the templates to apply. Figure 1 provides an overview of the entire system detailed in the following subsections.

* This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) Image Understanding Program

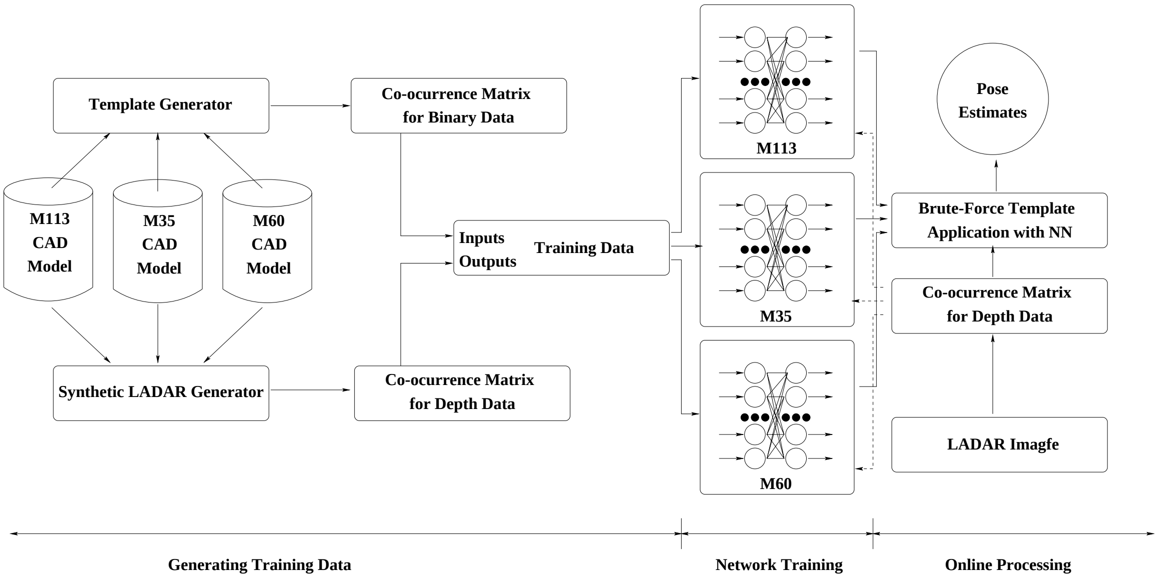
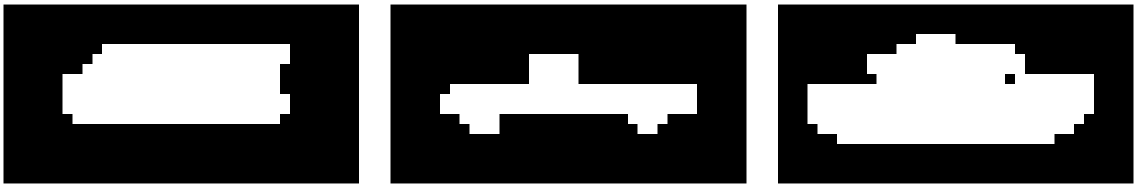


Figure 1: Overview of template generation, Neural Network Training, selection by neural networks, and application to LADAR image.

2.1 Generation and use of Templates

Template generation begins with the generation of synthetic images from three-dimensional models. The models used were reduced from highly-detailed BRLCAD models [11] to a level-of-detail more closely related to the granularity of a LAser raDAR (LADAR) sensor [10]. The reduction phase preserves salient features which are typically present in actual, coarse, range data that is obtained by a LADAR sensor as it scans a scene and produces an image of depth values.

Models of three vehicles, designated M113, M35, and M60[1], were used. In order to generate a template of a vehicle at a particular orientation, the model is placed into a scene and ray traced. The result is a binary image of ray hits and misses. Figure 2 shows templates generated for each vehicle in a sideways orientation; white represents a ray intersection with the vehicle and black represents a miss. Each vehicle is placed in 1,800 different orientations (72 changes in azimuth x 5 changes in elevation x 5 changes in up angle), resulting in 5,400 templates for the three vehicles. Our templates were all 40 x 20 pixels.



a. M113 **b.** M035 **c.** M060

Figure 2: Example templates for sideways orientations of each vehicle.

For each template, we calculate its *co-occurrence matrix*, which is a sparse, binary matrix of size $n \times n$ where n is the number of pixels in the template. Each entry in the matrix corresponds to one pair of pixels in the image. Nonzero entries are those for which the pair of pixels contain different values in the binary template and are within two rows and columns of each other in the template. The co-occurrence matrix is a commonly used representation, because it compactly designates only significant transition pixels which

under grants DAAH04-93-G-422 and DAAH04-95-1-0447, monitored by the U. S. Army Research Office.

[1]The M113 is an armored personnel carrier (APC), M35 a truck, and the M60 is a tank.

most likely correspond to an objects occluding contour. Since the template matching algorithm used here is based on the object's boundary, the co-occurrence matrix was the obvious representation choice.

To match the templates against a range image, the image is divided into all windows of the same shape as the templates and each window's co-occurrence matrix is calculated. First, the range data is scaled to the appropriate depth of the template. Median smoothing is also used to reduce sensor noise. Nonzero elements of the data co-occurrence matrix are those for which the difference in range data is greater than the expected depth difference between an on and off-vehicle pixel.

Once the co-occurrence matrix for a window has been generated, it is component-wise ANDed with the co-occurrence matrix for each template. The number of nonzero bits in the resulting matrix are counted and divided by the number of nonzero elements in the template co-occurrence matrix. The result is a number in the range $[0, 1]$ representing the quality of that template's match to the data in that window. The twenty-five templates which best match any sub-window across the entire range image are recorded. Templates whose degree of match less than 0.6 are discarded to reduce the number of false positives. The remaining templates' vehicle types and orientations along with the locations of the windows' center pixels are stored as predictions of the vehicles present in the LADAR data.

2.2 Training and Use of Neural Networks for Indexing

Typically, all templates are applied to a given window of an actual range image. Computational requirements can be reduced if we can limit the number of templates to compare to a given window using a decision process requiring less computation than that of the all-templates method. We used three feed-forward neural networks, one for each vehicle type, to implement such a decision process. Each network receives as input a portion (specified below) of the co-occurrence matrix of the window being processed. The outputs of the networks are predictions of the degree of match between the window and all templates for the corresponding vehicle. Rather than requiring the networks to learn accurate predictions, we only assume the predictions are approximately correct in relative ranking. The twenty-five templates predicted by the networks to best match the window are then directly compared to the data using the same component-wise AND operation described earlier.

The size of the input vectors to each network was reduced as follows. The co-occurrence matrices for the entire template set for each vehicle was examined and elements which were zero across all matrices were discarded. For our 40×20 pixel templates, a full co-occurrence matrix is 800×800 and contains 640,000 entries. Retaining only nonzero elements reduced this to $3,400$ values, still a large number but much more practical than $640,000$. Each network must have an output for each possible template. For each vehicle type there are $1,800$ templates, so each network has $1,800$ outputs.

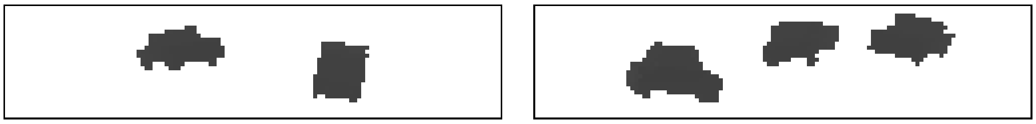


Figure 3: Two synthetic LADAR images.

The networks were trained with standard error back-propagation with momentum [9]. The training data was generated from 250 synthetic LADAR images with the various vehicles placed in the scene at a depth of $85m$ and at different orientations. Figure 3 shows two such images. For each window of these images, the data co-occurrence matrix is compared with the matrices for all templates and the degrees of match are recorded. The training data is formed by pairing each co-occurrence matrix for all windows with the degrees of match for all templates. This results in approximately $50,000$ examples per network. A cross-validation procedure was used for which the data was randomly divided into 80% for training, 10% for validating, and 10% for testing. The validation set was used to choose the best training epoch, but, for the results reported in the next section, the best epoch was always the last.

The best network topology and learning parameters were found empirically by training the system and observing performance. It was found that the best configuration had one hidden layer of five units. The learning rate was a constant 0.0001 and the momentum rate was 0.1. The networks were trained for 12 epochs, requiring two days on a Sun Sparc 20. The RMS error on the test data after 12 epochs was 0.056 for the M113 network, 0.063 for the M35 network, and 0.055 for the M60 network (recall that the network outputs are constrained to the range $[0, 1]$).

3 Results on Actual LADAR Data

As described in the previous section, the networks were trained only on synthetic imagery. After training, the networks were tested on 15 real LADAR images[2] from the Fort Carson data set [2]. Figure 4 shows four of the fifteen images tested. Shown in each figure are the original range image and the pixels at the center of windows for which the all-templates and the neural-network indexing methods predicted degrees of match greater than 0.6. For actual LADAR images, lighter pixels are those farther from the sensor whereas darker are closer. The template matching results are displayed as black for M60 templates, gray for M113 templates, and light gray for M35 templates.

To assess if the correct vehicles are found at the proper orientations, the top 25 templates were recorded for each image for both methods. This results in a total of 375 templates for all 15 images. When all templates were applied, the correct template was found for only 65 of the 375 cases, a success rate of 17.3%. A template was labeled *correct* if the template's center pixel was within 1 pixel of the ground truth and within 45° degrees of the true vehicle orientation. The all-template algorithm completely failed to detect any vehicles in 4 of the 15 images examined. Thus, better templates are needed in order to improve performance.

To assess if the neural-network was able to correctly predict which templates to use, values from the all-template method and the neural-network indexing method were recorded. Figure 4 shows the difference between the two methods. As can be seen in these images, neural-network indexing performs equally well when the vehicle is actually present in the window being processed. The largest discrepancies occur when no vehicle is in the vicinity of the pixel being processed. By adjusting the threshold value, the number of false positives in each case is reduced, thus removing some of the clutter in the difference image.

In addition to examining the difference images, we also looked at how many times the neural network failed to predict the best probe to apply for a given image. Out of the 375 templates, the network failed to predict the best probe in the top twenty five only 14 times for a success rate of about 96%. Thus the neural-network was able to correctly index the top twenty-five probes which should be applied at each pixel.

The greatest advantage in using the neural-network indexing approach is the substantial decrease in processing time. Figure 5 shows a graph of performance time for all 10,000 windows examined in the 15 Fort Carson data images. Figure 5a shows the cumulative time, and Figure 5b shows the time in seconds for each window using the two approaches, which were run simultaneously (on a heavily-loaded Sparc 20 with 96 MB of RAM) to obtain a more accurate measurement. The large drop in time per window for the all-templates method corresponds to the completion of the neural-network indexing method.

The relative computational savings of the neural-network indexing method make sense in light of the following calculation. The network for one vehicle with five hidden units requires 3,400 x 5 multiplications for the first layer and 1,800 x 5 for the output layer, for a total of 26,000. Applying all templates for one vehicle requires 3,400 x 1,800, or 6,120,000 Boolean operations. While the comparison is naive and neglects other important factors, such as the summation and ranking operations, the reduced processing for the neural-network is still quite obvious.

[2] The entire Fort Carson data collection is publicly available through our web site: `http://www.cs.colostate.edu/~vision`

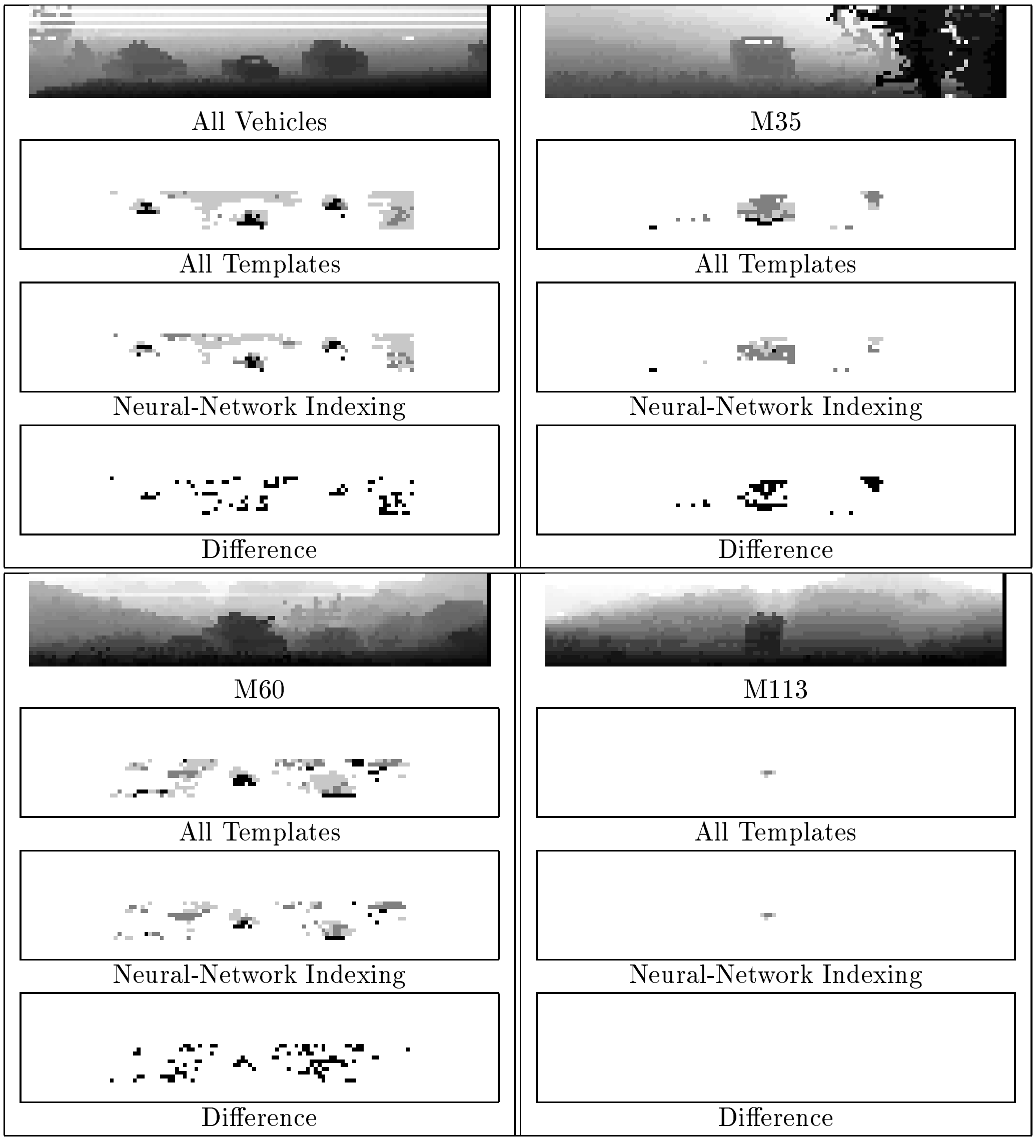
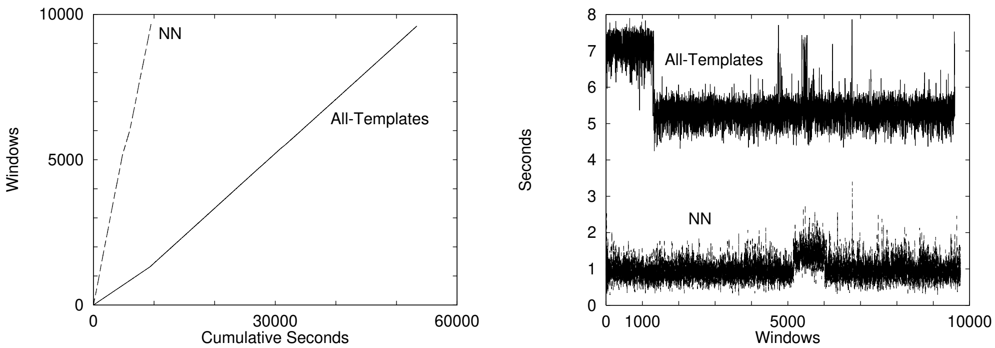


Figure 4: Sample Fort Carson data images and vehicle predictions made by applying all templates and only those selected by the neural-network indexing method. M60 matches are black, M113 matches are gray, and M35 matches are light gray.



a. Cumulative time **b.** Time per window

Figure 5: Execution time for all windows in 15 images.

4 Conclusion

The neural network approach is a viable method for predicting the templates that must be applied to each window of the range image. The weak link in the process appears to be in the all-templates algorithm which is unable to successfully detect, identify and determine the orientation of vehicles in the scene. In future work we will improve the performance of the all-templates method.

References

[1] J. Ross Beveridge, Allen Hanson, and Durga Panda. Integrated color ccd, flir & ladar based object modeling and recognition. Technical report, Colorado State University and Alliant Techsystems and University of Massachusetts, April 1994.

[2] J. Ross Beveridge, Durga P. Panda, and Theodore Yachik. November 1993 Fort Carson RSTA Data Collection Final Report. Technical Report CSS-94-118, Colorado State University, Fort Collins, CO, January 1994.

[3] James E. Bevington. Laser Radar ATR Algorithms: Phase III Final Report. Technical report, Alliant Techsystems, Inc., May 1992.

[4] Zhixi Fang, Xiabo Li, and Lionel M. Ni. Parallel algorithms for image template matching on hypercube SIMD computers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(6):835–841, November 1987.

[5] S. Ghosal and R. Mehrotra. Range surface characterization and segmentation using neural networks. *Pattern Recognition*, 28(5):711–727, may 1995.

[6] Alana Katz and Philip Thrift. Hybrid neural network classifiers for automatic target detection. *Expert Systems*, 10(4):243, nov 1993.

[7] V. K. Prasanna Kumar and Venkatesh Krishnan. Efficient parallel algorithms for image template matching on hypercube SIMD machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(6):665–669, June 1989.

[8] L.I. Perlovsky, J.A. Chernick, and W.H. Schoendorf. Multisensor atr and identification of friend or foe using mlans. *Neural Networks*, 8(7-8):1185–1200, 1995.

[9] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing, Volume 1: Foundations*, volume 1. The MIT Press, Cambridge, Massachusetts, 1986.

[10] Mark R. Stevens, J. Ross Beveridge, and Michael E. Goss. Reduction of BRL/CAD Models and Their Use in Automatic Target Recognition Algorithms. In *Proceedings: BRL-CAD Symposium*. Army Research Labs, June 1995.

[11] U. S. Army Ballistic Research Laboratory. *BRL-CAD User's Manual*, release 4.0 edition, December 1991.

[12] Allen M. Waxman, Micheal C. Seibert, Alan Gove, David A. Fay, Ann Marie Bernardon, Carol Lazott, William R. Steele, and Robert K. Cunningham. Neural processing of targets in visible, multispectral ir and sar imagery. *Neural Networks*, 8(7/8):1029, may 1995.