Computer Vision and Image Understanding **84,** 5–24 (2001)
doi:10.1006/cviu.2001.0938, available online at http://www.idealibrary.com on IDEAL®

Experimental Evaluation of FLIR ATR Approaches—A Comparative Study

B. Li[1]

Sharp Laboratories of America, 5750 NW Pacific Rim Boulevard, Camas, Washington 98607
E-mail: bli@sharplabs.com

R. Chellappa and Q. Zheng

Center for Automation Research, University of Maryland, College Park, Maryland 20742

and

S. Der, N. Nasrabadi, L. Chan, and L. Wang

Army Research Laboratory, Adelphi, Maryland 20783

Received June 13, 1999; accepted August 25, 2001

This paper presents an empirical evaluation of a number of recently developed Automatic Target Recognition algorithms for Forward-Looking Infrared (FLIR) imagery using a large database of real FLIR images. The algorithms evaluated are based on convolutional neural networks (CNN), principal component analysis (PCA), linear discriminant analysis (LDA), learning vector quantization (LVQ), modular neural networks (MNN), and two model-based algorithms, using Hausdorff metric-based matching and geometric hashing. The evaluation results show that among the neural approaches, the LVQ- and MNN-based algorithms perform the best; the classical LDA and the PCA methods and our implementation of the geometric hashing method ended up in the bottom three, with the CNN- and Hausdorff metric-based methods in the middle. Analyses show that the less-than-desirable performance of the approaches is mainly due to the lack of a good training set.

Key Words: automatic target recognition; Performance Evaluation.

1. INTRODUCTION

Many Automatic Target Recognition (ATR) algorithms have been proposed for FLIR imagery. Over the years, several classes of algorithms using neural, statistical, and model-based

[1] To whom correspondence should be addressed. Fax: (360)817-8436.

1077-3142/01 $35.00

approaches have been developed, and others continue to appear. The first two approaches use "learning" in the sense that they extract useful information from a given database (usually called the training set) for recognition/classification on a test data set not seen before. Information for recognition is implicitly extracted by the algorithm from the training data. On the other hand, model-based approaches build the target templates from models of the targets and then match them with target features from real images to fulfill the recognition task.

Although many approaches have been reported in the literature, in most cases they have been evaluated using only a small number of images. A direct comparison with a common database would therefore provide more insights for characterizing performance and measuring progress. Generally speaking, there are two approaches for ATR algorithm performance evaluation: theoretical and experimental methods. For FLIR ATR algorithms, theoretical evaluation would be difficult because underlying models used for characterizing target/clutter interactions are unrealistic, and good models for target/clutter signature prediction assumes the availability of too much information. Therefore theoretical evaluation results usually do not scale well to real image data, and commonly used techniques for estimating Bayesian error only provide loose bounds. Experimental evaluation, on the other hand, needs a reasonably large data set in order to be meaningful. In this paper we present an experimental evaluation of several prominent FLIR ATR algorithms.

The first algorithm we evaluated is the convolutional neural network (CNN). LeCun and Bengio have shown that a highly structured multilayer perceptron can be used to solve optical character recognition problems [1]. This highly structured multilayer perceptron, widely known as CNN, has also found application in face recognition and FLIR ATR. For example, researchers have been using CNN for ATR using FLIR images. Principal component analysis has been effectively used for face recognition [2]. PCA is potentially applicable to FLIR ATR provided a region of interest has been detected in the input image. Linear discriminant analysis is also a common tool for classification problems [3]. Two classifiers, based on learning vector quantization (LVQ) and modular neural networks (MNN), have been developed for FLIR ATR [4, 5]. The Hausdorff metric, developed for comparing two point sets, has been used by Doria and Huttenlocher in a distance-based matching technique for FLIR ATR problems [6]. As a parallelizable model-based recognition technique, geometric hashing, first introduced by Lamdan and Wolfson [7], has been applied to FLIR target identification by Akerman *et al*. [8]. CNN, LVQ, and MNN are obviously learning algorithms. PCA and LDA are also learning approaches since both the eigenspace structure in PCA and the discriminant functions (or discriminant hypersurfaces) in LDA are completely decided by the training data. Hausdorff metric and geometric hashing-based approaches are typically model-based in that they operate by first constructing a model database and then compare local features extracted from an image with the prestored target models. However, if the model database is constructed from a real training set rather than from a computer-aided design (CAD) model, then these two approaches could also be treated as learning algorithms in the sense that the models are *learned* from the training set.

In this paper, these approaches are evaluated using a database of more than 17,000 images (including the training and the testing sets) and their performances are compared. These approaches are chosen because some of them have been reported to be successful for FLIR ATR at least for small databases (for example, Hausdorff metric-based matching, geometric hashing, and MNN), and others such as PCA have been widely applied to appearance-based recognition (although not necessarily FLIR ATR). In our study, evaluation and comparisons

are characterized by the recognition rate, which is the primary metric for the ATR problem. Not much attention is paid to the space–time complexity of the algorithms. The experiments are based on our implementation of the algorithms reported in the literature.

It is worth pointing out that most ATR designs consist of two stages: target detection and target recognition. In this paper, our discussion is restricted only to the recognition stage since all images in the database are already the result of a detection stage. Our primary motivation in writing this paper is to encourage large-scale experimental evaluation of FLIR ATR algorithms.

The organization of the remainder of the paper is as follows. In Section 2, we give an overview of the approaches under discussion. The database used in this study is briefly described at the beginning of Section 3. After that, some implementation issues are discussed, and the experimental results are presented and compared, followed by analyses and discussion. Finally, Section 4 summarizes the paper.

2. OVERVIEW OF THE APPROACHES

We begin with a brief discussion of the approaches. Although most of the important and fundamental issues are addressed here, the interested reader should consult the original papers or books for further details on specific approaches evaluated in this paper.

2.1. Convolutional Neural Network

If we treat an image as a high-dimensional vector of gray level values formed by concatenating its rows together, then theoretically, we could train a large enough multilayer perceptron to perform any required mapping from this vector space to any other space [9], including the mapping that accurately classifies the targets in the FLIR ATR application. However, this problem is ill-posed in that there are many solutions that ideally fit the training data but do not generalize well to test images. In addition, a simple multilayer perceptron has little invariance to translation or local deformation of the images. CNNs are so designed that they allow some degree of translation and local deformation and generalize well by using three basic ideas: local receptive fields, weight sharing, and spatial subsampling.

An illustrative CNN structure is shown in Fig. 1. The network consists of an input layer, several hidden layers, and an output layer. Each hidden layer has one or more planes called feature maps. The input layer is of the same size as the input image, and each node corresponds directly to a pixel in the input image. For the hidden layers, each plane consists of nodes that receive input from a small neighborhood in the planes of the previous layer.

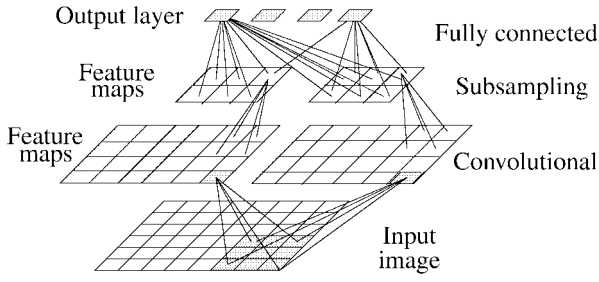

FIG. 1. A typical configuration for convolutional neural networks.

Weight sharing is achieved by assigning an identical set of synaptic weights to all the nodes within each plane. Each plane is called a feature map since it has a fixed feature detector (convolutional kernel) that is convolved with a local window that is scanned over the planes in the previous layer. Multiple planes are usually used in each layer so that multiple features can be detected. There are two types of hidden layers. One is convolutional as described above. The other is a subsampling layer that does a weighted local averaging and subsampling operation. The network is trained with the classical back-propagation algorithm. By using CNN, we avoid explicit feature extraction, and it is up to the neural network to figure out what are the useful features for classification. This is appealing especially in situations where we do not know what could be the right features.

2.2. Principal Component Analysis

A common statistical tool for analyzing the interrelations between variables is PCA, also known as the Karhunen–Loeve expansion. PCA's utility to image analysis is in part due to the correlation between nearby pixels in a real-world image. The idea of PCA is to compress most of the information about the data into a limited number of eigenvalues and eigenvectors. High-dimensional data are projected onto a smaller number of dimensions chosen so as to maximize the variances along the new axes which are mutually orthogonal. Given data with a 2-D probability distribution in the shape of an ellipse, PCA returns the two axes of the ellipse. The first principal component, corresponding to the longer axis, is the linear descriptor, which gives the most information about the data.

In a recognition problem, eigenspace approaches, first construct, by using PCA, a small set of basis images from the training set, and classification is actually done in the subspace spanned by the set of basis images. To perform PCA on training images, for each $n \times m$ image, an nm-D vector is formed by concatenating its rows together. By treating the vectors so obtained as random variable samples, and performing PCA on the vector space, we can get a set of orthogonal basis vectors, which are the eigenvectors of the sample covariance matrix R defined as

$$R = \frac{1}{N}\sum_{i=1}^{N}(X_i - M)(X_i - M)^{\mathrm{T}}, \tag{1}$$

where X_i is an image vector, N is the number of images, and $M = \frac{1}{N}\sum_{i=1}^{N} N_i$ is the sample mean. To achieve data reduction, we keep only the first k eigenvectors, where $k \ll nm$, corresponding to the k largest eigenvalues obtained in PCA. These k vectors form the eigenspace for encoding, reconstruction, or recognition.

To be more specific, a simple example of how recognition operates follows: each image in the training set is approximated by its projection on the subspace obtained above. All the projections of the training images are properly labeled, in the k-dimensional space, according to their classes. A new image of unknown class is first projected onto the eigenspace, and then its projection is compared with those of training images. The class of the closest training image is assigned to the class of the the new image.

2.3. Linear Discriminant Analysis

PCA is effective in capturing the most information about a given data set, and hence is useful for data reduction and reconstruction, but it is not necessarily suitable for discriminating

among classes. The reason is obvious: PCA describes the major variations of the data (all classes are considered together), but these variations may well be irrelevant to how the classes are separated from each other. On the other hand, LDA [3] projects the data onto a new (sub-)space where the between-class scatter is maximized and the within-class scatter is minimized. This ensures the optimality criteria is based on separability. Let *Proj* be a projection matrix that projects a vector X (high-dimensional, like that formed from an image) into a subspace: $Y = Proj^{\mathrm{T}} X$. X is a sample vector from the data set of N_{c} classes with class labels C_i, $i = 1, 2, \ldots, N_{\mathrm{c}}$. The within-class scatter matrix is defined as

$$S_{\mathrm{w}} = \sum_{i=1}^{N_{\mathrm{c}}} Prob(C_i) E[(X - M_i)(X - M_i)^{\mathrm{T}} \mid C = C_i], \tag{2}$$

where M_i is the mean for class C_i, $E[\cdot|\cdot]$ denotes conditional expectation, and $Prob(C_i)$ is the probability of ith class. Let M denote the grand mean vector of all samples from all classes, then the between-class scatter matrix is defined as

$$S_{\mathrm{b}} = \sum_{i=1}^{N_{\mathrm{c}}} Prob(C_i)(M_i - M)(M_i - M)^{\mathrm{T}}. \tag{3}$$

In LDA, the projection matrix *Proj* is chosen to maximize the ratio $det(S_{\mathrm{b}})/det(S_{\mathrm{w}})$, i.e., to maximize the between-class scatter while minimizing the within-class scatter. It has been shown that the ratio is maximized when the columns of *Proj* are the eigenvectors of $S_{\mathrm{w}}^{-1} S_{\mathrm{b}}$ associated with the largest eigenvalues. In practice, the mean, expectation, and probability are estimated from the samples.

2.4. Learning Vector Quantization

LVQ is one of the popular methods for globally modifying decision boundaries. A basic version of LVQ is as follows [10]. Assume that a number of codebook vectors m_i are placed into the input space to approximate various domains of the input vector x by their quantized values. Let $x(t)$ be a sample of the input and let $m_i(t)$ represent sequences of the m_i. The basic LVQ modifies the $m_i(t)$ according to the equation

$$\begin{cases} m_{\mathrm{c}}(t+1) = m_{\mathrm{c}}(t) + \eta(t)[x(t) - m_{\mathrm{c}}(t)] & \text{if } x \text{ and } m_{\mathrm{c}} \text{ belong to the same class} \\ m_{\mathrm{c}}(t+1) = m_{\mathrm{c}}(t) - \eta(t)[x(t) - m_{\mathrm{c}}(t)] & \text{if } x \text{ and } m_{\mathrm{c}} \text{ belong to different classes,} \end{cases} \tag{4}$$

where $m_{\mathrm{c}}(t)$ denotes the codebook vector matched to the input vector $x(t)$. No other codebook vectors are updated. Also, $0 < \eta(t) < 1$, and $\eta(t)$ may be constant or decrease monotonically with time. Many variants of LVQ have been proposed and applied for classification purposes [10, 11]. After the learning process is completed, the codebook vectors are used to define the class borders in the input space according to the nearest-neighbor rule.

2.5. Modular Neural Network

The modular neural network classifier employs a hierarchical neural network architecture, which contains several groups of modular neural networks [12], with each group consisting of a committee of neural networks [13]. Each group is trained for a particular subset of data vectors, for example, a particular region of an input image. A gating network is trained to

select or combine the outputs of each group of networks in order to form the final output. Partitioning (or decomposition) the data set into subsets depends on the application.

A committee of networks offers several advantages over a single neural net. First, it is robust to partial target occlusion if the input image is decomposed into subregions. Each neural network in the committee, receiving input information from only one of the local receptive fields on a target image, is trained independently. The occlusion of some portion of a target has no effect on the decisions of the other neural networks at the classification stage. Secondly, by dividing the input vector into subvectors, we allow each neural network classifier to have fewer inputs and hence less complexity. This allows better generalization with a smaller training set, and also reduces training time. Finally, stacked generalization [14] can be used to further improve the performance of the classifer by combining the outputs of each neural network in a nonlinear manner.

2.6. Hausdorff Metric-Based Matching for Recognition

The Hausdorff distance is a mathematical tool used to compare two sets of points in terms of their least similar members. The distance is defined as the maximum of the minimum distances from all members of point set A to point set B. Formally, given two finite point sets $A = \{a_1, a_2, \ldots, a_p\}$ and $B = \{b_1, b_2, \ldots, b_q\}$, the Hausdorff distance is defined as

$$H(A, B) = \max\{h(A, B), h(B, A)\}, \tag{5}$$

where

$$h(A, B) = \sup_{a \in A} \inf_{b \in B} \|a - b\| \tag{6}$$

and $\|\cdot\|$ is an underlying norm between two points. The function $h(A, B)$ is called the directed Hausdorff distance from A to B. Sometimes $h(A, B)$ is also referred to as Hausdorff distance for short. In effect, $h(A, B)$ ranks each point of A based on its distance to the nearest point of B; the largest ranked such point (the most mismatched point of A) specifies the value of the distance. Intuitively, if $h(A, B) = d$, then each point of A must be within distance d from the nearest point of B (the most mismatched point). Hence Hausdorff distance provides a measure of match between two point sets. Note that this distance measure differs from binary correlation and correspondence-based techniques such as point matching methods, in that there is no pairing of points in the two sets being compared.

A simple way to use the Hausdorff measure for recognition is as follows. First, a set of target models is formed by extracting and storing the intensity edges of corresponding target images. When an unknown target chip[2] is presented, its edge map is first extracted, then matched against the model set. The model with the highest matching score (i.e., smallest Hausdorff distance) will decide the class of the target chip. However, some implementation issues need to be taken into consideration, as discussed in detail in Section 3.

2.7. Geometric Hashing-Based Approach

First introduced by Lamdan and Wolfson [7, 15], the geometric hashing paradigm presents a unified approach to the representation and the matching problems, with applications to

[2] A target chip is a small image representing the region of interest. For the database used in this paper, a target chip contains a centered target. The Hausdorff measure can also be used to locate a target region in a large image as in [6].

object recognition under various geometric transformations. To illustrate the basic idea, let us consider a 2-D rigid object, represented by a set of geometric feature points, under affine transformation. Any triplet (**e0**, **e1**, **e2**) of noncollinear points forms an affine basis, and under this affine basis, any point P in the plane can be represented by a pair of scalars (α, β), such that

$$P = \alpha(\mathbf{e1} - \mathbf{e0}) + \beta(\mathbf{e2} - \mathbf{e0}) + \mathbf{e0}. \tag{7}$$

We call the pair (α, β) the new coordinates of P under the given affine basis. The good thing here is that the new coordinates are invariant under an affine transformation, which is critical to recognition. In practice, to remove the dependence of the representation on a specific affine basis, we can represent the object points by their coordinates in all possible affine basis triplets, resulting in the following algorithm for building the model base [15].

For each model object:

(a) Extract the feature points.

(b) For each ordered noncollinear triplet, compute the coordinates of all other points under this affine basis. Use each such coordinate as an index to a hash-table, and record the pair (*model*, *basis*) at the corresponding location in the table.

The matching or recognition step is essentially a voting procedure: every recorded (*model*, *basis*) pair will receive a vote if the object finds an entry into that location in the hash table, and the pair winning the maximum number of votes is taken as a candidate; finally, additional verification/rejection is done on the candidate. Details can be found in [15].

One thing common to the geometric hashing approach and the Hausdorff metric-based approach is that they need to store a model database before recognition can proceed. The model database can be constructed in two ways: from a CAD model or from a real image set. One disadvantage of constructing the model base from a CAD model is that it is difficult to include all the variations of the FLIR image. In this paper, we construct the model database from the real image set, and then apply it to recognition. The shortcoming of this approach is discussed in Section 3.

3. EXPERIMENTAL RESULTS AND DISCUSSION

We have examined the performance of the aforementioned approaches on a large real database, with our implementation. Before discussing the implementation techniques and experimental results, it is useful to describe the database used.

3.1. Database

The intensity image data consists of a training set and a test set. The training set has 13,862 image chips while there are 3,460 images in the test set. The training data were collected under very favorable conditions at different ranges. Then the images were normalized to a fixed range with the target put approximately in the center. The images are of size 40×75 pixels. In the training set there are 10 classes (object types) (denoted as TG1, TG2, . . . , and TG10, respectively), each with approximately 1,300 images. The objects are shown in Fig. 2. For each object type, there are 72 orientations, corresponding to aspect angles of $0^\circ, 5^\circ, \ldots, 355^\circ$ in azimuth. The test set, which contains five different objects (TG1, TG2, TG3, TG4, and TG7), is more challenging in that the images were taken under less

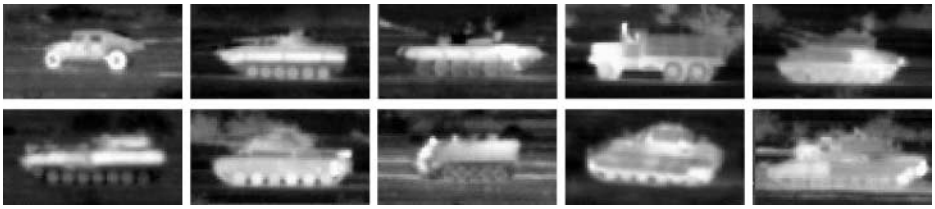

FIG. 2. Side view of all the 10 targets in the training set, denoted TG1, . . . , TG10, from left to right and top to bottom, respectively. TG1, TG2, TG3, TG4, and TG7 are those contained in the test set.

favorable conditions. The test set was similarly normalized, but the orientation was given very coarsely (every 45°). No other information about the database is available, such as operation history and environment temperature. The division of a benign training set and a more challenging test set is an attempt to simulate real applications, where a new image may have a signature pattern that is never seen in a training set. In addition to the intensity data, for every orientation of each target type, there is a binary silhouette image that was extracted from a CAD model, providing the contours of the objects at corresponding aspect angles. Figure 3 shows a TG4 at aspect angles of 0°, 45°, 90°, and 135°, respectively, and corresponding silhouettes.

3.2. Implementation Issues and Experiments

CNN approach. CNN has a highly structured configuration. Conventionally, the size of the receptive field is fixed within the same layer. Considering the fact that each feature map is in effect the result of a feature detector, we designed the CNN in such a way that the size of the receptive field can be specified for individual planes (even if they are in the same layer); thus we can choose, for example, 3×3, or 5×5 kernels. Furthermore, the kernels do not have to be squares; for example, they can be 3×5. This technique was motivated by the consideration that since the kernels act as feature detectors, kernels of different sizes should be used to detect features at different scales and orientations. In addition, the number of layers can be specified by the user, which adds flexibility to the network. Figure 4 shows one of the simulation results for a six-layered CNN and a four-layered CNN. The result using a six-layered CNN was provided by the U.S. Army Research Laboratory. The simpler four-layered CNN, designed at the University of Maryland, uses different kernel sizes as described above. Although the four-layered CNN has a slightly degraded performance on the test set, it needs much less time to train.

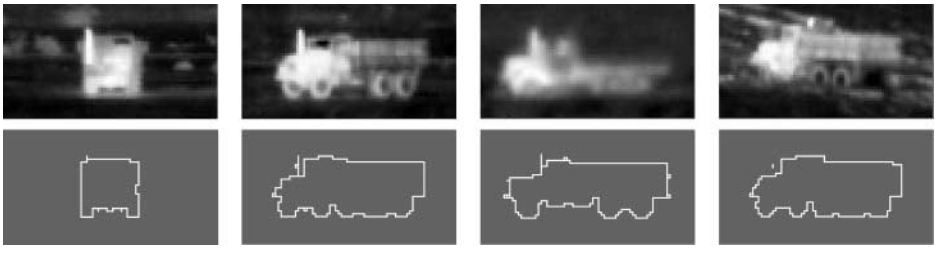

FIG. 3. Sample images of TG4 in the database and its corresponding silhouettes at aspect angles of 0°, 45°, 90°, and 135°, respectively.

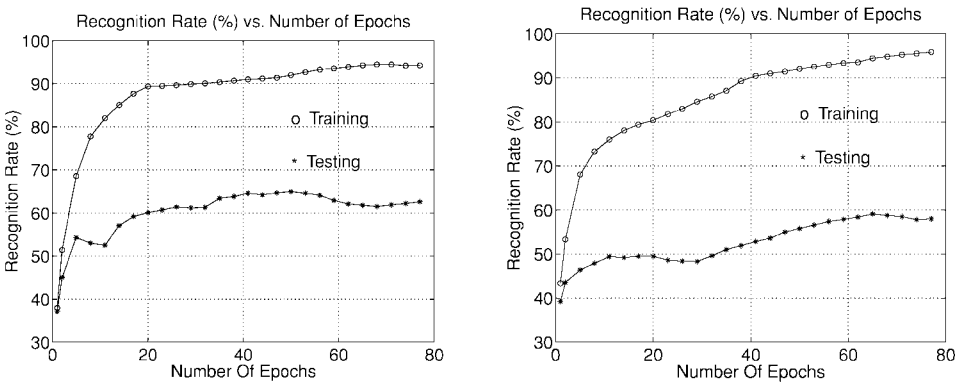

FIG. 4. Learning curves and recognition rates for a six-layered CNN (left) and a four-layered CNN (right), respectively (plotted every three epochs).

PCA-based approach. In the implementation of PCA, we first form the union of the silhouettes of all targets at different orientations, and then use this union to mask out background that is not desired in the PCA approach. Still, some background will be included especially for those pose at which the target appears smaller (e.g., the front view). The eigenspace was constructed by using subsampled training images (hence, the dimensionality of each image vector is around 600). To choose k, the dimensionality of the eigenspace, the residual mean-square error, or equivalently the sum of the k largest eigenvalues is used. In the experiments, different k values were tested. It was found that after k went beyond 40, there was no longer notable improvement on the recognition rate, and there was only a small difference when k varied between 20 and 40, which suggests that $k \leq 20$ would be sufficient for this database. After the eigenspace has been constructed from the training data, a codebook is formed by projecting all the training images onto the eigenspace. To classify a test image chip, we first project it onto the eigenspace and then find its closest match from the codebook (note that when testing with the training data, we need to discard the best choice and examine only the second best choice; otherwise we will get a meaningless 100% recognition rate). To improve the robustness of the recognition, instead of considering the best choice only (or the second best choice when using the training data to do the test), we find the top 3 (or 5, etc.) best choices and let them vote for the final recognition.

LDA-based approach. LDA operates in a similar fashion to PCA except that the projection matrix is formed differently. Of course, here we do not have the problem of choosing k since k is equal to or less than the number of classes. Similar test techniques used in PCA are adopted here.

LVQ-based approach. The LVQ-based scheme adopted in this paper is shown in Fig. 5, which was proposed by Chan *et al*. [4]. The algorithm creates templates from the training images and uses mean square error template matching to perform the recognition. The algorithm is designed by separating the training images into target-aspect groups. Each target-aspect group contains only one target type within a restricted range of viewing angles. The training images are then decomposed into wavelet subbands, and each wavelet subband of each target-aspect group is clustered using the well-known k-means algorithm [16] in order to create a set of target templates (code vectors). The LVQ algorithm is then applied to the templates to enhance discriminatory ability. Testing is performed by calculating the

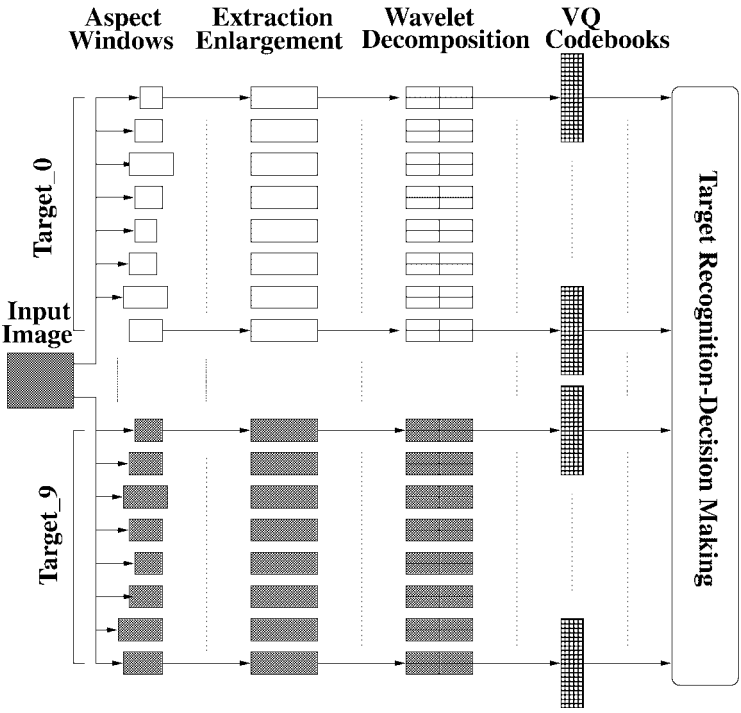

FIG. 5. The Learning Vector Quantization-based scheme evaluated in this paper.

mean square error between an input image and every code vector. The classifier achieves 99.72% accuracy on the training set and 75.12% on the testing set. The high training set performance is not surprising because VQ is a *universal classifier* in that it can classify targets arbitrarily well given a sufficiently large number of code vectors and an adequate training set, at the cost of high computational complexity. Figure 6 shows training and

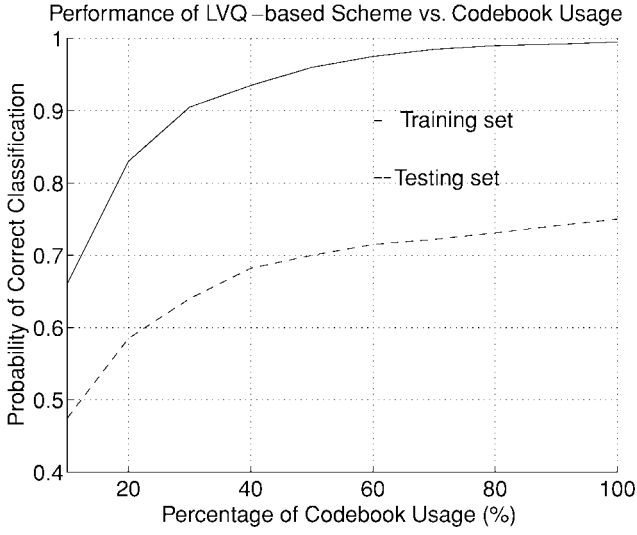

FIG. 6. The performance curve of the LVQ-based classifier as shown in Fig. 5.

TABLE 1
Heuristic Comparison of LVQ and MNN-Based Classifiers

Approach	LVQ-based	MNN-based
Training	Clustering	Discriminant function
Classification	Template matching	Posteriori probability
Implementation	VQ, LVQ	Neural network (MLP)
Feature type	Wavelet	Directional variance
Feature decomposition	Wavelet subbands	Local receptive field
Parameterized	Aspect window by target	Silhouette category
CPU time to test testing set	1936.0 s	197.0 s
Training set	99.72%	95.49%
Testing set	75.12%	75.58%

testing set performance as a function of the number of code vectors in the codebooks. The algorithm is described fully in [4]. The computational complexity is shown in Table 1 and is measured by the CPU time for running 3,460 target images on the testing set using a SUN Ultra 1.

MNN-based approach. Figure 7 shows the structure of an MNN-based scheme introduced by Wang *et al.* [5], where the partitioning of the input is realized by splitting the image into six disjoint regions (this is called *feature decomposition* by the authors). A subsampled version of the input image is also added as the input signal to the network, and it is also partitioned, as shown in the figure. With similar network structure, we also used another partitioning scheme called *data decomposition* where the partitioning of the data set into subsets is based on the similarity of the target silhouettes. The silhouettes are binary images formed from CAD models of the targets, and clustering is performed using the k-means algorithm on the binary silhouettes. Although data partitioning is important to the MNN-based approach, there is no optimal algorithm for that purpose. So the choice of partitioning was guided by intuition and confirmed by experiment. No doubt

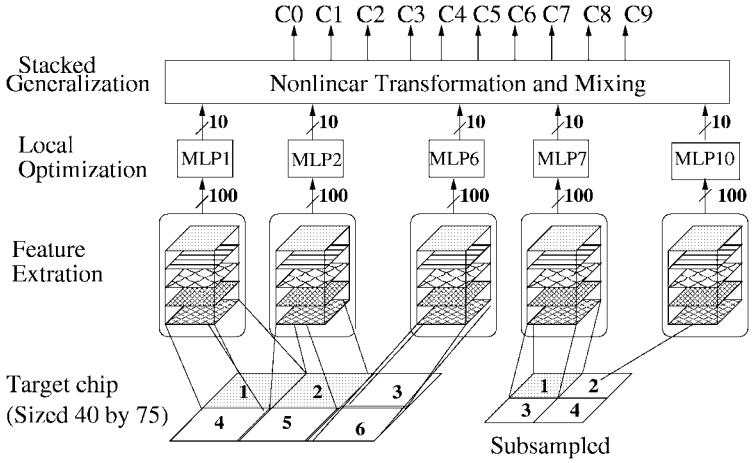

FIG. 7. The Modular Neural Network-based scheme evaluated in this paper.

further experimentation could lead to a superior partition. In the committee of networks, each member network receives distinct inputs, which are features extracted from one local region of the target image (in this study, the features are directional variance in 5×5 image blocks), and features are extracted at different resolutions. That is, the input of a member network in the committee of networks is a *subvector* (the input data vector is partitioned). This reduces the dimensionality of the data, allowing each neural net to have fewer weights. The technique requires that the targets be reasonably close to the center of the input image because the networks are not shift-invariant. It was determined experimentally that the centering error of the detection algorithm used in the acquisition of the database was small enough that it did not cause significant error in the recognition results. The classification decisions of the individual members of a committee of networks are combined using *stacked generalization* [14], which uses an additional neural network whose input is the output of the low-level neural nets. The classifier achieves good performance with much lower computational complexity than the LVQ-based scheme. The probabilities of correct classification for the training and testing sets are 95.49 and 75.58%, respectively. A heuristic comparison of the approaches of LVQ and MNN is given in Table 1.

Hausdorff metric-based approach. The following two methods were used to build models from the training set. Method 1: for each target type, we first compute one mean image for each orientation by averaging all the images with the same aspect angle, then extract edge features on the mean image (the averaging process makes sense since the target chips are all prealigned). Subsequently, the edge data are first masked by the corresponding silhouettes and then stored as one model for this target type. Therefore we have 72 models for each target type. Method 2: for every image in the training set, we extract the edge features, mask them with corresponding silhouette, then store the edge data as one model. Hence for each target type, we have more than 1000 models. Experiments have shown that the second method is more reliable, although this requires that we keep a large model base, hence slowing down the matching process.

Besides the basic Hausdorff metric, some generalized versions were also implemented. We recall from Eq. (1) that the largest distance is chosen, which makes Hausdorff distance very sensitive to noise: a single noise point could drag the distance too far from that desired. One alternative method is to choose the kth largest distance as the Hausdorff distance. Theoretically, doing so will improve robustness. Experiments on the database show that when k varies from 1 to 6, there is little change in the recognition rate, but when k goes beyond 7, the result degrades quickly. Another method is to use a weighted sum of the top k largest distances as the Hausdorff distance, which actually improves the result by about 1%. Also, in addition to edge positions, edge orientations were also used for matching, which improved the result by 3%, but made the matching more complex.

Canny's algorithm [17] was used for edge detection. Naturally this brings up the problem of how to choose good parameters for the edge detector, raising the question that the comparison might not be between the recognition algorithms themselves but instead between the edge detectors, which are only preprocessors to the recognition task. As long as feature detection algorithms are used, we eventually must choose some parameters. To alleviate the problem, we choose a set of proper parameters and the best recognition result is picked (like that in Fig. 8) for comparison with results from other approaches. The same argument is true for geometric hashing.

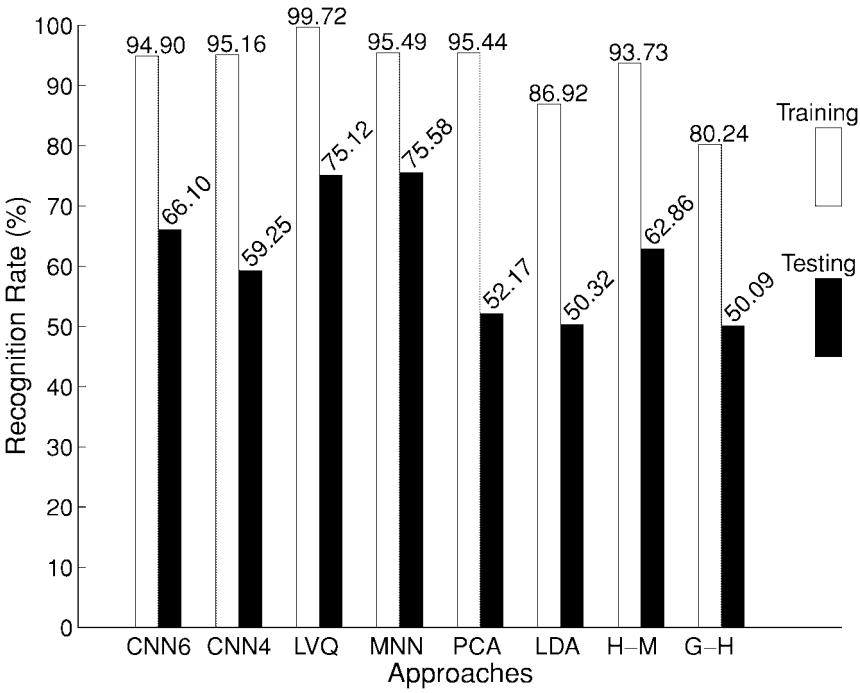

FIG. 8. Typical results obtained using different techniques. The gray bars are for the training set, and the black ones for the test set. CNN4 and CNN6 stand for four- and six-layered CNNs, respectively. H–M and G–H stand for Hausdorff metric-based and geometric hashing-based approaches, respectively.

Geometric hashing-based approach. To implement the geometric hashing approach, first we need to represent an object by a collection of feature points, which are then matched to similarly constructed models. Therefore the hash point model is essential to geometric hashing-based recognition. In the implementation, the image chip containing the detected potential target is processed to extract key point attributes. First edges are detected, and then lines are extracted. This line representation of the target is then processed to extract line endpoints, points of large curvature and line intersections. These extracted features, the hash points, are the basis for the geometric hashing algorithms. Note that this is different from the Hausdorff metric-based approach where whole edge pieces are used for matching. One appealing property of geometric hashing is that it can be efficiently implemented with parallel processing so that an unknown target chip can be simultaneously tested against tens or even thousands of targets, as well as specific orientations of each target. In the geometric hashing technique, feature points are obtained after line and point extraction. Usually, a line detector is constructed on the top of edge detection. Again we have a dilemma: we do not have a perfect line and point extractor, especially for FLIR imagery. Sensor fusion techniques may be helpful, as reported by Akerman *et al.* [8]. Since geometric hashing uses point features only, it requires feature points with good discrimination. For the database used, there are too many target types, each with abundant orientations, making geometric hashing less applicable. For example, it is possible that, when expressed in terms of point features, one target at a certain orientation can be quite similar to another target. However, one can expect that for fixed orientations (or with fewer target types), geometric hashing will perform better. If we can somehow first reduce the hypotheses to a small number of candidates, and then use the geometric hashing technique, we may achieve better results.

3.3. Summary of the Results

Figure 8 summarizes the results of the different approaches. As pointed out above, the recognition rate for each approach is the best one chosen from the output of a set of experiments with different parameters. For example, in the Hausdorff metric-based technique, different sets of thresholds and scales were tested for the edge detector. As shown in the figure, for the training set, most techniques perform well, but for the test set, none of them is satisfactory. The best recognition rate for the test set (75.58%) was given by the MNN-based approach. More analyses and discussion of the results are presented in the next subsection.

3.4. Analyses and Discussion

As presented in the previous section, although we have a very large database to learn from, the best results are still under 80%, which is not desirable for a real application. We now attempt to analyze the reasons of the less-than-desirable performance, mostly from experimental point of view. The discussion will be based on the experiments presented above and additional experiments designed in the following. For clarity of presentation, and to avoid loss of focus, we will mostly focus on the experiments based on the CNN-based approach (with four-layered CNNs). However, similar results were observed irrespective of the approaches; hence the conclusions are typical of the database, and not only for any specific approach.

Generalization of learning algorithm. CNN, PCA, LDA, LVQ, and MNN are learning algorithms. Although the Hausdorff metric-based approach and the geometric hashing technique are based on model matching, in the experiments reported in this paper, the models were constructed from the training data set; therefore they are also, in a sense, learning algorithms. For any learning algorithm, a critical issue is to obtain a representative training set. If the training set is underrepresentative, bad generalization is unavoidable. Of course, even with a good training set, problems could arise from the algorithms themselves. Following experiments will show, however, for this database, the training set and test set are mutually underrepresentative with respect to each other. In the first experiment, we train a four-layered CNN with the test set, and then test with the training set (with only those targets available in the test set). Figure 9 shows the learning curves of the experiment. Note that although the algorithm seems to be doing better (with 76.8% for testing compared with 59.25% in Fig. 8), the problem is now simplified—the algorithm is handling practically only five classes. Thus one cannot say that the test set is good (representative) with respect to the training set. When we only use five classes in the training set to train the network and test on the test set, we get similar results for the test set.

In the second experiments, we trained a CNN with 10,000 images from the training set and tested it with the remaining 3,862 images, which gave a recognition rate of more than 80%. This is indeed a big jump from 59.25%.

Similar experiments such as training and testing with disjoint subsets of the test set confirm the above claim about the mutual representativeness of the data sets. This observation is not surprising considering the different image acquisition conditions mentioned in Section 3.1.

It is unquestionable that a highly representative training set is essential for any learning algorithm. However, for FLIR imagery, it is expensive to gather such a training set by taking images of real targets in real scenes. For example, one must consider the variations due to ambient temperature, vehicle operation history, background, range, etc. One solution

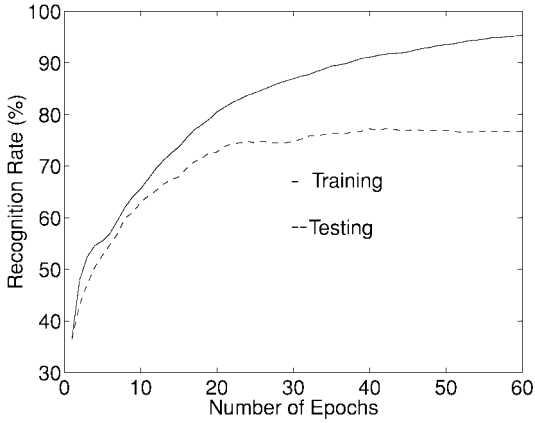

FIG. 9. The learning curves of a CNN, when trained with the test set, and tested with the training set (only with those five target types available in the test set).

could be what some researchers are doing now: simulation of infrared scenes [18, 19], but trying to address all possible variations in infrared imagery (in fact, in any imagery) by simulation is an extremely difficult task. Other alternatives include using video data. Video may provide more information through multiframe integration and, if the target is moving, through motion analysis, which is beyond the scope of this discussion.

The problem of pose. In the database, each target has abundant poses even though the orientation only changes horizontally. For most targets, since their length is usually larger than their width, the targets will have much more pixels in a side view than in a head view (or a tail view). It was conjectured that maybe some views suffer from misclassification worse than that of others. Our experiments do not give a strong indication about this, however. In Fig. 10, we plot the distribution of the number of misclassified targets with respect to pose angle for each target type. In this figure, the horizontal axis is pose angle (recall from Section 3.1, the test set only has coarsely labeled orientations), and the vertical axis is the number of misclassified target chips. The results are again from a four-layered CNN. From these plots, it is difficult to draw any universal conclusion about the relationship of pose and misclassification rate.

To take the pose problem one step further, we also designed a hierarchical pose estimator, which is intended to show if we can reliably estimate the pose of a target from the test set, given that we know the pose for each target in the training set. Pose information could be utilized in methods such as PCA and LDA to exclude background. Also, if we know the possible orientation of a target, we can validate the target type by using some features unique to the target in that orientation. A hierarchical system that combines CNN and PCA or LDA together to do pose estimation was constructed. A block diagram of the approach is shown in Fig. 11. This approach takes the top k outputs of the CNN as possible candidates, then uses PCA or LDA to get an estimated pose for each candidate. Two methods of estimating the pose have been studied:

Method 1. For each target type, use the corresponding training data to construct a single eigenspace. Pose estimation is then carried out in that eigenspace.

Method 2. For each target type, divide all the training images into eight groups according to their orientation (for example, those with aspect angle falling between -22.5° and 22.5°

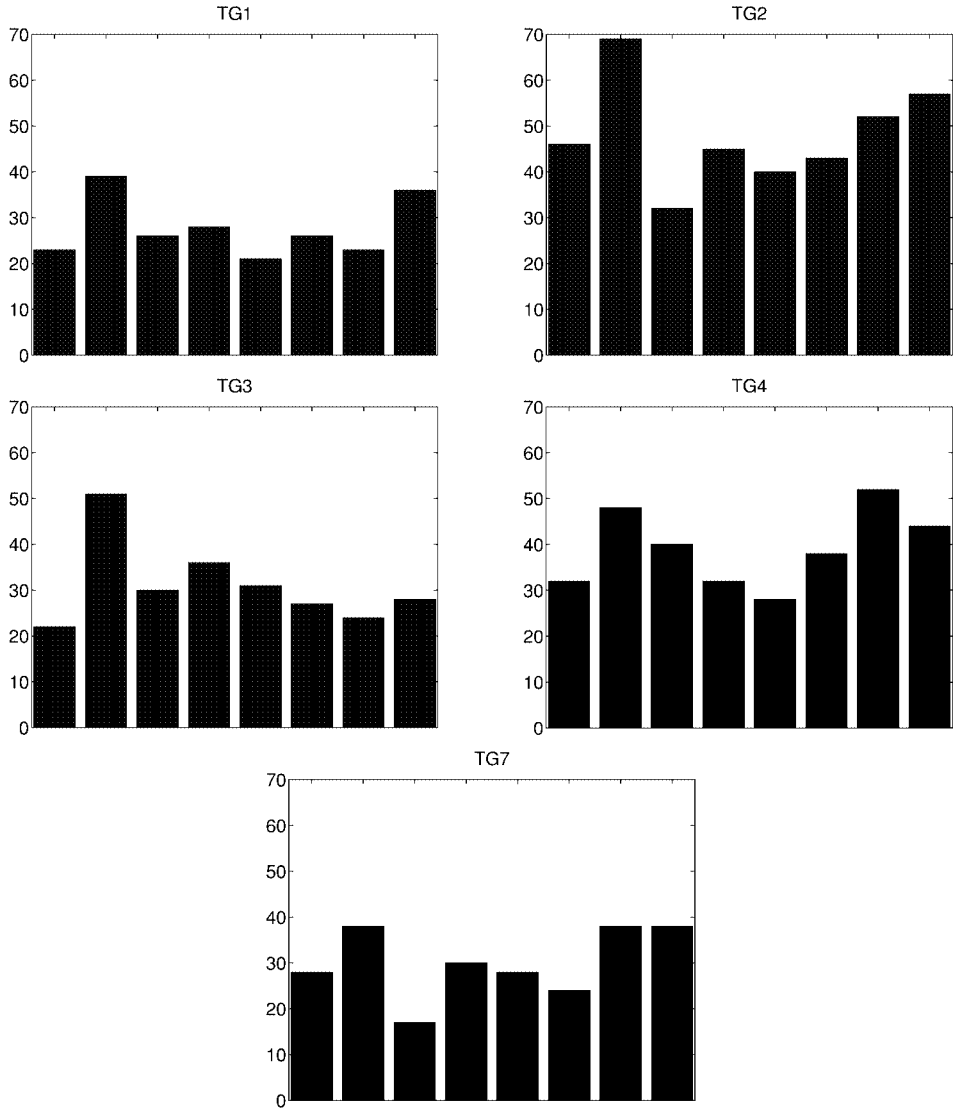

FIG. 10. The distribution of the number of incorrectly classified targets with respect to the target poses. The horizontal axis is pose angle ($0^\circ, 45^\circ, \ldots, 315^\circ$; see description of the database), and the vertical axis is the number of targets that are misclassified. Each figure is for one target type, as labeled.

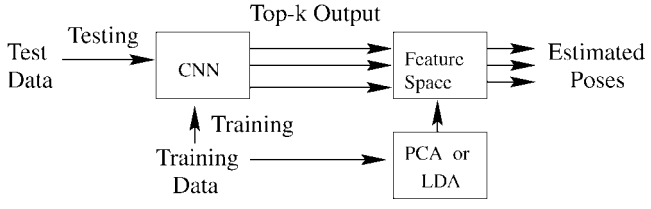

FIG. 11. A hierarchical pose estimator.

are classified as group 1, between 22.5° and 67.5° as group 2, and so on), then use the LDA method to do pose estimation.

Since we have a hypothesis as to the target type, we can use the corresponding silhouette to mask the image chip so that the background is excluded before applying PCA or LDA. Specifically, in method 1, we use the union of the silhouettes of all orientations of a certain target type to mask the image chip (in practice, the union is expanded a little before masking to allow a certain degree of translation of the target). In method 2, the union is taken only among those orientations in the same group, thus maximally excluding the background. Also, in practice, when forming the groups, a small overlap should be used to avoid sharp jumps between groups. Another slightly different method is, for each target type, to do the same division as in method 2, and then construct one eigenspace for each of the eight groups. This, however, would need comparison of distances in eight eigenspaces to give a final pose estimate.

This approach works very well for the training set (as before, when testing with a training image by PCA or LDA, the best response should be always discarded): when a bias of up to 10° is allowed, it gives correct pose estimates for over 90% of the input images. However, for the test set, the pose estimator gives a rate of correctness of around 80% even if a 45° bias is allowed. This result appears to support the conclusion that pose alone is not the reason for the less-than-desirable performance of the algorithms. Otherwise, after we have treated targets at different orientations separately (as in Method 2 above), we should have obtained good pose estimation results.

Is the number of classes an issue? All the experiments presented so far have used all the targets at the same time in training. Is it because we mix so many targets (5 classes for the experiment of Fig. 9, and 10 classes for all the rest) together? Can we do better if we "decompose" the problem into two-class problems such as TG1 and non-TG1? To answer this, we use the whole training set to train a CNN for this two-class problem. Note that due to the imbalance in the numbers of TG*x* and non-TG*x* targets, we must randomly reuse TG*x* targets such that the training will not be biased toward non-TG*x* targets. In this situation, we can compute the recognition rate together with false alarms. Figure 12 shows typical results, using "TG1 and non-TG1" as the example. It is obvious from the figure that, even if the problem is reduced to a binary classification task, there is no significant change in the test set performance.

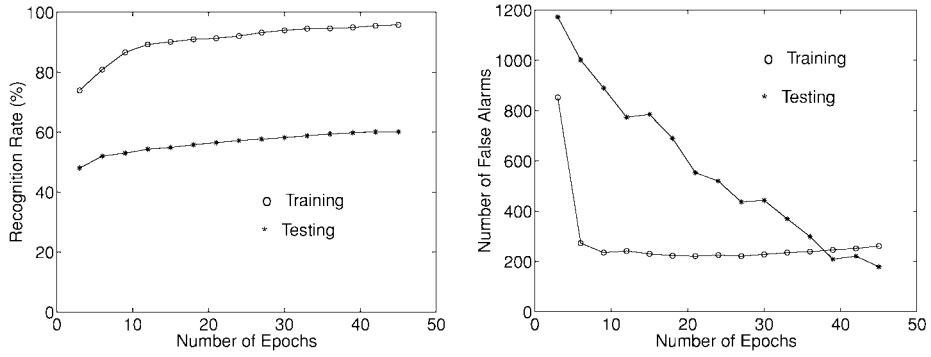

FIG. 12. Two-class problem: TG1 and non-TG1. Left: learning curves. Right: false alarms.

Another issue related to the number of classes is how one class is misclassified as other classes: is a target almost always misclassified as a particular target (or a particular set of targets) whenever a misclassification happens? Recall from our previous experiments, in many cases, we can choose the top k matches (or responses). We have found that with the four-layered CNN, we can obtain around 70% recognition rate for the test set if the top two responses are allowed (meaning that whenever the top two responses include the right target then it will be treated as correct recognition), and the recognition rate goes to 85% if the top three choices are allowed. This seems to suggest that there is some special pattern in misclassification. The so-called confusion matrix is a good reflection of this type of information. Given N_c classes, the confusion matrix is of size $N_c \times N_c$, whose diagonal entries are the number of targets that are correctly classified, while the off-diagonal entries are the number of misclassifications. The following is a typical confusion matrix obtained with the four-layered CNN when the top two choices are allowed. The matrix is organized in the order of TG1 $\cdots$ TG10. Since there are only five classes in the test set, five rows are all zeros, but we have kept them for clarity. Although one can find that, for example, TG7 is rarely classified as TG8 (only one case), the matrix is rather scattered in off-diagonal positions, meaning in general one target is likely misclassified as any other target:

$$\begin{pmatrix} 624 & 52 & 9 & 9 & 17 & 26 & 4 & 18 & 21 & 19 \\ 36 & 483 & 34 & 19 & 22 & 46 & 8 & 13 & 14 & 23 \\ 22 & 43 & 458 & 26 & 44 & 16 & 43 & 2 & 41 & 74 \\ 20 & 9 & 14 & 453 & 9 & 7 & 10 & 12 & 20 & 23 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 16 & 34 & 32 & 24 & 16 & 383 & 1 & 34 & 56 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The influence of background. Another important issue is the background. Even though we have a target chip, it always contains some background. Feeding the background into an algorithm may force the algorithm to figure out the differences between the backgrounds instead of the differences between the targets, or at least partially so. This becomes worse if we are using some approaches that totally depend on training without explicit feature extraction, such as CNN. In fact, this may also be true with other approaches such as PCA, which is indeed very sensitive to background. We have attempted to reduce the influence of the background by using silhouettes in some of the experiments. However, even if with accurate silhouettes alignment to help the exclusion of background, if the algorithm requires a fix-sized input (as CNN or PCA does), we still must feed into the algorithm some background region for those views where the target is smaller. To analyze the effect of the background, simulation may be easily done with synthetic data, but it is difficult for a real database. It remains to be future work to find an effective way for evaluating the influence of the background.

As we have mentioned, although we used CNN in the above analyses, similar results were observed when using other algorithms. All these experiments and analyses suggest that the less-than-desirable performance of the algorithms on the test set is mainly due to the mutual underrepresentativeness between the training and test sets, and cannot be easily attributed to

any other single source such as pose, the number of classes, and extreme similarity between any pair of specific target types. In fact, even though the database is reasonably large, it has not covered the variations in depression angle. Thus the conclusion about the mutual representativeness is then not to complain about the database itself. Instead, it suggests that, in general, it would be hard to expect these evaluated approaches to make a breakthrough on FLIR ATR, since a universally "good" training set is not yet available.

4. SUMMARY AND CONCLUSION

Several recently developed FLIR ATR approaches have been evaluated based on their performance on a common large database. We implemented several techniques including CNN, PCA, LDA, LVQ, MNN, the Hausdorff metric-based matching technique, and a geometric hashing method. Using a common database allows a comparison of performances of conceptually different techniques as well as conceptually similar approaches. For the database used, we found that neural network-based approaches (CNN, LVQ and MNN) gave better results than the PCA, LDA and model-based approaches. Also they are easy to use since explicit feature extraction is not required. Although not as good, the Hausdorff metric-based method also performed well. The PCA-based, LDA-based, and geometric hashing methods gave relatively low recognition rates for this database. We analyzed the possible reasons for the less-than-desirable performance of all the algorithms.

It is interesting to note that a human being can correctly recognize targets in many cases that baffle the approaches discussed above, even if he/she gets familiar with the targets only through learning using the same training set. This suggests that some human capability not yet understood is effective.

Note that this paper has focused on FLIR ATR from still imagery. There is a significant amount of work on performance evaluation of ATR using other cues (for example, see [20–21]). There are also efforts on fusing different cues, using multispectrum sensor, using video, etc., that are not covered by our evaluation. However, as long as a still FLIR image is the only cue, it seems that new and better approaches are still needed, which may probably have to be radically different from current algorithms.

ACKNOWLEDGMENTS

Comments from the anonymous reviewers are greatly appreciated, as they have considerably improved the presentation. The support of the Army Federated Laboratories/Lockheed Sanders under Contract DAAL-01-96-2-0001 is gratefully acknowledged. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

REFERENCES

1. Y. LeCun and Y. Bengio, Convolutional networks for images, speech, and time series, in *The Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.), pp. 255–258, MIT Press, Cambridge, MA, 1995.
2. A. Pentland, B. Moghaddam and T. Starner, View-based and modular eigenspaces for face recognition, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1994*, pp. 84–91.
3. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, New York, 1990.
4. L. A. Chan, N. M. Nasrabadi, and V. Mirelli, Multi-stage target recognition using modular vector quantizers and multilayer perceptrons, in *Proc. Computer Vision Pattern Recognition, 1996*, pp. 114–119.

5. L. C. Wang, S. Z. Der, and N. M. Nasrabadi, A committee of networks classifier with multi-resolution feature extraction for automatic target recognition, in *Proc. IEEE Int. Conf. Neural Networks, 1997*, Vol. 3, 1596–1600.
6. D. Doria and D. Huttenlocher, Progress on the fast adaptive target detection program, RSTA Technical Reports of the ARPA IU Program, 1996, pp. 589–594.
7. Y. Lamdan and H. Wolfson, Geometric hashing: A general and efficient model-based recognition scheme, in *Proc. 2nd Inter. Conf. Computer Vision, 1988*, pp. 238–249.
8. A. Akerman, R. Patton, W. Delashmit, and R. Hummel, Target identification using geometric hashing and FLIR/LADAR fusion, in RSTA Technical Reports of the ARPA IU Program, 1996, pp. 595–618.
9. K. Hornik, M. Stinchcombe, and H. White, Multi-layer feedforward networks are universal approximators, *Neural Networks* **2**, 1989, 359–366.
10. T. Kohonen, The self-organizing Map, *Proc. IEEE* **78**, 1990, 1464–1480.
11. L. C. Wang, S. Der, and N. Nasrabadi, Composite classifiers for automatic target recognition, *Opt. Eng*. **37**, 1998, 856–868.
12. R. Jacobs and M. Jordan, Learning piecewise control strategies in a modular neural networkarchitecture, *IEEE Trans. Systems Man Cybernet*. **23**, 1993, 337–345.
13. M. Perrone, General averaging results for convex optimization, in *Proc. Connectionist Models Summer School, 1993*, pp. 364–371.
14. D. H. Wolpert, Stacked generalization, *Neural Networks* **5**, 1992, 241–259.
15. H. Wolfson, Model-based object recognition by geometric hashing, in *Proc. European Conf. Computer Vision, 1990*, pp. 526–536.
16. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, 1994.
17. J. Canny, A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell*. **8**, 1986, 679–698.
18. M. Cooper, A. Lanterman, S. Joshi, and M. Miller, Representing the variation of thermodynamic state via principal components analysis, in *Proc. 3rd Workshop Conventional Weapon ATR*, 1996, pp. 479–490.
19. M. Cooper, U. Grenander, M. Miller, and A. Srivastava, Accommodating geometric and thermodynamic variability for forward-looking infrared sensors, in *Proc. SPIE, Algorithms for Synthetic Aperture Radar* (E. Zelnio, Ed.), Vol. 3070, pp. 162–172, 1997.
20. D. E. Dudgeon, ATR performance modeling and estimation, MIT Lincoln Labs Technical Report 1051, Lexington, MA, 1998.
21. M. Phillips and S. R. Sims, Signal-to-clutter measure for ATR performance comparison, *Proc. SPIE* **3069**, 1997, 74–81.
22. A. Mahalanobis, D. Carlson, B. Vijava Kumar, and S. R. Sims, Distance classifier correlation filter, *Proc. SPIE* **2238**, 1994, 2–13.