Synthetic Aperture Radar Automatic Target Recognition Using Adaptive Boosting

Yijun Sun, Zhipeng Liu, Sinisa Todorovic, and Jian Li

Dept. of Electrical and Computer Engineering
University of Florida, Gainesville, FL, USA

ABSTRACT

We propose a novel automatic target recognition (ATR) system for classification of three types of ground vehicles in the MSTAR public release database. First, each image chip is pre-processed by extracting fine and raw feature sets, where raw features compensate for the target pose estimation error that corrupts fine image features. Then, the chips are classified by using the adaptive boosting (AdaBoost) algorithm with the radial basis function (RBF) net as the base learner. Since the RBF net is a binary classifier, we decompose our multiclass problem into a set of binary ones through the error-correcting output codes (ECOC) method, specifying a dictionary of code words for the set of three possible classes. AdaBoost combines the classification results of the RBF net for each binary problem into a code word, which is then "decoded" as one of the code words (i.e., ground-vehicle classes) in the specified dictionary. Along with classification, within the AdaBoost framework, we also conduct efficient fusion of the fine and raw image-feature vectors. The results of large-scale experiments demonstrate that our ATR scheme outperforms the state-of-the-art systems reported in the literature.

Keywords: MSTAR, automatic target recognition, adaptive boosting, data fusion.

1. INTRODUCTION

Radar systems are important sensors due to their all weather, day/night, long standoff capability. Along with the development of radar technologies, as well as with increasing demands for target identification in radar applications, automatic target recognition (ATR) using synthetic aperture radar (SAR) has become an active research area. The latest theoretical developments in classification methodologies quickly find applications in the SAR ATR design. Joining these efforts, in this paper, we present a new ATR scheme based on the adaptive boosting (AdaBoost) algorithm,[1] which yields the best classification results reported to date in the literature on the MSTAR public release database.

The MSTAR data is a standard dataset in the SAR ATR community, allowing researchers to fairly test and compare their ATR algorithms. The literature abounds with reports on systems where the MSTAR database is used for validation. Among these systems, one most commonly used approach is the template based method,[2] ,[3] .[4] In this method, for each class, a set of filters (templates) is generated to represent target signatures around different aspect angles; then, a given target is identified as the class whose filters give the "best" output. The template based method, however, is computationally inefficient. Here, high classification accuracy conditions the creating of large sets of filters for each class, which results in extensive computation both in the training and testing stages. In addition, large memory-storage requirements, inherent to the method, are a major hindrance for its implementation in systems subject to stringent real-time constraints.

A more appealing approach is to design a classifier, which can be represented by a set of parameters. For example, in,[5] the authors use the Neural Network for their ATR system. Its parameters can be easily estimated through optimization of a specified cost function, as for example in the back-propagation algorithm. However, the Neural Network is prone to overfitting. Moreover, after training of the Neural Network, its parameter estimates are not guaranteed to be a global-minimum solution of a given optimization function. Recently developed large-margin based algorithms have been reported to successfully alleviate the overfitting problem, while at the same

Please address all correspondence to: Dr. Jian Li, Department of Electrical and Computer Engineering, P. O. Box 116130, University of Florida, Gainesville, FL 32611, USA. Phone: (352) 392-2642. Fax: (352) 392-0044. E-mail: li@dsp.ufl.edu.

time maintaining the classification efficiency. Two typical examples of the margin-based classifiers are the support vector machine (SVM)[6] and AdaBoost.[1] Contrary to the anticipated superior classification performance, SVM-based ATR systems (e.g.,,[7] [8]) perform only slightly better than the template based approaches. Concerning AdaBoost, the literature reports success of many AdaBoost-based systems for pattern classification (e.g.,[9]). Nevertheless, to our knowledge, it has not yet gained appropriate attention in the SAR ATR community.

In this paper, we propose a novel classification scheme for the MSTAR data using the AdaBoost algorithm. AdaBoost is originally designed for a binary problem. To categorize multiple types of vehicles in the MSTAR database, we first employ the error-correcting output codes (ECOC) method[10] to decompose the multiclass problem into a set of binary sub-problems. We refer to this decomposition as coding scheme, where each class is labeled with a code word. Then, for each binary problem, we build a binary classifier by using AdaBoost with the *radial basis function* (RBF) networks[11] as the base learner. When presented an unseen sample, each binary classifier produces a bit in a code word, which is then classified measuring the distance of that code word from the code words in the dictionary specified by the coding scheme. The experimental results, reported herein, show that the outlined classification system, without any preprocessing of the MSTAR data, achieves comparable performance to that of systems utilizing sophisticated pose estimators. This means that our SAR ATR system can be implemented for the applications with stringent real-time constraints, where estimation of target poses is computationally prohibitive.

The performance of the outlined system can be further improved by accounting for the randomly distributed poses of SAR image chips. We propose a pose estimation algorithm, whereby all targets are rotated to the same aspect angle. Although our pose estimation algorithm is quite accurate, it may fail when a target is severely shadowed and/or distorted. To compensate for the occasional pose estimation error, we extract two sets of image features to represent the MSTAR image chips. The first set, called *raw* features, represent pixel values of the original image chips that are not pre-processed. The second set of features, called *fine* features, comprises magnitudes of two-dimensional DFT coefficients computed on the aligned image chips, which are rotated to the same aspect angle by the pose estimator. Here, it is necessary to design an efficient data fusion method for combining the two highly correlated feature sets. Again, AdaBoost lends itself as a natural solution. Essentially, we utilize AdaBoost to choose the "better" feature set in each training iteration, such that the final ensemble classifier is optimized over the two feature sets. The resulting classification system, which incorporates the information from the pose estimator, can correctly classify 1360 out of 1365 testing samples, which, to our knowledge, is the best result ever reported in the literature.

2. DATA PREPROCESSING

2.1. Pose Estimation

Targets in the MSTAR data have randomly distributed poses, ranging from 0° to 360°. Eliminating variations in target poses can significantly reduce the classification error. However, accurate estimation of poses is a challenging task. Fortunately, available *a priori* knowledge can be exploited to mitigate the difficulty of the pose estimation problem. Herewith, we assume that the MSTAR data is rectangular shaped, such that the target pose can be determined by the longer edge, with the uncertainty of 180°.

Unlike optical images, targets in SAR images do not have clear edges. To extract edges, it is first necessary to segment targets from noisy background, and then detect edges of the segmented targets. In Figure 1, we present a block-diagram of our edge detection procedure. Since the radar reflection varies from chip to chip, we first equalize the intensity of the image to be in the interval from 0 to 1, by using a histogram equalization operator. Then, to the equalized image chip, we apply a mean filter, which reduces the image noise level. The "smoothed" image is then roughly segmented with a predefined threshold θ_1. As output, we obtain a roughly segmented target with pixel values larger than θ_1. Next, the median of this target is calculated. The median, denoted as θ_2, is used as a threshold for the subsequent fine-target segmentation. After the fine-target segmentation, target edges are detected by using the standard Sobel filter, and then dilated with a $d_2 \times d_2$ box. In Figure 2, we depict an example of the outlined edge-detection procedure for a T72 target. In our experiments, we set the parameters to the following values: $\theta_1 = 0.8$ and $d_2 = 2$.

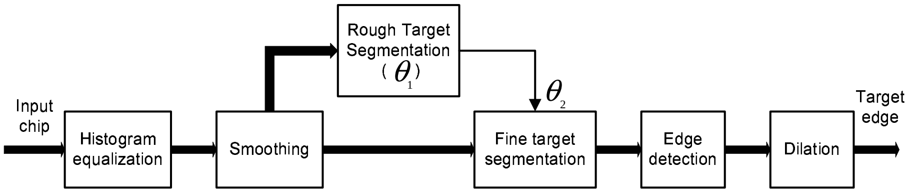


Figure 1. Flow chart of the edge detection.

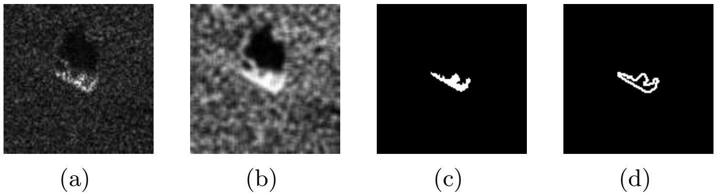


Figure 2. An example of the intermediate results in the edge-detection procedure: (a) original chip, (b) "smoothed" image, (c) fine target, and (d) target contour.

The above edge-detection algorithm poorly extracts straight edges for majority of the chips. To alleviate this problem, we perform an exhaustive target-pose search over different pose angles. Thus, for each pose angle α, we first draw a rectangle in α direction, such that the sides of the rectangle are tangent to the detected target contour. Here, α is the angle between the longer side of the rectangle and the horizontal image axis. Then, we dilate the sides of the rectangle to be d_2 pixels wide. Next, along the two longer sides of the rectangle, we compute the number of pixels that belong to both the target contour and the rectangle sides, and find the maximum of the two numbers. The maximum number of overlapped pixels is recorded as the edge weight in α direction, $w(\alpha)$. This procedure is iterated, rotating the tangent rectangle with a rotation step $\Delta\alpha$, until the entire interval $[0°, 180°]$ is covered. Since we assume that targets have a rectangular shape, there is no need to conduct search for angles greater than 180°. The angle α^*, characterized by the largest edge weight, $\alpha^* = \arg\max_\alpha w(\alpha)$, is selected as the target-pose estimate.

In our experiments, we set $\Delta\alpha = 5°$. To compute the pose estimation error we determine the ground truth by human inspection. For the majority of the chips, the pose estimation error is within $\pm 5°$. However, the method fails in some cases, where the longer target edge is shadowed or significantly distorted and the angle of the shorter target edge is estimated as α^*.

To identify potential outliers, we test each estimate against the following two criteria: (1) α^* is approximately 180°, and (2) $w(\alpha^*)$ is smaller than a certain threshold. If neither is true then the pose estimate is equal to α^*. Otherwise, if the outcome of either of these two tests is positive, we check yet another criterion: if the target length, measured along α^* direction, is smaller than the target width, measured along the direction perpendicular to α^*. If the last test is negative then we consider the pose estimate equal to α^*. If the last test is positive then the pose estimate is equal to the angle perpendicular to α^*.

To validate our approach, we calculate the pose estimation error with respect to manually estimated target poses. For three training data sets: T72 sn-132, BMP2 sn-c21, and BTR70 sn-c71, at 17° depression angle, the mean of the error is $-0.32°$ and the standard deviation of the error is $6.87°$. For seven test data sets: T72 sn-132/s7/812, BMP2 sn-c21/9566/9563, and BTR70 sn-c71, at 15° depression angle, the mean of the error is $0.07°$ and the standard deviation of the error is $7.82°$. Note that the standard deviation of the error is slightly greater than our search step, $\Delta\alpha$=5°, suggesting that our method is quite reliable.

2.2. Feature Extraction

In our approach, the MSTAR image chips are represented by two types of image features that we refer to as *raw* and *fine* features. The reason for extracting two feature sets is to reduce the occasional pose estimation error,

```
AdaBoost
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, maximum iteration number $T$, $d^{(1)}(n) = 1/N$
for $t = 1 : T$
    1. Train base learner with respect to distribution $\mathbf{d}^{(t)}$ and get hypothesis $h_t(\mathbf{x}) : \mathbf{x} \to [-1, +1]$.
    2. Calculate the edge $r_t$ of $h_t : r_t = \sum_{n=1}^N d^{(t)}(n) y_n h_t(\mathbf{x}_n)$,
    3. Compute the combination coefficient: $\alpha_t = \frac{1}{2} \ln \left( \frac{1+r_t}{1-r_t} \right)$,
    4. Update weights: $d^{(t+1)}(n) = d^{(t)}(n) \exp(-\alpha_t y_n h_t(\mathbf{x_n}))/Z_t$ where $Z_t$ is the normalization constant
       such that $\sum_{n=1}^N d^{(t+1)}(n) = 1$.
end
Output: $F(x) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$ .
```

Figure 3. The pseudo-code of the confidence-rated AdaBoost.

as we discuss below.

For extracting fine features from a given image, we use the estimated fine target pose to rotate the target to the referent aspect angle of 180°. Then, we crop the image by selecting an 80×80 window in the center of the image. Next, we compute the two-dimensional DFT for the cropped image. Finally, the magnitudes of the 2-D DFT coefficients are used as fine features. Note that by using the two-dimensional DFT and taking the magnitudes for extracting fine features, we alleviate the problem of target-center variations. In addition, the inherent 180° uncertainty of our pose estimation algorithm is also eliminated.

Relying only on the fine-feature set may cause poor classification performance, due to the limited accuracy of our pose estimation algorithm. To compensate for the pose estimation error in fine features, we also extract raw features. The raw features of a given image are equal to pixel values in an 80×80 window, positioned at the center of the image.

3. CLASSIFIER DESIGN

3.1. AdaBoost

AdaBoost[1] is considered as one of the most important recent developments in the classification methodology. It has been used in many applications with great success. Given a set of hypothesis functions $\mathcal{H} = \{h(\mathbf{x}) : \mathbf{x} \to \mathcal{R}\}$, called *base learners*, and a set of training samples $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \in \mathcal{X} \times \pm 1$, AdaBoost finds an ensemble function $F(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$ to optimize the following cost function:

$$C = \frac{1}{N} \sum_{n=1}^{N} \exp(-y_n F(\mathbf{x}_n)). \tag{1}$$

The original AdaBoost algorithm[1] uses binary-valued hypotheses, i.e., $h(\mathbf{x}) : \mathbf{x} \to \{\pm 1\}$, as the base learners. Schapire and Single[12] extended the original AdaBoost to a more general case, where real-valued hypotheses are used. Here, $h(\mathbf{x}) : \mathbf{x} \to [-1, +1]$, where the sign of $h(\mathbf{x})$ represents the class label assigned to the instance $\mathbf{x}$, and the magnitude $|h(\mathbf{x})|$ represents the prediction "confidence". The pseudo-code of the confidence-rated AdaBoost is presented in Figure 3.

The main idea of AdaBoost is to repeatedly apply the base learning algorithm to the re-sampled versions of the training data, to produce a collection of hypothesis functions, which are ultimately combined via a weighted linear vote to form the final decision. An intuitive idea in AdaBoost is that the examples, which are misclassified, get larger weights in the following iterations; hence, in the subsequent training steps, the base learner is forced to focus on these hard-to-classify cases, which, for instance, are close to the decision boundary.

Initially, the probability of selecting each sample is set to be uniform, that is, $d^{(1)}(n) = 1/N$, $n = 1, \dots, N$. In each iteration, t, the base learner is trained with respect to the distribution $\mathbf{d}^{(t)}=[d^{(t)}(1), \dots, d^{(t)}(N)]$. Then, the performance of the base learner, measured through the edge, r_t, conditions the value of the combination coefficient, α_t. Note that if r_t is equal to zero, there is no information that h_t contributes to the ensemble classifier; therefore, α_t is set to zero. Usually, it is assumed that $\mathcal{H}$ is negative close; that is, if $h \in \mathcal{H}$ then $-h \in \mathcal{H}$. Given this assumption, we can further assume that the combination coefficients $\boldsymbol{\alpha}$ are non-negative without loss of generality. If α_t happens to be negative, $-h$ can be used instead of h, in order to change the sign of negative r_t and α_t. After computing α_t, the distribution $\mathbf{d}^{(t)}$ is updated. Note that if the sign of $h_t(\mathbf{x}_n)$ agrees with y_n, $d^{(t)}(n)$ decreases, otherwise $d^{(t)}(n)$ increases.

One of the most important properties of AdaBoost is that, under a mild assumption that weak learners can achieve the error rate less than random guessing (i.e., < 0.5), AdaBoost exponentially reduces the training error to zero as the number of the combined base hypotheses increases.[12] Moreover, along with minimizing the cost function C, AdaBoost effectively maximizes the margin of the resulting ensemble classifier. Consequently, the ensemble classifier is characterized by small generalization error. In many cases, the generalization error continues to decrease, with each iteration step, even after the training error reaches zero.[13]

3.2. Base Learner: RBF Network

As the base learner in AdaBoost, we use the RBF network.[11] The RBF net is a multidimensional nonlinear mapping based on the distances between the input vector and predefined *center vectors*. Using the same notation as in Section 3.1, the mapping is specified as a weighted combination of J basis functions: $h(\mathbf{x}) = \sum_{j=1}^{J} \pi_j \phi_j(\|\mathbf{x}-\mathbf{c}_j\|_p)$ where $\phi_j(\|\mathbf{x}-\mathbf{c}_j\|_p)$ is a radial basis function (RBF), π_j is a weight parameter, and J is a pre-defined number of RBF centers. Basis functions $\phi_j(\cdot)$ are arbitrary nonlinear functions, $\|\cdot\|_p$ denotes the p-norm (usually assumed Euclidean), and vectors $\mathbf{c}_j$ represent RBF centers. In the literature, one of the most popular RBF nets is the Gaussian RBF network.[14] Here, the basis functions are specified as the un-normalized form of the Gaussian density function given by $g(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance matrix. For simplicity, $\boldsymbol{\Sigma}$ is often assumed to have the form $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Hence, the Gaussian RBF network is given by

$$h(\mathbf{x}) = \sum_{j=1}^{J} \pi_j g_j(\mathbf{x}) = \sum_{j=1}^{J} \pi_j \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_j\|_2^2}{2\sigma_j^2}\right),$$

where the $\boldsymbol{\mu}$'s represent center vectors, while the σ's can be interpreted as the width of basis functions.

The parameters of the Gaussian RBF network – namely, the means $\{\boldsymbol{\mu}_j\}$, the variances $\{\sigma_j^2\}$, and the weighting parameters $\{\pi_j\}$ – are learned on training samples. Herewith, we employ an iterative learning algorithm, where all the RBF parameters are simultaneously computed by minimizing the following error function[15]:

$$E = \frac{1}{2}\sum_{n=1}^{N}(y_n - h(\mathbf{x}_n))^2 + \frac{\lambda}{2N}\sum_{j=1}^{J}\pi_j^2 \ , \tag{2}$$

where λ is a regularization constant. In the first step, the means $\{\boldsymbol{\mu}_j\}$ are initialized by the standard K-means clustering algorithm, while the variances $\{\sigma_j\}$ are determined as the distance between $\boldsymbol{\mu}_j$ and the closest $\boldsymbol{\mu}_i$, $(i \neq j,\ i \in [1, J])$. Then, in the following iteration steps, a gradient descent of the error function in (2) is performed to update $\{\boldsymbol{\mu}_j\}$, $\{\sigma_j^2\}$, and $\{\pi_j\}$. In this manner, the network fine-tunes itself to training data.

3.3. Multiclass Classification

So far, we have presented the ensemble classifier capable of solving binary-class problems. Since our goal is to identify three types of targets in the MSTAR dataset, we are faced with the multiclass problem. Here, a label from a finite label space $\mathcal{Y} = \{1, \cdots, K\}$, $K > 2$, is assigned to each pattern. To design such a classifier by using a binary classifier, we first decompose the multiclass problem into several binary problems. For this purpose, one popular approach is to implement the error-correcting output codes (ECOC) method, where each class is represented by a codeword in a suitably specified *code matrix*.[10] Here, each label $y \in \mathcal{Y}$ is associated with a row of a pre-defined code matrix, $\mathbf{M} \in \{-1, +1\}^{K \times L}$. Below, we denote the k-th row of $\mathbf{M}$ as $\mathbf{M}_{k\cdot}$, then, the l-th column as $\mathbf{M}_{\cdot l}$, and finally the (k, l)-th entry of $\mathbf{M}$ as M_{kl}. Note that the l-th column of $\mathbf{M}$ represents a binary partition over the set of classes, labeled according to $\mathbf{M}_{\cdot l}$. This partition enables us to apply a binary base learner to multiclass data. For each binary partition (i.e., column of $\mathbf{M}$), the binary base learner produces a binary hypothesis f_l. It follows that the K-class problem can be solved by combining L binary hypotheses $f_l, l = 1, \ldots, L$. After learning L binary hypotheses, the output code of a given unseen instance $\mathbf{x}$ is predicted as $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_L(\mathbf{x})]$. This output code is "decoded" as label y, if the y-th row of the code matrix, $\mathbf{M}_{y\cdot}$, is "closest" to $\mathbf{f}(\mathbf{x})$ with respect to a specified distance metric $d(\mathbf{M}_{y\cdot}, \mathbf{f}(\mathbf{x}))$.

A more general framework for decomposing the multiclass problem can be formulated by using a code matrix from the following set of matrices: $\{-1, 0, +1\}^{K \times L}$.[16] Here, a zero entry, $M_{kl}{=}0$, indicates that the classification of hypothesis f_l is irrelevant for label k. Consequently, f_l is learned only for those samples $(\mathbf{x}, M_{yl})$ where $M_{yl}{\neq}0$. Such definition of the code matrix provides for a unifying formulation of a wide range of methods for decomposing the multiclass problem, including the well-known *one-against-others* (OAO), and *all-pair* (AP) approaches. For the OAO scheme, $\mathbf{M}$ is a $K \times K$ matrix with all diagonal elements equal to $+1$ and all other elements equal to -1. For the AP scheme, $\mathbf{M}$ is a $K \times \binom{K}{2}$ matrix, where the number of columns is equal to the number of distinct label pairs (k_1, k_2). In the column l, which is assigned to the label pair (k_1, k_2), only the elements in rows k_1 and k_2 are nonzero. The code matrices of the OAO and AP schemes are given in $\mathbf{M}_o$ and $\mathbf{M}_a$, respectively:

$$\mathbf{M}_o = \begin{pmatrix} +1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \end{pmatrix} , \quad \mathbf{M}_a = \begin{pmatrix} +1 & 0 & -1 \\ -1 & +1 & 0 \\ 0 & -1 & +1 \end{pmatrix} . \tag{3}$$

In our multiple-target classification experiments, we use both coding approaches.

Recall that for the ultimate classification it is necessary to specify the distance metric $d(\mathbf{M}_{y\cdot}, \mathbf{f}(\mathbf{x}))$. One possible approach is to use the Hamming distance where a sample $\mathbf{x}$ is classified as label $\hat{y} = \arg\max_y \sum_{l=1}^{L} \mathrm{sign}(M_{yl} f_l(\mathbf{x}))$ where $\mathrm{sign}(z)$ is 1 if $z > 0$, -1 if $z < 0$ and 0 otherwise. A drawback of using the Hamming distance for decoding is that it does not make use of the confidence level indicated by the magnitude of $f_l(\mathbf{x})$. This problem can be alleviated by employing the Euclidean distance in conjunction with the following decoding rule: $\hat{y} = \arg\min_y \|\mathbf{M}_{y\cdot} - \mathbf{f}(\mathbf{x})\|_2$. If codewords are symmetrically spaced in the L-dimensional space, as is the case in the OAO and AP methods, the decoding rule can be simplified as $\hat{y} = \arg\max_y \sum_{l=1}^{L} M_{yl} f_l(\mathbf{x})$. We refer to this decoding rule as *max-correlation decoding*.

3.4. AdaBoost as a Fusion Method

Recall that in our approach each MSTAR image chip is represented by two sets of image features – namely, fine and raw features. In Section 2.2, we discuss that the reason for extracting the raw features is to compensate for the target pose estimation error in the fine features. The extraction of two different feature sets gives rise to the feature fusion problem. The simplest solution to combining the features is to stack the two feature vectors into a high-dimensional vector, which leads to a significant increase in classifier complexity and computational costs in both training and testing stages. In our case, where the information contained in the two feature sets is highly correlated, this simple method is not justified. We propose to use AdaBoost as a feature fusion method. Essentially, the idea is to choose the "better" set in each AdaBoost training iteration, such that the final ensemble function is optimized over the two feature sets. To this end, we use a fundamental theoretical result, presented in,[12] that the upper bound of the training error in Equation (1) is equal to the product of the normalizing constants $\{Z_t\}$, defined in Fig. 3:

$$\frac{1}{N}\sum_{n=1}^{N} \mathbf{I}\{y_n \neq \mathrm{sign}(F(\mathbf{x}_n))\} \leqslant \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n F(\mathbf{x}_n)) = \prod_{t=1}^{T} Z_t \; . \tag{4}$$

The pseudo-code of the fusion algorithm is shown in Figure 4. There are two separate branches of learning processes: Steps 1a-3a and Steps 1b-3b in Figure 4. In one branch the ensemble classifier is iteratively trained on

```
Data fusion using AdaBoost
Initialization: $\mathcal{D} = \{((\mathbf{x}_n^f, \mathbf{x}_n^r), y_n)\}_{n=1}^N$, $T$, $d^{(1)}(n) = 1/N$
for $t = 1 : T$
    1a. Train $h_t$ on $\{\mathbf{x}_n^f\}$ with respect to $\mathbf{d}^{(t)}$.      | 1b. Train $h_t$ on $\{\mathbf{x}_n^r\}$ with respect to $\mathbf{d}^{(t)}$.
    2a. Calculate $\alpha_t$ using the results from 1a.               | 2b. Calculate $\alpha_t$ using the results from 1b.
    3a. Calculate $Z_t$ using the results from 1a and 2a.           | 3b. Calculate $Z_t$ using the results from 1b and 2b.
    4. Choose the branch that gives smaller $Z_t$, and update $\mathbf{d}^{(t+1)}$, as in AdaBoost, using $h_t$, $\alpha_t$,
       and $Z_t$ from the selected branch.
end
Output: $F(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$.
```

Figure 4. The pseudo-code of AdaBoost as a fusion method.

fine feature vectors $\mathbf{x}_n^f$, and in the other, on raw feature vectors $\mathbf{x}_n^r$. More precisely, in each training step t, for both feature sets, the base learner h_t, combination coefficient α_t, and normalization constant Z_t are calculated with respect to the data distribution $\mathbf{d}^{(t)}$. Since the empirical training error is bounded by the product of $\{Z_t\}$ (Equation (4)), we choose Z_t as a criterion for selecting between the two feature sets. Thus, in iteration step t, the branch that produces smaller Z_t would yield a smaller upper bound of the empirical training error, and, as such, is selected. Then, for the following step $t+1$, we use h_t, α_t, and Z_t of the selected branch to update the data distribution $\mathbf{d}^{(t+1)}$. Note that it is necessary to keep track of the feature set used for computing each hypothesis h_t in the ensemble classifier $F(\mathbf{x})$. In experiments on test images, a given base hypothesis h_t should operate on the same feature set as that selected in the training process.

The outlined strict mathematical formulation of the AdaBoost fusion algorithm has a more appealing intuitive interpretation. Recall that AdaBoost has the capability of identifying difficult-to-classify examples in the training data. During training, a given target sample with erroneous pose estimation, will be categorized as a hard example, if fine features are used to represent that sample. To prevent the propagation of pose-estimation error in AdaBoost training, we simply discard the information of fine features and use raw features instead for those difficult-to-classify training data.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

We validate the proposed ATR system on the MSTAR public release database.[2] Here, the task is to classify three distinct types of ground vehicles: BTR70, BMP2, and T72. There are seven serial numbers (i.e., seven target configurations) for the three target types: one BTR70 (sn-c71), three BMP2's (sn-9596, sn-9566, and sn-c21), and three T72's (sn-132, sn-812, and sn-s7). For each serial number, the training and test sets are provided, with the target signatures at the depression angles 17° and 15°, respectively. The sizes of the training and test datasets are given in Table 1.

We conduct experiments for two different settings. In the first setting, which we refer to as $\mathcal{S}_1$, our classifier is optimized by using all seven training sets and tested on all seven test sets. To balance the number of training data per target class, we augment the training set of BTR70 sn-c71 by triplicating each image chip. In the second setting, referred to as $\mathcal{S}_2$, our classifier is learned using only a subset of training data, and tested on all available test data. More precisely, for training we use training datasets of BTR70 sn-c71, BMP2 sn-c21, and T72 sn-132, while for testing, all seven test sets. By doing so, we are in a position to examine the generalization properties of our method.

	Training set		**Testing set**	
	serial number	size	serial number	size
BTR70	sn-c71	233	sn-c71	196
BMP2	sn-9563	233	sn-9563	195
	sn-9566	232	sn-9566	196
	sn-c21	233	sn-c21	196
T72	sn-132	232	sn-132	196
	sn-812	231	sn-812	195
	sn-s7	228	sn-s7	191

Table 1. Summary of the MSTAR database.

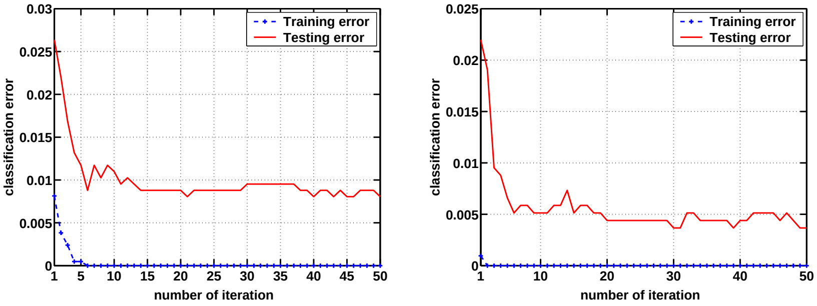
Throughout, to reduce computational complexity, the dimensionality of each data is reduced to 150 by principal component analysis. The projection of the original 80×80=6400-dimensional onto a 150-dimensional feature space is justified by significant processing-time savings, the price of which is nearly negligible in the ultimate classification performance of our system. The maximum number of iterations in AdaBoost is fixed at $T = 50$. The number of Gaussian RBF centers and the number of iterations are optimized through cross validation for each binary problem. Also, the regularization constant λ is set to 10^{-6}.

4.2. Experiments for $\mathcal{S}_1$ Setting

In this section, we present the performance of our classifier trained on all seven training datasets and tested on all seven test datasets. To decompose the multiclass problem into a set of binary problems, we use the one-against-others and all-pair encoding approaches, specified by the code matrices $\mathbf{M}_o$ and $\mathbf{M}_a$ (Equation (3)), respectively. Each binary problem is classified by the RBF net that is trained through the AdaBoost algorithm. The outcomes of the RBF nets are combined into a code word. This code word is then interpreted as one of possible target classes by using either the Hamming decoding or the max-correlation decoding. In the experiments, the training data is always classified with zero classification error after $T = 50$ iteration steps in AdaBoost. In Table 2, we report the correct-classification rate obtained for the test dataset, where we use different coding and decoding methods. The best performance is achieved when the all-pair coding and max correlation decoding are used. The probability of correct classification is 99.63%, which means that only 5 out of the 1365 test chips are misclassified. The confusion matrix of the best classification result is given in Table 3.

To validate the efficiency of AdaBoost, we now present the classification results obtained by using only the RBF base learner, discarding the boosting. As before, after decomposing the multiclass problem into a set of binary ones, the code word of the RBF outcomes is decoded as a target class. Note that the examined system corresponds to a classifier generated after the first AdaBoost iteration. The classification results, when AdaBoost is not used, are reported in Table 4. Comparing the results in Tables 2 and 4, we observe that AdaBoost significantly improves the performance of the RBF networks. This phenomenon is also illustrated by the training and testing error curves in Figures 5(a) and 5(b). From the figures, we observe that the training error exponentially decreases to zero as the number of iterations increases, which is in agreement with the theoretical result discussed in Section 3.1. Also, the classification error on the test dataset becomes smaller with the increase in the number of iterations. Moreover, it keeps decreasing even after the training error becomes zero. From Figures 5(a) and 5(b), we also see that setting the maximum number of iterations of AdaBoost to 50 is conservative. From our experience, approximately 20 iteration steps are enough for AdaBoost to yield a sufficiently accurate classifier. We point out that the specification of the optimal maximum number of AdaBoost iteration steps is not critical for the overall performance of our ATR system.

In the above experiments, both the fine and raw features are used for data representation. To demonstrate the effectiveness of our data fusion method, in Table 5, we present the correct-classification rate, when either the fine features, or the raw features are used. In comparison to the results when both feature sets are used, we observe that our feature fusion procedure significantly improves the probability of correct classification. Note that extracting only raw features gives a better classification performance than extracting only fine features. We defer a detailed explanation of this observation for the next subsection.



(a) (b)

Figure 5. Classification error curves of (a) the one-against-others approach, and (b) the all-pair approach. Both use the max correlation decoding rule. The classifiers are learned by AdaBoost on all training datasets and tested on all testing datasets.

We also compare our ATR algorithm with the following existing approaches: (1) the matched-filter based approach,[17] (2) the MACH filter based approach,[18] and (3) the Neural-Network based approach.[5] For all these methods, the experimental set-up is the same as ours. The classification results are summarized in Table 6. From the table, we observe that our ATR system, when both feature sets are used, outperforms all three methods. Also, when only raw features are extracted, our system achieves equal performance to that of the MACH filter based approach, while still proves better than the Neural-Network based approach. The closest classification result to ours is achieved by the matched-filter based method,[17] where 504 templates are used for the three target classes. However, in terms of the computational complexity and the memory usage, the matched-filter based method may not be the most suitable for implementation in real-time systems. For those systems, our approach has clear advantage.

4.3. Experiments for $\mathcal{S}_2$ Setting

To demonstrate the generalization capability of our ATR scheme, we present experimental results for the $\mathcal{S}_2$ setting. Here, we train our classifier only on a subset of training datasets, representing three serial numbers: T72 sn-132, BMP2 sn-c21, and BTR70 sn-c71 at the depression angle of 17°. After training, we test the classifier on all seven test datasets with target signatures at the depression angle of 15°.

The experimental results are reported in Table 7. We note that, for $\mathcal{S}_2$ setting, our system achieves correct classification at the rate of 96.12% over all seven test datasets. This means that our method has a very good generalization property. The best result is accomplished by using the all-pair encoding approach and the max correlation decoding rule, as for $\mathcal{S}_1$ setting.

To demonstrate the effectiveness of our data fusion method, herein, we also report classification results when either of the feature sets is used. In these experiments, both coding approaches are employed for decomposition of the multiclass problem into a set of binary problems. The max-correlation rule is used for decoding. The correct-classification rate is presented in Table 8. Note that the AdaBoost-based fusion of features improves the classification performance.

In contrast to the results obtained for $\mathcal{S}_1$ setting, now, extracting only fine features gives a higher correct-classification rate than using only raw features. This indicates that, along with the variations in target aspect angles, poor diversity of training data is another key factor which conditions the classification error. Thus, for $\mathcal{S}_1$ setting, where the training dataset represents well all target variations, the pose-estimation error is the dominant factor. Consequently, for $\mathcal{S}_1$ setting, using only raw features gives a more successful classifier than extracting only fine features. On the other hand, for $\mathcal{S}_2$ setting, the training dataset lacks information on certain target aspect angles. If the pose estimator is not employed to mitigate the variations in target poses, the missing information on target poses hinders the optimal training of our classifier. As a result, for $\mathcal{S}_2$ setting, poor diversity of training data becomes the dominant factor leading to an increased classification error. Therefore, for $\mathcal{S}_2$ setting, using only fine features yields a more successful classifier than extracting only raw features. In practice, however, it

	Hamming decoding	max correlation decoding
one-against-all	98.68%	99.19%
all-pair	99.05%	99.63%

Table 2. Correct classification rates using both feature sets for $\mathcal{S}_1$ setting.

	BTR70	BMP2	T72
BTR70	195	1	0
BMP2	0	584	3
T72	0	1	581

Table 3. Confusion matrix of the all-pair encoding and the max correlation decoding rules using both feature sets for $\mathcal{S}_1$ setting.

is *never* known *a priori* which setting our classifier is required to operate in. Fortunately, as our experimental results demonstrate, regardless of the setting, $\mathcal{S}_1$ or $\mathcal{S}_2$, the proposed fusion method always leads to an improved classification performance.

We also compare the classification performance of our ATR system with that of the template based, Neural-Network based, and SVM based approaches presented in.[7] For these methods, the same training and test sets are used as ours. As reported in,[7] for each of the three methods, a threshold is set to keep the probability of detection on the whole test dataset equal to 0.9. That is, the classifier first rejects hard-to-categorize samples up to 10% of the total number of test samples, and then the correct-classification rate is calculated only for the remaining 90% of test samples. In contrast, when experimenting with our ATR system, no test data is discarded. In Table 9, we report the correct-classification rate for the three benchmark approaches, as well as the results for our ATR system, where the all-pair encoding approach and max correlation decoding rule are used. From the table, we observe that our algorithm significantly outperforms the other three, even though our ATR system is tested on the entire test set.

5. CONCLUSIONS

We have proposed a novel AdaBoost-based ATR system for the classification of three types of ground vehicles in the MSTAR public database. Targets in the MSTAR data have randomly distributed poses, ranging from 0° to 360°. To eliminate variations in target poses we have proposed an efficient pose estimator, whereby all targets are rotated to the same aspect angle. To compensate for the occasional pose estimation error, we have proposed to represent MSTAR images by two correlated feature vectors, called fine and raw features. We formulate the ATR problem as a machine-learning problem, where the AdaBoost algorithm is used for two purposes: as a classifier and as a fine/raw-feature fusion method. Through feature fusion, we have efficiently reduced the redundant information contained in the two feature vectors. As the base learner in AdaBoost, we have used the RBF network as a binary classifier. Since identification of multiple types of vehicles in the MSTAR database represents the multiclass recognition problem, we have employed the error-correcting output codes (ECOC) method to decompose it into a set of binary problems, which can be then classified by AdaBoost. The outcomes of each binary classification are finally combined into a code word, which is then "decoded" by measuring the distance of that code word from the pre-defined code words representing the three ground-vehicle classes in the MSTAR database.

Further, we have reported the results of our large-scale experiments. When all the available training data is used, our system achieves 99.63% correct classification rate. To our knowledge, this is the best result ever reported in the literature. When only a subset of the training data is used, our system achieves 96.12% correct classification rate. This means that our system has a very good generalization capacity.

REFERENCES

1. Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences* **55**(1), pp. 119–139, 1997.

	Hamming decoding	max correlation decoding
one-against-all	94.58%	97.36%
all-pair	97.44%	97.80%

Table 4. Correct classification rates of the RBF base learner using both feature sets for $\mathcal{S}_1$ setting.

	fine features	raw features	both features
one-against-all	97.07%	97.95%	99.19%
all-pair	97.44%	98.10%	99.63%

Table 5. Correct classification rates for different feature sets using the max correlation decoding for $\mathcal{S}_1$ setting.

	Matched Filter	MACH Filter	Neural Network	AdaBoost	
				raw features	both features
Pcc	98.97%	98.10%	93%	98.10%	99.63%

Table 6. Correct classification rate of different ATR systems for $\mathcal{S}_1$ setting. Pcc - probability of correct classification.

	Hamming decoding	max correlation decoding
one-against-all	92.45%	95.53%
all-pair	95.46%	96.12%

Table 7. Correct classification rates using both feature sets for $\mathcal{S}_2$ setting.

	fine features	raw features	both features
one-against-all	94.65%	92.45%	95.53%
all-pair	95.31%	92.82%	96.12%

Table 8. Correct classification rates for different feature sets using the max correlation decoding for $\mathcal{S}_2$ setting.

	Template Matcher[7]	Neural Network[7]	SVM[7]	AdaBoost
Pcc	89.7%	94.07%	94.87%	96.12%
Pd	90%	90%	90%	100%

Table 9. Correct classification rate of different ATR systems for $\mathcal{S}_2$ setting; Pcc - probability of correct classification, Pd - probability of detection. Note that the classification rate of our AdaBoost ATR system is calculated for a much higher (100% vs. 90%) detection probability than for the other systems.

2. T. Ross, S. Worrell, V. Velten, J. Mossing, and M. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery V*, pp. 566–573, (Orlando, Florida), 1998.
3. Q. H. Pham, A. Ezekiel, M. T. Campbell, and M. J. T. Smith, "A new end-to-end SAR ATR system," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery VI*, pp. 293–301, (Orlando, Florida), 1999.
4. A. K. Shaw and V. Bhatnagar, "Automatic target recognition using eigen-templates," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery V*, pp. 448–459, (Orlando, Florida), 1998.
5. Q. H. Pham, T. M. Brosnan, M. J. T. Smith, and R. M. Mersereau, "An efficient end-to-end feature based system for SAR ATR," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery VI*, pp. 519–529, (Orlando, Florida), 1998.
6. N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines (and other kernel-based learning methods)*, Cambridge University Press, 2000.
7. Q. Zhao, J. S. Principe, V. Brennan, D. Xu, and Z. Wang, "Synthetic aperture radar automatic target recognition with three strategies of learning and representation," *Optical Engineering* **39**, pp. 1230–1244, May 2000.
8. M. Bryant and F. Garber, "SVM classifier applied to the MSTAR public data set," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery VI*, pp. 355–360, (Orlando, Florida), 1999.
9. R. Meir and G. Rätsch, "An introduction to boosting and leveraging," *In S. Mendelson and A. Smola, editors, Advanced Lectures on Machine Learning, LNCS, Springer* , pp. 119–184, 2003.
10. T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research* **2**, pp. 263–286, Jan. 1995.
11. J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation* **1**(2), pp. 281–294, 1989.
12. R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated prediction," *Machine Learning* **37**(3), pp. 297–336, 1999.
13. R. E. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics* **26**(5), pp. 1651–1686, 1998.
14. C. Bishop, *Neural Networks for Pattern Recognition*, Claredon Press, Oxford, 1995.
15. G. Rätsch, T. Onoda, and K. R. Müller, "Soft margins for AdaBoost," *Machine Learning* **42**, pp. 287–320, 2001.
16. E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Jouranl of Machine Learning Research* **1**, pp. 113–141, Dec. 2000.
17. L. M. Kaplan, R. Murenzi, E. Asika, and K. Namuduri, "Effect of signal-to-clutter ratio on template-based ATR," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery VI*, pp. 408–419, (Orlando, Florida), 1998.
18. A. Mahalanobis, D. W. Carlson, and B. V. Kumar, "Evaluation of MACH and DCCF correlation filters for SAR ATR using MSTAR public data base," in *Proc. SPIE: Algorithms for Synthetic Aperture Radar Imagery V*, pp. 460–468, (Orlando, Florida), 1998.