

# Modeling Images With Multiple Trace Transforms for Pattern Analysis

Nan Liu, *Student Member, IEEE*, and Han Wang, *Senior Member, IEEE*

**Abstract**—Taking advantage of the various available Trace transforms generated from a single image, the multiple Trace feature (MTF) is proposed as a new image representation. In the process of MTF construction, genetic algorithms (GAs) play a key role as an information fusion tool. The systematic evaluations on a combo face data set comprising ORL, Yale, and UMIST databases reveal that MTF presents high discriminative power in terms of outperforming features extracted from principal component analysis (PCA) and linear discriminant analysis (LDA). In addition, the proposed Bagging-based extension of fitness guides GAs achieving more fitting features for classification.

**Index Terms**—Face recognition, genetic algorithms, multiple trace feature, trace transform.

## I. INTRODUCTION

IN the applications of pattern classification, people are not always satisfied with the quantity of training samples, especially in human face authentication where very few images per person are available. Therefore, numerous feature extraction methods are proposed to select as many discriminative features as possible for recognition. However, when the instances are limited, and no matter how powerful the features are, the increase on generalization performance has an upper bound.

A natural question is, are there any alternative image representations? The answer is yes. Trace transform [1] was proposed several years ago but received little attention in the pattern recognition community. Given an image, Trace transform is able to generate several alternative representations by choosing different Trace functionals [2]. Consequently, a new set of two-dimensional virtual images are created. Srisuk *et al.* [3] have done some pioneering works on using Trace transform for face recognition. Their method extracts shape information from the Trace transforms and classifies patterns through reinforcement learning. However, the transforms obtained from different Trace functionals are handled separately. Thus, our focus is on utilizing multiple Trace transforms and fusing them to construct discriminatory features that are able to improve the overall generalization performance.

In this paper, multiple Trace feature (MTF), a novel image representation constructed through evolutionary process, is proposed. Moreover, a new type of fitness function based on both cross-validation [4] and Bagging [5] is proposed to guide the evolving procedure to achieve better testing result. Details of the algorithm and supporting results on the performance evaluations are described in the remaining sections.

Manuscript received October 17, 2008; revised January 10, 2009. Current version published March 26, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vince D. Calhoun.

The authors are with the School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: nliu@pmail.ntu.edu.sg; hw@ntu.edu.sg).

Digital Object Identifier 10.1109/LSP.2009.2016450

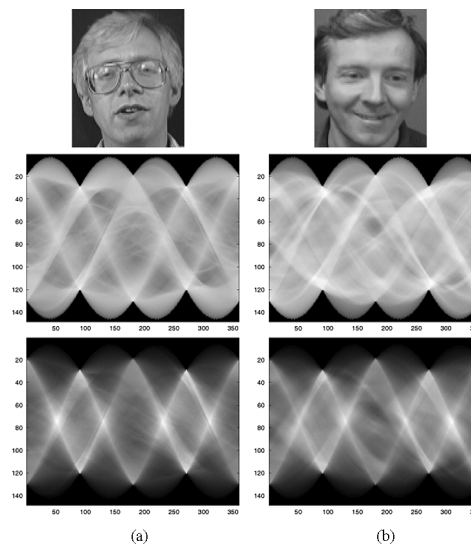


Fig. 1. Face images and their Trace transforms. The transforms in row 2 and row 3 correspond to Trace functionals  $T_1$  and  $T_2$ , respectively.

## II. IMAGE REPRESENTATION USING SINGLE TRACE TRANSFORM

In Randon transform, it is proposed that for some time, an image can be reconstructed from integrals along straight lines defined in its domain. As a generalization, Trace transform [1] extends the reconstruction by applying more functionals rather than integrals only. Given the original coordinate system of the image  $C_1$ , Trace transform  $g$  can be obtained through scanning an image with lines in all directions. Suppose a line  $L(C_1; \phi, p, t)$  with parameters  $t, \phi, p$  which are the directions of tracing line, the orientation of the normal to the line, and the distance from the line to the origin, respectively. Trace transform is defined as  $g(F; C_1; \phi, p) = T(F(C_1; \phi, p, t))$  with the help of  $T$  which is called Trace functional. As  $F(C_1; \phi, p, t)$  are the values of an image along tracing lines, variable  $t$  can be eliminated. Thus 2-D function of  $\phi$  and  $p$  is created and can be interpreted as a new image.

In summary, when an image is transformed to the Trace transform space, its values are proportional to grey scales whose value represents the physical meaning of the face image. The resultant Trace transform depends on the functional that we use. Therefore, by applying different functionals to a single image, various Trace transforms can be produced. Two face images and their Trace transforms using two functionals are shown in Fig. 1, in which transforms in second and third rows correspond to  $T_1$  and  $T_2$ . It can be observed that large variations exist within different Trace transforms of the same image. In this paper, besides  $T_1$  and  $T_2$ , other six functionals are used in our implementation.

These functionals can be found in [3] with function index 3, 4, 6, 8, 10, 12, 18, 20.

$$T_1 : \left( \int_0^\infty |f(t)|^p dt \right)^q, p = 4, q = 1/p \quad (1)$$

$$T_2 : \int_c^\infty r^p f(t) dt, p = 0.5, r = |l - c|, l = 1, 2, \dots, n, \quad (2)$$

$$c = \frac{1}{S} \int_0^\infty r |f(t)| dt, S = \int_0^\infty |f(t)| dt.$$

### III. PROPOSED MULTIPLE TRACE FEATURE

Given that Trace functional  $T_t$  is implemented to the original image,  $g^t(F; C_1, \phi, p)$  is generated where  $t$  is the index of the Trace functional. In order to extract features for further processing such as classification, the 2-D image  $g^t$  is divided into  $\phi$  vectors to form 1D string as shown in (3).

$$\mathbf{x}^t = [g^t(F; C_1, 1, p)^T, \dots, g^t(F; C_1, \phi, p)^T]^T. \quad (3)$$

However, the vector dimension is fairly large ( $\sim 50000$ ) that redundant or irrelevant information may exist. Thus, dimensionality reduction algorithms are required to decrease the feature complexity. To simplify the description, the superscript  $t$  is omitted thereafter.

#### A. Dimensionality Reduction

Principal component analysis (PCA) and linear discriminant analysis (LDA) are implemented to remove redundancy within the Trace transforms. PCA generates a set of orthonormal basis vectors that maximize the scatter of all the projected samples. Suppose that there are  $N$  vectorized Trace transforms  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$  and each has  $M$  observations in the feature set, the covariance matrix of the image set is defined as  $\Sigma = (1)/(N) \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{X}})(\mathbf{x}_i - \bar{\mathbf{X}})^T$  where  $\bar{\mathbf{X}} = (1)/(N) \sum_{i=1}^N \mathbf{x}_i$ . Its eigenvector and eigenvalue matrices  $\mathbf{U}$  and  $\Lambda$  are computed by solving  $\Lambda \mathbf{U} = \Sigma \mathbf{U}$ . Assume that the eigenvalues are sorted in decreasing order, the first  $m$  leading eigenvectors define the feature space as  $\tilde{\mathbf{U}} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m]$ . Then the feature vector  $\mathbf{y}_k$  of pattern  $\mathbf{x}_k$  is calculated as  $\tilde{\mathbf{U}}^T \mathbf{x}_k$ .

LDA is a technique for finding an optimal subspace in which the ratio of the between-class scatter and the within-class scatter is maximized. Assume that each pattern  $\mathbf{x}_k$  belongs to one of  $c$  classes  $\{l_1, l_2, \dots, l_c\}$ , we define the between-class matrix as  $\mathbf{S}_B = \sum_{i=1}^c N^i (\bar{\mathbf{X}}^i - \bar{\mathbf{X}})(\bar{\mathbf{X}}^i - \bar{\mathbf{X}})^T$  and the within-class matrix as  $\mathbf{S}_W = \sum_{i=1}^c \sum_{\mathbf{x}_k \in l_i} (\mathbf{x}_k - \bar{\mathbf{X}}^i)(\mathbf{x}_k - \bar{\mathbf{X}}^i)^T$  where  $\bar{\mathbf{X}}^i$  is the mean image of class  $l_i$ .  $N^i$  is the number of samples in class  $l_i$ . If  $\mathbf{S}_W$  is nonsingular, the optimal subspace can be calculated as  $\mathbf{W}_{\text{opt}} = \arg \max_{\mathbf{W}} (|\mathbf{W}^T \mathbf{S}_B \mathbf{W}|) / (|\mathbf{W}^T \mathbf{S}_W \mathbf{W}|)$ .

#### B. Genetic Algorithms-Based Multiple Trace Feature

Single Trace transform can be used instead of the original image for classification such as in [1] and [3], but there is no prior knowledge on how efficient a specific functional  $T_t$  is. In other words, Trace transform using one functional may perform well in application A while unsatisfied in accomplishing task B. In face recognition, specific Trace functional  $T_t$  can be found during a training phase in terms of successfully suppressing variations in pose, illumination and expression [3]. Although the low-dimensional vector  $\mathbf{y}^t$  captures the most expressive features, its high variance does not necessarily lead to good discriminatory ability unless the corresponding distribution is

multimodal and the modes correspond to the classes to be discriminated [4].

Taking advantage of available Trace functionals, multiple transforms of different models are obtained. In this paper, we propose a novel image representation by fusing multiple Trace feature vectors. To clarify the terminology, the new representation is called the multiple Trace feature. Given a single image, a series of feature vectors  $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^8$  are generated by implementing Trace functionals  $T_1, T_2, \dots, T_8$ . Suppose that the feature dimension is  $m$ , we have

$$\begin{cases} \mathbf{y}^1 = [\mathbf{y}^1(1), \mathbf{y}^1(2), \dots, \mathbf{y}^1(m)]^T \\ \mathbf{y}^2 = [\mathbf{y}^2(1), \mathbf{y}^2(2), \dots, \mathbf{y}^2(m)]^T \\ \vdots \\ \mathbf{y}^8 = [\mathbf{y}^8(1), \mathbf{y}^8(2), \dots, \mathbf{y}^8(m)]^T. \end{cases} \quad (4)$$

According to the proposed feature fusion mechanism, the new representation is  $\mathbf{y} = [\mathbf{y}^{I_1}(1), \mathbf{y}^{I_2}(2), \dots, \mathbf{y}^{I_m}(m)]^T$  where  $I_i = 1, 2, \dots, 8$  and  $i = 1, 2, \dots, m$ . Therefore,  $8^m$  possible new features are created. Obviously, a huge number of feature combinations are there, from which we should select proper features for generating best classification performance. Hence, the problem can be considered as a task of discovering the global optimum. It is a challenge to tackle such a complex task, as most optimization algorithms may get into local minimum during convergence.

Genetic algorithms (GAs) [6], inspired by the biological mechanisms of reproduction, are a class of robust optimization method. GAs have been used extensively in pattern recognition, for example, face recognition [7]. Therefore, we want to take the merit of GAs to solve our problems. Genetic algorithms start the search by randomly generating a set of scalars to form the initial population. Then the GAs make choices of evolution via genetic operators selection, crossover, and mutation, driven by the fitness function. The fitness plays a key role in controlling the convergence to global optimum.

The chromosome in the population is coded as a set of  $m$  (feature dimension) scalars  $\alpha_1, \alpha_2, \dots, \alpha_m$  where  $\alpha_i \in (0, 8)$  and  $i = 1, 2, \dots, m$ . The scalars can be interpreted as:

$$\begin{cases} \text{if } \alpha_i \in (0, 1], & t = 1 \Rightarrow \mathbf{y}(i) = \mathbf{y}^1(i) \\ \text{if } \alpha_i \in (1, 2], & t = 2 \Rightarrow \mathbf{y}(i) = \mathbf{y}^2(i) \\ \vdots \\ \text{if } \alpha_i \in (7, 8), & t = 8 \Rightarrow \mathbf{y}(i) = \mathbf{y}^8(i). \end{cases} \quad (5)$$

Since eight Trace functionals are used, the upper limit of  $\alpha_i$  is 8. In general, a set of  $N_p$  (population size) chromosomes are randomly generated to constitute the initial population. In order to get a trade-off between the computational cost and the diversity of the population, a population size of 100 is used in the experimental validation. Then, for each generation  $k$ ,  $\mathbf{P}(k)$  indicates the population of individuals (chromosomes). During each generation, individuals are evaluated by calculating their fitness values, based on which the chromosome selection is done probabilistically for creating a mating pool  $\tilde{\mathbf{P}}(k)$ . The fitness related selection mechanism can ensure that fitter chromosomes have higher probability of survival. Classification accuracy (CA) is usually employed as the fitness function to guide the search in the field of evolutionary pattern recognition [7]. In order to avoid the over-training that may degrade the overall generalization ability,  $K$ -fold cross-validation scheme is applied for measuring fitness values. Thus the fitness function  $\zeta(\mathcal{F})$  is computed as

(1)/(K)  $\sum_{i=1}^K CA_i$ , where  $CA_i$  is the recognition rate in one of  $K$  validation sets of the training data.

After evaluating the fitness, GAs move to create a new generation  $\mathbf{P}(k+1)$ . Firstly,  $(1-p_c)N_p$  members of  $\hat{\mathbf{P}}(k)$  with top fitness values are probabilistically selected to be part of  $\mathbf{P}(k+1)$ . Then, the crossover operator is applied to the remaining individuals in  $\hat{\mathbf{P}}(k)$ . The offsprings after crossover together with selected chromosomes form the new generation  $\mathbf{P}(k+1)$ . Finally, a small portion of the population,  $p_m \cdot N_p$  candidates in the new generation undergo mutation so as to introduce diversity into the population to prevent a premature convergence.  $p_c$  and  $p_m$  are the crossover and mutation probabilities, respectively. The evolutionary process will be repeated several times until the termination criteria is achieved. In our work, the evolution will stop after 200 generations. Once the training is finished, the scalars encoded in the optimal individual of the last generation are used for testing on unseen data.

### C. Cross-Validation Plus Bagging: A New Fitness Function

Bagging predictor [5] generates multiple models using bootstrap samples and votes for class prediction in classification task. By implementing such scheme, classification accuracy and stability are improved, and overfitting may be avoided due to reduced variance. Therefore, we propose incorporating the Bagging scheme to enhance the cross-validation based fitness function so that higher recognition rate can be achieved.

Let  $L$  denote the training data set, in calculating the fitness values using  $K$ -fold cross-validation, approximately  $((K-1)N)/(K)$  instances  $L_{\text{trn}}$  are used for training and the remaining  $(N)/(K)$  samples  $L_{\text{val}}$  are used for validation during the evolutionary procedure. As the number of instances in  $L_{\text{val}}$  is small in face recognition, high classification accuracy could be achieved in the evolving stage, while poor performance is probably observed in real testing. To investigate possible solutions to above difficulties, Bagging method is combined with the cross-validation scheme to facilitate the training in which validation is done with  $L$  rather than  $L_{\text{val}}$ . Technical details are described as follows. Given data set  $L_{\text{trn}} = \{(\mathbf{y}_n, l_n)\}, n = 1, 2, \dots, ((K-1)N)/(K)$  with labels  $l_n \in \{l_1, l_2, \dots, l_c\}$ ,  $h(\mathbf{y}, L_{\text{trn}})$  is defined as the predictor, i.e., classifier. Suppose  $B$  bootstrap replicates are to be created, we repeat the following process  $B$  times. Firstly, we take a bootstrap sample  $L_{\text{trn}}^b$  by randomly drawing  $p_b$  percent of  $L_{\text{trn}}$  with replacement, where  $b = 1, 2, \dots, B$ . Then we call the learner  $h_b(\mathbf{y}, L_{\text{trn}}^b)$  and receive the hypothesis; add  $h_b$  to the ensemble. So far we have  $B$  hypotheses in the ensemble. The next action is to make a final decision using plurality majority voting. For any test instance  $\mathbf{y}' \in L$ , a series of predictions  $h_1, h_2, \dots, h_B$  are available. Then the vote given to class  $l_j$  by predictor  $h_b$  is defined as

$$D_{b,j} = \begin{cases} 1, & \text{if } h_b \text{ picks class } l_j \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

After obtaining the total votes received by each class, we select the class with the highest vote as the predicted label using (7).

$$h(\mathbf{y}', L_{\text{trn}}) = \arg \max_{j=1}^c \sum_{b=1}^B D_{b,j}, \quad j = 1, 2, \dots, c \quad (7)$$

Since there are  $K$  sets of  $L_{\text{trn}}$ , the above training procedures are repeated  $K$  times, and the mean value of all classification accuracies is used as the new fitness. After pre-defined generations, a set of scalars  $\alpha_1^{\text{opt}}, \alpha_2^{\text{opt}}, \dots, \alpha_m^{\text{opt}}$  are selected, from which the

optimal feature vector  $\mathbf{y}^{\text{opt}}$  can be extracted. Finally, we apply the GAs optimized outputs to the original training and testing sets for experimental validations.

## IV. EXPERIMENTS

### A. Database

The proposed MTF representation is validated with a data set of 1130 samples consisting of 555 instances for training and 575 examples for testing. The data set is a combo database comprising ORL, Yale and UMIST databases where large variations of illuminations and poses are presented. The ORL database contains 400 images of 40 individuals with variations in facial expressions and poses. In the experiments, we randomly select 40 persons with each having five samples as the training set, and the remaining 200 samples are used for testing. The Yale database contains 165 frontal face images covering 15 individuals taken under 11 different facial expression and lighting conditions. The training set consists of 75 randomly selected images, and the testing set includes the remaining 90 images. The UMIST face database comprises 564 images of 20 people covering a range of race, sex, appearance. In validation, 280 images are used for training, and the rest of the images are used for testing.

### B. Experimental Results

Having obtained the face data set, another key task is to choose a proper classifier. Among numerous powerful predictors, a widely used instance-based method,  $k$ -nearest neighbor algorithm ( $k$ -NN) [4], is selected. Because this method simply stores the training examples rather than constructs an explicit description of the target function, so the emphasis of validation can be put onto the proposed features. Given the training set  $L = \{(\mathbf{y}_n, l_n)\}$  where  $n = 1, 2, \dots, N$  and a query feature vector  $\mathbf{y}'$ ,  $k$ -NN algorithm firstly calculates the standard Euclidean distance between test sample and stored training examples, then categorizes  $\mathbf{y}'$  to the class having most instances among the  $k$  neighbors. The distances  $d(\mathbf{y}_n, \mathbf{y}')$  are computed as  $\sqrt{\sum_{i=1}^m (\mathbf{y}_n(i) - \mathbf{y}'(i))^2}$ . Furthermore, in the evolving system, 10-fold cross-validation is adopted for fitness estimation and  $p_c$  and  $p_m$  are set as 0.65 and 0.004.

In the experiments, the proposed MTF is compared with the facial features extracted by PCA and LDA to evaluate its effectiveness. The classification accuracies shown in Table I reveal that MTF outperforms other features in all dimensions.  $\text{MTF}_P^c$  and  $\text{MTF}_L^c$  use the cross-validation based fitness, in which P and L represent dimensionality reduction methods PCA and LDA, respectively.  $\text{MTF}_P^b$  and  $\text{MTF}_L^b$  implement the Bagging-based fitness function and lead to higher accuracies compared with the cross-validation based strategy.

### C. Computational Complexity

The analysis of computational complexity of the proposed algorithm is straightforward. The whole procedure can be divided into several components for discussion because the generating of Trace transform and the subsequent feature extraction are independent from the evolutionary process. Assume that  $n_p, n_\phi, n_t$  are the number of samples of parameters  $p, \phi$ , and  $t$ , respectively. The complexity of creating Trace transforms is calculated as  $O((N_T C_T n_t n_p n_\phi)/(2))$  [1], where  $C_T$  is the number of operations per sample for each Trace functional and  $N_T$  is the total number of functionals. Next, in the  $N_T$  training sets comprising  $N$  instances each,  $O(n_\phi N_T N)$  operations are needed for the conversion from 2-D Trace transform to a string

TABLE I  
RECOGNITION RATES ON THE COMBO FACE DATABASE

$m$	PCA	LDA	MTF <sub>P</sub> <sup>c</sup>	MTF <sub>L</sub> <sup>c</sup>	MTF <sub>P</sub> <sup>b</sup>	MTF <sub>L</sub> <sup>b</sup>
70	0.9148	0.9078	0.9165	0.9270	0.9200	0.9322
65	0.9130	0.9078	0.9217	0.9339	0.9252	0.9252
60	0.9113	0.9078	0.9235	0.9304	0.9165	0.9339
55	0.9009	0.9096	0.9183	0.9304	0.9252	0.9304
50	0.9078	0.9026	0.9252	0.9304	0.9183	0.9443
45	0.8974	0.9113	0.9235	0.9391	0.9322	0.9322
40	0.8974	0.9078	0.9252	0.9304	0.9304	0.9357
35	0.8974	0.9148	0.9252	0.9426	0.9287	0.9443
30	0.9043	0.9183	0.9217	0.9252	0.9217	0.9322
25	0.9009	0.9130	0.9113	0.9270	0.9130	0.9426
20	0.8748	0.8922	0.9200	0.9217	0.9165	0.9339
15	0.8643	0.8957	0.9026	0.9165	0.9078	0.9235
10	0.8157	0.8122	0.8852	0.9026	0.9061	0.9130
05	0.6296	0.6696	0.8087	0.8487	0.8452	0.8539

in the training set, and  $O(N^3 N_T)$  computations are required for implementing PCA [4].

Above operations are done prior to the training process. Because  $k$ -NN classifier is implemented for prediction in the evolutionary procedure, it is important to know the computational cost of  $k$ -NN. In general, the computational complexity of  $k$ -NN classifier is  $O(\text{training sample} \times \text{feature dimension})$ . Recall the parameters in GAs,  $N_p, p_c, p_m$  are population size, crossover probability and mutation probability. In addition,  $N_G$  is defined as the number of expected generations. The calculation of complexity follows the framework of Salomon's works [8]. The bit coding used in our work means that the scalars  $\alpha_1, \alpha_2, \dots, \alpha_m$  are encoded by  $q$  bits. Thus, each chromosome is a bit string of length  $l = mq$  on which the genetic strategy is applied. As a result, a complexity of  $O(l \ln l)$  is derived [8]. As  $K$ -fold cross-validation scheme is used to estimate the fitness values, approximately  $((K-1)N)/(K)$  samples are used to train the  $k$ -NN classifier. Consequently, the overall complexity of evolving process is  $O((l \ln l + N_p K ((K-1)N)/(K) m) N_G)$  where  $O(((K-1)N)/(K) m)$  is the complexity of a single classifier. The complexity of the system can be computed as

$$\text{Complexity} = O\left(\frac{N_T C_T n_t n_p n_\phi}{2} + n_\phi N_T N + N^3 N_T + N_G l \ln l + N_G N_p (K-1) N m\right). \quad (8)$$

The values of parameters used in this paper are application-specific. In the experiments,  $N_T = 8$  Trace functionals are applied to  $N = 555$  training samples. The parameters  $C_T, n_t, n_p, n_\phi$  are set as 10, 100, 72, and 360, respectively.  $n_p$  varies according to image dimensions. As GAs use 10-bit coding,  $q$  is 10. The algorithm was implemented in MATLAB and run on a Pentium 4

2.4 GHz processor. It took around 20 hours for Trace transform generation and several hours for training. Testing on a single feature vector finished within seconds.

In the face recognition system proposed by Srisuk *et al.* [3], it took about 100 hours for training with Pentium 4 processor. Since their algorithms seek for optimal parameters to extract shapes through reinforcement learning, the training relies on 2-D Trace transforms. In our proposal, the Trace transforms are converted to low-dimensional vectors before training, thus the operations on 2-D transforms are not involved in the learning process. Therefore, it is reasonable that the computing time is saved when feature vectors with reduced dimension rather than two-dimensional Trace transforms are used for training.

## V. CONCLUSION

In this paper, various Trace transforms are fused by genetic algorithms to generate the multiple Trace feature for face recognition. Our contribution is the proposal of integration of Trace transforms in the framework of evolutionary computation. In addition to cross-validation based fitness function, bootstrap aggregating algorithm is employed to enhance the estimation of fitness. The experimental results have been presented to demonstrate the robustness and discriminatory power of MTF. Ideally, the proposed feature has a wide range of applications, not just limited to face recognition.

## ACKNOWLEDGMENT

The authors would like to thank M. Petrou, A. Kadyrov, and S. Srisuk, who kindly provided the codes for Trace transform.

## REFERENCES

- [1] A. Kadyrov and M. Petrou, "The trace transform and its applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 811–828, Aug. 2001.
- [2] M. Petrou and A. Kadyrov, "Affine invariant features from the trace transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 30–44, Jan. 2004.
- [3] S. Srisuk, M. Petrou, W. Kurutach, and A. Kadyrov, "A face authentication system using the trace transform," *Pattern Anal. Applicat.*, vol. 8, pp. 50–61, 2005.
- [4] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.
- [5] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, pp. 123–140, 1996.
- [6] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
- [7] C. Liu and H. Wechsler, "Evolutionary pursuit and its application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 570–582, 2000.
- [8] R. Salomon, "The influence of different coding schemes on the computational complexity of genetic algorithms in function optimization," in *Proc. Parallel Problem Solving from Nature*, Berlin, Germany, Sep. 1996, pp. 227–235.