Fingerprint Classification Based on Learned Features

Xuejun Tan, Bir Bhanu, *Fellow, IEEE*, and Yingqiang Lin

***Abstract*—In this paper, we present a fingerprint classification approach based on a novel feature-learning algorithm. Unlike current research for fingerprint classification that generally uses well defined meaningful features, our approach is based on Genetic Programming (GP), which learns to discover composite operators and features that are evolved from combinations of primitive image processing operations. Our experimental results show that our approach can find good composite operators to effectively extract useful features. Using a Bayesian classifier, without rejecting any fingerprints from the NIST-4 database, the correct rates for 4- and 5-class classification are 93.3% and 91.6%, respectively, which compare favorably with other published research and are one of the best results published to date.**

***Index Terms*—Composite operators, feature learning, fingerprint classification, genetic programming.**

I. Introduction

THE Henry system is a systematic method for classifying fingerprints into five classes: Right Loop (R), Left Loop (L), Whorl (W), Arch (A), and Tented Arch (T). Fig. 1 shows an example of each class. This system of fingerprint classification is commonly used by most of the developers and users, although the scheme adopted by the FBI defines eight classes [1]. The most widely used approaches for fingerprint classification are based on the number and relations of the singular points (SPs), which are defined as the points where a fingerprint's orientation field is discontinuous. Using SPs as reference points, Karu and Jain [2] present a classification approach based on the structural information around SPs. Most other research uses a similar method: first, find the SPs and then use a classification algorithm to find the difference in areas, which are around the SPs for different classes. Several representations based on principal components analysis (PCA) [3], a self-organizing map (SOM) [4], and Gabor filters [5] are used. The problems with these approaches are

a) it is not easy to detect the SPs and some fingerprints do not have SPs;
b) the uncertainty about the location of SPs is large, which has great effect on the classification performance since the features around the SPs are used.

Cappelli *et al.* present a structural analysis of a fingerprint's orientation field [6]. Based on the orientation field, they also present a fingerprint classification system based on multispace KL transform [7]. It uses a different number of principal components for different classes and it is not clear how the number of these components is determined. Jain and Minut propose a classification algorithm based on finding the kernel that best fits the flow field of the given fingerprint [8]. In both approaches it is unnecessary to find the SPs. Researchers have also tried different methods to combine different classifiers to improve the classification performance. Senior [9] combines hidden Markov models (HMM), decision trees, and PCASYS (a standard fingerprint classification algorithm) [3]. Yao*et al.* [10] present new fingerprint classification algorithms based on two machine learning approaches: support vector machines (SVMs) and recursive neural networks (RNNs). Table I summarizes representative fingerprint classification approaches. The features used in these approaches are well defined, conventionally known features. Thus, it is clear that most current approaches in fingerprint classification are based on the extraction of reference points and conventional transforms.

Some researchers use learning algorithms to extract minutiae features from fingerprint images. Prabhakar *et al.* [11] propose a feedback system that learns the characteristics of minutiae in gray-scale images and can be used to verify each detected minutia. They show that a minutiae verification stage, which is based on reexamining the gray-scale profile in a detected minutia's spatial neighborhood in a fingerprint image, could improve the matching performance. Bhanu and Tan [12] present a learned template based algorithm for minutiae extraction. Templates are learned from examples by optimizing a criterion function. Using Lagrange's method to detect the presence of minutiae in fingerprints, templates are applied with appropriate orientations to the binary fingerprints only at selected potential minutia locations. However, the above three approaches are for learning minutiae, which are well-defined structure features in fingerprints and are commonly used in fingerprint verification. To the best of our knowledge, unconventional features discovered by the computer are never used in fingerprint classification.

In most imaging applications, the approach used to extract feature vectors from images can often be dissected into some primitive operations on a set of selected features in images. Generally, the task of finding a good feature is equivalent to finding a good point in the search space of composite operators, where a composite operator consists of primitive operators, and it can be viewed as a selected combination of primitive operations applied to primitive feature images. Our Genetic Programming (GP) based approach may try many unconventional ways of combining primitive operations that may never be imagined by humans and may yield exceptionally good results. The parallelism of GP and the speed of computers allow the search space explored by GP to be much larger than that by human experts. As the search goes on, GP gradually shifts the population of composite operators to the portion of the space containing good composite operators.

Manuscript received March 1, 2003; revised December 14, 2003. This work is supported in part by a grant from SONY, DiMI, I/O Software and F49620-02-1-0315. The contents and information do not necessarily reflect the positions or policies of the sponsors. This paper was recommended by Guest Editor D. Zhang.

The authors are with the Center for Research in Intelligent Systems, University of California, Riverside, CA 92521 USA (e-mail: xtan@cris.ucr.edu; bhanu@cris.ucr.edu; yqlin@cris.ucr.edu).

Digital Object Identifier 10.1109/TSMCC.2005.848167

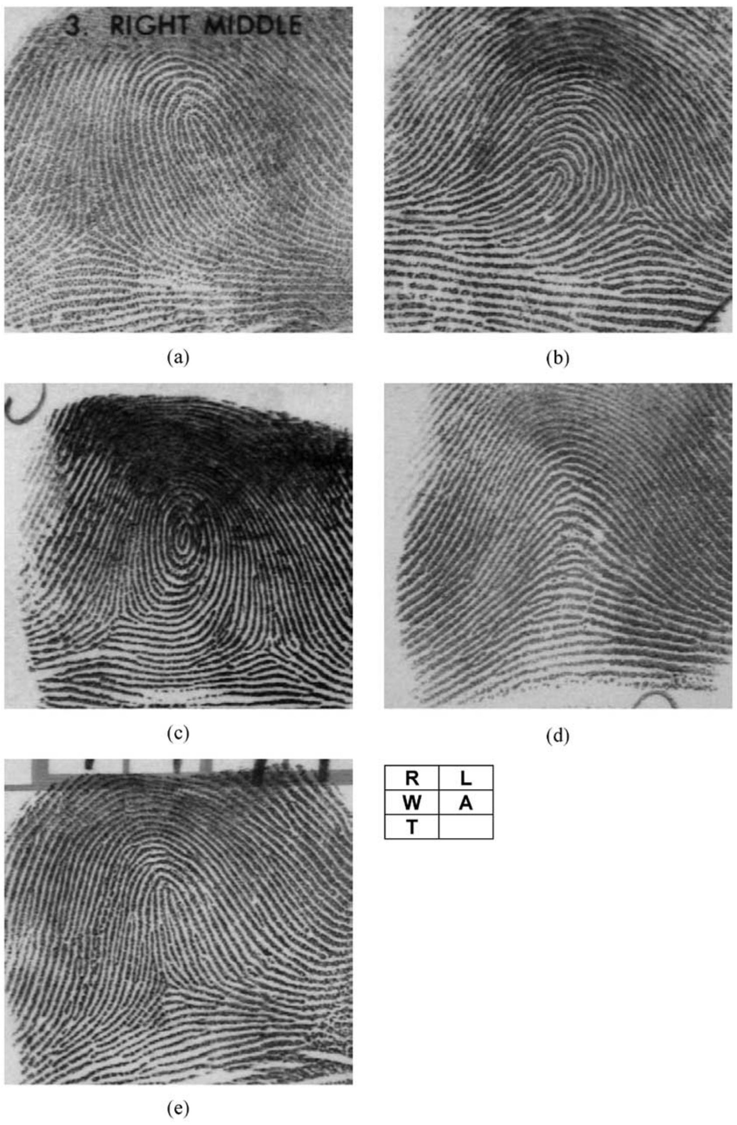


Fig. 1. Examples of fingerprints from each class of the Henry System for fingerprint classification: (a) Right Loop; (b) Left Loop; (c) Whorl; (d) Arch; and (e) Tented Arch.

Genetic programming, an extension of genetic algorithms, was first proposed by Koza in [13]. In GP, the individuals can be binary trees, graphs, or some other complicated structures of dynamically varying size. Poli [14] used GP to develop effective image filters to enhance and detect features of interest or to build pixel-classification-based segmentation algorithms. Stanhope and Daida [15] used GP paradigm for the generation of rules for target/clutter classification and rules for the identification of objects. To perform these tasks, previously defined feature sets are generated on various images and GP is used to select relevant features and methods for analyzing these features. Howard*et al.* [16] applied GP to automatic detection of ships in low-resolution SAR imagery using an approach that evolves detectors. Roberts and Howard [17] used GP to develop automatic object detectors in infrared images.

The contributions of our work are as follows

a) An approach that learns composite operators based on primitive features automatically. It helps to find some useful unconventional features, which are difficult for humans to comprehend and visualize. The primitive operators and features defined in this paper are very basic and easy to compute.
b) Primitive operators are separated into computation operators and feature generation operators. Features are computed wherever feature generation operators are used. These features are used to form a feature vector that represents a particular fingerprint image and it is used for subsequent fingerprint classification.
c) Results are shown on the entire NIST-4 fingerprint database and they are compared with the other published research.

TABLE I
REPRESENTATIVE FINGERPRINT CLASSIFICATION APPROACHES

Approaches	Characteristics
Kamijo 1993 [18]	A four-layered neural network integrated in a two-step learning method
Candela *et al.* 1995 [3]	Probabilistic neural network (PNN)
Karu and Jain 1996 [2]	Rule-based classification
Maio and Maltoni 1996 [6]	Classification based on partitioning of orientation image
Halici and Ongun 1996 [4]	Neural network based on self-organizing feature maps (SOM)
Qi *et al.* 1998 [19]	Probabilistic neural network based on Genetic Algorithm (GA) and feedback mechanism
Jain *et al.* 1999 [5]	K – nearest neighbor + Neural network based on Gabor features (FingerCode)
Su *et al.* 2000 [20]	Fractal analysis
Pattichis *et al.* 2001 [21]	Probabilistic neural network + AM–FM representation for fingerprints
Bernard *et al.* 2001 [22]	Kohonen topologic map
Senior 2001 [9]	Hidden Markov model + Decision tree + PCASYS
Jain and Minut 2002 [8]	Model-based method based on hierarchical kernel fitting
Mohamed and Nyongesa 2002 [23]	Fuzzy neural network
Yao *et al.* 2003 [10]	Support vector machine + Recursive neural network based on FingerCode
Cappelli *et al.* 1999 [7]	Multispace principal component analysis

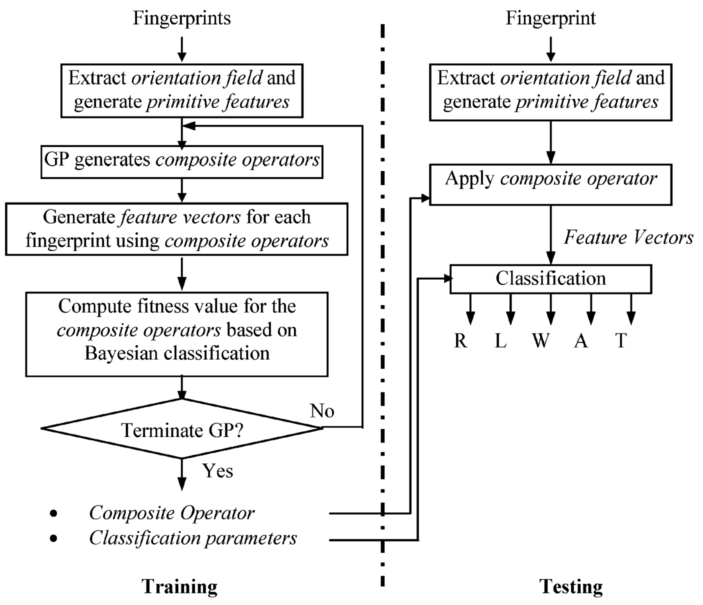

Fingerprints
Extract *orientation field* and generate *primitive features*
GP generates *composite operators*
Generate *feature vectors* for each fingerprint using *composite operators*
Compute fitness value for the *composite operators* based on Bayesian classification
Terminate GP?
No
Yes
• *Composite Operator*
• *Classification parameters*
Training
Fingerprint
Extract *orientation field* and generate *primitive features*
Apply *composite operator*
Feature Vectors
Classification
R L W A T
Testing

Fig. 2. Block diagram of our approach.

II. Technical Approach

Fig. 2 shows the block diagram of our approach. During the training, GP is used to generate composite operators, which are applied to the primitive features generated from the original orientation field. Feature vectors used for fingerprint classification are generated by composite operators. A Bayesian classifier is used for classification. During training, the fitness value is computed according to the classification result and is monitored during evolution. During testing, the learned composite operator is applied directly to generate feature vectors. Note that, in our approach, we do not need to find the reference points.

In our GP-based approach, individuals are composite operators, which are represented by binary trees. The search space of GP is the space of all possible composite operators. The space is very large. In order to illustrate this, consider only a special kind of binary tree, where each tree has exactly 30 internal nodes and one leaf node and each internal node has only one child. For 17 primitive operators and only one primitive feature image, the total number of such trees is 17^{30}. It is extremely difficult to find good operators from this vast space unless one has a smart search strategy.

A. Design Considerations

The major design considerations of GP are explained in the following.

- **The Set of Terminals:** For a fingerprint, we can estimate the orientation field [24]:

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{m} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{m}\sum_{j=1}^{m}\left(G_x^2(i,j) - G_y^2(i,j)\right)}\right) \tag{1}$$

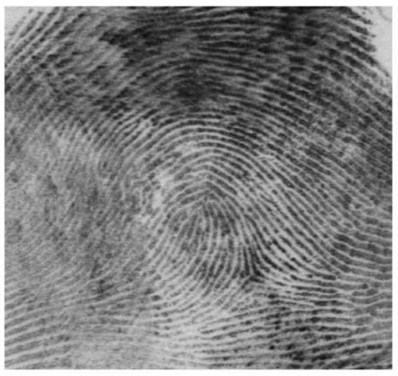
where G_x and G_y are the gradient magnitudes of sobel operators in x and y directions, respectively. And m is the block size $m = 32$ in our experiments. $\theta \in [0, 180)$ and is measured in a clockwise direction.

The set of terminals used in this paper are called primitive features, which are generated from the orientation field. They capture some structural relationships of the orientation field in different areas of a fingerprint. Primitive features used in our experiments are as follows.

  - Original orientation image (primitive feature 1). The orientation image contains important structural information about a fingerprint.
  - Mean, standard deviation, min, max, and median images obtained by applying 3×3 and 5×5 templates on the orientation image (primitive features 2–11). These images contain information in the neighborhood of every pixel in the orientation image.
  - Edge images obtained by applying sobel filters along horizontal and vertical directions on the orientation image (primitive features 12–13). Both images contain information about the changes of orientation along different directions.
  - Binary image obtained by thresholding the orientation image with a threshold of 90 (primitive feature 14). Since $\theta \in [0, 180)$, the threshold is chosen as 90. If the pixel value in the orientation image is greater than 90, the corresponding pixel in the binary image is set to 1, otherwise, 0.
  - Images obtained by applying sine and cosine operations on the orientation image (primitive features 15–16). Both images contain information about the changes in orientation.

These 16 images are input to the composite operators. The size of these images is 12×13. GP determines which operations are applied to them and how to combine the results. Fig. 3 shows an example of a fingerprint image from the NIST-4 fingerprint database and its corresponding primitive feature images. Note that, in order to show primitive feature images clearly, in each primitive image, maximum and minimum values in the image are mapped to 255 and 0, respectively, and other values are linearly mapped to a value between 0 and 255.

- **The Set of Primitive Operators:** A primitive operator takes one or two input images, performs a primitive operation on them, and outputs a resultant image. Suppose: 1) A and B are images of the same size and c is a constant of real number, $c \in [-100, +100]$; 2) for operators, which take two images as input, the operations are performed on a pixel-by-pixel basis. Currently, there are two kinds of primitive operators in our approach: computation operators and feature generation operators. Table II explains the meaning of these operators in detail. For computation operators, the output is an image, which is generated by applying the corresponding operations to the input image. However, for feature generation operators, the output includes an image and a real number or vector. The output image is the same as the input image and passed as the input image to the next node in the composite operator. The real numbers or the vectors are the elements of the feature vector, which is used for classification. Thus, the size of the feature vectors depends on the number of the feature generation operators that are a part of the composite operator. Fig. 4 shows an example of a composite operator, which includes three computation operators and three feature generation operators. Computation operators do computation and the feature vector is generated by feature generation operators: SPE_MAX_OP, SPE_U3_OP, and SPE_STD_OP.


(a)

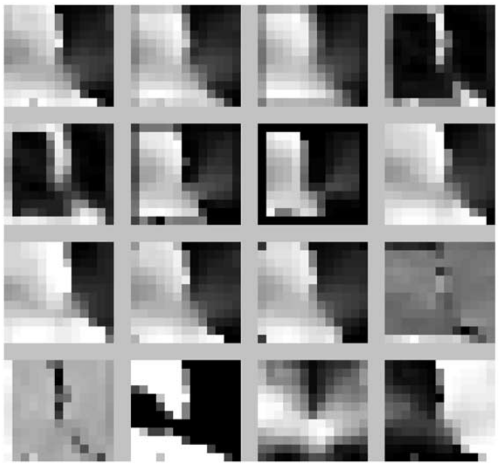

(b)

Fig. 3. Example of a fingerprint image from the NIST-4 fingerprint database and the primitive feature images derived from the original image: (a) original image, f0760_06; and (b) primitive feature images. Note that 16 primitive feature images are sorted from left to right and top to bottom.

- **The Fitness Measure:** During training, at every generation for each composite operator proposed by GP, we compute the feature vector and estimate the Probability Distribution Function (PDF) for each class using all the

TABLE II
PRIMITIVE OPERATORS USED IN OUR APPROACH

Primitive Operator		Meaning
Computation Operators	ADD_OP, SUB_OP, MUL_OP, and DIV_OP	A+B, A–B, A×B, and A/B. If the pixel in B has value 0, the corresponding pixel in A/B takes the maximum pixel value in A.
	MAX2_OP and MIN2_OP	max(A,B) and min(A,B)
	ADD_CONST_OP, SUB_CONST_OP, MUL_CONST_OP, and DIV_CONST_OP	A+c, A-c, A×c, and A/c
	SQRT_OP and LOG_OP	$sign(A) \times \sqrt{\lvert A \rvert}$ and $sign(A) \times \log(\lvert A \rvert)$
	MAX_OP, MIN_OP, MED_OP MEAN_OP, and STD_OP	max(A), min(A), med(A), mean(A), and std(A) replace the pixel value by the maximum, minimum, median, mean, or standard deviation in a 3×3 block.
	BINARY_ZERO_OP and BINARY_MEAN_OP	threshold/binarize A by zero or mean of A
	NEGATIVE_OP	-A
	LEFT_OP, RIGHT_OP, UP_OP, and DOWN_OP	left(A), right(A), up(A) and down(A). Move A to the left, right, up, or down by 1 pixel. The border is padded by zeros.
	HF_DERIVATIVE_OP and VF_DERIVATIVE_OP	HF(A) and VF(A). Sobel filters along horizontal and vertical directions
Feature Generation Operators	SPE_MAX_OP, SPE_MIN_OP, SPE_MEAN_OP, SPE_ABS_MEAN_OP, and SPE_STD_OP	max2(A), min2(A), mean2(A), mean2(|A|), and std2(A)
	SPE_U3_OP and SPE_U4_OP	μ_3(A) and μ_4(A). Skewness and kurtosis of the histogram of A
	SPE_CENTER_MOMENT11_OP	μ_{11}(A). First order central moments of A
	SPE_ENTROPY_OP	H(A). Entropy of A
	SPE_MEAN_VECTOR_OP and SPE_STD_VECTOR_OP	mean_vector(A) and std_vector(A). A vector contains the mean or standard deviation value of each row/column of A.

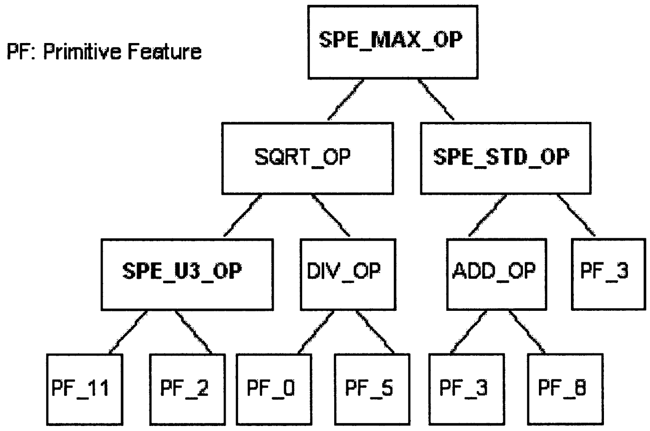


Fig. 4. Example of a composite operator, which includes three computation operators and three feature generation operators. PFs are primitive features.

available feature vectors. For simplicity, we assume feature vectors for each class have normal distribution, $v_{i,j}$, where $i = 1, 2, 3, 4, 5$ and $j = 1, 2, \ldots n_i, n_i$ is the number of feature vectors in the training for class i, ω_i. Then, for each i, we estimate the mean μ_i and covariance matrix Σ_i by all $v_{i,j}$:

$$\mu_i = E[x], \quad \sum_i = E[(x - \mu_i)(x - \mu_i)^T] \tag{2}$$

where $x \in \{v_{i,1} \ v_{i,2} \ \cdots \ v_{i,n_i}\}$.

Thus, the PDF of ω_i can be expressed as

$$p(x \mid \omega_i) = \frac{1}{(2\pi)^{n/2} \lvert \sum_i \rvert^{1/2}} \times \exp\left(-\frac{1}{2}(x - \mu_i)^T \sum_i^{-1} (x - \mu_i)\right). \tag{3}$$

According to Bayesian theory, we have

$$v \in \omega_k, \quad \text{iff} \cdot p(v \mid \omega_k) \cdot p(\omega_k) = \max_{i=1,2,3,4,5} (p(v \mid \omega_i) \cdot p(\omega_i)) \tag{4}$$

where n is the size of the feature vector and v is a feature vector for classification.

During training, we estimate $p(x \mid \omega_i)$, then use the entire training set to do the classification. The Percentage of Correct Classification (PCC) is taken as the fitness value of the composite operator.

$$\text{Fitness Value} = \frac{n_c}{n_s} \times 100\% \tag{5}$$

where n_c is the number of correctly classified fingerprints in the training set and n_s is the size of the training set.

Note that, if $\lvert \Sigma_i \rvert = 0$ for ω_i in (3), we simply let the fitness value of the composite operator be 0. During testing, we still use (4) to obtain the classification results of the testing set; however, none of the testing fingerprints is used in the training.

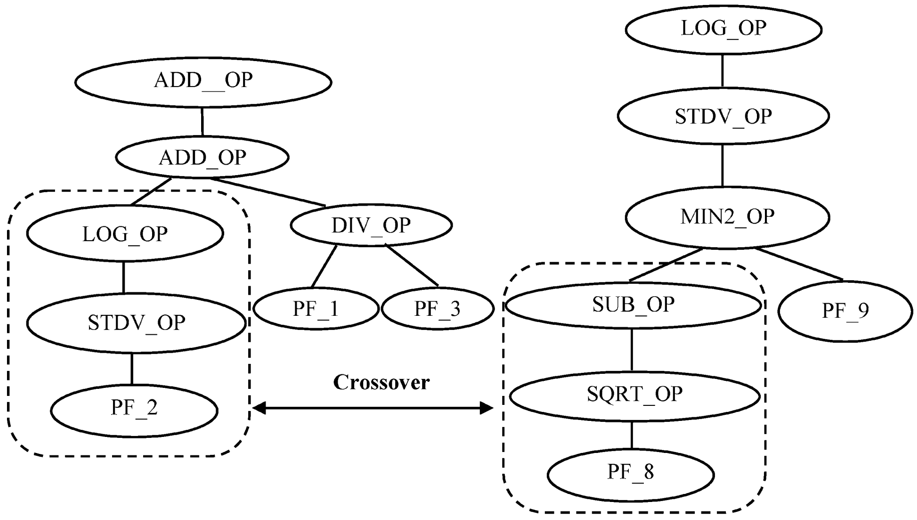


Fig. 5. Example of crossover between two composite operators.

- **Parameters and Termination:** The key parameters are maximum size of composite operator (150), population size (100), number of generations (100), crossover rate (0.6), and mutation rate (0.05). The GP stops whenever it finishes the prespecified number of generations.

B. *Reproduction, Crossover, and Mutation*

The GP searches through the space of composite operators to generate new operators, which may be better than the previous ones. By searching through the composite operator space, GP gradually adapts the population of composite operators from generation to generation and improves the overall fitness of the whole population. More importantly, GP may find an exceptionally good operator during the search. The search is done by performing reproduction, crossover, and mutation operations. The initial population is randomly generated and the fitness of each individual is evaluated.

- **Reproduction:** The reproduction operation involves selecting a composite operator from the current population. In this research, we use tournament selection, where a number of individuals are randomly selected from the current population and the one with the highest fitness value is copied into the new population.
- **Crossover:** To perform crossover, two composite operators are selected on the basis of their fitness values. These two composite operators are called parents. One internal node in each of these two parents is randomly selected, and the two subtrees with these two nodes as root are exchanged between the parents. In this way, two new composite operators, called offspring, are created. Fig. 5 shows an example of crossover between two composite operators.
- **Mutation:** In order to avoid premature convergence, mutation is introduced to randomly change the structure of some of the individuals to help maintain the diversity of the population. Once a composite operator is selected to perform a mutation operation, an internal node of the binary tree representing this operator is randomly selected, then the subtree rooted at this node is deleted, including the node selected. Another binary tree is randomly generated and this tree replaces the previously deleted subtree. The resulting new binary tree represents a new composite operator. This new composite operator replaces the old one in the population. Fig. 6 shows an example of the mutation of a composite operator.

C. *Steady-state and Generational Genetic Programming*

In steady-state GP, two parental composite operators are selected on the basis of their fitness for crossover. The children of this crossover, perhaps mutated, replace a pair of composite operators with the smallest fitness values. The two children are executed immediately and their fitness values are recorded. Then another two parental composite operators are selected for crossover. This process is repeated until the crossover rate is satisfied. In generational GP, two composite operators are selected on the basis of their fitness values for crossover. Then, the two composite operators with the smallest fitness values, among those that have not been selected for replacement, are selected. They will be replaced by the children of the crossover. At this time, the replacement has not occurred. The above process is repeated until the crossover rate is satisfied. A composite operator may be repeatedly selected for crossover, but it cannot be repeatedly selected for replacement. After crossover operations are finished, all the children resulting from the crossover operations replace all the composite operators selected for replacement at once. In addition, we adopt an elitism replacement method that copies the best composite operator from generation to generation. The steady-state and generational genetic programming algorithms are given in Figs. 7 and 8, respectively. For simplicity, we use steady-state GP in our experiments.

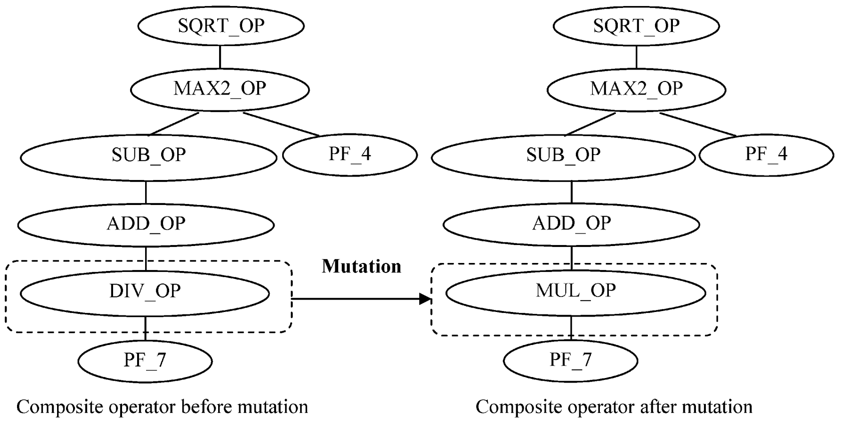


Fig. 6. Example of mutation.

- **Steady-state Genetic Programming:**

```
0. randomly generate population P and evaluate each composite operator in P.
1. for gen = 1 to generation_num do
2.   keep the best composite operator in P.
3.   perform reproduction to generate population P' from P.
4.   number_of_crossover = population_size * cross_over_rate / 2.
5.   for i = 1 to number_of_crossover do
6.     select 2 composite operators from P' based on their fitness values for crossover.
7.     select 2 composite operators with the lowest fitness values in P' for replacement.
8.       perform crossover operation and let the 2 offspring composite operators replace the 2
           composite operators selected for replacement.
9.     if  mutation is performed on the composite operators from the crossover  then
10.      perform mutation on the 2 offspring operators with probability mutation_rate.
       end.
11.    execute the 2 offspring composite operators and evaluate their fitness values.
     end // loop 5
12.  if  mutation is performed on the composite operators from the whole population P'  then
13.    perform mutation on each composite operator with probability mutation_rate.
14.      execute and evaluate mutated composite operators.
     end
15.  let the best composite operator from population P replace the worst composite operator in P'.
16.  let P = P'
17.  if  the fitness value of the best composite operator in P is above fitness threshold value then
18.    stop.
     end
   end  // loop 1
```

Fig. 7. Steady-state genetic programming.

- **Generational Genetic Programming:**

```
0. randomly generate population P and evaluate each composite operator in P.
1. for gen = 1 to generation_num do
2.   keep the best composite operator in P.
3.   perform reproduction to generate population P' from P.
4    number_of_crossover = population_size * crossover_rate / 2.
5    perform crossover number_of_crossover times and record 2 * number_of_crossover composite
       operators to be replaced.
6    perform mutation on the composite operators generated from crossover or on the composite
       operators from the whole population. If a composite operator is mutated, recorded it for later
       execution.
7    execute offspring composite operators from crossover and the mutated composite operators and
       evaluate their fitness values.
8    put offspring composite operators from crossover in P' and remove the composite operators
       selected for replacement from P'.
9.   let the best composite operator from population replace the worst composite operator in P'.
10.  let P = P'
11.  if  the fitness value of the best composite operator in P is above fitness threshold value then
12.    stop.
     end
   end  // loop 1
```

Fig. 8. Generational genetic programming.

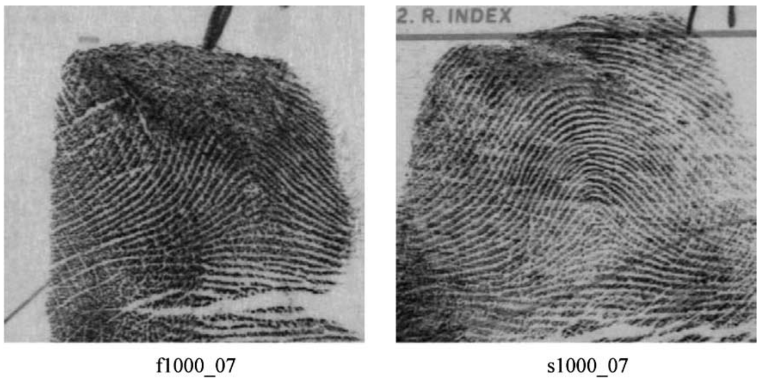


Fig. 9. Sample fingerprints from NIST-4.

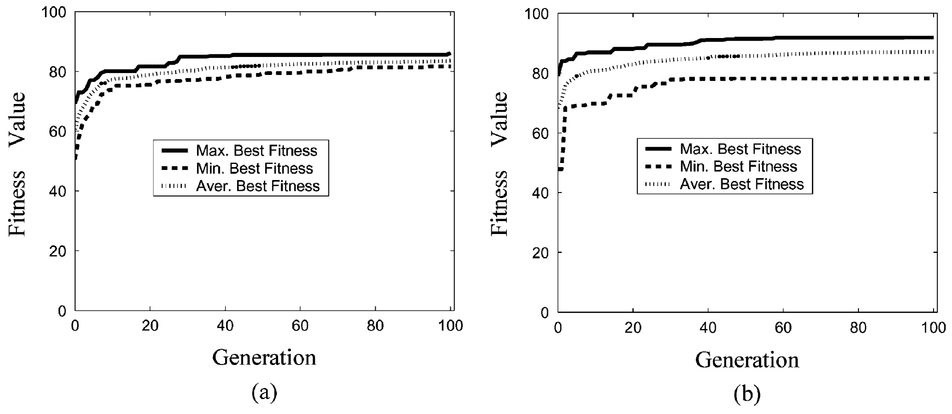


Fig. 10. Average fitness values based on the number of generations: (a) 5-class and (b) 4-class.

Composite Operator for 5-class classification, size 61:

```
( (SUB_OP) ( (MIN_OP) ( (HF_DERIVATIVE_OP) ( (HF_DERIVATIVE_OP) ( (ADD_CONST_OP) (
(MUL_OP) ( (SPE_STD_VECTOR_OP) ( (STDV_OP) ( (SPE_CENTER_MOMENT11_OP) ( (SQRT_OP) (
(SUB_CONST_OP) ( (VF_DERIVATIVE_OP) ( (MEAN_OP) ( (PF_OP: 0) ) ) ) ) ) ) ) ) ( (SUB_CONST_OP) (
(HF_DERIVATIVE_OP) ( (SUB_CONST_OP) ( (HF_DERIVATIVE_OP) ( (ADD_CONST_OP) (
(SUB_CONST_OP) ( (ADD_CONST_OP) ( (MUL_OP) ( (SPE_STD_OP) ( (MEAN_OP) ( (LOG_OP) (
(SPE_MEAN_VECTOR_OP) ( (SQRT_OP) ( (RIGHT_OP) ( (SPE_MIN_OP) ( (ABS_OP) ( (MEAN_OP) (
(PF_OP: 0) ) ) ) ) ) ) ) ) ) ) ( (SUB_CONST_OP) ( (SPE_MEAN_VECTOR_OP) ( (SPE_STD_VECTOR_OP) (
(SPE_MIN_OP) ( (STDV_OP) ( (SPE_CENTER_MOMENT11_OP) ( (SPE_U3_OP) ( (SPE_STD_VECTOR_OP)
( (SPE_MIN_OP) ( (STDV_OP) ( (SPE_CENTER_MOMENT11_OP) ( (SPE_U3_OP) ( (UP_OP) (
(SPE_MEAN_OP) ( (PF_OP: 1) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ( (SUB_CONST_OP) ( (SPE_MEAN_OP)
( (SQRT_OP) ( (SUB_CONST_OP) ( (SPE_U3_OP) ( (SPE_U4_OP) ( (SPE_STD_VECTOR_OP) (
(SPE_MIN_OP) ( (STDV_OP) ( (SPE_CENTER_MOMENT11_OP) ( (SQRT_OP) ( (SUB_CONST_OP) (
(VF_DERIVATIVE_OP) ( (PF_OP: 13) ) ) ) ) ) ) ) ) ) ) ) ) ) ) )
```

Fig. 11. Learned composite operators for 5-class with size of 61.

III. Experiments

A. Database

The database used in our experiments is the NIST Special Database 4 (NIST-4) [3]. The size of the fingerprint images is 480×512 pixels with a resolution of 500 DPI. Since fingerprints' borders do not have much useful information, we only use the 384×416 pixels around the center of fingerprints. Thus, the size of primitive feature images is 12×13 pixels. Every pixel represents the orientation value in a local 32×32 block. NIST-4 contains 2000 pairs of fingerprints. One pair of sample fingerprints is shown in Fig. 9. We use the first 1000 pairs of fingerprints for training and the second 1000 pairs of fingerprints for testing. In order to reduce the effect of overfitting, for the 1000 pairs of fingerprints in the training set, we use the first 500 pairs to estimate the parameters for each class and use the entire training set to evaluate the training results. Note that the second 500 pairs in the training set are not used in the estimation of distribution parameters for each class.

B. Experimental Results

We performed the experiments 10 times and took the best result as the learned composite operator. Fig. 10 shows the fitness values based on the number of generations in GP. Since the NIST-4 fingerprint database is a difficult database and includes many low-quality fingerprints, even in the training, the classification performance can not reach 100%. Fig. 11 shows the best

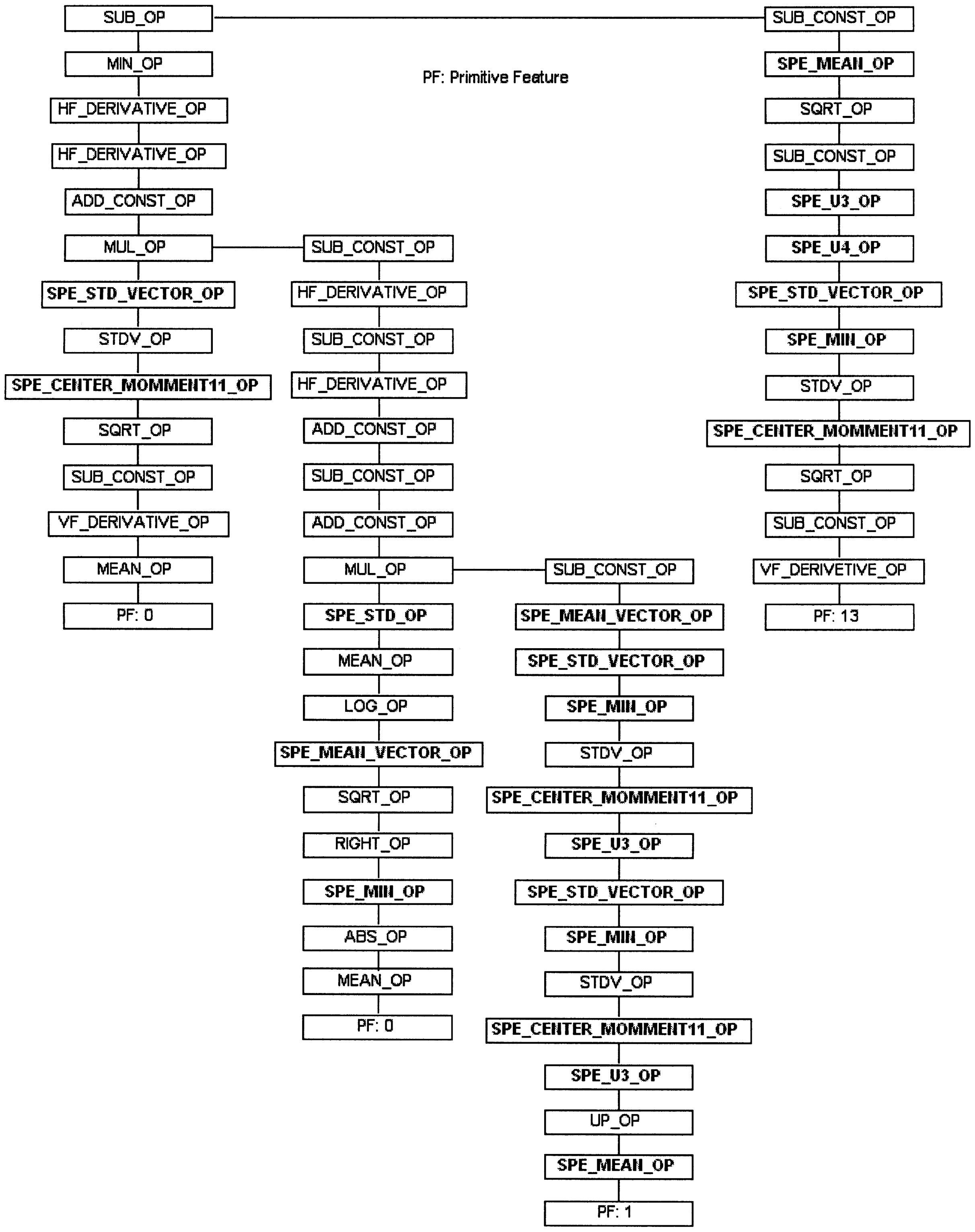


Fig. 12. Tree structure for the composite operator in Fig. 11. All these operators are defined in Table II, and the size of the tree is 61. Feature generation operators are shown in bold font and start with SPE, and all the others are computation operators. PFs are primitive features.

composite operators for 5-class classification. For 5-class, the composite operator's size is 61, out of which there are 21 feature generation operators and the length of the feature vector is 87. The tree, which represents this composite operator, is shown in Fig. 12. For 4-class classification, the composite operator's size is 149, out of which there are 23 feature generation operators and the length of the feature vector is 102. Obviously, these composite operators are not easily constructed by humans. Note that it is possible to perform feature selection to reduce the size of feature vectors by using genetic algorithms (GAs) or carrying

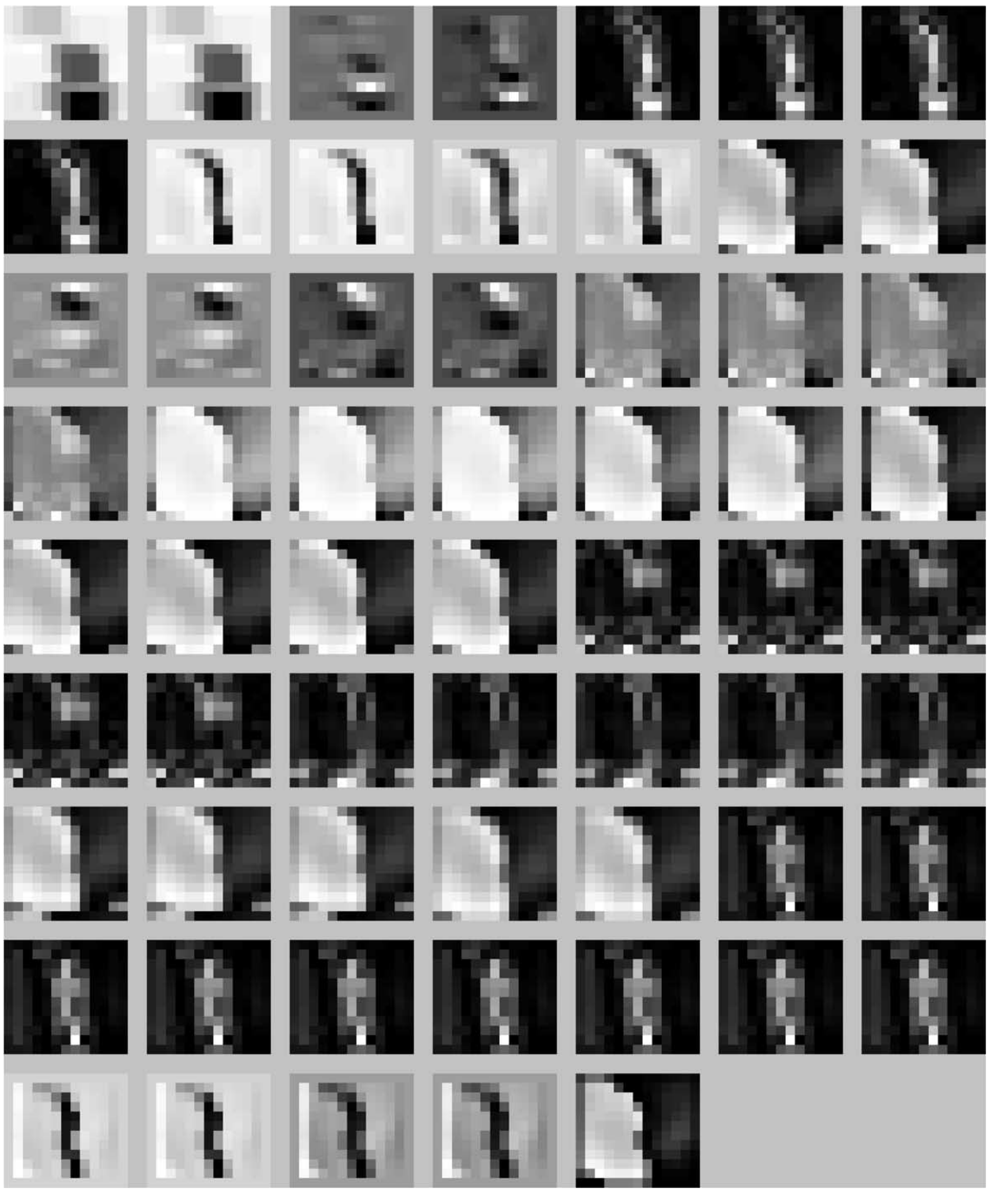


Fig. 13. Output images of each node for the composite operator in Fig. 11. The input fingerprint is s1000_07 as shown in Fig. 8. Those images are sorted from left to right and top to bottom according to the preorder traversal of the composite operator. The size of the composite operator is 61.

out an analysis like principal component analysis (PCA) or factor analysis (FA). Fig. 13 shows the output images of each node for the composite operator shown in Fig. 11. The input image is s1000_07 and is shown in Fig. 9. Note that those images are sorted from left to right and top to bottom according to the preorder traversal of the composite operator. Accordingly, the feature vector extracted by the composite operator is shown in Fig. 14. Note that feature vectors are multidimensional vectors; for simplicity, we show them as signal sequences.

During the training step, since we use GP, the experiments run slowly. Usually, it takes about 60 minutes to evolve one generation. However, once training is finished, applying a composite operator is simple and it runs fast. On a SUN Ultra II workstation with a 200 MHZ CPU, without any code optimization, the average run-times for one testing for 5-class and 4-class classification are 40 ms and 71 ms, respectively.

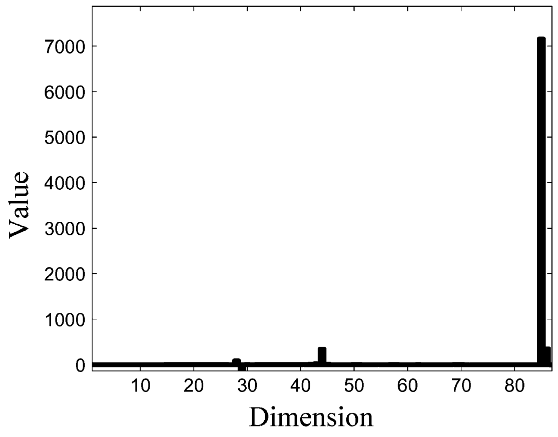


Fig. 14. Feature vectors generated by the composite operators as shown in Fig. 11 on fingerprint s1000_07.

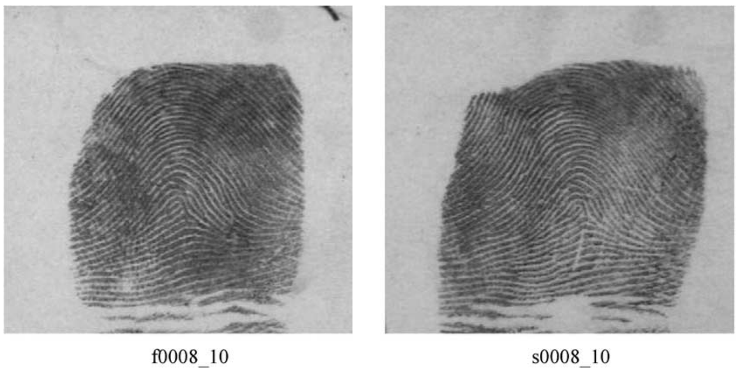


Fig. 15. Fingerprints with two ground-truth labels: T and L.

TABLE III
CONFUSION MATRIX OF THE TESTING RESULTS FOR 5- AND 4-CLASS CLASSIFICATIONS

True Class	Assigned Class				
	R	L	W	A	T
R	**356**	12	4	8	23
L	4	**372**	6	9	3
W	16	15	**369**	0	1
A	3	3	0	**416**	8
T	10	24	2	19	**337**

True Class	Assigned Class			
	R	L	W	A/T
R	**381**	11	1	20
L	4	**375**	4	12
W	11	5	**382**	3
A/T	23	40	1	**741**

Table III shows the confusion matrix of our testing results on the second 1000 pairs of fingerprints in NIST-4. Note that, because of bad quality, the ground-truths of some fingerprints provided by the NIST-4 fingerprint database contain 2 classes, i.e., the ground-truths of f0008_10 and s0008_10 include classes T and L. Fig. 15 shows these two images together. As other researchers did in their experiments, we use only the first ground-truth label to estimate the parameters of the classifier. However, in testing, we use all the ground-truth labels and consider a test as correctly classified if the output of the system matches to one of the ground-truths. However, if the output of the system does not match any one of them, then we consider it as two incorrect classifications and each of them has an entry in the confusion matrix. Note that some published research work, such as [7], has only one entry in the confusion matrix when the input fingerprint has two ground-truths and the classification result is incorrect, which inevitably reduces the error rate. Based on the confusion matrix in Table III, the PCC is 93.3% and 91.6% for 4- and 5-class classifications, respectively. Because bad quality image areas do not provide any useful information, they result in misclassifications. Some examples of misclassifications are shown in Fig. 16. Figs. 17 and 18 show all the fingerprints that are misclassified in our approach for 5-class and 4-class, respectively.

Classes R, L, W, A, and T are uniformly distributed in NIST-4. However, in nature, the frequencies of their occurrence are 31.7%, 33.8%, 27.9%, 3.7%, and 2.9%, respectively. From Table III, we observe that most of the classification errors are related to classes A and T. Considering that A and T occur less frequently in nature, our approach is expected to perform better in the real world. Table IV shows the results from the NIST-4 database reported by other researchers. Considering that we have not rejected any fingerprints from NIST-4, our results are one of the best. For the 5-class classification, our result has a 1.6% advantage over the result shown in [5], although in [5] the reject rate is 1.8%. The results reported in [7] are better than ours. One of the important reasons is that they have only one entry in the confusion matrix when the input fingerprint has two ground-truths and the classification result is incorrect. They do not report the number of incorrect classifications for which the input fingerprint has two ground-truths, so we can not compute the error rate after adjustment. In our experimental results shown in Table III, there are a total of 20 and 14 of this kind of incorrect classification for 5-class and 4-class classification, respectively. There are a total of 2020 and 2014 entries for 5-class and 4-class, respectively. If we count correct classification when any of the ground-truth classes match and count incorrect classification when none of the ground-truth classes match, the PCC for 5-class and 4-class classifications are 92.5% and 93.9%, respectively.

IV. CONCLUSION

In this paper, we proposed a learning algorithm for fingerprint classification based on GP. Our experimental results show that the primitive operators selected by us are effective and GP can find good composite operators, which are beyond humans' imagination, to extract the feature vectors for fingerprint classification. The experimental results from the NIST-4 fingerprint database show that our approach is one of the best approaches. Without rejecting any fingerprints, the experimental results show that our approach is efficient and promising, and shows one of the best results reported in the literature.

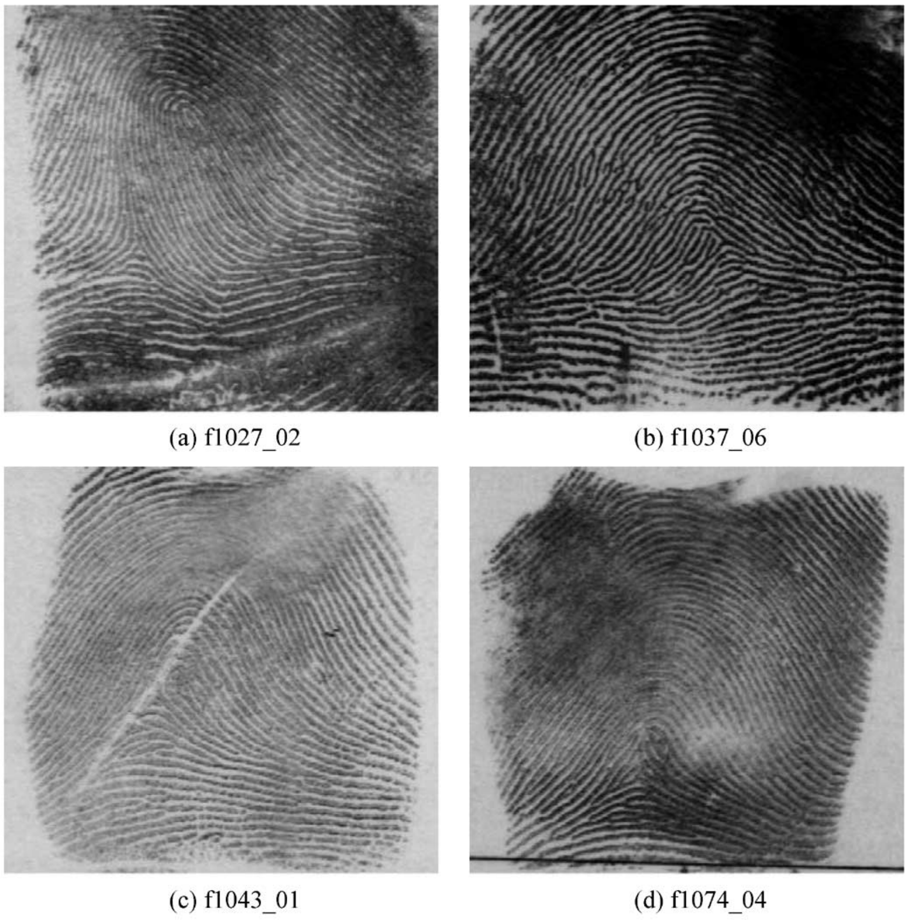


Fig. 16. Errors in classification: (a) ground-truth: R, classification: W; (b) ground-truth: T, classification: A; (c) ground-truth: R, classification: T; (d) ground-truth: T/R, classification: A. (a) fl027_02, (b) fl037_06, (c) fl043_01, (d) fl074_04.

f1027_02(2, 3), f1037_06(1, 0), f1043_01(0, 3), f1061_01(2, 3), f1070_07(1, 0), f1074_04(1, 0 & 3), f1076_09(0, 1), f1130_09(3, 4), f1131_02(4, 0), f1142_03(4, 0), f1173_08(4, 0), f1199_08(4, 0), f1202_09(2, 4), f1204_06(1, 4), f1217_05(3, 2), f1237_09(4, 2), f1259_08(4, 0), f1260_04(3, 2), f1266_08(3, 4), f1268_02(1, 0 & 3), f1269_04(4, 3), f1290_10(4, 2), f1292_07(1, 3 & 0), f1297_09(2, 4), f1347_09(2, 4), f1349_04(0, 1), f1362_01(3, 2), f1376_08(4, 2), f1401_06(4, 1), f1405_07(3, 0), f1417_02(0, 3), f1437_07(1, 0 & 4), f1438_09(4, 2), f1440_05(4, 3), f1441_02(0, 3), f1459_09(4, 2), f1462_09(3, 2), f1473_08(0, 1), f1475_09(4, 3), f1480_02(4, 0), f1492_09(0, 4), f1502_04(4, 0 & 3), f1521_02(4, 0), f1522_03(4, 2), f1527_02(3, 2), f1529_03(3, 0), f1545_03(3, 2), f1552_04(0, 3), f1572_07(1, 0), f1576_06(1, 0 & 4), f1580_09(4, 2), f1653_07(4, 0), f1659_04(0, 3), f1668_07(0, 3), f1700_07(4, 1), f1702_03(1, 0 & 3), f1706_07(4, 0), f1711_02(1, 0), f1724_01(4, 2), f1727_02(4, 0), f1745_09(1, 0 & 4), f1746_09(1, 4), f1755_04(0, 3), f1757_08(4, 3), f1774_03(2, 3), f1780_03(3, 0), f1785_10(4, 0), f1791_03(4, 0 & 3), f1802_08(4, 0), f1812_09(3, 4), f1822_03(1, 3), f1833_05(3, 2), f1852_01(3, 2), f1863_04(3, 2), f1889_04(4, 3), f1897_07(1, 0 & 4), f1902_04(0, 3), f1907_04(0, 3), f1915_07(4, 2), f1921_04(1, 3), f1930_03(3, 0), f1953_03(3, 0 & 1), f1966_07(1, 0 & 3), f1980_04(3, 2), f1998_07(0, 1), s1009_10(4, 2), s1024_02(2, 0), s1037_06(1, 0), s1057_06(4, 2), s1093_05(4, 3), s1100_09(0, 4), s1131_02(4, 0), s1172_01(3, 1), s1193_05(0, 3), s1199_08(4, 0), s1210_08(4, 0), s1217_05(3, 2), s1242_08(4, 0), s1259_08(4, 0), s1269_04(0, 3), s1278_06(4, 2), s1299_07(1, 0), s1338_03(0, 3), s1363_08(2, 0 & 3), s1371_08(4, 0 & 3), s1372_04(4, 2), s1391_10(2, 4), s1405_07(4, 0), s1441_02(0, 3), s1443_03(3, 2), s1459_09(4, 2), s1462_09(3, 2), s1486_03(1, 0 & 3), s1516_02(0, 3), s1521_02(3, 0), s1527_02(3, 2), s1528_09(2, 4), s1529_03(3, 0), s1546_08(2, 4), s1568_08(0, 1), s1583_04(0, 3), s1594_07(4, 2), s1595_08(0, 4), s1611_07(4, 0), s1625_03(3, 0), s1646_04(0, 3), s1668_07(0, 3), s1711_02(1, 0), s1712_04(4, 3), s1727_02(3, 0), s1740_07(4, 1 & 0), s1745_09(1, 0 & 4), s1746_09(1, 4), s1755_04(0, 3), s1759_09(1, 0 & 4), s1760_04(0, 3), s1772_04(0, 2 & 3), s1783_04(0, 3), s1785_10(4, 0), s1822_03(0, 3), s1850_04(4, 3), s1855_02(3, 1), s1863_04(3, 2), s1868_02(3, 0 & 4), s1873_05(4, 3), s1921_04(0, 3), s1948_05(0, 1), s1956_10(0, 1), s1957_05(3, 2), s1998_07(0, 1)

Fig. 17. Fingerprints that are misclassified in our approach for 5-class classification: Fingerprint_ID (class, ground-truth). 0, 1, 2, 3, and 4 represent classes T, A, W, R, and L, respectively. The total number of misclassification is 150. There are 20 incorrect classifications which have two ground-truths.

f1018_05(0, 2), f1024_02(2, 0), f1049_08(3, 0), f1081_05(0, 2), f1117_02(2, 0), f1129_06(3, 0), f1131_02(3, 0), f1142_03(3, 0), f1144_05(0, 2), f1172_01(2, 0), f1199_08(3, 0), f1202_09(1, 3), f1225_04(2, 1), f1241_07(3, 2), f1259_08(3, 0), f1284_05(0, 2), f1290_10(3, 1), f1316_09(2, 3 & 0) , f1347_09(0, 3), f1362_01(2, 1), f1393_01(3, 1), f1401_06(3, 0), f1405_07(3, 0), f1406_03(2, 0), f1424_10(0, 3), f1440_05(3, 2), f1462_09(2, 1), f1480_02(2, 0), f1486_03(3, 0 & 2), f1492_09(0, 3), f1494_04(2, 0), f1510_06(3, 0), f1521_02(3, 0), f1527_02(2, 1), f1529_03(2, 0), f1552_04(3, 2), f1554_10(0, 3), f1611_07(3, 0), f1623_04(0, 2), f1655_07(3, 0), f1659_04(0, 2), f1666_10(0, 3), f1668_07(0, 2), f1693_08(2, 0 & 3), f1700_07(3, 0), f1727_02(3, 0), f1741_06(1, 3), f1746_09(0, 3), f1772_04(0, 1 & 2), f1794_08(0, 2), f1807_10(0, 1), f1822_03(0, 2), f1835_02(3, 0), f1841_05(2, 0), f1863_04(2, 1), f1868_02(2, 0 & 3), f1907_04(0, 2), f1930_03(2, 0), f1957_05(2, 1), f1980_04(2, 1), f1997_04(2, 0), s1024_02(3, 0), s1025_09(0, 3), s1057_06(3, 1), s1064_06(3, 1), s1109_04(0, 2), s1129_06(3, 0), s1131_02(3, 0), s1142_03(2, 0), s1172_01(2, 0), s1175_08(0, 3), s1185_03(3, 0 & 2), s1193_05(0, 2), s1199_08(3, 0), s1206_02(1, 0 & 2), s1210_08(3, 0), s1211_10(0, 3), s1255_07(3, 0 & 2), s1259_08(3, 0), s1260_04(2, 1), s1290_10(3, 1), s1296_07(2, 0), s1346_05(0, 2), s1371_08(3, 0 & 2), s1405_07(3, 0), s1406_03(2, 0), s1427_10(3, 0), s1433_07(3, 0 & 2), s1443_03(2, 1), s1462_09(0, 1), s1480_02(3, 0), s1492_09(0, 3), s1506_02(2, 0), s1521_02(2, 0), s1527_02(2, 1), s1528_09(1, 3), s1529_03(2, 0), s1538_04(0, 2), s1549_08(3, 0), s1557_10(1, 3), s1588_07(3, 0 & 2), s1611_07(3, 0), s1659_04(0, 2), s1668_07(0, 2), s1706_07(3, 0), s1722_10(0, 3), s1727_02(2, 0), s1746_09(0, 3), s1751_08(3, 0 & 0), s1755_04(0, 2), s1769_07(3, 0), s1785_10(3, 0), s1794_08(0, 2), s1817_07(3, 0 & 2), s1862_09(2, 3 & 0), s1863_04(2, 1), s1902_04(3, 2), s1921_04(0, 2), s1923_06(3, 0), s1930_03(2, 0), s1955_08(3, 0)

Fig. 18. Fingerprints that are misclassified in our approach for 4-class classification: Fingerprint_ID (class, ground-truth). 0, 1, 2, and 3 represent classes T/A, W, R, and L, respectively. The total number of misclassification is 121. There are 14 incorrect classifications which have two ground-truths.

TABLE IV
CLASSIFICATION RESULTS FROM NIST-4

Approaches	Class #	Error rate %	Reject rate %	Dataset	Comments
Karu and Jain 1996 [2]	5	14.6	zero	4000 images, no training	Decision based on topological information
	4	8.6			
Jain and Minut 2002 [8]	4	8.7	zero	Same as above	Hierarchical kernel fitting
Jain et al. 1999 [5]	5	14.6	**1.8**	Training: First 2000 images Testing: Second 2000 images	KNN
	4	8.5			
	5	13.6			Neural Network
	4	7.9			
	5	**10.0**			KNN+NN, two stage classifier
	4	**5.2**			
Senior 2001 [9]	4	Average 8.5	zero	Same as above	Neural Network fusion with priors
Yao et al. [10]	5	10.0	1.8	Same as above	SVM+RNN
	4	5.3			
Cappelli et al. [7], see text Section III-B	5	**7.8**	zero	Same as above	Multispace KL transform
	4	**5.5**			
This paper	5	**8.4**	**zero**	Same as above	GP-based learned features + Bayesian classifier
	4	**6.7**			

REFERENCES

[1] FBI National Academy, U.S. Department of Justice, Federal Bureau of Investigation. Advanced Latent Fingerprint School., Quantico, VA, 1983.

[2] K. Karu and A. K. Jain, "Fingerprint classification," *Pattern Recognit.*, vol. 29, no. 3, pp. 389–404, Mar. 1996.

[3] G. T. Candela, P. J. Grother, C. I. Watson, R. A. Wilkinson, and C. L. Wilson, "PCASYS—A pattern-level classification automation system for fingerprints," NIST, Gaitherburg, MD, Tech. Rep. NISTIR 5647, Apr. 1995.

[4] U. Halici and G. Ongun, "Fingerprint classification through self-organizing feature maps modified to treat uncertainties," *Proc. IEEE*, vol. 84, no. 10, pp. 1497–1512, Oct. 1996.

[5] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 348–359, Apr. 1999.

[6] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, "Fingerprint classification by directional image partitioning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 402–421, May 1999.

[7] R. Cappelli, D. Maio, and D. Maltoni, "Fingerprint classification based on multi-space KL," in *Proc. Workshop Autom. Identific. Adv. Tech.*, Oct. 1999, pp. 117–120.

[8] A. K. Jain and S. Minut, "Hierarchical kernel fitting for fingerprint classification and alignment," in *Proc. Int. Conf. Pattern Recog.*, vol. 2, Aug. 2002, pp. 469–473.

[9] A. Senior, "A combination fingerprint classifier," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 10, pp. 1165–1174, Oct. 2001.

[10] Y. Yao, G. L. Marcialis, M. Pontil, P. Frasconi, and F. Roli, "Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines," *Pattern Recognit.*, vol. 36, no. 2, pp. 397–406, Feb. 2003.

[11] S. Prabhakar, A. K. Jain, J. G. Wang, S. Pankanti, and R. Bolle, "Minutia verification and classification for fingerprint matching," in *Proc. Int. Conf. Pattern Recog.*, vol. 1, Sep. 2000, pp. 25–29.

[12] B. Bhanu and X. Tan, "Learned templates for feature extraction in fingerprint images," in *Proc. IEEE Conf. Computer Vision and Pattern Recog.*, vol. 2, 2001, pp. 591–596.

[13] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs.*. Cambridge, MA: MIT Press, 1994.

[14] R. Poli, "Genetic programming for feature detection and image segmentation," in *AISB Workshop Evolutionary Computing*, T. C. Forgarty, Ed. Brighton, UK, 1996, pp. 110–125.

[15] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Proc. Evolutionary Programming VII*, 1998, pp. 735–744.

[16] D. Howard, S. C. Roberts, and R. Brankin, "Target detection in SAR imagery by genetic programming," *Adv. Eng. Soft.*, vol. 30, no. 5, pp. 303–311, May 1999.

[17] S. C. Roberts and D. Howard, "Evolution of vehicle detectors for infrared line scan imagery," in *Proc. Evolutionary Image Anal., Signal Process., and Telecommunic.*, 1999, pp. 110–125.

[18] M. Kamijo, "Classifying fingerprint images using neural network: Deriving the classification state," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 3, Apr. 1993, pp. 1932–1937.

[19] Y. Qi, J. Tian, and R. W. Dai, "Fingerprint classification system with feedback mechanism based on genetic algorithm," in *Proc. Int. Conf. Pattern Recog.*, vol. 1, Aug. 1998, pp. 163–165.

[20] F. Su, J. A. Sun, and A. Cai, "Fingerprint classification based on fractal analysis," in *Proc. Int. Conf. Signal Process.*, vol. 3, 2000, pp. 1471–1474.

[21] M. S. Pattichis, G. Panayi, A. C. Bovik, and S. P. Hsu, "Fingerprint classification using an AM-FM model," *IEEE Trans. Image Process.*, vol. 10, no. 6, pp. 951–954, Jun. 2001.

[22] S. Bernard, N. Boujemaa, D. Vitale, and C. Bricot, "Fingerprint classification using kohonen topologic map," in *Proc. Int. Conf. Image Process.*, vol. 3, 2001, pp. 230–233.

[23] S. M. Mohamed and H. O. Nyongesa, "Automatic fingerprint classification system using fuzzy neural techniques," in *Proc. IEEE Int. Conf. Fuzzy Systems*, vol. 1, May 2002, pp. 358–362.

[24] A. M. Bazen and S. H. Gerez, "Systematic methods for the computation of the directional fields and singular points of fingerprints," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 905–919, July 2002.

[25] C. I. Watson and C. L. Wilson, *NIST special database 4*, U.S. National Institute of Standards and Technology, 1992.

Bir Bhanu (S'72–M'82–SM'87–F'95) received the S.M. and E.E. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, the M.B.A. degree from the University of California, Irvine, and the Ph.D. degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles.

Currently, he is the Director of the Center for Research in Intelligent Systems at the University of California, Riverside, where he has been a professor and Director of Visualization and Intelligent Systems Laboratory since 1991. Previously, he was a Senior Honeywell Fellow at Honeywell, Inc., in Minneapolis, MN. He has been on the faculty of the Department of Computer Science at the University of Utah, Salt Lake City, and has worked at Ford Aerospace and Communications Corporation, CA, INRIA-France, and IBM San Jose Research Laboratory, CA. He has been the principal investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO, and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, biometrics, navigation, image databases, and machine vision applications. He is the coauthor of several books and many published articles. He has received two outstanding paper awards from the Pattern Recognition Society and industrial and university awards. He has been on the editorial board of various journals and has edited special issues of several IEEE transactions and other journals. He holds 11 U.S. and international patents. He has been General Chair for IEEE Workshops on Applications of Computer Vision, Chair for the DARPA Image Understanding Workshop, General Chair for the IEEE Conference on Computer Vision and Pattern Recognition, and Program Chair for the IEEE Workshops on Computer Vision Beyond the Visible Spectrum.

Dr. Bhanu is a Fellow of the American Association for the Advancement of Science, International Association of Pattern Recognition, and The International Society for Optical Engineering.

Xuejun Tan received the B.S. degree in automation from Tian Jin University, Tian Jin, China, in 1995, the M.S. degree in pattern recognition and artificial intelligence from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1998, and the Ph.D. degree in electrical engineering from the University of California, Riverside, in 2003.

Dr. Tan's research interests include biometrics, image processing, pattern recognition, and machine learning.

Yingqiang Lin received the B.S. and M.S. degrees in computer science from Fudan University, Shanghai, China, in 1991 and 1994, respectively, and the Ph.D. degree in computer science from the University of California, Riverside, in June 2003.

Dr. Lin's research interests include image processing, pattern recognition and machine learning.