Extraction and Optimization of B-Spline PBD Templates for Recognition of Connected Handwritten Digit Strings

Zhongkang Lu, *Member, IEEE*,
Zheru Chi, *Member, IEEE*, and
Wan-Chi Siu, *Senior Member, IEEE*

Abstract—Recognition of connected handwritten digit strings is a challenging task due mainly to two problems: poor character segmentation and unreliable isolated character recognition. In this paper, we first present a rational B-spline representation of digit templates based on Pixel-to-Boundary Distance (PBD) maps. We then present a neural network approach to extract B-spline PBD templates and an evolutionary algorithm to optimize these templates. In total, 1,000 templates (100 templates for each of 10 classes) were extracted from and optimized on 10,426 training samples from the NIST Special Database 3. By using these templates, a nearest neighbor classifier can successfully reject 90.7 percent of nondigit patterns while achieving a 96.4 percent correct classification of isolated test digits. When our classifier is applied to the recognition of 4,958 connected handwritten digit strings (4,555 2-digit, 355 3-digit, and 48 4-digit strings) from the NIST Special Database 3 with a dynamic programming approach, it has a correct classification rate of 82.4 percent with a rejection rate of as low as 0.85 percent. Our classifier compares favorably in terms of correct classification rate and robustness with other classifiers that are tested.

Index Terms—Connected handwritten digit recognition, pixel-to-boundary distance map, B-spline fitting, digit templates, template optimization, nearest neighbor classifier, multilayer perceptron classifier, evolutionary algorithm.

1 Introduction

Recognition of connected handwritten character strings is a challenging task due mainly to two problems: poor character segmentation and unreliable isolated character recognition. Character segmentation is often ambiguous, error prone, and subject to noncharacter patterns produced by incorrect segmentation. Many handwritten digit classifiers can produce a high rate of correct classification on well-isolated digits. However, the classifiers may produce rather poor results on connected handwritten digit strings because they cannot reliably reject noncharacter patterns. The techniques for recognizing connected digit strings can be categorized into two classes: segmentation-based algorithms [1], [2], [3] and segmentation-free algorithms [4]. Recently, techniques that combine the above two approaches have been proposed for achieving more reliable performance [5]. In a segmentation-based approach, a segmentation step is carried out before recognition is performed. In a segmentation-free approach, however, the segmentation and recognition are tightly coupled. Segmentation-free techniques have been attracting increasing attention because 1) character segmentation is error-prone and unreliable and 2) lexical information can be used to improve recognition [6]. However, little lexical information is available in the recognition of connected digit strings. A segmentation-based approach appears to be a better choice for digit string recognition. Based on the experience gained from previous research [3], [7], it is reasonable to conclude that a reliable and robust classifier that can distinguish true digits from nondigit patterns is of the utmost importance to handwritten digit string recognition.

- *Z. Lu is with the School of Electronics and Electrical Engineering, Nanyang Technological University, Singapore 639798. E-mail: ezklu@ntu.edu.sg.*
- *Z. Chi and W.-C. Siu are with the Center for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. E-mail: {enzheru, enwcsiu}@polyu.edu.hk.*

Manuscript received 13 Sept. 1999; revised 23 Oct. 2000; accepted 23 Apr. 2001.
Recommended for acceptance by A. Kundu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 110575.

Many new techniques have been presented in recent years for the recognition of isolated digits. They differ in the feature extraction and classification methods employed. Two comprehensive reviews were given by Govindan and Shivaprasad [8] and Impedovo et al. [9]. Trier et al. also provided an exhaustive survey of the feature extraction methods for offline recognition of isolated digits/characters [10].

A number of studies have applied template-based techniques to isolated digit recognition. The template-matching-based technique is one of the earliest pattern recognition techniques for digit recognition. The technique has come to the attention of many researchers in recent years with the development of advanced computational algorithms such as artificial neural networks and evolutionary algorithms. Yan proposed an Optimized Nearest-Neighbor Classifier (ONNC) for the recognition of hand-printed digits [11]. Wakahara used iterated Local Affine Transformation (LAT) operations to deform binary images to match templates [12]. Cheung et al. proposed a template representation based on splines [13] which modeled a digit image using a spline, assuming that the spline parameters had multivariate Gaussian distributions. Jain and Zongker presented a deformable templates matching algorithm based on object contours [14]. In their approach, the recognition procedure minimized an objective function by iteratively updating the transformation parameters to alter the shapes of the templates so that the best match between the templates and unknown digits is determined. The recognition performance of deformable template matching depends very much on the ability of the "deformations" in covering the variations of objects in the real world.

The objective of our research is to develop a robust classifier that recognizes the true digits of a large number of variations and reliably rejects nondigit patterns. These capabilities are highly desirable for connected handwritten digit string recognition. In this paper, we will present a B-spline template representation scheme, a neural network approach for extracting templates, and an evolutionary algorithm for template optimization. In our approach, each template represented by a Pixel-to-Boundary Distance (PBD) map is approximated by a rational B-spline surface with a set of control knots. In the training, a cost function that takes into account both the amplitude and gradient of the PBD map is first minimized by a neural network to extract a set of templates. The extracted templates are further optimized by an evolutionary algorithm. At the matching stage, a similarity measure that takes into account both the amplitude and gradient of the PBD map is adopted.

The rest of this paper is organized as follows: The representation of B-spline templates is discussed in Section 2. Section 3 discusses templates extraction by using a neural network. Templates optimization based on an evolutionary algorithm is presented in Section 4. Experimental results are reported in Section 5. Finally, conclusions are made in Section 6.

2 B-Spline Representation of Templates

Curve and surface representation and manipulation, utilizing nonrational or rational B-splines, are widely used in geometric design. In 1975, Versprille proposed a rational B-splines for the geometric design of curves and surfaces [15]. The work outlined the important properties of rational B-splines, such as continuity,

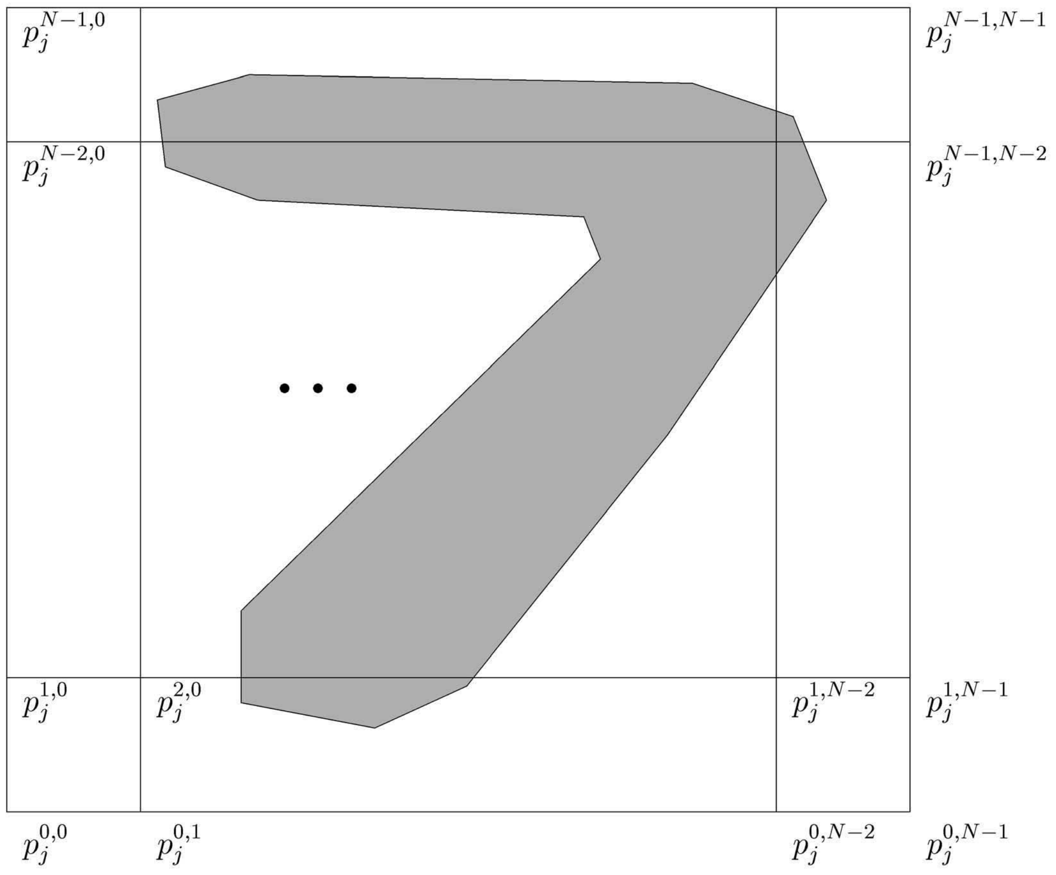

Fig. 1. Control points of a B-spline digit template.

local control property, etc. Recently, B-spline curves (snakes) have been used to describe digit images [13], [16]. With B-spline fitting, information concerning the shapes of the desired objects can be incorporated into the control parameters of the curve or surface-based templates. However, B-spline fitting of a digit image involves some uncertainties and, therefore, is prone to error. This is because the input data and corresponding parameters of a template should reflect the information in the same aspect of the image. In our approach, we use a B-spline surface to fit the Pixel-to-Boundary Distance (PBD) map of a digit image. The number of control points of the surface and their locations are predetermined; hence, for each control point, the control area is certain. Therefore, the problem can be partially solved.

2.1 Representation of Binary Digit Images

A binary digit image can be converted to a PBD map. The distance of pixel (x, y) to the nearest foreground/background boundary is measured. We use (x, y) instead of normally discrete (i, j) for its consistency with the definition of the rational B-spline surface. The foreground pixel O_{fg} has $d(x, y) \geq 0$, but, for the background pixel O_{bg}, we assume it has a negative value, that is, $d(x, y) < 0$.

Let $g(x, y)$ denote the value of pixel (x, y) in the PBD map. It is defined as

$$g(x, y) = \frac{e}{2} e^{-\frac{[(d(x,y)-d_m)]^2}{d_m^2}}, \tag{1}$$

where d_m is the maximum pixel-to-boundary distance in the image and $(e/2)$ is a constant to make a point on the boundary O_b have $g(x_{O_b}, y_{O_b}) = 0.5$.

Assume that the rational B-spline surfaces for templates are $\{S(\mathbf{P}_j), 0 \leq j \leq N_h\}$, where N_h is the number of templates and $\mathbf{P}_j$ is an $N \times N$ matrix of the control points (parameters) for template j, as shown in Fig. 1. $\mathbf{P}_j$ is given by

$$\mathbf{P}_j = \begin{pmatrix} p_j^{0,0} & p_j^{0,1} & \cdots & p_j^{0,N-1} \\ p_j^{1,0} & p_j^{1,1} & \cdots & p_j^{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_j^{N-1,0} & p_j^{N-1,1} & \cdots & p_j^{N-1,N-1} \end{pmatrix}_{N \times N.} \tag{2}$$

A rational B-spline surface is defined as

$$S_j(x, y) = \mathbf{B}_{r_1}^t(x)\mathbf{P}_j\mathbf{B}_{r_2}(y), \tag{3}$$

where $\mathbf{B}_r(x)$ is a base function vector and $\mathbf{B}_r^t(y)$ is the transpose of $\mathbf{B}_r(y)$. The parameters r_1 and r_2 are the orders of the B-spline bases. Normally, we have $r_1 = r_2 = r$. $\mathbf{B}_r(x)$ is defined as

$$\mathbf{B}_r(x) = \begin{pmatrix} B_{0,r}(x) \\ B_{1,r}(x) \\ \cdots \\ B_{N-1,r}(x) \end{pmatrix}_{N \times 1}, \tag{4}$$

where $B_{i,1}(u)$ is given by

$$B_{i,1}(u) = \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and

$$B_{i,r}(u) = \frac{u - u_i}{u_{i+r-1} - u_i} B_{i,r-1}(u) + \frac{u_{i+r} - u}{u_{i+r} - u_{i+1}} B_{i+1,r-1}(u), \tag{6}$$

where $U = \{u_i | 0 \leq i \leq N_u\}$ is a knot vector of length N_u. It is assumed that $0/0 = 0$. The value of u_i determines the location of

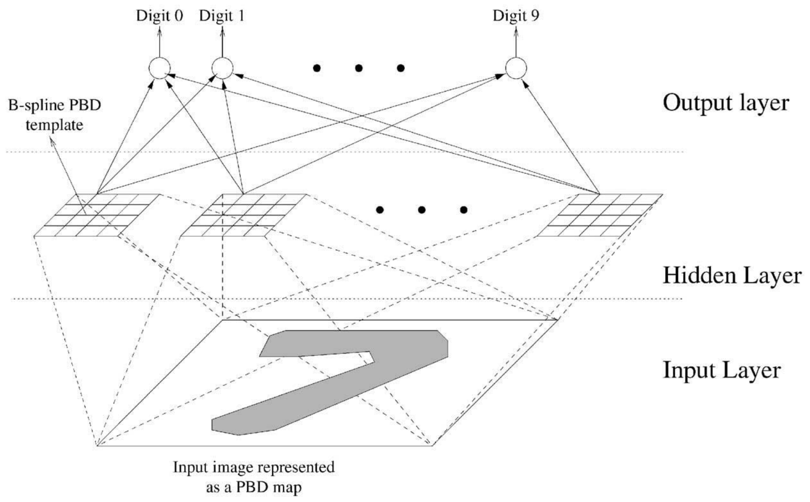

Fig. 2. Template extraction using a multiplayer feedforward neural network.

the control knots and must be predefined. To simplify the B-spline surface, we define a knot vector as

$$U = \{\overbrace{0, \cdots, 0}^{r}, \frac{1}{(N-3)}, \frac{2}{(N-3)}, \cdots, \overbrace{1, \cdots, 1}^{r}\},$$

so $N_u = N + 2 \times (r-2)$. In most applications, the number of input patterns, N_i, is much greater than the number of control points in a template, $N \times N$.

3 Template Extraction by Training a Neural Network

B-spline templates are extracted by training a multilayer feedforward neural network, as shown in Fig. 2. The input to the neural network is a Pixel-to-Boundary Distance (PBD) map $G = \{g(x,y), 0 \le x \le 1,\ 0 \le y \le 1\}$. Note that both x and y have been normalized to the range of $[0,1]$. The hidden layer consists of templates to be extracted. Each hidden node corresponds to a template. $Z = [z_0, z_1, \cdots, z_{N_o}]$ is the output vector of the network, corresponding to N_o output classes. The output of the jth hidden node is given by

$$\phi_j = f(G, \mathbf{P}_j) = \gamma_1 \phi_{1,j} + \gamma_2 \phi_{2,j}, \tag{7}$$

where γ_1 and γ_2 are two weighting factors with $\gamma_1 + \gamma_2 = 1$. Function ϕ_j measures the similarity between an input PBD map G and the jth template $\mathbf{P}_j$, and considers not only the magnitude but also the gradient at a point. $\phi_{1,j}$ is defined as

$$\phi_{1,j} = \frac{2}{1 + e^{c_s v_j}}, \tag{8}$$

where c_s is a smoothing factor, and v_j is defined as

$$\begin{aligned} v_j &= \frac{\int_0^1 \int_0^1 [S_j(x,y) - g(x,y)]^2\, dx\, dy}{\int_0^1 \int_0^1 g^2(x,y)\, dx\, dy} \\ &= \frac{\int_0^1 \int_0^1 [\mathbf{B}_r^t(x) \mathbf{P}_j \mathbf{B}_r(y) - g(x,y)]^2\, dx\, dy}{\int_0^1 \int_0^1 g^2(x,y)\, dx\, dy}. \end{aligned} \tag{9}$$

Denote $\int_0^1 \int_0^1 g^2(x,y)\, dx\, dy$ by g_s^2, then

$$v_j = \frac{1}{g_s^2} \int_0^1 \int_0^1 [\mathbf{B}_r^t(x) \mathbf{P}_j \mathbf{B}_r(y) - g(x,y)]^2\, dx\, dy. \tag{10}$$

We defined $\phi_{2,j}$ as

$$\phi_{2,j} = \int_0^1 \int_0^1 \cos^2\big(\beta_j(x,y)\big)\, dx\, dy, \tag{11}$$

where $\beta_j(x,y)$ is the angle between the gradient of the jth template S_j at (x,y) and the gradient of the input PBD map G at the same point. It is easy to verify that $0 \le \phi_{1,j} \le 1$ and $0 \le \phi_{2,j} \le 1$. If $\phi_{1,j} = 1$, then $S_j(x,y) \equiv g(x,y)$.

The connection weight from hidden node j to output node m is denoted w_{mj}. The output of the mth output node is given by

$$z_m = \sum_{j=1}^{N_h} w_{mj} f(G, \mathbf{P}_j), \tag{12}$$

where N_h is the number of templates. Here, weights $\{w_{mj}\}$ are predefined, which will be discussed later.

Assume that the desired and actual activities of the output node m are z_m^t and z_m, respectively, for the input $G = \{g(x,y)\}$. For each j, we have to find $p_j^{k,l}$ (the control parameters for the B-spline templates) so that the following energy function is minimized:

$$E = \frac{1}{2} \sum_{m=1}^{N_o} (z_m - z_m^t)^2. \tag{13}$$

The generalized delta rule is adopted to train the multilayer perceptron for extracting a set of templates.

In order to simplify the extraction algorithm, at iteration t of the template extraction procedure, only the template(s) with maximum ϕ are updated, instead of updating all templates. So, w_{mj} is defined as

$$w_{mj} = \begin{cases} 1 & j = \arg\max_i(\phi_i),\ \ i \in D_m \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

where D_m is a set in which each index corresponds to a hidden node that represents a template belonging to the class represented by output node m. Further, the input PBD map belongs to the class m.

4 Template Optimization by Evolutionary Algorithm

Evolutionary algorithms for optimization have been studied for more than three decades. Many research results and applications have demonstrated that evolutionary algorithms, which emulate

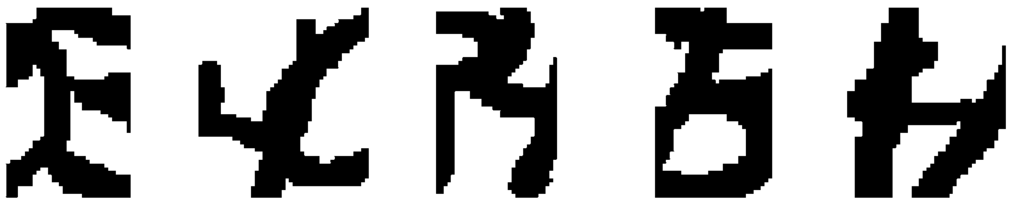
Fig. 3. Examples of nondigit patterns.

the process of natural evolution, are powerful tools for global optimization [17], [18].

Several applications of evolutionary template optimization have been reported in [19], [20]. However, in most of these algorithms, only the best template is extracted for each class from one set of training samples. Obviously, in most object recognition problems, a single template for a class of objects is not sufficient to achieve reliable recognition. More templates are required in order to achieve better performance. Sarkar et al. presented a *fitness*-based clustering algorithm, which can be utilized for template optimization [21]. In the algorithm, one opponent (parents or offspring) contains a variable number of clustering centers. In its selection procedure, the fitness values of the parents and offspring are compared, and a number of opponents are selected arbitrarily as reference opponents. The opponent that has a better performance with respect to each reference opponent receives a *win*. Based on the number of *wins*, opponents are selected as the parents of the next generation. One disadvantage of this algorithm is that the computational requirement has an exponential relationship with the cluster number. If the number of clusters is very large, the algorithm ceases to be feasible.

Unlike Sarkar et al.'s approach of using a set of cluster centers as a component in evolution and selecting only one set as the winner [21], we use templates directly as the components, and the selected survivors are a group of templates. The evolutionary process emulates the evolution of a social system, such as ants, bees, and human society. In our algorithm, we define a small, simple, and homogeneous social system. Each template is like an individual in the system. The society can only accommodate a limited number of individuals, but the individuals can generate an increasing number of offspring in each generation. Therefore, only the capable offspring are kept and the others are discarded, that is, "survival of the fittest." Moreover, the relationship between individuals is not only competitive (only winners can survive), but also cooperative (the properties of individuals are reflected by their collective performance). After a number of generations in the process of evolution, a group of templates is selected. These templates, as a whole, can achieve a good recognition performance. The procedure is given below:

1. A population of N_P templates (called parents) is initially generated. We use the templates extracted by the multilayer feedforward neural network presented in Section 3.
2. $N_O = N_O^r + N_O^m + N_O^c$ offspring are generated from the parents by evolutionary operations which include replication ($N_O^r = N_P$), mutation ($N_O^m = nN_P$, where n is a positive integer), and recombination (N_O^c).
3. A subset of N_P offspring is selected as the parents of the next generation based on a fitness measure given in (15).
4. If the number of generations is less than the preset number or the recognition performance of the selected templates is not satisfactory, go to Step 2.

In our algorithm, the templates in each digit class are optimized independently. The parallel processing nature of our proposed method allows fast templates optimization on parallel computers or multiple computers.

The task of the selection procedure is to select a subset NPS of N_P templates from N_O generated offsprings for the parents of the next generation. Based on the similarity function $\phi_k = f(G_j, \mathbf{P}_k)$ (G_j is the jth input PBD map) defined in (7), we define the fitness of NPS as

$$\mathrm{fitness}(NPS) = \sum_{j=1}^{N_t} \max_{P_k \in NPS} f(G_j, \mathbf{P}_k), \tag{15}$$

where N_t is the number of training samples. The fitness (NPS) of the selected subset must be greater than the fitness of any other subset of N_P templates from the offspring set OS. The number of possible subsets of N_P templates from N_O offspring templates is $C_{N_O}^{N_P}$, which is an extremely large number if we set $N_P = 100$ and $N_O = 1,000$. In order to make this algorithm practical, a fast selection procedure directly based on the function $\phi_k = f(G_j, \mathbf{P}_k)$ is adopted. The computing time of our fast selection procedure is proportional to the number of templates, which is an improvement over the existing methods in which the computational requirement has an exponential relationship with the cluster number.

5 Experimental Results and Discussion

5.1 Experimental Settings

In total, 10,426 digit samples from the NIST Special Database 3 were extracted as Training Set 1 (TS1). Each digit was normalized into a 48×48 binary image with 8-pixel-wide background borders for constructing the Pixel-to-Boundary Distance (PBD) map. Hence, the actual image size was 64×64. For the purpose of comparison with other classifiers, training set TS1 and 2,000 nondigit patterns generated from the digit samples of TS1 (see the examples shown in Fig. 3) formed Training Set 2 (TS2). The nondigit patterns used in the experiment were generated by merging two parts of the isolated digits, the left part of the right digit and the right part of the left digit. The selection of the two digits, the width and the relative y position of each part, and the overlapping degree were set randomly. It appeared that, with one more output for nondigit patterns, the rejection performance of MLP could be improved. Another set of independent 10,426 digit patterns was extracted from the NIST Special Database 3 as the test set.

Preliminary experimental results with different γ (7) values showed that the best performance was achieved when $\gamma_1 = \gamma_2 = 0.5$. In all reported experiments in this paper, $\gamma_1 = \gamma_2 = 0.5$. One thousand templates (100 for each digit class) were first extracted from the training set by the neural network approach discussed in Section 3. Each template contained 11×11 control points. Since the amplitude of a pixel on four borders of a PBD map is close to zero, the control points on four borders were set to zero. Therefore, only the inner $9 \times 9 = 81$ control points were involved in the training. With these 1,000 extracted templates, a nearest neighbor classifier was adopted to classify the input patterns based on the ϕ function defined in (7).

The $1,000$ extracted templates were then used as the initial parents of our evolutionary optimization algorithm to obtain $1,000$ optimized templates. Because each digit class has 100 templates, $N_P = 100$ and $N_O^r = 100$ in the optimization of templates for each

TABLE 1
Experimental Settings

Experiment	Technique	Training set	Feature set	Classifier parameters
BSTNN	NNC	TS1	B-spline of PBDs	1,000 templates
BSTEC	NNC	TS1	B-spline of PBDs	1,000 templates
MLP1	MLP	TS1	FS1	64-75-10
MLP2	MLP	TS1	FS2	81-80-10
MLP3	MLP	TS1	FS3	145-100-10
MLP4	MLP	TS2	FS1	64-75-11
MLP5	MLP	TS2	FS2	81-80-11
MLP6	MLP	TS2	FS3	145-100-11
ONNC1	NNC	TS1	FS1	1,000 templates
ONNC2	NNC	TS1	FS2	1,000 templates

BSTNN: B-spline Templates extracted from training a neural network; BSTEC: Optimized templates from evolutionary computation.

class. We set $N_O^m = 2N_P = 200$ and $N_O^c = 7N_P = 700$. As a result, $N_O = 1,000$.

For comparison purposes, we applied two other techniques, namely, a three-layer Multilayer Perceptron (MLP) classifier and Optimized Nearest-Neighbor Classifier (ONNC) proposed by Yan [11] to recognize the same digit samples with different feature sets. Feature Set 1 (FS1) was defined as 64 intensities on the 8×8 image, rescaled from the original binary image. The 9×9 intensities extracted from the PBD maps were used as Feature Set 2 (FS2). Sixty-four directional features extracted from the PBD maps, together with the 81 intensities in FS2 (a total of 145 features), formed Feature Set 3 (FS3). Directional features were extracted by using four directional Kirsch masks (horizontal, vertical, right-diagonal, and left-diagonal). Each of the directional feature vectors were compressed into a 4×4 normalized feature vector. The experimental settings with different techniques, training sets, feature sets, and classification parameters are summarized in Table 1.

5.2 Recognition Performance on Isolated Digits

Fig. 4 shows the Figure-of-Merit (FOM) measures of our classifiers BSTNN (using the B-spline PBD templates from training a neural network) and BSTEC (using the B-spline PBD templates from evolutionary computation) and classifiers MLP5 (the best among MLP1 - MLP6) and ONNC2 (the better one of ONNC1 and ONNC2). The FOM is defined as

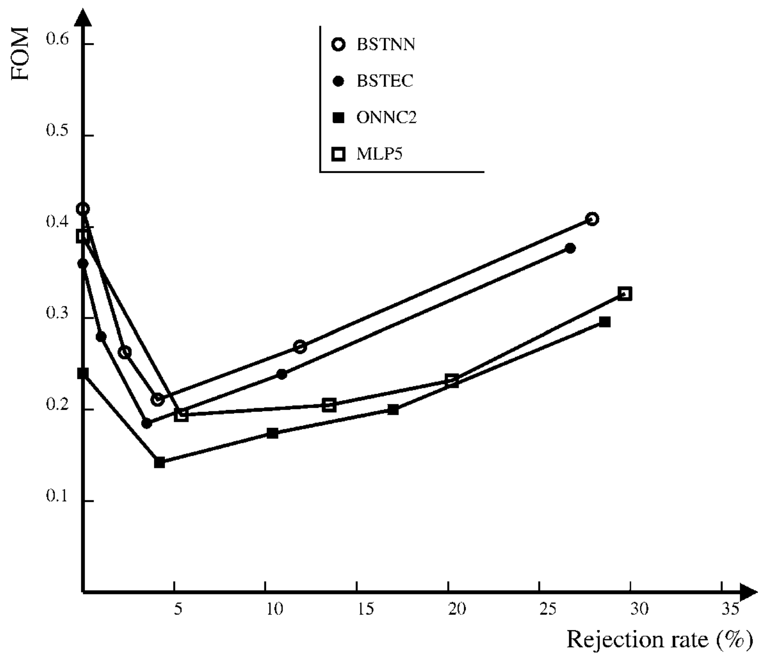

Fig. 4. A performance comparison of our classifiers BSTNN and BSTEC with classifiers MLP5 and ONNC2 in terms of the Figure-of-Merit (FOM) defined in (16).

TABLE 2
A Performance Comparison of Our Classifiers BSTNN and BSTEC with Classifiers MLP5 and ONNC2

Experiment	With rejection		Without rejection	
	Rejection (%)	Correct (%)	Rejection (%)	Correct (%)
BSTNN	27.9	98.7	0.0	95.8
BSTEC	26.7	98.9	0.0	96.4
MLP5	29.7	99.7	0.0	96.1
ONNC2	28.6	99.9	0.0	97.6

$$\mathrm{FOM} = 10\mathrm{E} + \mathrm{R}, \tag{15}$$

where E is an error rate and R is a rejection rate. By using such a definition, we weigh the error rate more than the rejection rate. Experimental results show that there is little difference among the FOMs of MLP1, MLP2, and MLP3 classifiers that use the same training set TS1 but different feature sets (FS1, FS2, and FS3, respectively). The same is true for MLP4, MLP5, and MLP6 classifiers that use the same training set TS2 but different feature sets (FS1, FS2, and FS3, respectively). This suggests that 64 intensity-based features contain sufficient information required for the classification. Adding more features does not appear helpful. This also suggests that directly using PBD maps as features contributes little to the performance enhancement of the system. From the results, we do observe that the FOM performance of an MLP classifier trained by using both digit and nondigit patterns (training set TS2) is better than that of an MLP classifier trained by using digit patterns only (training test TS1), particularly in more conservative cases of high rejection rates. This also suggests that including negative examples (nondigit patterns in our case) is important for training an MLP classifier. Our classifiers can do slightly better than an MLP classifier at a low rejection rate, although their performance is slightly lower than MLP classifiers at a high rejection rate. Yan's ONNC classifiers perform slightly better than both MLP classifiers and our classifiers in the recognition of isolated digits.

Table 2 shows the experimental results of different classifiers at a high rejection rate or without rejection. As expected, the performance with the templates optimized by the evolutionary algorithm (BSTEC) is better than that without evolutionary optimization (BSTNN: the templates extracted by training a neural network). This can also be observed from the FOM measures shown in Fig. 4. As we know, it is easy for neural network training to get stuck to a local minimum, in particular, for a complex algorithm to train B-spline templates (dealing with both intensity and directional features) in our case. It is believed that evolutionary algorithms have a better chance of achieving global optimization. This may explain why BSTEC achieves better results than BSTNN.

5.3 Rejection Capability on Nondigit Patterns

To further compare these classifiers, we use another set of 10,426 nondigit patterns to verify the reliability of the classifiers in rejecting nondigit patterns. This capability is of utmost importance for connected character recognition. The 2,000 nondigit patterns used in TS2 are not included in this data set. Table 3 shows the experimental results of our classifier together with an MLP classifier, MLP5 (the best among MLP classifiers), and Yan's ONNC2 (the better one of ONNC1 and ONNC2). The thresholds used here are the same as the settings used in Table 2 for the "with rejection" experiments on isolated digits.

We can see that the rejection rate of using evolutionally optimized B-splice PBD templates (BSTEC) is the highest on nondigit patterns. In other words, BSTEC can achieve a more reliable performance in rejecting nondigit patterns, which is a very desirable property for connected handwritten digit recognition. Two factors contributed to the enhanced performance: 1) a new B-spline representation of PBD templates based on PBD maps and 2) template extraction and optimization by using a neural network and an evolutionary algorithm. We can also see from the experimental results that BSTEC can achieve a better performance than BSTNN (using the templates from training a neural network).

In a template-based approach, the distance from an unknown pattern to a cluster center more precisely reflects the similarity between the pattern and the cluster center. Template matching is also more predictable than a neural network model. Normally, a nondigit pattern will have a larger distance than a digit pattern. The training of an MLP classifier decides the boundaries among different classes in the feature space. To better train an MLP classifier, we need to have all the representative patterns in the training set. In our case, it is difficult, if not impossible, to include all the nondigit patterns in the training set. Even if we could collect all the nondigit patterns, the nondigit set would be much larger than the digit set and, therefore, it would be an overwhelming training to nondigit patterns. As a result, the classifier might be very good in rejecting nondigit patterns but it may also reject some true digit patterns. Unlike an MLP classifier, we used a neural network to extract initial B-spline PBD templates and an evolutionary algorithm to further optimize the templates. In our B-spline templates, both intensity and gradient (directional) features are used and that is why we use continuous PBD maps to model digit images. This approach to improving the capability of rejecting nondigit patterns is used for the following reasons:

TABLE 3
Experimental Results on Nondigit Patterns with Different Classifiers

Experiment	Rejection rate (%)
BSTNN	87.5
BSTEC	90.7
MLP5	63.0
ONNC2	68.7

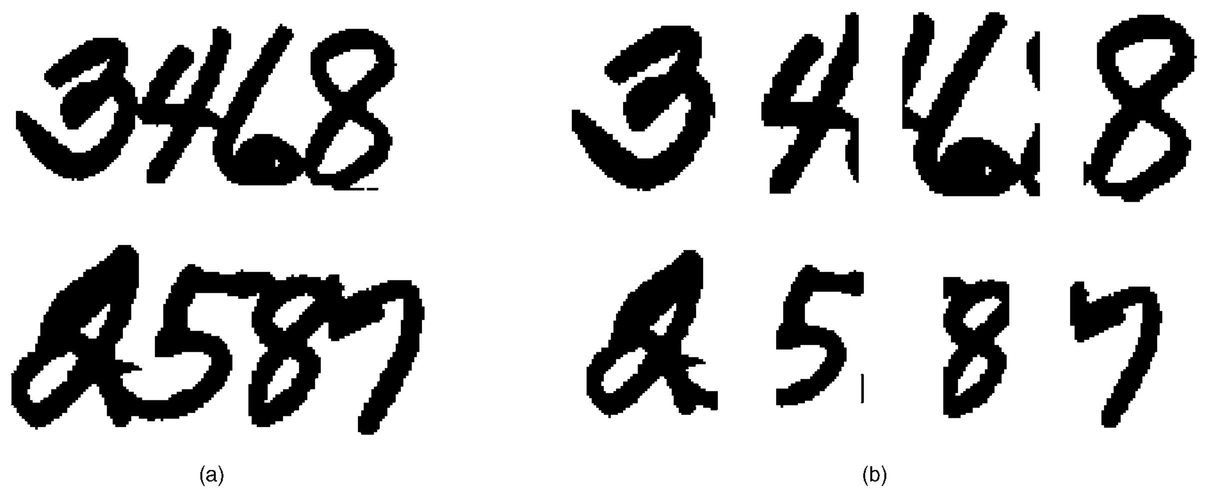

Fig. 5. Examples of correctly recognized digit strings: (a) original connected digit strings and (b) digits after separation.

1. Because of the poor rejecting capability of an MLP classifier, we adopt a template matching technique.
2. In order to deal with directional features, we use B-spline templates.
3. Since a B-spline function is continuous, we use Pixel-to-Boundary Distance (PBD) maps to transfer a binary digit or nondigit image to a gray-level PBD map.

5.4 Segmentation-Free Recognition of Connected Handwritten Digit Strings

To further verify the performance of these classifiers on connected digit string recognition, we also apply them to the recognition of 4,958 connected digit strings (4,555 2-digit, 355 3-digit, and 48 4-digit strings from the NIST Special Database 3) with a dynamic-programming approach. The main idea of the dynamic-programming approach adopted here is to apply the classifier on a rectangle window of a variable width (nine different widths proportional to the height of the digit string are applied) sliding along a digit string image. Let O_i denote the recognition score of the ith segment in a composition set (a possible set of digits in the string), where there are a total of L segments. The confidence of the composition set, CF, is defined as

$$CF = \sqrt[L]{\prod_{i=1}^{L} O_i}. \quad (17)$$

Because each recognition is given a score in our approach, the postprocessing of recognized strings using nonstationary Markovian models presented by Bouchaffra et al. [22] can be performed for improved performance if certain context information is available in digit strings.

TABLE 4
A Comparison of Different Classifiers on the Segmentation-Free Recognition of Connected Handwritten Digit Strings

Classifier	Rejection rate (%)	Recognition rate (%)
BSTEC	0.85	82.4
MLP5	7.73	70.2
ONNC2	3.65	76.9

Fig. 5 provides examples of correctly recognized digit strings. The one on the left is the original connected digit string and the one on the right is the separated and correctly recognized digits. Table 4 shows the recognition performance of our classifier BSTEC with the other two classifiers when a segmentation-free approach based on a dynamic programming algorithm is adopted. Our classifier can achieve a correct classification rate of 82.4 percent with a rejection rate of as low as 0.85 percent. Our classifier compares favorably with two other classifiers (a classification rate of 76.9 percent with a rejection rate of 3.65 percent for Yan's ONNC2 and a classification rate of 70.2 percent with a rejection rate of 7.73 percent for the MLP5 classifier).

6 Conclusion

In this paper, we first propose a rational B-spline representation of handwritten digit templates based on Pixel-to-Boundary Distance (PBD) maps. The new representation aims at building a classifier that can reliably reject nondigit patterns while achieving a high recognition rate on true digits. We then present a two-step approach for extracting and optimizing templates using this new template representation. In the first step, a multilayer feedforward neural network is adopted to extract templates from training samples. An evolutionary algorithm is then applied in the second step to optimize the extracted templates. In applying the evolutionary algorithm, we directly used a function for offspring selection that measures the difference between a template and a training sample. Experimental results on the NIST Special Database 3 show that the evolutionally optimized templates can achieve a better performance than those without evolutionary optimization. When compared with an MLP classifier and Yan's Optimized Nearest Neighbor Classifier (ONNC), a nearest neighbor classifier using the templates generated by our approach can perform more reliably in rejecting nondigit patterns and therefore achieves a better performance in the recognition of connected handwritten digit strings.

Acknowledgments

The work described in this paper was partially supported by two research grants from the Hong Kong Polytechnic University (Project Codes A408 and G-T048).

REFERENCES

[1] M. Shridhar and A. Badreldin, "Context-Directed Segmentation Algorithm for Handwritten Numeral Strings," *Image Vision Computing,* vol. 5, no. 1, pp. 3-9, 1987.

[2] Z. Shi and V. Govindaraju, "Segmentation and Recognition of Connected Handwritten Numeral Strings," *Pattern Recognition,* vol. 30, no. 9, pp. 1501-1504, 1997.

[3] Z.K. Lu, Z. Chi, W.C. Siu, and P.F. Shi, "A Background-Thinning Based Algorithm for Separating Handwritten Touching Digit Strings," *Pattern Recognition,* vol. 32, no. 6, pp. 921-933, 1999.

[4] J. Rocha and T. Pavlidis, "Character Recognition without Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 9, pp. 903-909, Sept. 1995.

[5] D.M. Ha, M. Zimmermann, and H. Bunke, "Off-Line Handwritten Numeral String Recognition by Combining Segmentation-Based and Segmentation-Free Methods," *Pattern Recognition,* vol. 31, no. 3, pp. 257-272, 1998.

[6] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 366-376, Apr. 1997.

[7] P.D. Gader, M. Mohamed, and J. Chiang, "Handwritten Word Recognition with Character and Inter-Character Neural Networks," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 27, no. 1, pp. 158-164, 1997.

[8] V.K. Govindan and A.P. Shivaprasad, "Character Recognition—A Review," *Pattern Recognition,* vol. 23, no. 6, pp. 671-683, 1990.

[9] R. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical Character Recognition—A Survey," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 5, nos. 1 and 2, pp. 1-24, 1991.

[10] Q.D. Trier, A.K. Jain, and T. Taxt, "Feature Extraction Methods for Character Recognition—A Survey," *Pattern Recognition,* vol. 29, no. 4, pp. 641-662, 1996.

[11] H. Yan, "Prototype Optimization of a Nearest Neighbor Classifier Using a Multi-Layer Neural Network," *Pattern Recognition,* vol. 26, pp. 317-324, 1993.

[12] T. Wakahara, "Shape Matching Using LAT and Its Application to Handwritten Numeral Recognition," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 16, no. 6, pp. 618-629, June 1994.

[13] K.W. Cheung, D.Y. Yeung, and R.T. Chin, "A Unified Framework for Handwritten Character Recognition Using Deformable Models," *Proc. Second Asian Conf. Computer Vision,* vol. 1, pp. 344-348, 1995.

[14] A.K. Jain and D. Zongker, "Representation and Recognition of Handwritten Digits Using Deformable Templates," *IEEE Trans. Pattern Analysis and Machine Analysis,* vol. 19, no. 12, pp. 1386-1391, Dec. 1997.

[15] K.J. Versprille, "Computer-Aided Design Applications of The Rational B-spline Approximation Form," PhD Thesis, Syracuse Univ., 1975.

[16] M. Revow, C.K.I. Williams, and G.E. Hinton, "Using Generative Models for Handwritten Digit Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 6, pp. 592-606, June 1996.

[17] T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation,* vol. 1, no. 1, pp. 1-23, 1993.

[18] D.B. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Trans. Neural Networks,* vol. 5, no. 1, pp. 3-14, 1994.

[19] T. Kozek, T. Roska, and L.O. Chua, "Genetic Algorithm for CNN Template Learning," *IEEE Trans. Circuits and Systems—I: Fundamental, Theory, and Applications,* vol. 40, no. 6, pp. 392-402, 1993.

[20] K. Delibasis and P. Undrill, "Genetic Algorithm and Deformable Geometric Models for Anatomical Object Recognition," *Proc. IEE Colloquium Genetic Algorithm in Image Processing and Vision,* vol. 8, pp. 1-7, 1994.

[21] M. Sarkar, B. Yegnanarayana, and D. Khemani, "A Clustering Algorithm Using an Evolutionary Programming-Based Approach," *Pattern Recognition Letters,* vol. 18, pp. 975-986, 1997.

[22] D. Bouchaffra, V. Govindaraju, and S.N. Srihari, "Postprocessing of Recognized Strings Using Nonstationary Markovian Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 10, pp. 990-999, Oct. 1999.

A Note on Park and Chin's Algorithm

Ronaldo Fumio Hashimoto and Junior Barrera

Abstract—A finite subset of $\mathbb{Z}^2$ is called a structuring element. A decomposition of a structuring element A is a sequence of subsets of the elementary square (i.e., the 3×3 square centered at the origin) such that the Minkowski addition of them is equal to A. Park and Chin [1] developed an algorithm for finding the optimal decomposition of simply connected structuring elements (i.e., 8-connected structuring elements that contain no holes), imposing the restriction that all subsets in this decomposition are also simply connected. In this paper, we show that there exist infinite families of simply connected structuring elements that have decompositions but are not decomposable according to Park and Chin's definition.

Index Terms—Simply connected set, structuring element, decomposition, Minkowski addition.

1 INTRODUCTION

A finite subset of $\mathbb{Z}^2$ is called a *structuring element* (SE). In this paper, we just consider nonempty SEs. The problem of decomposing a SE as a sequence of Minkowski additions of smaller subsets has been studied by several researchers [2], [3], [4], [5], [6], [1], [7], [8], [9] and many different algorithms have arisen to generate decompositions.

Park and Chin [1] developed an algorithm for decomposing a *simply connected* SEs (i.e., an 8-connected SE that contains no holes) as a sequence of Minkowski additions of a minimal number of simply connected subsets of the *elementary square* (i.e., the 3×3 square centered at the origin). Their algorithm is very complex and can be divided in three steps: 1) verify if a simply connected SE A satisfies some necessary conditions for decomposition, 2) find a decomposition of A by solving a system of linear inequalities, and 3) search for an optimal decomposition using an extension of an intricate optimization process, given in [4], that is subdivided in many complex cases to join, by Minkowski addition, the subsets found in Step 2. Moreover, in their work, Park and Chin [1] did not mention the time complexity of their algorithm.

Now, we observe that there exists some simply connected SEs, undecomposable by Park and Chin's definition, but that can be decomposed as a sequence of subsets of the elementary square without the restriction that these subsets must be simply connected. For example, in Fig. 1a, we present a simply connected SE A that is not decomposable according to Park and Chin's definition [1, example 3, p. 10]. But, if we do not impose that the subsets in the decomposition must be simply connected, then A is decomposable. Fig. 1b presents a decomposition of A. Note that the first subset in this decomposition is not simply connected.

In this paper, we show that infinite families of simply connected SEs that have decompositions but are not decomposable according to Park and Chin's definition exist. This result is very important and relevant because it implies that Park and Chin's algorithm is not a real algorithm for decomposing simply connected SEs; that is, if one applies their algorithm for finding a decomposition of a given simply connected SE A and its output is "A has no decomposition,"

• R.F. Hashimoto is with the Departamento de Ciência da Computaçãao, Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508-900, São Paulo, SP, Brasil, and also with the National Genome Research Institute, Texas A&M University, 1109 Southwest Parkway 2007, College Station, TX 77840. E-mail: ronaldo@ime.usp.br.
• J. Barrera is with the Departamento de Ciência da Computaçãao, Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508-900, São Paulo, SP, Brasil. E-mail: jb@ime.usp.br.

Manuscript received 4 Oct. 1999; revised 26 Oct. 2000; accepted 28 Feb. 2001.
Recommended for acceptance by A. Kundu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 110697.