ELSEVIER

Pattern Recognition Letters 23 (2002) 381–394

Pattern Recognition Letters

www.elsevier.com/locate/patrec

StrCombo: combination of string recognizers

Xiangyun Ye [a,b], Mohamed Cheriet [a,b,*], Ching Y. Suen [a]

[a] *Centre for Pattern Recognition and Machine Intelligence, Concordia University, Suite GM606, 1455 de Maisonneuve Blvd. West, Montréal, Qué., Canada H3G 1M8*

[b] *Laboratory for Imagery, Vision and Artificial Intelligence, École de Technologie Supérieure, University of Québec, 1100 Notre-Dame West, Montréal, Qué., Canada H3C 1K3*

Abstract

In this paper, we contribute a new paradigm of combining string recognizers and propose generic frameworks for hierarchical and parallel combination of multiple string recognizers. The frameworks are open to any new achievement in either recognizers or combination algorithms, and can be applied to both machine-printed and handwritten string recognition problems. A parallel combination system, *StrCombo*, is implemented based on three independent alphanumeric handwritten string recognizers that act as black boxes. We propose a graph-based approach that regards each segment from individual string recognizers as nodes of a graph, and choose the optimal path with the lowest cost to output a combined result. All factors such as the agreement of size, classification, and the position are converted into a measurement resulting in a soft decision. *StrCombo* has achieved a substantial improvement over any one of the individual recognizers, as demonstrated by experimental results on standard numeral string databases and a non-standard alphanumeric string database from real-life applications.

Keywords: Handwritten string recognition; Combination of multiple experts; Alphanumeric string recognizers

1. Introduction

In recent years, a new trend called "Combination of Multiple Experts" (CME) has been intensively investigated to solve complex pattern recognition problems. This idea is based on the intuition that classifiers with different methodologies and features can complement each other, and therefore a higher performance can be achieved if their results are combined properly. With the emergence of the theory and related methods for combining multiple classifiers, promising results have been obtained in many diverse domains, such as handwriting recognition, fingerprint recognition, disease diagnosis, remote sensing data analysis, etc.

In the context of handwriting recognition, the concept of combining multiple classifiers has been proposed as a new direction for the development of highly reliable character recognition systems (Suen et al., 1990; Suen et al., 1993). During the last decade, the industry has been trying to increase the optical

* Corresponding author.
E-mail address: cheriet@gpa.etsmtl.ca (M. Cheriet).

0167-8655/02/$ - see front matter
PII: S0167-8655(01)00171-4

character recognition (OCR) accuracy by voting the outputs from multiple engines (Spencer, 2000). Up till now, several companies have already marketed their products based on combination at the character level, which substantially increased the accuracy and decreased the substitution in processing typical business forms.

While intensive research efforts have been put into the combination of character recognizers, little work has been done on combining handwritten string recognizers, which serve as the basic module in most applications. In a trivial case when the users are required to write carefully in pre-defined positions, the characters are usually detached from each other, and the combination can be done at the character level since one-to-one correspondence is guaranteed. However, when dealing with unconstrained strings that are composed of alphabetic or numeric characters without context, faulty segmentation results may introduce *m–n* correspondence problems, rendering a direct combination at the character level impossible. Some recent publications have addressed this problem, and provided preliminary solutions such as combination based on intuitive voting rules (Wang et al., 1999) or by merging the layouts (Klink and Jäger, 1999) obtained by different commercial OCR devices. While Wang et al. (1999) assumed that the number of characters in the string is known a priori and applied a simple majority voting rule based on the agreement of multiple string recognizers, Klink and Jäger (1999) utilized more information of layout analysis results of machine printed documents. Their combination process starts with line segments, usually based on words, and ends up with a result preserving the original page layout as precisely as possible, including a combination of recognition results. This method is designed for machine printed documents, which generally enables consensus on segmentation points. Due to their basic assumption of the characteristics of the input strings, these methods are not directly applicable to the recognition of unconstrained handwritten strings that contain an unknown number of characters.

This paper proposes a generic framework for combining multiple string recognizers. Although the individual recognizers and the combination rules are not optimal, the combined recognizer, *StrCombo*, can achieve a substantial improvement over any one of the individual recognizers. As this framework is open to new recognizers and combination rules, a higher performance of *StrCombo* can be expected whenever the individual recognizers upgrade their performance.

The paper is organized as follows. Section 2 will introduce the state-of-art methods of character level combination, which are the basic components of *StrCombo*. Sections 3 and 4 will introduce generic frameworks and an implementation of the parallel combination strategy; Section 5 will discuss the experimental results; and Section 6 will conclude the research work.

2. Combination at character levels

The combination rules applicable to a set of classifiers is dependent to the types of information available from their outputs. The most commonly adopted categorization was proposed by Xu et al. (1992), in which the outputs of various classifiers are generalized into one of the following three levels: *abstract*, *rank* and *measurement levels*. *Abstract level* outputs only a unique class label or a subset of labels for the input pattern. *Rank level* outputs a sorted list of class labels, with the top one corresponding to the class to which the input pattern most likely belongs. *Measurement level* classifiers assign measurement values to indicate the probability or confidence that the input pattern belongs to each class. Depending on the types of information produced by the individual classifiers, many different combination methods have been proposed. Detailed surveys can be found in (Suen and Lam, 2000; Lam and Suen, 1995; Lu and Yamaoka, 1997; Tax et al., 2000; Ho, 1994; Impedovo and Salzo, 1999; Jain et al., 2000) and are skipped here due to the limit on the length of this paper. For the sake of

consistency, we follow the outline of Suen and Lam (2000) and summarize below a brief review of these three types of combination methods.

- Combination of *abstract level* outputs. For *abstract level* classifiers, representative combination methods are Majority Voting (Ho, 1998; Ji and Ma, 1997; Lam and Suen, 1997), Weighted Majority Voting (Lam and Suen, 1995), Bayesian formulation (Xu et al., 1992), Dempster-Shafer theory of evidence (Mandler and Shuermann, 1998), the Behavior–Knowledge Space method (Huang and Suen, 1995), and a dependency-based framework for optimal approximation of the product probability distribution (Kang and Lee, 1999).
- Combination of *rank level* outputs. Borda Counts has been widely used in combining *rank level* classifiers (Ho et al., 1994). Weighting the rank scores according to their relative importance in the decision making can modify the Borda Counts to account for different levels of performance. The rank scores can be used to denote the quality of the input pattern.
- Combination of *measurement level* outputs. The simplest ways of combining *measurement level* outputs are the Max, Min, Sum (Avg) and Median rules. A common theoretical framework for these combination rules has been established in (Kittler, 1996; Kittler, 1998; Kittler et al., 1998). Authors of some recent studies have proposed unified combination frameworks (Al-Ghoneim and Kumar, 1998; Cordella et al., 1999) that treat the three types of combination algorithms as special cases, and allow for new combination methods.

3. Generic frameworks for the combination of string recognizers

Implemented with different segmentation and classification strategies, different string recognizers may provide different recognition as well as segmentation results for touching or broken characters, which make it impossible to directly combine the multiple outputs on a one-to-one basis. With proper combination techniques, faulty segmentation can be avoided and correct string recognition results can be expected, even if none of the individual recognizers is able to provide a fully recognized string. Depending on the performance and adjustable parameters of individual string recognizers, the combination can be conducted in two manners: hierarchical and parallel, which are similar to the combination of multiple classifiers (Jain
et al., 2000). The concept of cascading or serial combination of multiple classifiers is not applicable to the combination of string recognizers, since string recognition results are more complicated than classification results, which usually indicate to which class the input pattern belongs.

3.1. Hierarchical combination strategy

In the ideal case of combining isolated character recognizers, we can find an *oracle* that is able to predict the best classifier for each input pattern, and direct the dynamic selection of classifiers (Ho, 1994). Similar principles can be applied to combine string recognizers if the recognizers are available at both character and string levels. A framework is illustrated in Fig. 1, in which a pre-segmentation module can be designed to distinguish isolated characters from touching character strings. For isolated characters, a set of character recognizers can be activated and combined by conventional combination rules (Xu et al., 1992) or dynamic selection (Ho et al., 1994). For strings composed of touching characters, a set of string recognizers can be activated and combined by voting rules as proposed in (Wang et al., 1999). Most segmentation-based or segmentation-free string recognition methods fall into this framework, and can be viewed as special cases if only partials are available. For example, if the input images contain only touching strings, the framework can be simplified as the multi-expert method in (Wang et al., 1999), which forms the right part of the system. In another case when only one

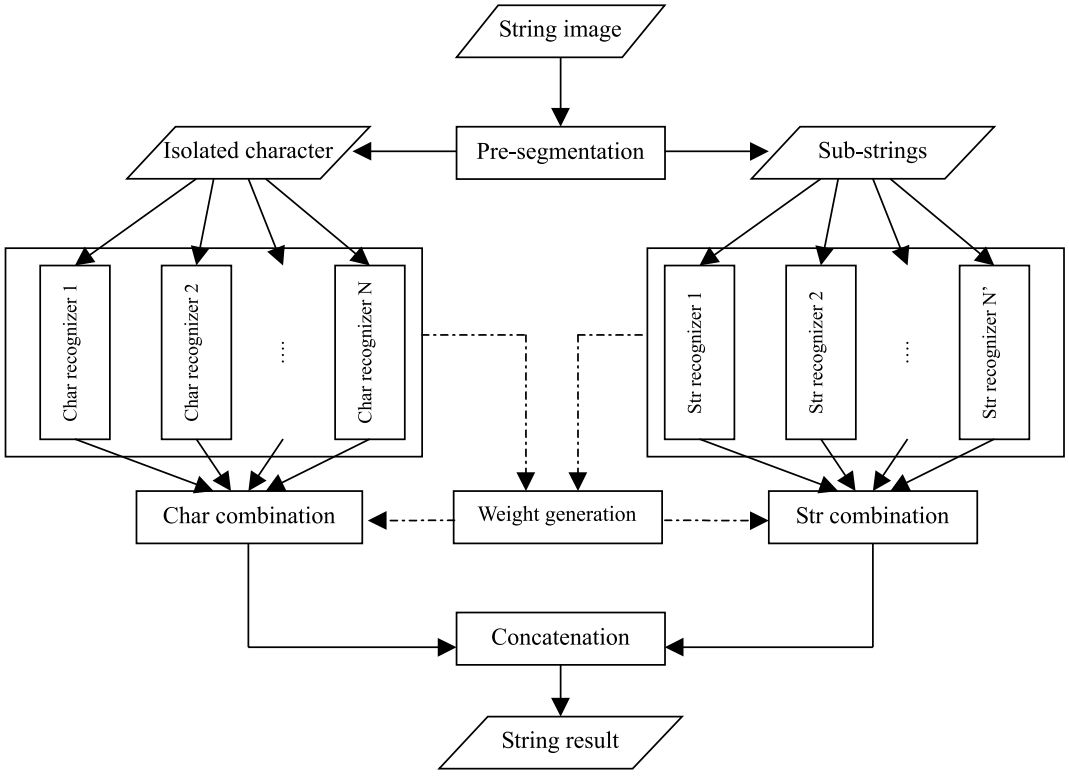

Fig. 1. A framework for hierarchical combination of string recognizers. The dotted lines depict the training procedure in which the performance of the different recognizers are analyzed and weights for combination purposes are generated.

segmentation-based and one segmentation-free string recognizers are available, the framework can be simplified as the combination method in (Ha et al., 1998), in which the segmentation-based module only attempts to segment the input string into groups of digits, and the segmentation-free module recognizes groups of broken and/or touching digits.

3.2. Parallel combination strategy

Commercial string recognizers are often wrapped up as black boxes. The users do not have control over the segmentation and recognition modules. In this case, a parallel combination framework that uses only the outputs from individual string recognizers can be very useful in optimizing the recognition performance. In this framework, each individual string recognizer works independently on the input image, and the recognition results are evaluated regardless of the features extracted from the input images. The combination can be conducted in a batch mode to post-process large quantities of string recognition results.

As we have mentioned in Section 1, faulty segmentation results make direct use of the conventional combination methods impossible. Therefore, we propose to take both segmentation and recognition into account, and construct a directed graph by taking each output character of the individual recognizers as a node. Each node and edge is weighted by a comprehensive evaluation based on recognition and segmentation procedures. A combined output can be obtained by searching the optimal path from the starting

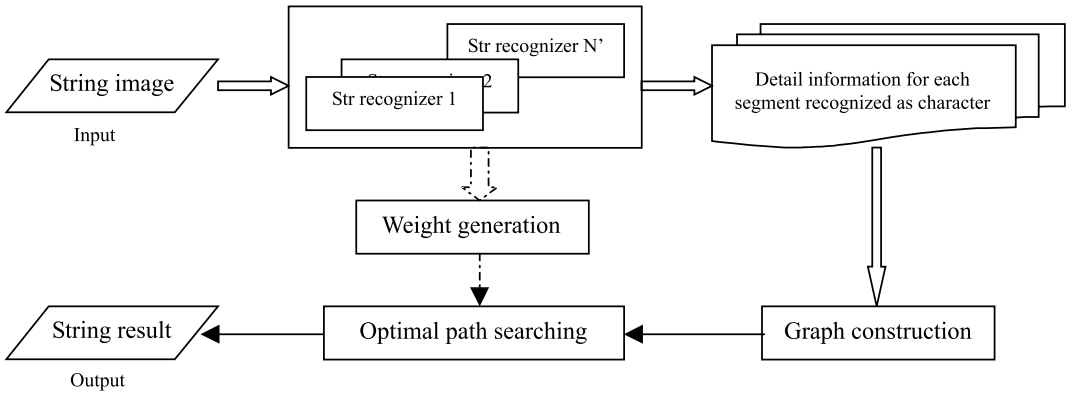

Fig. 2. A framework for parallel combination of string recognizers. Similar to Fig. 1, the dotted lines depict the training procedure in which the performance of the different recognizers are analyzed and weights for combination purposes are generated.

character to the end. Fig. 2 shows a generic framework for parallel combination of string recognizers. Actually, many methods in the literature (Xu et al., 1992; Suen and Lam, 2000; Lam and Suen, 1995; Lu and Yamaoka, 1997; Tax et al., 2000; Ho, 1994; Impedovo and Salzo, 1999; Jain et al., 2000; Ho, 1998; Ji and Ma, 1997) can be modified and adopted to adjust the weights of nodes and edges in the graph during the training stage (dotted line in the figure), and enhance the performance of the entire system.

4. Graph-based parallel combination strategy

Limited to the string recognizers available in this study, we investigated a parallel combination method based on graph theory. Although all recognizers used here belong to the *measurement level* (Xu et al., 1992), generalization to *abstract level* and *rank level* recognizers is possible by using unified criteria, such as the Pooled Ranking Figure of Merit (Al-Ghoneim and Kumar, 1998).

Typical outputs from a *measurement level* string recognizer consist of a sequence of characters of the top N choices, their corresponding confidence values, and the minimal bounding boxes:

$$S = \{s_1, s_2, \ldots, s_L\}, \quad s_i = \langle \vec{C}_i, \vec{P}_i, Rect_i, CS_i \rangle, \ \ 1 \leqslant i \leqslant L, \ \ L \text{ is the length of the output string.} \tag{1}$$

$\vec{C}$: A sorted list of classes for the segment that is recognized as a character. $\vec{P}$: A sorted list of confidence values corresponding to $\vec{C}$. $Rect = \langle L(eft), T(op), R(ight), B(ottom) \rangle$: The minimal bounding box of the segment. CS: The confidence value of the segmentation procedure.

A parallel combination of string recognizers can be presented as constructing a combined sequence of characters $S_{\text{combo}} = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{L_c}\}$ from multiple strings $S_k = \{s_{k1}, s_{k2}, \ldots, s_{L_k}\}$, and $\bar{s}_j \in Y_{k=1}^{K} S_k$ $(1 \leqslant j \leqslant L_c)$ are the combined results of characters recognized by K string recognizers.

To solve this problem, we construct a directed graph $G = \{V, E\}$, in which each node (vertex) $V \in Y_{k=1}^{K} S_k$ represents a segment recognized as meaningful by one of the string recognizers, and each edge E indicates a possible linkage between segments. Two special nodes, i.e., *Start* and *End*, correspond to the beginning and the end of the combined string, and the path between the *Start* and *End* with the best score (or the lowest cost) describes the combined character sequence. No loop is permitted in a string combination graph. This is assured by the structure of the graph and the score assigned to the edges.

The cost of an edge from node u to v is defined as

$$Cost(u \to v) = \{Score(v) * Score(E(u, v))\}^{-1}, \tag{2}$$

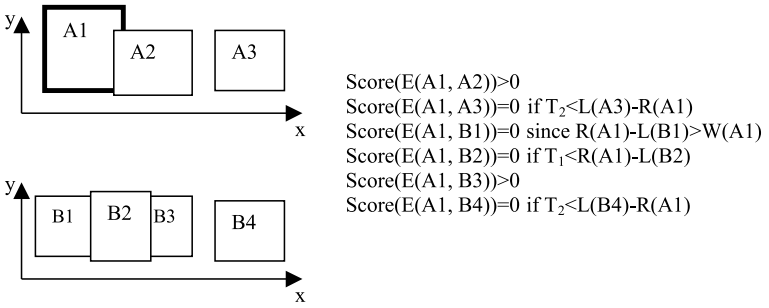

Fig. 3. An abstract example of combining two string recognizers A and B. The scores of the edges depend on the spatial relationship between two nodes. Scores of edges from A1 are shown.

in which

$$Score(E(u,v)) = \begin{cases} 0 & \text{if } R(u) - L(v) > \max(W(u), T_1) \text{ or } L(v) - R(u) > T_2, \\ f_{\text{size}} * f_{\text{ID}} * f_{\text{overlap}} * f_{\text{strlen}} & \text{otherwise.} \end{cases} \tag{3}$$

The upper condition in Eq. (3) prevents any path from constructing a loop in the graph or skipping a valid character. An abstract example of combining two string recognizers, A and B, is shown in Fig. 3. Assume the outputs from the two recognizers are located at the shown positions, then the scores of the edges from node A1 would depend on their positional relationship, as listed in the figure. The threshold T_1 in Eq. (3) is the tolerance of overlap between two neighboring characters in a string. In the experiments described later, we choose T_1 as $W(u)/2$. In applications such as machine-printed string recognition, T_1 can be set to zero. Threshold T_2 can be set to the nominal character width in the application, so that a path cannot skip a valid character. For example, a path from A1 to A3 should not be allowed, since A2 is a valid candidate.

A very important definition here is a set, $Peer(v) = \{v' | v' \neq v \text{ and } precedent(v') = precedent(v)\}$, that is composed of nodes (referred as peers from now on) that have the same precedents as v during the construction of a path from *Start* to the *End*. The precedents of v are a set of nodes that are defined as $precedent(v) = \{u | Score(E(u,v)) > 0\}$. A comparison of attributes among node v and its peers will help to evaluate the score of a transition from u to v. The evaluation scores are denoted as f_{size}, f_{ID}, f_{overlap}, and f_{strlen}. The notion behind this is that if node v is well recognized by one or more recognizers, and its position is well aligned with those of the peers, v is very likely to be a correct choice.

- Size agreement score

$$f_{\text{size}} = \prod_{v' \in Peer(v)} \frac{\min(W(v), W(v'))}{\max(W(v), W(v'))} * \frac{\min(H(v), H(v'))}{\max(H(v), H(v'))}. \tag{4}$$

It evaluates the size of segment v compared with the peers. $W(v) = R(v) - L(v)$ and $H(v) = B(v) - T(v)$ are the width and height of v. High values indicate that the segment v has similar width and height to those of its peers, and justifies that a path through v has a high score.

- Identity agreement score

$$f_{\text{ID}} = \prod_{\substack{v' \in Peer(v) \\ ID(v') = ID(v)}} (1 + e_1). \tag{5}$$

It evaluates the number of peers that have the same identification as segment v. $e_1 > 0$ is an adjustable parameter to increase the score of edge $u \to v$ if v has the same recognition output as the peers. The higher the value, the less is the cost of choosing v among the peers.

- Bounding box agreement score

$$f_{\text{overlap}} = \prod_{v' \in Peer(v)} \frac{Area(v \cap v')}{Area(v \cup v')}. \tag{6}$$

It evaluates the overlapping ratio among the bounding boxes of the peers. The higher the value, the more likely that segment v is a correct segmentation result.

- String length agreement score

$$f_{\text{strlen}} = \prod_{\substack{v' \in Peer(v) \\ L(v')=L(v)}} (1 + e_2). \tag{7}$$

It evaluates the number of peers that come from character sequences of the same length. $L(v)$ denotes the length of the optimal path from *Start* to node v. $e_2 > 0$ helps to increase the score of edge $u \to v$ if v comes from a recognition output of the same length as that from other recognizers. A path of length k from a vertex u to a vertex u' in a graph $G = \{V, E\}$ is a sequence of vertices $\langle v_0, v_1, \ldots, v_k \rangle$, such that $u = v_0,\ u' = v_k$, and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, \ldots, k$.

- Segmentation and recognition confidence score

$$Score(v) = Rel_{\text{seg}} * CS(v) + Rel_{\text{rec}} * P(v). \tag{8}$$

It evaluates the confidence and reliability of the segmentation and recognition of node v by a given recognizer. Depending on the different features and classification methods, the outputs from various recognizers usually need to be normalized before being combined. By evaluating the performance of a given recognizer on a training set, and the output on the input patterns, a reliability parameter can be defined (Xu et al., 1992; Cordella et al., 1999) and used to weight the vote of node v. Generally, high reliability and confidence of the segmentation and recognition will produce a higher score and therefore, high probability of node v.

The combination is a procedure of searching for the optimal path from the *Start* node to the *End* node at the lowest cost. Let u be the last chosen node, and $v_1, v_2, \ldots, v_k$ be the nodes with $Score(E(u, v_i)) > 0$ then the next node to be chosen in the optimal path is node v_j that incurs the lowest cost:

$$j = \arg \min_{i=1}^{k} (Cost(u \to v_i)).$$

This is an application of Dijkstra's algorithm (Cormen et al., 1990) on searching for the path at the lowest cost. Unlike conventional combination methods that evaluate the possibility of an input pattern belonging to each class and choose the class of highest possibility to be the output, this combination method evaluates the score of each node based on contextual information, and convert all factors such as the agreement of size, classification, and the position into measurements that result in a soft decision. In practice, most digit recognizers are not able to distinguish more than three top choices in the ten classes, therefore, we derive only three nodes for each segment, which can save much processing time.

Similar to the combination schemes at the character level, the performance of the method we proposed relies on the performance and the independence of the individual recognizers. Two examples are shown in Fig. 4. Fig. 4(a) is an example when none of the three recognizers is able to recognize the full string, the combined recognizer, *StrCombo*, can take advantage of all of them and obtain the right answer. For Fig. 4(b) recognizer C is able to recognize the string with low confidence. However, since recognizer A has higher recognition reliability than that of C, *StrCombo* is biased by recognizer A on the third character

Fig. 4. Two string images taken from NIST database. (a) cdf2097_31_21; (b) cdf2069_42_30.

Table 1
Recognition results of Fig. 4(a) from individual recognizers and the combination result (*ID*, *P*, and *Rect* are the outputs from the individual recognizers, as defined in Eq. (1))

String Recognizer A			String Recognizer B			String Recognizer C		
ID	*P*	*Rect*	*ID*	*P*	*Rect*	*ID*	*P*	*Rect*
6	0.80	(92, 63, 120, 112)	1	0.89	(93, 63, 180, 112)	**0**	0.99	(92, 63, 132, 111)
2	0.80	(121, 63, 151, 112)				**3**	0.97	(133, 64, 184, 111)
3	0.88	(152, 63, 179, 112)	**7**	0.99	(185, 63, 228, 112)	1	0.92	(185, 63, 226, 111)
7	0.70	(184, 63, 227, 112)	0	0.57	(231, 54, 316, 112)	4	0.98	(230, 54, 270, 111)
4	0.83	(230, 54, 264, 112)				0	0.75	(271, 61, 314, 111)
0	0.46	(265, 54, 315, 112)						

Table 2
Recognition results of Fig. 4(b) from individual recognizers and the combination result

String Recognizer A			String Recognizer B			String Recognizer C		
ID	*P*	*Rect*	*ID*	*P*	*Rect*	*ID*	*P*	*Rect*
0	0.78	(124, 48, 161, 87)	**0**	0.99	(124, 47, 162, 84)	0	0.97	(124, 48, 158, 82)
3	0.72	(160, 44, 193, 87)	4	0.85	(160, 43, 268, 88)	3	0.94	(159, 44, 194, 86)
5	0.54	(194, 44, 217, 87)				2	0.57	(195, 44, 223, 81)
4	0.43	(218, 44, 248, 87)				**8**	0.94	(224, 33, 262, 85)
5	0.68	(249, 44, 267, 87)						
5	0.58	(249, 29, 280, 43)	7	0.61	(249, 28, 283, 44)			
3	0.80	(267, 43, 309, 84)	3	0.99	(267, 42, 310, 85)	**3**	0.97	(263, 29, 308, 83)

although the raw score from A is lower than from C, and output a wrong result. Future work including optimizing the design of the cost function, or implementing a hierarchical combination system may help to bring down this type of failures.

Tables 1 and 2 list the top-choice output strings from three independent recognizers and their corresponding attributes. The optimal path is found to be composed of the nodes printed in boldface. Fig. 5 illustrates the graph composed of segments output from three independent string recognizers; the arrows indicate the optimal paths. Details of the individual string recognizers will be discussed in the next section.

5. Experiments and discussion

To evaluate the parallel combination method, we have carried out experiments using three independent string recognizers. They act as black boxes, which accept a binary string image as input, and give a sorted list of classes and corresponding confidences as output. In the following discussion, they are referred simply as string recognizers A, B, and C. The tests are made on the NIST SD3 database (Wilkinson et al., 1992), the CEDAR database (Hull, 1994), and an alphanumeric string database acquired from a local company, DOCImage Inc. The number of characters in a string is not known to any of the string recognizers

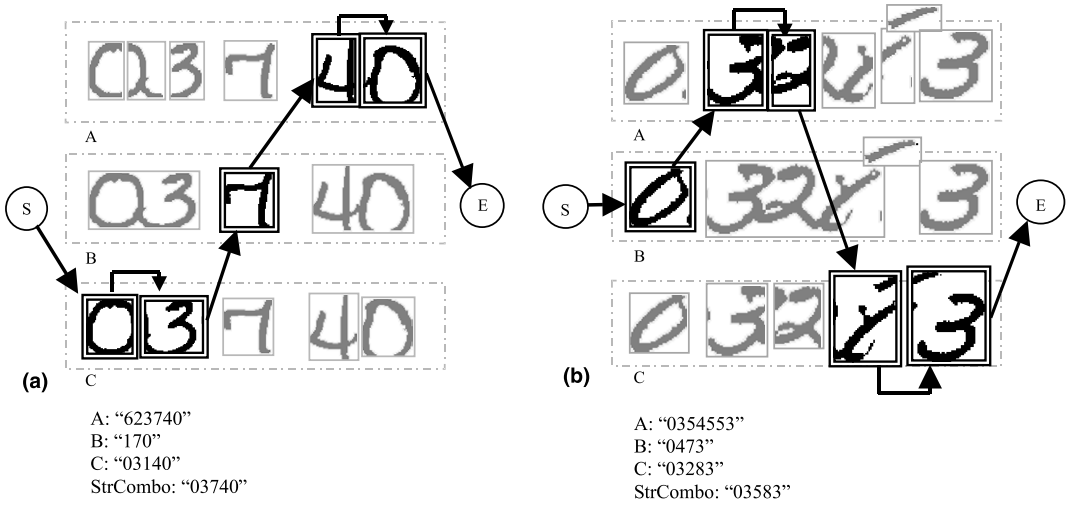

Fig. 5. Combination of string recognizers by searching for the optimal path in a graph. The nodes of the graph denote the segments recognized as meaningful characters by individual string recognizers. *S* and *E* denote the *Start* and *End* nodes, respectively. (a) Combination of the numeral string in Fig. 4(a), *StrCombo* provides a correct string while none of the three individuals does; (b) Combination of the numeral string in Fig. 4(b), Recognizer C provides a correct string but *StrCombo* does not.

involved. We use the following definitions of the recognition rate, error rate, and rejection rate at the string level (Ha et al., 1998; Junker et al., 1999). Let N be the total number of string images in a testing set. If N_{rej} strings are rejected, N_{cor} strings are correctly recognized, and N_{err} are mis-recognized, then $N_{\text{cor}} + N_{\text{err}} + N_{\text{rej}} = N$.

$$\text{string recognition rate} \quad StrRec = 100 * \frac{N_{\text{cor}}}{N}\%, \tag{9}$$

$$\text{string error rate} \quad StrErr = 100 * \frac{N_{\text{err}}}{N}\%, \tag{10}$$

$$\text{string rejection rate} \quad StrRej = 100 * \frac{N_{\text{rej}}}{N}\%, \tag{11}$$

$$\text{string reliability} \quad StrRel = 100 * \frac{N_{\text{cor}}}{N_{\text{cor}} + N_{\text{err}}}\%, \tag{12}$$

$$StrRec + StrErr + StrRej = 100\%. \tag{13}$$

Moreover, if N_{seg} strings are segmented into the right number of characters, we define the string segmentation rate as

$$StrSeg = 100 * \frac{N_{\text{seg}}}{N}\%. \tag{14}$$

If these N_{seg} strings are composed of n characters, and among them n_{cor} are correctly recognized, n_{err} are mis-recognized, and n_{rej} are rejected, we can define the following rates at the character level:

$$\text{character recognition rate} \quad CharRec = 100 * \frac{n_{\text{cor}}}{n}\%, \tag{15}$$

$$\text{character error rate} \quad CharErr = 100 * \frac{n_{\text{err}}}{n}\%, \tag{16}$$

$$\text{character rejection rate} \quad CharRej = 100 * \frac{n_{\text{rej}}}{n}\%, \tag{17}$$

$$\text{character reliability} \quad CharRel = 100 * \frac{n_{\text{cor}}}{n_{\text{cor}} + n_{\text{err}}}\%. \tag{18}$$

For the total n_{total} characters that form the total N strings, we define a character extraction rate as follows:

$$\text{character extraction rate} \quad CharExtr = 100 * \frac{n_{\text{cor}}}{n_{\text{total}}}\%. \tag{19}$$

In our tests, a string is counted as correctly recognized only if all characters composing it are correctly recognized.

5.1. Training on the combination weights

Since the string recognizers in this test use different features and classification methods, their outputs are not normalized and cannot be compared directly. In order to compare their outputs fairly, two important parameters need to be determined before combination can be conducted: the reliability of recognition Rel_{rec} and segmentation Rel_{seg} in Eq. (8). These values can be estimated based on the individual recognizers' performance on a training set. We call this procedure the training of combination weights, in the sense that these parameters are obtained prior to any combination procedures, and they are used in weighting the nodes and edges in the graph that will be used in the combination. Meanwhile, we set both e_1 and e_2 in Eqs. (5) and (7) to 0.25.

We tested string recognizers A, B, and C with a set of 500 numeral strings chosen from the NIST string database, and 100 alphanumeric strings chosen from DOCImage database. The results in non-rejection case are shown in Table 3, and the thresholds of recognizers A, B, and C at 1% *StrErr* rate are listed in Table 4.

Therefore, we set the reliability parameters as $Rel_{\text{seg}} = StrSeg$, $Rel_{\text{rec}} = StrRec/Threshold_{1\%\text{err}}$ (Table 5).

The reliability parameters determined by the recognition and segmentation rates on the training set follows a well-known approach for combining isolated character recognizers (Xu et al., 1992). More sophisticated approaches can be derived from the output of the recognizers (Impedovo and Salzo, 1999).

Table 3
Recognition results of string recognizers A, B, and C on testing sets

Testing set	Numeral string			Alphanumeric strings		
String recognizer	A	B	C	A	B	C
StrSeg (%)	93.6	89.7	94.7	87.7	57.6	61.3
CharRec (%)	96.9	99.2	92.1	81.4	87.3	81.2

Table 4
Recognition results of string recognizers A, B, and C on testing sets

Testing set	Numeral string			Alphanumeric strings		
String recognizer	A	B	C	A	B	C
Threshold at 1% *StrErr*	0.81	0.81	0.83	0.94	0.85	0.66

Table 5
Reliability parameters for string recognizers A, B, and C

Testing set	Numeral string			Alphanumeric strings		
String recognizer	A	B	C	A	B	C
Rel_{seg}	0.936	0.897	0.947	0.916	0.799	0.879
Rel_{rec}	1.196	1.225	1.110	0.866	1.03	1.23

These reliability parameters are dependent on the specific database that is used for the training. We do not extend our discussion to these sophisticated approaches because our framework is an open structure that can incorporate any theoretical achievement in the literature, and this paper is focused only on the validity and feasibility of the framework instead of comparing different combination methods.

5.2. Tests on numeral strings

The combination method is tested on the following databases of numeral strings, which have also been used as testing sets in (Ha et al., 1998), which obtained the highest performance on these databases.

- NIST, 5195 numeral strings are selected (files f1800-f1899 and f2000-f2099). In total 23840 digits are included.
- CEDAR, two sets, i.e., "BinZips" including 495 numeral strings composed of 2711 digits, and "ZipCodes" including 435 numeral strings composed of 2318 digits.

The combination results are listed in Tables 6 and 7, respectively. Following the conventions in (Ha et al., 1998), the column "*StrRej* 0%" gives the string recognition rates at zero-rejection level, and columns "*StrErr* = 2%", "1%", "0.5%" present the string recognition rate at the corresponding error rate levels. Although the performance of *StrCombo* is not comparable to Ha et al. (1998), substantial improvement has been obtained over the best individual recognizer except for 1% error case in "ZipCodes" set. Here, the improvement of any rate is defined as the percentage of the difference between that of *StrCombo* and the best individual over the best individual

$$improvement_{\text{Rate}} = \frac{(Rate_{\text{Combo}} - \max_{i\in\{A,B,C\}} Rate_i)}{\max_{i\in\{A,B,C\}} Rate_i}. \tag{20}$$

Table 6
Combination results (string recognition rates) on CEDAR database

	BinZips				ZipCodes			
	StrRej = 0%	*StrErr*			*StrRej* = 0%	*StrErr*		
		2%	1%	0.5%		2%	1%	0.5%
String length		5, 9				5, 9		
# of strings		495				435		
Ha et al. (1998)	83.6	60.0	51.5	48.0	72.9	49.5	44.5	43.0
A	60.3	29.7	19.0	5.3	48.4	18.4	16.1	10.6
B	61.1	22.6	17.2	11.7	56.0	20.0	13.6	9.0
C	44.4	9.9	8.1	5.7	43.8	11.7	10.1	5.7
StrCombo	77.4	35.6	26.7	15.8	68.6	30.3	14.9	12.9
Improvement	26.8	19.9	40.5	35.1	22.5	51.5	−7.5	21.7

Table 7
Combination results (string recognition rates) on NIST database

			StrRej	*StrErr*		
Str recognizer	String length	# of strings	0%	2%	1%	0.5%
Ha et al. (1998)	2–6	4925	92.7	86.0	82.0	74.0
A	2–10	5196	84.7	65.8	56.2	41.2
B			85.9	66.1	53.3	39.4
C			69.5	30.4	24.4	17.9
StrCombo			93.4	77.4	65.3	53.3
Improvement			8.7	17.1	16.2	35.3

Table 8
Combination results at the character level (CEDAR and NIST databases)

	BinZips (495 strings)			ZipCodes (435 strings)			NIST (5196 strings)		
Str recognizer	*StrSeg*	*CharRec*	*CharExtr*	*StrSeg*	*CharRec*	*CharExtr*	*StrSeg*	*CharRec*	*CharExtr*
A	77.8	93.9	72.4	65.5	92.3	64.4	94.7	97.1	92.1
B	71.9	95.6	62.5	68.0	93.9	65.4	89.3	94.6	88.1
C	77.0	88.7	65.9	74.9	88.2	73.9	95.7	92.9	88.5
StrCombo	88.7	96.7	84.5	83.4	95.5	81.9	96.8	99.1	96.2
Improvement	12.3	1.1	14.3	11.3	1.7	10.8	1.2	2.1	4.5

The combination results measured at the character level are listed in Table 8. It is clearly shown that the parallel combination method improved not only the character recognition, but also string segmentation abilities.

5.3. Tests on alphanumeric strings

Another test is conducted on a set of alphanumeric strings, including 690 strings that contain 4255 characters in total. The string lengths are between 6 and 7. Tables 9 and 10 list the combination results at

Table 9
Combination results at the string level (alphanumeric string databases)

Str Recognizer	A	B	C	*StrCombo* (%)	*Improvement* (%)
StrRec (%) when *StrRej* = 0%	21.9	27.0	22.8	43.9	62.6

Table 10
Combination results at the character level (alphanumeric string databases)

	690 alphanumeric strings		
Str recognizer	*StrSeg*	*CharRec*	*CharExtr*
A	89.0	77.6	69.2
B	69.9	83.9	58.7
C	80.0	78.8	63.2
StrCombo	90.4	87.5	79.4
Improvement	1.6	4.3	14.7

the string and character levels respectively. In the experiments, we noticed that the output of alphanumeric string recognizer A is either 0 or +1.0, which result in difficulties in obtaining meaningful results at "2%", "1%" and "0.5%" error rates. Due to the poor quality of the string images, the performance of the string recognizers is far from satisfactory. However, the string recognition rate of StrCombo has improved 62.6% over the best one among the three recognizers that are combined. The improvement from individual recognizers to *StrCombo* reveals a promising way of combining measurement level string recognizers that provide positional information of the characters as well.

6. Conclusion

Combination of multiple experts is found to be effective in solving pattern recognition problems. Encouraging results have been obtained in the combination of isolated handwritten character recognizers. However, direct application to handwritten strings is non-trivial due to faulty segmentation results that occur often on touching strings. In this paper, we propose general frameworks for hierarchical and parallel combination of string recognizers. All theoretical achievements on combining character recognizers can be readily adapted to these frameworks, and some existing methods of string recognition can be considered as special cases of these frameworks. We have investigated the parallel combination of string recognizers, and proposed a graph-based approach that regards each segment from individual string recognizers as a node of a graph, and the optimal path from the *Start* to the *End* nodes according to specific evaluation scores corresponding to the best combined result. Experimental results on standard numeral string databases and a non-standard alphanumeric string database demonstrate the effectiveness of the proposed approach. Although the combined results are still far from the best published over the years, the improvement obtained over the individuals is proven to be helpful in constructing a high-performance string recognizer from multiple recognizers with medium level performance. Since we have found few references in the literature that address the problem of combining string recognizers, a numerical comparison with the existing methods is not practical here. We do believe that more combination methods of isolated character recognition can be applied to the proposed frameworks and better performance can be obtained.

Acknowledgements

This research was supported by the grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), and Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) program of the Ministry of Education of Quebec. The authors would like to thank Mr. Claude Rheault of DOCImage Inc. for providing training and testing alphanumeric string images, and Ms. Christine P. Nadal for her assistance in data collection.

References

Al-Ghoneim, K., Kumar, B.V.K.V., 1998. Unified decision combination framework. Pattern Recognition 31 (12), 2077–2089.

Cordella, L.P., Foggia, P., Sansone, C., Tortorella, F., Vento, M., 1999. Reliability parameters to improve combination strategies in multi-expert systems. Pattern Anal. Appl. 2, 205–214.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. In: Introduction to Algorithms. MIT Press, Cambridge, MA, pp. 527–531.

Ha, T.M., Zimmermann, M., Bunke, H., 1998. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. Pattern Recognition 31 (3), 257–272.

Ho, T.K., 1994. Adaptive coordination of multiple classifiers. In: Hull, J.J., Taylor, S.L. (Eds.), Document Analysis Systems II. World Scientific, Singapore, pp. 371–384.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Machine Intell. 20 (8), 832–844.

Ho, T.K., Hull, J.J., Srihari, S.N., 1994. Decision combination in multiple classifier systems. IEEE Trans. Pattern Anal. Machine Intell. 16 (1), 66–75.

Huang, Y.S., Suen, C.Y., 1995. Combination of multiple experts for the recognition of unconstrained handwritten numerals. IEEE Trans. Pattern Anal. Machine Intell. 17, 90–94.

Hull, J.J., 1994. A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Machine Intell. 16, 550–554.

Impedovo, S., Salzo, A., 1999. Evaluation of combination methods. In: Proc. ICDAR, Bangalore, India, pp. 713–716.

Jain, A., Duin, P., Mao, J., 2000. Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Machine Intell. 22 (1), 4–37.

Ji, C., Ma, S., 1997. Combination of weak classifiers. IEEE Trans. Neural Networks 8 (1), 32–42.

Junker, M., Hoch, R., Dengel, A., 1999. On the evaluation of document analysis components by recall, precision and accuracy. In: Proc. ICDAR, Bangalore, India, pp. 713–716.

Kang, H.-J., Lee, S.-W., 1999. Combining classifiers based on minimization of a Bayes error rate. In: Proc. ICDAR, Bangalore, India, pp. 398–401.

Kittler, J., 1996. Improving recognition rates by classifier combination. In: Fifth Internat. Workshop on Frontiers in Handwriting Recognition, Colchester, UK, pp. 81–101.

Kittler, J., 1998. Combining classifiers: a theoretical framework. Pattern Anal. Appl. 1, 18–27.

Kittler, J., Hatef, M., Duin, R.P.W, Matas, J., 1998. On combining classifiers. IEEE Trans. Pattern Anal. Machine Intell. 20 (3), 226–239.

Klink, S., Jäger, T., 1999. MergeLayouts – Overcoming faulty segmentation by a comprehensive voting of commercial OCR devices. In: Proc. ICDAR, Bangalore, India, pp. 386–389.

Lam, L., Suen, C.Y., 1995. Optimal combination of patter classifiers. Pattern Recognition Lett. 16, 945–954.

Lam, L., Suen, C.Y., 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Trans. Systems Man Cybernet. 27 (5), 553–568.

Lu, Y., Yamaoka, F., 1997. Fuzzy integration of classification results. Pattern Recognition 30 (11), 1877–1891.

Mandler, E., Shuermann, J., 1998. Combining the classification results of independent classifiers based on the Dempster/Shafer theory of evidence. In: Geselma, E.S., Kanal, L.N. (Eds.), Pattern Recognition and Artificial Intelligence. North Holland, Amsterdam, pp. 381–393.

Spencer, H., 2000. OCR Update: using voting in document imaging solutions. Advanced Imaging Magazine, April, 17–21.

Suen, C.Y., Lam, L., 2000. Multiple classifier combination methodologies for different output levels. In: Proc. First Internat. Workshop on Multiple Classifier Systems, Cagliari, Italy, pp. 52–66.

Suen, C.Y., Legault, R., Nadal, C., Cheriet, M., Lam, L., 1993. Building a new generation of handwriting recognition systems. Pattern Recognition Lett. 14 (4), 303–315.

Suen, C.Y., Nadal, C., Mai, T.A., Legault, R., Lam, L., 1990. Recognition of totally unconstrained handwritten numerals based on the concept of multiple experts. In: Internat. Workshop on Frontiers in Handwriting Recognition, Montreal, Canada, pp. 131–143.

Tax, D.M.J., van Breukelen, M., Duin, R.P.W., Kittler, J., 2000. Combining multiple classifiers by averaging or by multiplying? Pattern Recognition 33, 1475–1485.

Wang, X., Govindaraju, V., Srihari, S., 1999. Multi-experts for touching digit string recognition. In: Proc. ICDAR, Bangalore, India, pp. 800–803.

Wilkinson, R.A., Geist, J., Janet, S., Grother, P.J., Burges, C.J.C., Creecy, R., Hammond, Hull, B.J.J., Larsen, N.W., Vogl, T.P., Wilson, C.L., 1992. The First Census Optical Character Recognition Systems Conf. The US Bureau of Census and the National Institute of Standards and Technology, Technical Report # NISTIR 4912, Gaithersburg, MD.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. Systems Man Cybernet. 22 (3), 418–435.