

# Recognizing Face Profiles in the Presence of Hairs/glasses Interferences

Weiping Chen\* and Yongsheng Gao\*†

\*School of Engineering, Griffith University, Australia

†National ICT Australia, Queensland Research Lab

{luke.chen, yongsheng.gao}@griffith.edu.au

**Abstract**—Facial profile provides a complementary structure of the face that is not present in frontal faces, which has been used in personal identification, face perception research and 3D face construction. In this paper, we present a novel local attributed string matching (LAsTrM) approach to recognize face profiles in the presence of interferences. The conventional profile recognition algorithms heavily depend on the accuracy of the facial area cropping. However, in realistic scenarios the facial area may be difficult to localize due to interferences (e.g., glasses, hairstyles). The proposed approach is able to efficiently find the most discriminative local parts between face profiles addressing the recognition problem with interferences. Experimental results have shown that the proposed matching scheme is robust to interferences compared against several primary approaches using two profile image databases (Bern and FERET). It has potential capability for partially occluded shape classification.

**Index Terms**—string matching, interference, profile recognition, facial area cropping, partially occluded.

## I. INTRODUCTION

Human face recognition is of practical importance in many applications, such as access control and security monitoring. Most automatic face recognition approaches are based on frontal images. Facial profiles, on the other hand, provide a complementary information of the face that is not present in frontal faces. Fusion of frontal and profile views can lead the overall personal identification technique more foolproof and efficient. Furthermore, in many scenarios, a frontal facial image is not available (e.g., the situation of a driver entering a gated area). Face profile recognition can be roughly classified into two categories: the fiducial mark-based approaches [1][2][3] and the profile outline-based methods [4][5][6]. Fiducial mark-based methods rely on the correct detection of fiducial points. However, automatic detection of fiducial marks is not always reliable, particularly when detecting features such as concave nose, flat chin, etc. Hence, profile outline-based methods have achieved much attention over past few years.

In [6], the profile curve was represented as a histogram, where histogram metric such as  $\chi^2$  distance and PDF  $L_N$  norms were employed to measure profiles. This approach is heavily dependent on empirically chosen threshold to filter profile outlines. Bhanu and Zhou [4] proposed a method using dynamic time warping (DTW) to match face profiles based on the curvature value of each point on the profile. Gao and Leung [5] approximated the face profiles using line segments described by attributed strings. Then an attributed string matching approach was proposed to compute the profile similarity, which

is more appropriate and robust than point matching methods. However, [4][5] assume that the two compared profiles have the same curve details and fails to work when an object has large local shape deformations or occlusions.

The effect of interferences has not been previously investigated in the face profile recognition, as most previous work has assumed that facial area required for recognition are visible and precisely localized. However, in realistic scenarios the facial area may be difficult to localize due to interferences (e.g., glasses, hairstyles). In this paper, we present a novel local attributed string matching (LAsTrM) approach that can recognize face profiles with hairs or glasses. A face profile outline is represented by an attributed string. The matching of two profiles is done by matching two attributed strings through a *string-to-string* matching scheme. The proposed method is able to efficiently find the most discriminative local parts for recognition without making any assumption on the distributions of the facial interferences regions. An example is illustrated in Figure 1, where (a) is original profile outlines; (b) is matched result using proposed approach; (c) is matched result using global method (i.e. AStrM [5]).

The rest of this paper is organized as follows. A brief overview of the string matching method is described in Section 2. Section 3 presents a new attributed string matching approach to address the face profile under partial conditions. Experimental results are given in Section 4. Finally, Section 5 concludes the paper.

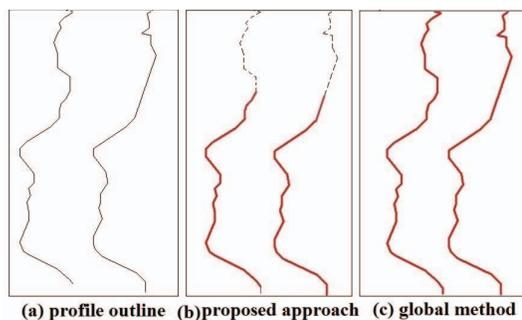


Fig. 1. An example of matching results. Solid red line represents matched parts and dash line represents non-match parts.

## II. STRING MATCHING

String matching is a syntactic and structural method for similarity measurement between strings or vectors, which has been widely used for researches in molecular biology [7], speech recognition [8], and file comparison [9]. Strings can be classified into two categories: symbolic strings and attributed strings. The symbolic string matching is widely used for shape recognition [10][11][12]. In these methods, shapes are described by string representation (see Fig. 2) and primitives are described by symbols (see Fig. 3). In Fig. 2, the string representations for these three shapes are  $S1 = aabccdd$ ,  $S2 = aabcccd$  and  $S3 = abccdd$ .

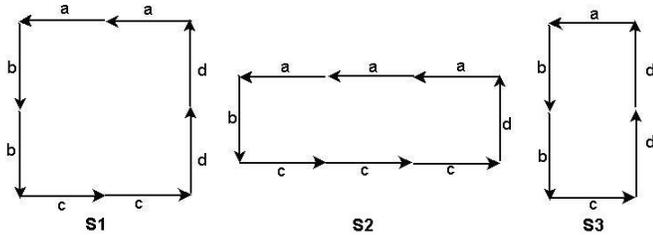


Fig. 2. Symbolic string representation for three shapes

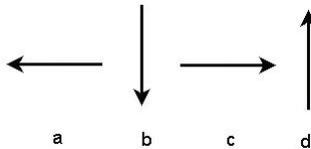


Fig. 3. Primitives and corresponding symbols used in Fig. 1

The goal of string matching algorithms is to find a sequence of elementary edit operations which transform one string into another at a minimal cost. The elementary operations for symbolic string matching are deletion, insertion, and substitution.

- 1) **Substitution (or Change):** to replace a symbol (e.g.  $a$  in  $S1$ ) with the other (e.g.  $b$  in  $S2$ ), denoted as  $a \rightarrow b$ .
- 2) **Insert:** to insert a symbol (e.g.  $a$  in  $S1$ ) into a string (e.g.  $S2$ ), denoted as  $\lambda \rightarrow a$ , where  $\lambda$  is a symbol used to denote nothing (called null symbol).
- 3) **Delete:** to delete a symbol (e.g.  $a$  in  $S1$ ) from a string (e.g.  $S1$ ), denoted as  $a \rightarrow \lambda$ .

Let  $C$  be a cost function which defines a cost  $C(a \rightarrow b)$  for each edit operation  $a \rightarrow b$ . Hence, the cost of a sequence of edit operations  $S = s_1, s_2, \dots, s_m$  is defined as:

$$C(S) = \sum_{i=1}^m C(s_i)$$

The matching between two strings (e.g.  $S1$  and  $S2$ ) is to calculate the edit distance  $d(S1, S2)$  from  $S1$  to  $S2$ , which is the minimum of the costs of all the edit sequences taking  $S1$  to  $S2$ .

From Fig. 3, we can see that the primitives used in symbolic string matching are pre-defined symbols. However, symbols

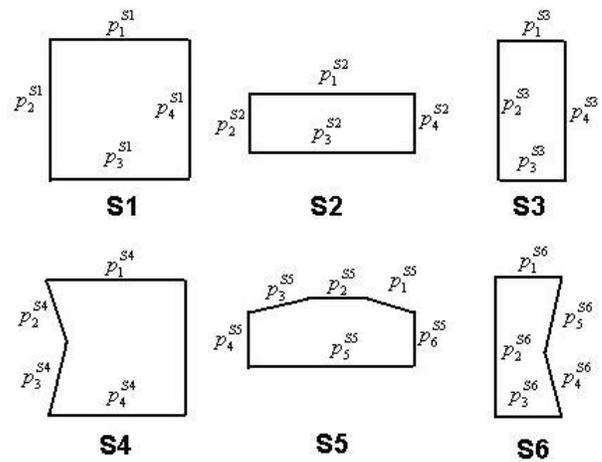


Fig. 4. Attributed string representation for shapes

are discrete in nature while most problems of pattern recognition deal with attributes that are basically continuous in nature. It was found inadequate to use symbols as primitives for complex pattern recognition [13]. Hence, the attributed string matching [5][13][14] are proposed. In attributed string matching, an object is represented by a string which is composed of attributed primitives (see Fig. 4). For example, in Fig. 4, the string representation of the shape  $S1$  is  $S1 = p_1^{s1} p_2^{s1} p_3^{s1} p_4^{s1}$ , where  $p_j^{s1}$  is the  $i$ th primitives with attributes (e.g. the length, orientation and location). Compared with the symbolic strings, the attributed strings are object-oriented, which is suitable for complex pattern recognition. In addition, a new edit operation, merge, is introduced in attributed string matching, which can address the noise and distortion issues (shapes  $S4, S5$  and  $S6$  in Fig. 4). The merge operation is used to combine and match any number of consecutive primitives in one string with those in the other. An example of merge operation is illustrated in Fig. 5, where three primitives  $P1, P2$  and  $P3$  are combined into a new primitive  $P$ .

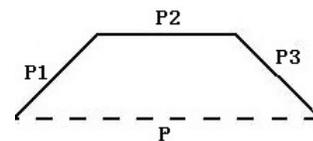


Fig. 5. An example of the merge operation

However, the conventional attributed string matching methods [5][13][14] have two major drawbacks. Firstly, their method is based on Needleman-Wunsch algorithm [15], which performs a global alignment on two sequences. Their algorithm fails to work when an object has large local shape deformations or occlusions (e.g. Shapes  $S2$  and  $S3$  in Fig. 6). In addition, the conventional attributed string matching methods need to search the start points (the first primitives).

Global method works well when the sequences are globally similar and there is no large occlusion. However, when unexpectedly long matching segments have been located between

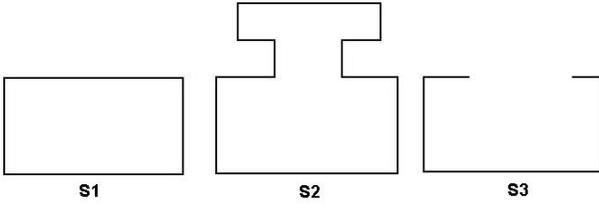


Fig. 6. An example of partial shapes

two sequences, the global alignment fails to work because every part of the sequences is taken into consideration. Smith and Waterman proposed a local alignment algorithm (SW algorithm) for identification of common molecular subsequences [16]. Given two molecular sequence  $A = a_1a_2 \dots a_n$  and  $B = b_1b_2 \dots b_m$ .  $s(a_i, b_j)$  is a similarity between two elements  $a_i$  and  $b_j$  in sequences  $A$  and  $B$ , respectively.  $W_d$  is weight to delete a element and  $W_i$  is weight to insert a element.  $H_{i,j}$  is the maximum similarity score of any two segments ending in  $a_i$  and  $b_j$ . The similarity is obtained using dynamic programming [16]:

$$H(i, j) = \max \begin{cases} H(i, j-1) - W_d \\ H(i-1, j) - W_i \\ H(i-1, j-1) + s(a_i, b_j) \\ 0 \end{cases} \quad (1)$$

The Smith-Waterman (SW) algorithm is similar to Needleman-Wunsch algorithm but includes an extra zero (Eq. 1), which allows termination of subsequence alignments that perform poorly and is thus able to find similar subsequences without exhaustive search. But SW algorithm is based on symbolic strings, which was found inadequate for complex pattern recognition. In this study, we propose a novel local attributed string matching approach and applied for partial face profile recognition.

### III. PROPOSED STRING MATCHING ALGORITHM

The algorithm in [5] calculated similarity between two entire face profile strings. However, the algorithm cannot ignore areas that show little similarity when dealing with face profiles under partial conditions(see Fig. 1). To overcome this limitation, we propose a new attributed string matching algorithm to perform partial matching for face profile in the presence of hairs/glasses interferences.

#### A. Cost functions

Let  $S_1(i)$  and  $S_2(j)$  be the  $i$ th and the  $j$ th primitives in strings  $S_1$  and  $S_2$  with attributes  $(l_i, \theta_i, x_i, y_i)$  and  $(l_j, \theta_j, x_j, y_j)$  respectively, where  $l_i$  and  $l_j$  are the lengths,  $(x_i, y_i)$  and  $(x_j, y_j)$  are the midpoint locations and  $\theta_i$  and  $\theta_j$  are the orientations of the line segments measured between the line and the reference line. The reference line is the line between the nose tip and the chin point. Let  $S(i, j)$  denote the substring of  $S$  from the  $i$ th to the  $j$ th primitives of  $S$ .

The cost function for change operation from  $S_1(i)$  to  $S_2(j)$  is defined as:

$$Cost[Change(S_1(i), S_2(j))] = |l_i - l_j| + f(\Delta(\theta_i, \theta_j)) + \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad (2)$$

where  $\Delta(\theta_i, \theta_j)$  is the angle difference between two primitives ( $0 \leq \Delta(\theta_i, \theta_j) \leq 90$ ), and  $f()$  is a non-linear penalty function to map the angle to a scalar using  $f(x) = x^2/W$ , and  $W = 50$  is the weight to balance the angle and length.

The costs of delete and insert operations can be derived from the above change cost function by introducing a null primitive  $\phi$  with zero length and indefinite angle and location. The cost functions of these two operations are defined as

$$Cost[deleteS_1(i)] = f(K_\theta) + l_i + K_{loc}, \quad (3)$$

$$Cost[insertS_2(j)] = f(K_\theta) + l_j + K_{loc}, \quad (4)$$

where  $K_\theta$  and  $K_{loc}$  are constants to represent the indefinite orientation and location of the line segment. For the purpose of penalization,  $K_\theta = 90^\circ$  represents the maximum angle difference and the maximum location difference (the diagonal distance of the input image) is used for  $K_{loc}$ .

Next, we consider the merge operation. The merge operation is used to combine and match any number of consecutive primitives in one string with those in the other. An example of merge operation is illustrated in Fig. 5, where three primitives  $P1$ ,  $P2$  and  $P3$  are combined into a new primitive  $P$ . Let  $S(i-k+1, i) = S(i-k+1)S(i-k+2) \dots S(i)$  be a substring with  $k$  primitives in  $S$  to be merged, and  $S^k(i)$  be the merged primitive of these  $k$  primitives. The merge operation is denoted as  $merge(S(i-k+1, i) \rightarrow S^k(i))$ . If  $k = 1$ ,  $S^k(i) = S(i)$ , which is the case without any merge operation. The merge cost is defined as:

$$Cost[merge(S(i-k+1, i) \rightarrow S^k(i))] = f\left(\frac{k-1}{l^k}\right) \sum_{p=i-k+1}^i \Delta(\theta^k, \theta_p) \times l_p, \quad (5)$$

where  $k$  is the number of merged primitives in  $S$ .  $l^k$  and  $\theta^k$  are the length and the line orientation of merged primitive  $S^k(i)$ ,  $l_p$  and  $\theta_p$  are the length and the line orientation of primitive in  $S$  before merging.

Now, by considering  $S^k(i)$  as a single primitive, the cost function for a change operation after merge can be rewritten as:

$$Cost[Change(S_1^k(i), S_2^l(j))] = |l^k - l^l| + f(\Delta(\theta^k, \theta^l)) + \sqrt{(x^k - x^l)^2 + (y^k - y^l)^2}, \quad (6)$$

which is performed after the  $k$  primitives in  $S_1(i-k+1, i)$  are merged as  $S_1^k(i)$  and the  $l$  primitives in  $S_2(j-l+1, j)$  are merged as  $S_2^l(j)$ . If  $k = 1$  and  $l = 1$ , no merge is performed and the above change operation reduces to the conventional one-to-one change operation  $Change(S_1(i), S_2(j))$  (see Eq.2)

### B. Similarity measure via dynamic programming

The similarity between the two profiles can be characterized by the edit operation cost using Dynamic Programming (DP) between the two strings. Let  $S_1 = S_1(1) \dots S_1(m)$  and  $S_2 = S_2(1) \dots S_2(m)$  be string representations of input profile face and model profile face, respectively, where  $m$  and  $n$  are numbers of primitives in  $S_1$  and  $S_2$ . To find pairs of strings with high degrees of similarity, we set up a similarity matrix  $D$ . Let the input string (i.e.  $S_1$ ) has  $m$  primitives represented by the rows of the similarity matrix  $D$ , and let the model string (i.e.  $S_2$ ) has  $n$  primitives represented by the columns of the similarity matrix  $D$ . First we initialize

$$D(i, 0) = D(0, j) = 0 \quad (0 \leq i \leq m, 0 \leq j \leq n). \quad (7)$$

$D(i, j)$  is the similarity of two strings ending at  $S_1(i)$  and  $S_2(j)$ . It is defined as:

$$D(i, j) = \max \begin{cases} 0 \\ D(i, j-1) - \text{Cost}[\text{insert}(S_2(j))] \\ D(i-1, j) - \text{Cost}[\text{delete}(S_1(i))] \\ \max_{k,l} \{ D(i-k, j-l) \\ + c(S_1(i-k+1 \rightarrow i), \\ S_2(j-l+1 \rightarrow j)) \} \end{cases} \quad (8)$$

where  $c(S_1(i-k+1 \rightarrow i), S_2(j-l+1 \rightarrow j))$  is defined as follows:

$$c(S_1(i-k+1 \rightarrow i), S_2(j-l+1 \rightarrow j)) = \lambda - \text{Cost}[S_1(i-k+1, i), S_2(j-l+1, j)], \quad (9)$$

where  $\text{Cost}[S_1(i-k+1, i), S_2(j-l+1, j)]$  is the cost of merge and change edit operations between substring  $S_1(i-k+1, i)$  and  $S_2(j-l+1, j)$  (see Eq. 5 and Eq.6).  $k$  and  $l$  are numbers of the merged primitives. In Eq.9,  $\lambda$  is used to decide the similarity between primitives  $S_1(i-k+1, i)$  and  $S_2(j-l+1, j)$ . If the cost value  $\text{Cost}[S_1(i-k+1, i), S_2(j-l+1, j)]$  is less than  $\lambda$ , these primitives are considered as similar elements.

For two profiles,  $S_1$  and  $S_2$ , with primitives  $S_1(i)$  ( $i = 1, 2, \dots, m$ ) and  $S_2(j)$  ( $j = 1, 2, \dots, n$ ), we compute all the similarity costs between their primitives and obtain the similarity matrix  $S$  as

$$D = \begin{pmatrix} D(0,0) & D(0,1) & \dots & D(0,n) \\ \vdots & \vdots & \vdots & \vdots \\ D(m,0) & D(m,1) & \dots & D(m,n) \end{pmatrix} \quad (10)$$

The pair of substrings with maximum similarity is found by first locating the maximal element in  $D$ . The other matrix elements leading to this maximal value are then sequentially determined with a traceback procedure ending with an element of  $D$  equaling to zero. The pair of substrings with the next best similarity is found by applying the traceback procedure to the second largest element in  $D$  not associated with the first traceback.

The proposed attributed string matching is conducted according to Algorithm 1, where  $\text{merg\_limitA}$  and  $\text{merg\_limitB}$  are controlling upper limits on the number

of primitives to be merged into a new one in  $S_1$  and  $S_2$ , respectively. The similarity of associating a group of segments from string  $S_1$  with a group of segments from string  $S_2$  is computed as

$$s(S_1, S_2) = \sum_{i=1}^f D_i, \quad (11)$$

The term  $D_i$  is the similarity (the  $i$ th maximal element in  $D$  matrix table) of the  $i$ th best similar substrings between two strings and  $f$  is the number of best similar substrings (in this study,  $f$  is the number of all the similar substrings).

```

D(0,0) := 0
for i:=1 to m do D(i,0) := 0;
for j:=1 to n do D(0,j) := 0;
for i:=1 to m
  for j:=1 to n
    begin
      m1 := D(i,j-1) - Cost[insert(S2(j))];
      m2 := D(i-1,j) - Cost[delete(S1(i))];
      for k:=1 to merge_limitA
        for l:=1 to merge_limitB
          M[k,l] := D(i-k,j-l) + lambda
            - {Cost[merge(S1(i-k+1,i) -> S1^k(i))
              + Cost[merge(S2(j-l+1,i) -> S2^l(j))
              + Cost[change(S1^k(i), S2^l(j))]}
          m3 := max(M[k,l]);
      D(i,j) := max(0, m1, m2, m3);
    end
  end
end

```

Algorithm 1. Proposed attributed string matching.  $m$  and  $n$  are the line segment numbers of strings  $S_1$  and  $S_2$

## IV. EXPERIMENTAL RESULTS

### A. Datasets and preprocessing

Two well-known face profile databases are used to evaluate the proposed approach. The first database is from University of Bern [17] containing 30 individuals with two profile view images for each person. The size of images is 342x512 pixels. No individual wear glasses in this database. The facial areas were cropped to eliminate hairs interference. Hence, after cropping, all profiles from this database are without interferences. An example is shown in Fig. 7.

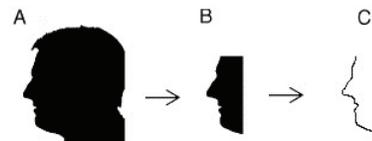


Fig. 7. A profile view of a face in the database(A), the cropped silhouette (B), and the profile contour (C)

Another database we used to test the proposed method is FERET database [18]. We selected 150 individuals with two side face images per individual. The size of images is 256x348 pixels. Compared with images in the database from University of Bern, images in FERET database are greatly affected by hairs or glasses interferences. Side images of the same person may have different facial areas (see Fig. 8 (b))

and (d)) or a large deformation on their profile outlines (see Fig. 8 (a) and (c)), which will affect the performance of the conventional face profile approach heavily. Hence, in this dataset, all information (including hairs or glasses) on profile outline was extracted to evaluate the performance of proposed approach under interference conditions.

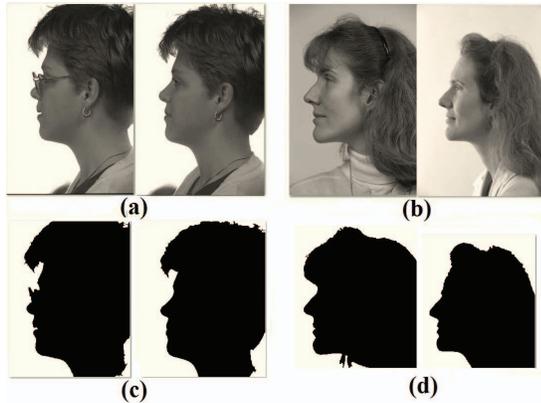


Fig. 8. Samples from FERET. (a)(or (c)) two side images of the same person with a large deformation on their profile outlines because of glasses interference. (b) (or (d)) two side images of the same person with different facial area because of hairs interference.

All face images were normalized using the line between the nose tip and the chin points. Then face profile outline curves were obtained using the algorithm proposed in [19]. In our experiments, *merge\_limit* was set as 10.

### B. Determination of $\lambda$

The effect of  $\lambda$  in Algorithm 1 was investigated on recognition accuracy using Bern database. Fig. 9 shows the curve of recognition rate against the  $\lambda$  in normal condition. The recognition rate increases greatly from  $\lambda = 5$  to  $\lambda = 12$ . Between  $\lambda = 12$  and  $\lambda = 14$ , the recognition rate is steady. Then it decreases with the increase of  $\lambda$ . For the rest of the experiments,  $\lambda$  was set as 13.

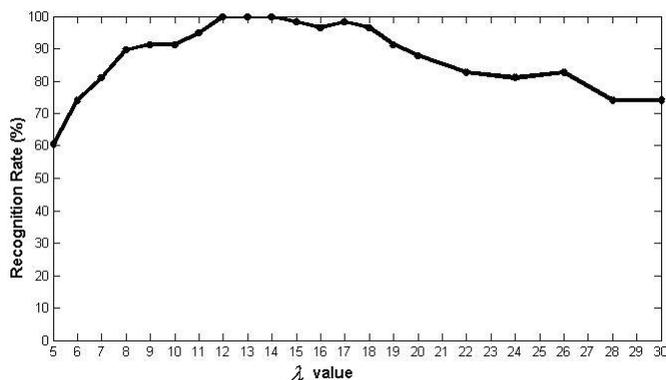


Fig. 9. The effect of  $\lambda$  on the recognition rate

### C. Profile Recognition without and with interferences

In order to evaluate the effectiveness and feasibility of the proposed approach, a system performance investigation

TABLE I  
PERFORMANCE COMPARISON BASED FACE PROFILE WITHOUT AND WITH HAIRS/GLASSES INTERFERENCE

Methods	Recognition rate(%)			
	Profile without interferences	Profile with interferences		
	rank1	rank1	rank3	rank5
LAstrM	98.2%	82.6%	88.3%	95.2%
AstrM	98.3%	33.3%	43.9%	48.4%
DTW	90.0%	25.7%	40.9%	46.9%
hist_ $\chi^2$	96.1%	72.7%	83.3%	90.9%
hist_pdf	98.0%	71.2%	83.3%	86.4%

is conducted, which covers face profile recognition without and with hairs/glasses interference using the first database and the second database (see Section 3.1), respectively. The system performances are compared with several benchmark approaches, i.e. the dynamic time warping (DTW) [4], histogram with  $\chi^2$  distance (hist\_ $\chi^2$ ) and PDF  $L_N$  norms (hist\_pdf) with abnegated 20% of points whose distances with the corresponding is large [6] and attributed string matching (AstrM) [5]. The experimental results are illustrated in Table I. It can be observed that the proposed method achieved the second best result in cropped face profile recognition test and significantly outperformed other benchmark methods in non-crop face profile recognition experiment.

## V. CONCLUSION

A local attributed string matching method for human face profile recognition has been proposed. The matching of two profiles is done by matching two attributed strings through a *string-to-string* matching scheme which is able to efficiently find the most discriminative local parts for recognition without making any assumption on the distributions of the facial interference regions. This is believed to be the first piece of work on face profile analysis to address the interferences problem, which is one of the most challenging problem in real applications. Furthermore, our method is based on line segments, which are more reliable and accurate than fiducial points to represent shapes. Through preliminary experiments on two popular face database our matching method produces promising results for both face profiles with and without interferences. It also has potential capability for partially occluded shape classification.

## REFERENCES

- [1] L. D. Harmon and W. F. Hunt, "Automatic recognition of human face profiles," *Computer Graphics and Image Processing*, vol. 6, pp. 135–156, 1977.
- [2] L. D. Harmon, M. K. Khan, R. Lasch, and P. F. Ramig, "Machine identification of human faces," *Pattern Recognition*, vol. 13, no. 2, pp. 97–110, 1981.
- [3] C. J. Wu and J. S. Huang, "Human face profile recognition by computer," *Pattern Recognition*, vol. 23, pp. 255–259, 1990.
- [4] B. Bhanu and X. Zhou, "Face recognition from face profile using dynamic time warping," *The 17th International Conference on Pattern Recognition*, vol. 4, pp. 499–502, 2004.
- [5] Y. Gao and M. K. Leung, "Human face profile recognition using attributed string," *Pattern Recognition*, vol. 35, pp. 353–360, 2002.
- [6] G. Pan, L. Zheng, and Z. Wu, "Robust metric and alignment for profile-based face recognition: An experimental comparison," *The 17th IEEE Workshop on Applications of Computer Vision*, 2005.

- [7] S. C. Chan and A. K. C. Wong, "Synthesis and recognition of sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 12, pp. 1245–1255, 1991.
- [8] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
- [9] P. Heckel, "A technique for isolating differences between files," *Communications of ACM*, vol. 21, no. 4, pp. 264–268, 1978.
- [10] L. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 4, pp. 404–411, 1975.
- [11] L. Fung and K. Fu, "Stochastic syntactic decoding for pattern classification," *IEEE Transactions on Computers*, vol. C-24, no. 6, pp. 662–667, 1975.
- [12] S. Lu and K. S. Fu, "A sentence-to-sentence clustering procedure for pattern analysis," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 5, pp. 381–389, 1978.
- [13] W. H. Tsai and S. S. Yu, "Attributed string matching with merging for shape recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7(4), pp. 453–462, 1985.
- [14] S. W. Chen, S. T. Tung, C. Y. Fang, S. Cheng, and A. K. Jain, "Extended attributed string matching for shape recognition," *Computer Vision and Image Understanding*, vol. 70, no. 1, pp. 36–50, 1998.
- [15] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J Mol Biol*, vol. 48, no. 3, pp. 443–453, 1970.
- [16] T. Smith and M. Waterman, "Identification of common molecular subsequences," *J. molec. Biol.*, vol. 147, pp. 195–197, 1981.
- [17] Face database, University of Bern. [Online]. Available: <ftp://iamftp.unibe.ch/pub/Images/FaceImages>
- [18] J. P. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face-recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000. [Online]. Available: <http://citeseer.ist.psu.edu/24549.html>
- [19] M. Leung and Y. Yang, "Dynamic two-strip algorithm in curve fitting," *Pattern Recognition*, vol. 23, no. 1-2, pp. 69–79, 1990.