A dynamic Time Warping Algorithm for Recognition of Multi-Stroke On-Line Handwriten Characters

Harold Mouchère, Jinpeng Li, Christian Viard-Gaudin, Zhaoxin Chen

A Dynamic Time Warping Algorithm for Recognition of Multi-Stroke On-Line Handwritten Characters*

Mouchère Harold [1] Li Jinpeng [1] Viard-Gaudin Christian [1] Chen Zhao-xin [1,2]

(1. IRCCyN laboratory, L'UNAM-Université de Nantes, Nantes 44300, France;

2. School of Electronics and Information Engineering，South China University of Technology, Guangzhou 510640, Guangdong, China)

Abstract: Dynamic time warping (DTW) is a famous distance to compare two mono-stroke symbols online. It obeys boundary and continuity constraints. The extension of DTW to multi-stroke symbols raises specific problems. A naive solution is to convert the multi-stroke symbol into a single one via a direct concatenation in the handwriting order. However, people may write a symbol with different stroke directions and orders. Applying a brute force method by searching all the possible directions and orders leads to prohibitive calculation times. To reduce the searching complexity, DTW-A* algorithm, which keeps the continuity constraint during each partial matching and reduces the searching complexity by using the A* algorithm, is proposed. Experimental results on a flowchart dataset mainly containing multi-stroke symbols indicate that DTW-A* algorithm helps to achieve the highest recognition rate and stability in cross-validation, as compared with such two other algorithms as DTW and modified Hausdorff distance.

Key words: pattern recognition; handwriting recognition; dynamic time warping

CLC number:(补 to be completed by publisher) doi:10.3969/j.issn.1000-565X.2013.07.000

Received date:

*Foundation item:

Biography: Mouchère Harold (born in 1981), male, Associate Professor in University of Nantes, FR, mainly researches on machine learning, on-line handwriting recognition, document analysis and currently on on-line handwritten math expressions. E-mail: harold.mouchere@univ-nantes.fr

收稿日期：

*基金项目：

作者简介：

多笔画联机手写字符识别的动态时间归正算法*

Mouchère Harold [1] 李锦鹏 [1] Viard-Gaudin Christian [1] 陈肇欣 [1,2]

(1. 法国南特市南特大学 IRCCyN 实验室, 邮政编码: 41300;

2. 中国广东省广州市华南理工大学电子与信息学院, 邮政编码: 510640)

摘要:在单笔画符号（或字符）联机手写识别中，动态时间规正（DTW）算法遵循时间次序约束和边界约束，并具有较高的识别率。 为了将此算法应用于多笔画符号识别，常用而简单的方法是按照人们的手写顺序连接多笔画符号为单笔画符号。 但此方法存在一问题: 人们常使用不同的笔画顺序和笔画方向书写同一个符号，用朴素（Brute Force）方法寻找所有笔画可能性非常耗时。为了降低计算复杂度，文中提出了 DTW-A*算法。在部分笔画匹配时，此算法保留着次序约束，并用 A*算法降低计算复杂度。文中还通过流程图数据库多笔画符号识别实验对比了 DTW-A*算法、 DTW 算法、 改良豪斯多夫距离 3 种算法的性能，结果表明 DTW-A*算法具有最好的识别率和稳定性

关键词: 模式识别; 手写识别; 动态时间规正

中图分类号:补 **doi:10.3969/j.issn.1000-565X.2013.07.018**

文章编号:1000-565X（2013）07-0000-00

0 Introduction

In this paper, a distance applied to on-line handwriting, whose basic elements are temporal point sequences (strokes), is proposed. A sequence is started from the pen-down point and is ended at the pen-up point, with variable point number. Based on the elastic point-to-point matching, the famous DTW algorithm computes the distance between two sequences[1], i.e. two single-stroke symbols, and obeys continuity and boundary constraints during the matching.

Many works [2] extend the temporal continuity constraint to a spatial continuity constraint in two spatial dimensions. It aims at finding a mapping between two sets of points (pixels). The contribution of this paper is to focus on how to design a matching between two sets of sequences, i.e. two multi-stroke symbols.

Each instance of a handwritten graphical symbol is different from the other because of the variability of human handwriting. Different people may write a visually same symbol with different stroke directions and orders. In writer identification, these characteristics help to efficiently distinguish writers [3]. However, to understand or communicate the same symbol written by different writers, stroke direction and order could be ignored. For instance, a symbol containing a horizontal stroke "—" can be written by two different approaches, namely the way from left to right "→"or an inverse way "←". Comparing two opposite direction strokes, the DTW distance $dist_{DTW}(\rightarrow, \leftarrow)$ naturally produces a large value because of two inverse directions. A simple solution is to choose the smaller distance between two possible directions of one stroke: $\min(dist_{DTW}(\rightarrow, \leftarrow), dist_{DTW}(inv(\rightarrow), \leftarrow))$, where inv(.) is an operator for reversing stroke trajectory direction.

However, when comparing two multi-stroke symbols, the number of possible directions and orders increases very fast as a function of the stroke number. Table 1 illustrates an example of how to write "*E*" within four strokes. With this example, 384 different writing sequences are possible. This example shows the complexity of the combination of different stroke directions and orders. In general, the number of different temporal writing paths for a symbol is given by

$$S_N = N! \times 2^N = 2 \times N \times S_{(N-1)} \quad (1)$$

where N is the stroke number of a symbol. For calculating the DTW distance between two multi-stroke symbols, a simple solution is to concatenate the strokes using different stroke directions and orders.

Table 1 Variability of stroke direction and order of on-line handwritten symbol

Stroke Number (N)	Example	Combination Number (S)	Writing Method Temporal Illustration
1	—	2	→ ←
2	=	8	1→2→ 1←2← 2→1→ 2←1←
3	王	48	1→2→3→ 1←2←3← ..... 3→2→1→ 3←2←1←
4	𠕋	384	.....

For example, for the DTW distance between 𠕋 (4 strokes) and 匚 (2 strokes), 384 × 8 = 3092 possible matching should be calculated. This large combination number is due to different writing orders of N strokes ($N!$) and due to the two directions of each written order (2^N).

In a more extreme case, we can get rid of all the temporal information and consider the symbol as a set of points ignoring the sequences they produce. This leads to the use of the Hausdorff distance [4]. This metric is mainly used in image processing domain. Furthermore, another varied version is the modified Hausdorff distance (MHD) [5].

However, there exists one disadvantage that the temporal continuity property of sequences is ignored. In this paper, a distance between two multi-stroke symbols is proposed, which is called DTW A* and preserves the temporal continuity constraint. In the investigation, the classical DTW between two sequences is first recalled, and is then extended to process two sets of sequences. Finally, the corresponding experimental results are presented and a conclusion is drawn.

1 DTW Between Two Point Sequences

We start with the simple case in which the two characters are composed of only one stroke, respectively. In this case, two strokes (two time-varying-data sequences), denoted as $S_1 = (p_1(1), p_1(2),\dots, p_1(N_1))$ and $S_2 = (p_2(1), p_2(2),\dots, p_2(N_2))$, are compared. Giving a warping path $P(h) = (i(h),j(h))$, $1 \le h \le H$, defining the point-to-point associated pairs where h is a pair index from the $i(h)$th point in S_1 and from the $j(h)$th point in S_2. $P(h)$ should consider the boundary constraint and the continuity constraint. It means that the first two beginning points should be matched in the two strokes, and so do the two ending points. The second temporal continuity constraint implies that the point-to-point matching shift is equal to one. In addition, all the points are matched at least for one time. Calculating the distance between two sequences involves the search of a warping path that minimizes the sum of the point-to-point associated cost function:

$$D(S_1,S_2) = \min_{P(h)} \sum_{h=1}^{H} dist\left(p_1\left(i(h)\right), p_2\left(j(h)\right)\right), \tag{2}$$

where dist(.,.) is the Euclidean distance in the point feature space.

The solution to Eq. (2) can be resolved by means of dynamic programming. The dynamic programming searches the minimum warping path from a cumulative distance matrix

$$D(i,j;0) = dist\big(p_1(i), p_2(j)\big) + min\begin{cases} D(i-1,j;h-1) \\ D(i,j-1\,;\,h-1) \\ D(i-1,j-1\,;h-1). \end{cases} \quad (3)$$

with $D(i, j; 0) = 0$ for initialization. Once the cumulative distance matrix is computed, we can use backtracking to find the minimum warping path.

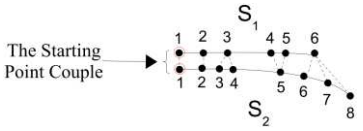


Fig.1 Two point sequences (two single-stroke symbols)

Fig.1 illustrates an example of matching two single-stroke symbols. The starting point couple is marked with the two circled points. The search for the next couple is obtained with Eq. (3). We first compute a cumulative distance matrix as explained in Fig. 2. The best warping path can be found by means of backtracking from the ending point couple to the starting point couple to obtain: $P(1)$, $P(2)$,..., $P(9)$ = (1,1), (2,2), (3,3), (3,4), (4,5), (5,5), (6,6), (6,7), (6,8). We can see that, once we define the starting point couple and the ending point couple, the best warping path will be found. In Section 2, a comparison between two sets of point sequences will be introduced.

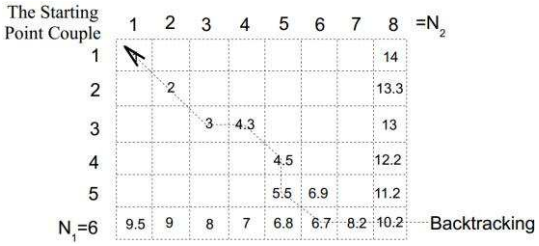


Fig. 2 The cumulative distance matrix $D(i, j; h)$ of Eq. (3) illustration and the best warping path

2 DTW Between Two Sets of Sequences

Here, we propose to extend the DTW algorithm to compare two multi-stroke symbols (i.e. two sets of sequences). A traditional method is to concatenate the strokes in the handwritten order [6]. Therefore, the distance between two multi-stroke symbols can be computed using DTW [1]. We call this method the classical DTW. However, because of the stroke-order and stroke-direction variation among writers, the classical DTW cannot easily match some symbols as discussed in the introduction. The MHD is a possible solution to cope with this variation. In this paper, the MHD between two symbols ($S_1 = \{pt_{i1}\}$ and $S_2 = \{pt_{i2}\}$) is defined as

$$MHD(S_1, S_2) = \frac{1}{|S_1| + |S_2|}\big(hd(S_1, S_2) + hd(S_2, S_1)\big) \quad (4)$$

where $hd(S_1, S_2) = \sum_{pt_{i1} \in S_1} \underset{pt_{i2} \in S_2}{min} \big(dist(pt_{i1}, pt_{i2})\big)$. The MHD here is slightly different from that defined in Ref. [5]. We choose an average distance rather than a maximum distance between two point sets to prevent the effect of outliers. However, the MHD does not consider the continuity constraint.

We now introduce a new distance called DTW A* to compare two multi-stroke symbols by keeping the continuity constraint.

Basically, as with the classical DTW, a point-to-point distance matrix is built via the dynamic programming. Giving two multi-stroke symbols, rows and columns of this matrix represent the

two symbols, respectively, as shown in Fig. 3. The strokes of one symbol are placed in one side (rows or columns). The respective positions of the strokes in the two sequences are irrelevant and their matching has not to respect the temporal order.

The main idea insists in iteratively constructing a small warping path until all the points are used. Once we choose a starting point couple, four possible directions of warping path are possible. Each direction represents a point-to-point distance matrix for matching two strokes or two sub-parts from two strokes. In each iteration, we search a warping path that minimizes warping cost and finishes at least one stroke. To find the best warping path, four cumulative distance matrices (see Fig. 3) are explored in four directions, respectively.

For example, giving two symbols, one contains two strokes while the other contains one stroke, the two strokes of the first symbol are placed in rows and the stroke of the second symbol is placed in columns (one for each point), as shown in Fig. 3. Once we define a starting point (the small rectangle in the middle of Fig. 3), there are four possible matching directions (four possible warping paths) corresponding to four cumulative matrices. In each cumulative matrix, we can apply the classical DTW algorithm described in Fig. 2 to find the minimum cost warping path. We allow, however, DTW algorithm to not stop at the diagonal opposed point (the ending point) in the cumulative distance matrix but along the borders of the matrix (red cross signs).

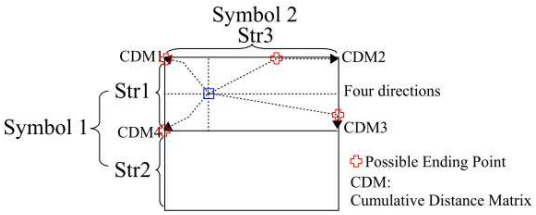


Fig. 3 Defining a starting point couple (the rectangle) and finding a warping path between the two-stroke symbol (symbol 1) and the single-stroke symbol (symbol 2) in four directions

In fact, from Fig. 2, it can be found that the warping path stopping at $P(9)=(6, 8)$ is not the best as the distance increases after $P(7)$. This warping path can be cut by choosing the minimum distance among the points of the cumulative matrix edges: $D(1, N_2)$, $D(2, N_2)$, ..., $D(N_1, N_2)$ and $D(N_1, 1)$, $D(N_1, 2)$, ..., $D(N_1, N_2)$. In reality, we first calculate the whole cumulative distance matrix till to the end of both two strokes. Then the warping path stops when finishing with at least one of two strokes. Thus, a new ending point couple is obtained. For example， in Fig. 2, we choose the sub-path: (1, 1), (2, 2), (3, 3), (3, 4), (4, 5), (5, 5), (6, 6).

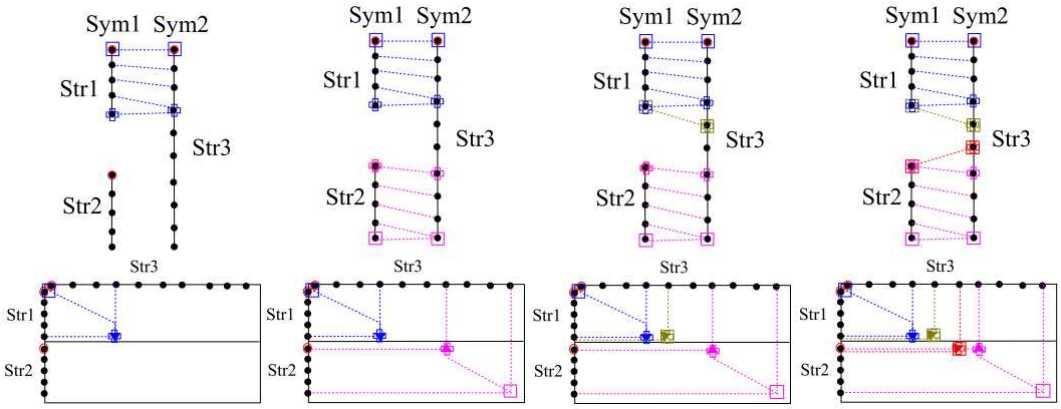
With this strategy, starting point couples are chosen to associate the two sub-sequences with respect to the continuity constraint in each step. In each step, we choose a starting point couple again from non-used points of the two strokes. The searching procedure finishes until all the points are associated in the warping path. Our objective is to find the warping path that minimizes the associated cost in Eq. (2). The distance of DTW A* is normalized by the number of couples.

Fig.4 shows the best warping path for associating two sets of point sequences. This solution contains four DTW sub-warping paths, which are obtained from Step 1 to Step 4. The matching directions are not necessary to be the same. A set of sub-warping paths which minimizes the associated cost (the sum of point-to-point distances) are searched.

However, there are a large number of possibilities. To search the best warping path, an A* algorithm [7] is used to accelerate the search as discussed in Section 2.1.

2.1 A* Algorithm

In this section, the A* algorithm (A star) [7] to limit some futureless explorations is introduced. It iteratively searches the best path in a graph from the starting node (empty associated point) to the ending node (all associated points). However, not all the possible nodes are generated because of a heuristic function in the A* algorithm. In each step, only the best hypothesis is explored for the next step.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Fig.4 A solution of warping path between two symbols (graphic and matrix views)

The A* algorithm uses a distance-plus-cost heuristic function $f(x) = g(x) + h(x)$ at each step x. The cost $g(x)$ represents the cost of the best warping path from the starting step to the current step x, and the heuristic cost $h(x)$ estimates the minimum distance to the ending step. Ref. [7] described the A* algorithm in detail. In this section, we define the two functions, namely $g(x)$ and $h(x)$, for our problem. Considering the heuristic distance $h(.)$, it should be as large as possible but equal to or less than the real optimal distance for reaching the ending step, which means that $h(.)$ is admissible.

We first define each step x by a warping path $P_x(h)=(i_x(h), j_x(h))$, $1 \le h \le H_x$, between the two symbols $S_1 = (p_1(1), p_1(2), ..., p_1(N1))$ and $S_2 = (p_2(1), p_2(2), ..., p_2(N_2))$. The warping path is a sequence of associated index pairs. Its cost is defined by the sum of pair costs:

$$g(x) = \sum_{h=1}^{H_x} dist\left(p_1(i_x(h)), p_2(j_x(h))\right). \tag{5}$$

Defining a set of non-used points NUPt(Sym, x) for a symbol Sym in step x, the heuristic cost $h(.)$ therefore can be defined by

$$h(x) = \frac{1}{2}\left(h_{sub}(x, S_1, S_2) + h_{sub}(x, S_2, S_1)\right) \tag{6}$$

where

$$h_{sub}(x, S_a, S_b) = \sum_{p_1(i) \in NUPt(S_a, x)} dist\left(p_1(i), nnp(p_1, S_b)\right) \tag{7}$$

$$nnp(p_1, S_b) = \underset{p_2(j) \in NUPt(S_b, x)}{argmin} dist\left(p_1, p_2(j)\right)$$

and . This heuristic distance $h(.)$ is admissible because we always choose the minimum distance between the two sets of non-used pair points during the association of the point pairs.

Even though the A* algorithm is used to accelerate the searching, the number of combinations is still large. In order to further reduce the combination numbers, we try to limit the number of starting point couples rather than using all the non-used point couples. This strategy

will be developed in Section 2.2.

2.2 Choosing Starting Points

In order to generate next steps from step *x*, one have to choose a non-used starting point couple, which is used for starting up two sequences with a matching in four directions in maximum. For each direction, a new step is obtained. Although the A* algorithm can reduce the searching complexity, there are still many possibilities when using all the non-used points for a next step. In this section, we propose a strategy to limit the starting point couple generation.

Defining the non-used segments in step *x* for each stroke in a symbol Sym by Segs(Sym, *x*), the boundary points of these segments are defined by FSeg(Sym, *x*). A set of new starting point couples $\{(p_i, p_j)\}$ between two symbols, S_1 and S_2, are produced from FSeg(S_1, *x*) to the closest points in Segs(S_2, *x*), and vice versa:

$$\begin{aligned}\{(p_i,p_j)\} = \{&\forall p_i \in Fseg(S_1,x), \quad \forall seg \in Segs(S_2,x), (p_i, nnp(p_i,seg))\} \\ &\cup \\ \{&\forall p_j \in Fseg(S_2,x), \quad \forall seg \in Segs(S_1,x), (nnp(p_j,seg), p_j)\}\end{aligned} \tag{8}$$

Fig.5 shows the possible starting point couples of the first step in Fig.4 which is as considered as the step *x*. In this case, there exist three starting couples, namely (P1, P6) with only one direction, (P1, P8) with two possible directions and (P5, P11) with only one possible direction. In the general case, up to four directions have to be considered. All the possibilities are explored by the A* algorithm for searching the best warping path.

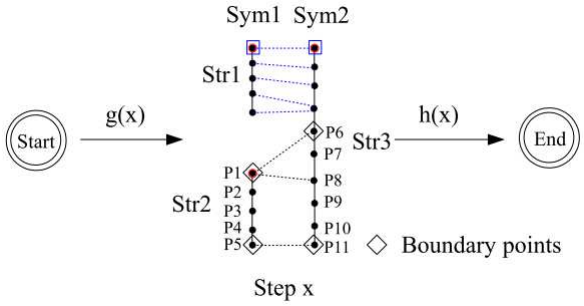


Fig.5 Three starting point couples (P1, P6), (P1, P8) and (P5, P11) for starting the second step in Fig.4

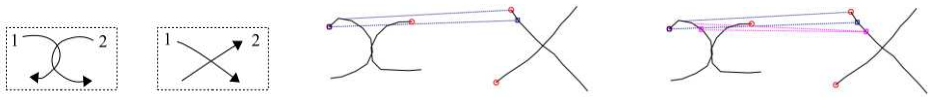
By limiting the number of starting couples, fewer branches are explored from the current step. It has two consequences: the system is faster but we cannot guarantee to select the best solution. In the next section, we will present experimental results of DTW A* algorithm.

3 Experiments

In this section, we first display a qualitative matching between two patterns so that we can intuitively understand how it works. Second, a handwriting dataset is presented. Then, the performances of three distances (DTW A*, Classical DTW and MHD) are compared on a handwriting dataset.

Fig. 6 illustrates the matching between two allographs of "x". Our algorithm finds the best solution in 5 steps which are 5 sub-warping paths. The first two steps show the point-to-point matching of the top-left branch of "x". The bottom-right branch is matched in the third step, etc.

Our algorithm can cut the strokes into sub-graphemes which minimize the DTW distance between segments from two symbols.



(a) Two different “x” (b) Step 1 (c) Step 2

(d) Step 3 (e) Step 4 (f) Step 5

Fig. 6 The best solution of two “x”

In order to assess the recognition efficiency of DTW A*, we first present a handwriting dataset evaluated by a *k*-NN classifier. The handwriting dataset is a realistic handwritten flowchart dataset named FC dataset [8]. We also use six different graphical symbols (six classes) that represent the basic operations (data, terminator, process, decision, connection and arrow) without any handwritten text, as displayed in Fig. 7. This data set contains a training part (3641 symbols) and a test part (2494 symbols). In average, each symbol contains 2.4 strokes. As we can see, most symbols on the \textit{FC} dataset are composed of more than one stroke. People may write strokes in a symbol with different orders and directions.

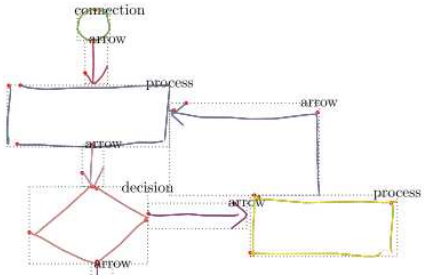


Fig. 7 An example of flowchart on the FC dataset

Table 2 shows the recognition rates of three algorithms. DTW A* is slightly better (97.47%) than the classical DTW (96.79%), and more surprisingly MHD is also very efficient (97.31%). One explanation is that, for flowcharts, the sequence information is very irrelevant, and that it is better not to rely on. With the classical DTW, the strokes from one symbol are just concatenated, so that the time information is strongly kept. Thus, when a symbol is written in a different stroke order, the same stroke order should exist in the train dataset of the *k*-NN. To study the sensitivity with respect to the training size set, a 5-fold cross validation with a smaller training set size is also proposed. We can notice that DTW A* is quite stable (96.90%) although only one fifth of the training samples are available. Conversely, the performance of the classical DTW drops to 91.33%, in that case missing samples of the training set are not compensated by the flexibility of the matching process.

Even though we have optimized a lot for the A* algorithm in terms of time, it is still time-consuming and memory-consuming (storing a large number of hypotheses). In average, each comparison takes 0.04 s, but some of them consume several seconds.

Table 2 Results of k-NN classification (k = 5) and cross-validation on the FC dataset %

	DTW A*	Classical DTW	MHD
Normal *k*-NN	97.47	96.79	97.31
Cross-Validation	**96.90**	**91.33**	**95.65**
Decrease	**0.57**	**4.86**	**1.66**

4 Conclusions

As the brute force method produces a large number of possibilities, a distance between two sets of sequences is proposed, which keeps the continuity during each small matching. The proposed distance, namely DTW A*, uses the A* algorithm to reduce the complexity and only the promising candidates are considered. In addition, we limit the starting point couples. By the test on the FC dataset which contains flowchart symbols, it is found that the proposed DTW A* distance slightly outperforms the classical distances of DTW and MHD. The DTW A* remains quite stable during the cross-validation test. However, it is still not fast enough in practical usage. Further limitation of the starting point couples is a possible solution.

References:

[1] Vuori V. Adaptive methods for on-line recognition of isolated handwritten characters[D]. Espoo:Department of Computer Science and Engineering, Helsinki University of Technology，2002.

[2] Uchida S , Sakoe H. A survey of elastic matching techniques for handwritten character recognition[J]. The Institute of Electronics, Information and Communication Engineers (IEICE) Transactions, 2005， 88-D(8):1781–1790.

[3] Tan G X, Viard-Gaudin C, Kot A C. Automatic writer identification framework for online handwritten documents using character prototypes[J]. Pattern Recognition, 2009, 42(12): 3313–3323.

[4] Huttenlocher D P, Klanderman G A, Rucklidge W A. Comparing images using the Hausdorff distance[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,1993, 15(9): 850–863.

[5] Dubuisson M P, Jain A K.A modified Hausdorff distance for object matching[C]// Proceedings of the 12th IAPR International Conference onPattern Recongnition. Jerusalem:IEEE, 1994:566-568.

[6] Ding K, Deng G, Jin L. An investigation of imaginary stroke techinique for cursive online handwriting Chinese character recognition[C]// Proceedings of the 10th International Conference on Document Analysis and Recognition. Barcelona:IEEE Computer Society, 2009:531–535.

[7] Hart P, Nilsson N, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science and Cybernetics,1968, 4(2):100–107.

[8] Awal A M, Feng G, Mouchère H, et al.First experiments on a new online handwritten flowchart database[C]// Document Recognition and Retrieval XVIII. San Fransisco: États-Unis, 2011:1-10.