Recovering Dynamic Stroke Information of Multi-Stroke Handwritten Characters with Complex Patterns

Takayuki Nagoya
Humanities Center
Tottori University of Environmental Studies
Tottori 689-1111, Japan
Email: nagoya@kankyo-u.ac.jp

Hiroyuki Fujioka
Department of System Management
Fukuoka Institute of Technology
Fukuoka 811-0295, Japan
Email: fujioka@fit.ac.jp

***Abstract*—In this paper, we consider the problem of recovering dynamic stroke information from multi-stroke handwritten character images with complex patterns. The characters are assumed to be of a class of characters whose strokes are recursively formulated. By employing the so-called graph theoretic approach, we develop a systematic algorithm for recovering dynamic stroke information from character images in the class of our interest. It is shown that the correctness of algorithm is guaranteed mathematically. Moreover, we show that the time complexity becomes $O(n)$, where n denotes the number of stroke-intersections on characters. Some recovery examples are included.**

***Keywords*-stroke recovery; graph theoretic approach; multi-stroke handwritten character; recursive formulation**

I. Introduction

Recovering the dynamic stroke information from static character images is a key problem with wide range applications of off-line handwritten character recognition (see e.g.[1]). A standard approach of studying such a problem is by using the so-called graph theoretic approach (e.g. [2] and [3]). That is, by representing the input characters as some undirected graph, the dynamic stroke information is recovered by analyzing the graph based on the global or local criteria. Thus, a variety of recovery algorithms have been developed.

For example, Kato and Yasuhara in [4] have developed a hybrid-type algorithm using both the global and local criteria. The time complexity is $O(n^2)$, where n denotes the number of stroke-intersections on characters. However, the recovery algorithm often fail to recovery the character strokes with complex pattern – such as the case where the loop structure of characters recursively contains smaller ones, etc. Then, one of interesting issues in this regard may be to reveal a question "*What class of characters can or not be recovered with a minimum set of heuristic rules ?*".

The main purpose of this paper is to explore such a question based on the Kato and Yasuhara's work in [4], so that a class of characters with complex patterns can be recovered well. Specifically, we consider a class of characters whose strokes are recursively formulated, in which the class of characters treated in [4] is included. We first present a recursive formulation of single- and multi-stroke characters. Based on the results, we develop a systematic algorithm which can recover dynamic stroke information from character images in the class of our interest. Then, it is shown that the correctness of our algorithm is guaranteed mathematically. Moreover, we show that the time complexity of our algorithm becomes $O(n)$. Clearly, our method improves Kato and Yasuhara's work [4], and then may become a powerful tool for the recovery problems of character stroke. Some recovery examples are included.

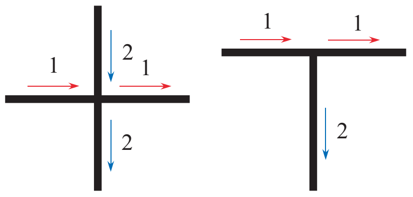
This paper is organized as follows. In Section II, we briefly review some assumptions on the structure of character images. The formal definition on single- and multi-strokes is presented in Section III. In Section IV, we show how a graph model is constructed from a given character image. In Section V, we then develop an algorithm for recovering dynamic stroke information of multi-stroke character images, and some main theorems are summarized in Section VI. Concluding remarks are given in Section VII.

II. General Assumptions

Supposing that handwritten characters are scanned and stored as binary image data, we here assume the structure of character images as follows:

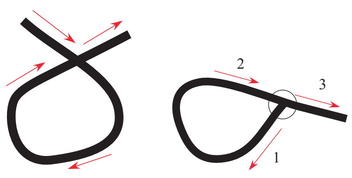
Assumption 1. (i) *Characters consist of combinations of single-strokes, i.e. multi-strokes.*
(ii) *Any single-stroke has start and end points of writing, but the location of their points are nonidentical and unknown.*
(iii) *A single-stroke may intersect with other single-strokes.*
(iv) *A single-stroke may intersect with itself.*

By Assumption 1, we restrict ourselves to consider the character with the following types of stroke structure. First, in Assumption 1 (iii), it may be natural to consider two types of intersection in Figure 1. S-crossing in Figure 1 (a) illustrates a case where a stroke intersects with another stroke. T-crossing in Figure 1 (b) is a special case of S-crossing, in which a terminal point, i.e. start or end point,



(a) S-crossing (b) T-crossing

Figure 1. Basic crossings.



(a) S-loop (b) T-loop

Figure 2. Loop structures.

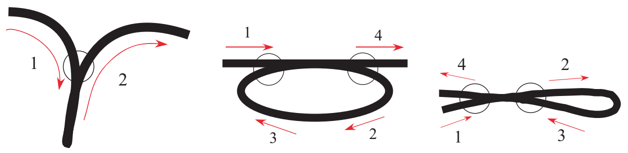
is on another stroke. We then assume that two strokes in T-crossing cross almost perpendicularly. We refer these intersections as 'basic crossings'.

On the other hand, Assumption 1 (iv) allows us that a single-stroke generates the loop structures in Figure 2. We then see that the loop structures are same as ones of basic crossings, and the difference between them is that intersection is constructed by a single-stroke self or two single-strokes. As special cases of loop structure, we may face the case where some part of a single-stroke may be drawn twice. Such lines are called 'double-traced lines' [4]. We call the crossing structures derived from double-traced lines 'double-traced structures', and we here consider three types in Figure 3. D-line in Figure 3 (a) is yielded by a pen movement that is immediately followed by its reverse movement. M-loop in Figure 3 (b) and L-loop in Figure 3 (c) are special cases of S-loop where the crossing angle between two lines is very small. In addition, we consider special types of S-loop and T-loop. That is, we allow S-loops at which three or more lines intersect as shown in Figure 4 (a). Also, a double-traced line of D-line can be a part of a S-loop and T-loop as shown in Figure 4 (b) and (c). As you can see from Figure 2 and Figure 3, there are three-way branches in T-loop, D-line, M-loop and L-loop as denoted by circle marks. These points are called 'odd point' in the follows.

III. Formal Definition

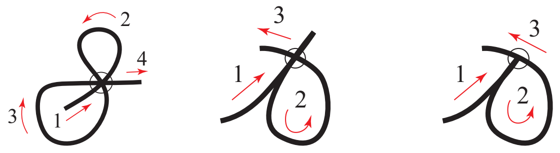
We next present formal definitions on single- and multi-strokes considered in this paper. Such definitions may enable us to recover the dynamic stroke information of multi-stroke characters with complex patterns.

From Section 2, we see that each crossing structure is the most fundamental single-stroke. Then, we may compose a



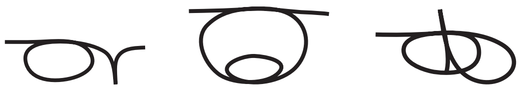
(a) D-line (b) M-loop (c) L-loop

Figure 3. Double-traced structures.



(a) S-loop (b) S-loop with D-line (c) T-loop with D-line

Figure 4. Special types of loop structures.



(a) S-composition (b) P-composition (c) U-composition

Figure 5. Compositions of crossing structures.

single-stroke with larger size by combining some crossing structures. As types of such composition, we here consider the following three types of composition: (a) series composition, (b) parallel composition and (c) unified composition, as shown in Figure 5. First, (a) series composition, denoted as S-composition, is to combines two crossing structures by connecting their terminals. In Figure 5 (a), note that S-composition consists of M-loop and D-line. Next, (b) parallel composition, denoted as P-composition, is to construct parent-child relationship between two crossing structures. That is, a crossing structure may exists in loop segment of other crossing structure, but any loop and double-traced structures except D-line could be a parent. Also, any loop and double-traced structures except T-loop could be a child. Figure 5 (b) is an example of a P-composition with two M-loops. Finally, (c) unified composition, denoted as U-composition, is composition such that a crossing structure intersects with other crossing structures at some non-odd points by S-crossing or T-crossing. Figure 5 (c) illustrates an example of an U-composition of M-loop and D-line which intersect by S-crossings and one T-crossing.

From the above results, we get the following formal definition on single-stroke and multi-stroke.

Definition 1. *Single-strokes are defined recursively as follows:*

(i) *Each loop structure in Figure 2 and Figure 4 and each double-traced structure in Figure 3 is a single-stroke.*

(ii) *If I_1 and I_2 are two single-strokes, then S-, P-, and U-composition of them is a single-stroke.*

Definition 2. *Multi-strokes are defined recursively as follows:*

(i) *Any single-stroke is a multi-stroke.*

(ii) *If I_1 and I_2 are two multi-strokes, then their combination is multi-stroke if I_1 and I_2 intersect arbitrary times at some non-odd points with S-crossing.*

(iii) *If I_1 and I_2 are two multi-strokes, then their combination is multi-stroke if I_1 and I_2 intersect with T-crossing such that a terminal of I_1 is on a line segment (i.e. non-odd point) of I_2.*

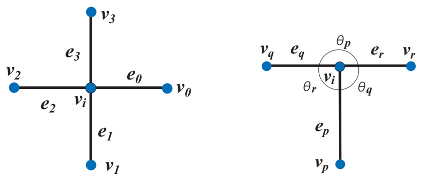
IV. Constructing Graph

For recovering dynamic stroke information from a given handwritten image I, we need to construct an undirected graph G (see [5]). Such graph G is constructed as skeleton image I_s by employing the thinning method of image processing techniques. We here construct a graph G from input I by employing the Zhang-Suen's thinning algorithm together with Stentiford preprocessing. A set of edges and vertices of graph G is defined by $E(G) = \{e_i,\ i = 1, 2, \cdots, m\}$ and $V(G) = \{v_i,\ \ i = 1, 2, \cdots, n\}$ respectively. Each edge $e_i \in E(G)$ corresponds a line segment in the skeleton, where the the sequence of coordinate points on the line segment is labeled. Each vertex $v_i \in V(G)$ corresponds to a geometrical feature point, i.e. terminal point and crossing point. We here refer the sequence of coordinate points on e_k from v_i to v_j as $\mathrm{c}(v_i, e_k, v_j)$. The concatenation of two coordinate point data $\mathrm{c}(v_i, e_j, v_k)$ and $\mathrm{c}(v_s, e_t, v_u)$ is denoted as $\mathrm{c}(v_i, e_j, v_k)|\mathrm{c}(v_s, e_t, v_u)$.

Figures 6-8 show the graphs constructed from the crossing structures in Figures 1-3 respectively. From Figure 6 (a) and (b), we see that the graphs of S-crossing and T-crossing have a vertex and edges that incident to the vertex. In Figure 7 (a) and (b), we observe that the graphs of S-loop and T-loop have single vertex with a self-loop. In Figure 8 (a), the graph of D-line may have two vertices and an edge, where the edge is a double-traced line. Similarly, M-loop in Figure 3 (b) may yield the graph with two vertices and multiple-edges connecting them as shown in Figure 8 (b), where one of the edges is a double-traced line. L-loop in Figure 3 (c) also yield the graph with two vertices as shown in Figure 8 (c) where the edge connecting them is a double-traced line and one of the vertices has a self-loop.

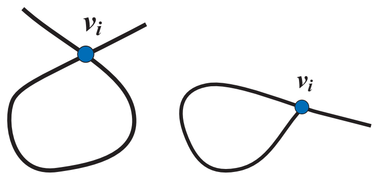
Since there is no branch at vertex of degree two, without loss of generality, we assume that there is no vertex with degree two in the graph. The next lemma is useful to prove the correctness of our algorithm which is developed in next section.

Lemma 1. *Let G be a graph obtained from a given handwritten character image I. Then, the graph G is planar. Furthermore, G has neither a vertex of odd degree greater than three nor a vertex of degree two.*



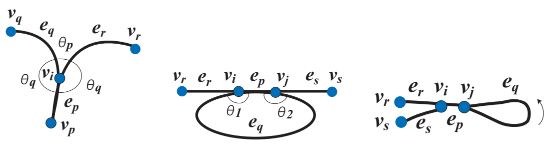
(a) S-crossing (b) T-crossing

Figure 6. Graph representations corresponding to Figure 1.



(a) S-loop (b) T-loop

Figure 7. Graph representations corresponding to Figure 2.



(a) D-line (b) M-loop (c) L-loop

Figure 8. Graph representations corresponding to Figure 3

Sketch of proof: The former assertion is obvious, i.e. G is planar. That is, G can be drawn in a plane without any edge intersections. The latter assertion can be proved by using the mathematical induction on the number of composition operations. □

V. Recovery Algorithm of Dynamic Stroke Information

Based on the foregoing development, we here develop an algorithm for recovering dynamic stroke information of multi-stroke character image with complex patterns. The algorithm consists of the following two steps: (S1) global labeling and (S2) local labeling. Main objective of (S1) is to detect each single-stroke and separate the graph G into connected graphs U_k, $k = 1, 2, \cdots$, where each U_k corresponds to a single-stroke. Then the recovering problem reduces to the problems for single-strokes. We thus recover drawing order of each single-stroke by (S2). Specifically, by detecting all the types of graph structure in U_k – such as D-line, M-loop and L-loop, we recover drawing order of them by using the edge contraction operation which merges two adjacent edges to an edge. As a result, all the U_k is transformed into a graph that has exactly two vertices and one edge connecting them such that the edge represents whole of the single-stroke and the vertices correspond to its start and end points.

We are now in the position to develop (S1) global labeling and (S2) local labeling methods.

A. Global labeling

From Section III, we may see that any single-strokes intersect with other strokes by only the basic crossings, i.e. S-crossing and T-crossing in Figure 1. Then, in order to decompose the graph G into some connected graphs $U_k,\ k = 1, 2, \cdots$ corresponding to single-strokes, we have only to detect S-crossing and T-crossing. Then, the connected graphs $U_k,\ k = 1, 2, \cdots$ are labeled.

First, we detect S-crossings from the graph G. Let $v_i \in V$ be a vertex with even degree, i.e. $\deg(v_i) = 0 (\bmod\ 2)$. Moreover, letting $v_j, j = 0, 1, \ldots, \deg(v_i) - 1$ be a vertex adjacent to v_i, an edge e_j between v_i and v_j is written as $e_j = (v_i, v_j)$. Without loss of generality, we here assume that $(e_0, e_1, \ldots, e_{\deg(v_i)-1})$ is a sequence of incident edges to v_i in clockwise order around v_i. Then, our task is to detect the pair of e_{k_1} and e_{k_2} $(k_1 \neq k_2)$ for $k_1, k_2 = 0, 1, \ldots, \deg(v_i) - 1$such that natural strokes is achieved. Such pairs of edges can be readily detected by employing an idea of "middle-edge" in [4], and we have only to find a pair $(e_k, e_{\deg(v_i)/2+k}),\ k = 0, 1, \cdots, \deg(v_i)/2 - 1$. By directly connecting the edges e_k and $e_{\deg(v_i)/2+k}$ without v_i, we consequently eliminate any branch of stroke at v_i. Hence, graph structure of S-crossings is detected and the corresponding dynamic stroke information is recovered. Note that, by this procedure, graph structure of S-loops is also recovered at the same time since it is quite same as one of S-crossing.

Next, we detect T-crossing in Figure 6 (b). By the above procedure, any vertices with even degree are removed. Thus, the degree of remaining vertices is one or three. From Section IV, it is clear that such a vertex is one of the five types of crossing structure, i.e. T-crossing, T-loop, D-line, M-loop and L-loop. However, we notice that T-crossing satisfies the condition so that three angles θ_p, θ_q, and θ_r constituted by the three edges e_p, e_q and e_r are approximately equal to π, $\frac{\pi}{2}$ and $\frac{\pi}{2}$ respectively (see Figure 6 (b)). Thus, by employing the condition, the graph structure of T-crossings can be detected and the corresponding stroke information is recovered. Moreover, the graph structure of T-loop is simultaneously detected and recovered by this procedure.

B. Local labeling

By the global labeling, the graph G is decomposed into connected graphs $U_k,\ k = 1, 2, \cdots$ corresponding to single-strokes of a given character image. Therefore, our problem is reduced to one for each single-stroke $U_k,\ k = 1, 2, \cdots$. Now, note that the graph structures corresponding to S-loop and T-loop on U_k have already been recovered by global labeling. Thus, the remaining types of graph structure on the graph U_k may be ones of D-line, M-loop and L-loop. But, when the handwritten characters have complex patterns, M-loop and L-loop may be contained as a part of P-composition in

Algorithm 1 Stroke order of D-line

```
Require: connected graphs $U_k, k = 1, 2, \ldots$.
Ensure: recover stroke order of D-lines.
  for all $U_k$ in arbitrary order do
    Compute the vertex set $V_3 \subseteq V(U_k)$ of degree three that
    has at least one neighbor of degree one.
    for all vertex $v_i \in V_3$ in arbitrary order do
      Let $e_p, e_q, e_r, \theta_p, \theta_q$ and $\theta_r$ be as shown in Figure 8
      (a). Let $\theta_p$ be the smallest among them.
      if $\deg(v_p) = 1$ then
        Add new edge $e'$ that connects $v_q$ and $v_r$.
        Set $c(v_q, e', v_r) = c(v_q, e_q, v_i)|c(v_i, e_p, v_p)$
        $|c(v_p, e_p, v_i)|c(v_i, e_r, v_r)$.
        Remove $e_p, e_q, e_r, v_p$, and $v_i$.
      end if
    end for
  end for
```

Figure 5. We here propose the algorithms for detecting and recovering D-line, M-loop and L-loop structures including even the cases where P-composition exists. Then, the whole dynamic stroke information of $U_k,\ k = 1, 2, \cdots$, i.e. a given character I, can be recovered.

1) Detecting and Recovering D-line: Let v_i be a vertex with $\deg(v_i) = 3$ and let v_p, v_q, and v_r be vertices adjacent to v_i respectively as shown in Figure 8 (a). Also, let $e_p = (v_i, v_p), e_q = (v_i, v_q)$, and $e_r = (v_i, v_r)$ be incident edges to v_i. Then, it is obvious that the vertex v_i is an odd point on a graph structure of D-line, M-loop or L-loop. However, as shown in Figure 4, such an odd point in D-line does not have self-loop by itself or multiple-edge between itself and another vertex. Moreover, noting that at least one of v_p, v_q, and v_r is a vertex with degree one, we see that the connecting edge between such a vertex and v_i is double-traced line. Utilizing this feature, the graph structure of D-line is readily detected and recovered as follows: Let us consider the case where a vertex v_i has $\deg(v_i) = 3$ and at least one of adjacent vertices v_p, v_q, and v_r is with degree one. If only one of v_p, v_q, and v_r is of degree one, the connecting edge between such a vertex and v_i may be double-traced line. Otherwise, their vertices may include terminal point (i.e. start or end points) of a single stroke. In order to determine the edge-double traced line, we then evaluate the three angles θ_p, θ_q and θ_r constituted by the three edges e_p, e_q and e_r. For example, if $\theta_p \ll \theta_q, \theta_r$ and $\deg(v_p) = 1$, then e_p is double-traced line. We thus get the algorithm in Algorithm 1 for detecting and recovering the graph structure of D-line from the connected graphs $U_k,\ k = 1, 2, \cdots$.

2) Detecting and Recovering M-loop and L-loop: By the procedure in Section V-B1, the remaining types of graph structure in $U_k,\ k = 1, 2, \cdots$ may be those of M-loops and L-loops. Thus, when the vertex v_i with $\deg(v_i) = 3$ exists on

U_k, it may indicate that the graph structures of M-loop or L-loop is contained.

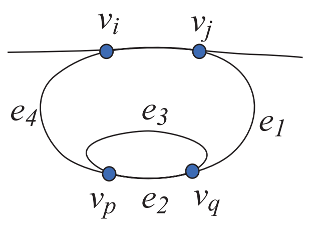
It is clear that a vertex v_i is one in M-loop if and only if v_i is the vertex with $\deg(v_i) = 3$ and there exists a vertex $v_j (j \neq i)$ with $\deg(v_j) = 3$, where v_j is adjacent to v_i with multiple-edges. Let e_p and e_q be the multiple-edges connecting with v_i and v_j, where $|\mathrm{c}(v_i, e_p, v_j)| < |\mathrm{c}(v_i, e_q, v_j)|$. Then, letting θ_1 and θ_2 be two angles between e_p and e_q, we note that both θ_1 and θ_2 are not small. On the other hand, if v_i does not only have just an adjacent vertex, i.e. v_j with $\deg(v_j) = 3$ but also self-loop, it is clear that v_i is of L-loop. Thus, for detecting the graph structures of M-loop and L-loop from U_k, we have only to detect v_i with multiple edge and self-loop, respectively.

However, when the character images I are given as ones with complex patterns, M-loop and L-loop may contain other smaller ones. That is, the graph structure of P-composition is contained in U_k, and then the so-called parent-child relationship is constructed by the combinations of $\{\mathrm{M-loop}, \mathrm{L-loop}\}$. One of way for solving such a case may be to detect and recover the graph structure corresponding to the children, i.e. smaller M-loop or L-loop, in first step. We then detect and recover the graph structure corresponding to the parent, i.e. larger M-loop or L-loop.

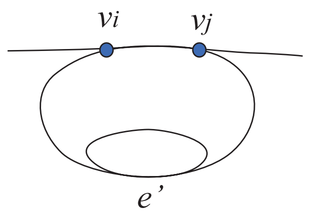
Figure 9 illustrates an example of the above procedures, where both parent and children are M-loop. In this example, we here notice that the parent M-loop on v_i and v_j has no multiple-edges as in shown Figure 9 (a). Thus, we firstly need to detect child crossing structures and remove their vertices by merging edges e_1, e_2, e_3 and e_4 to an edge e' as shown in Figure 9 (b). Hence, the stroke information of child M-loop is recovered. By the same procedure, the parent M-loop is also recovered. The case of other combinations of $\{\mathrm{M-loop}, \mathrm{L-loop}\}$ can be similarly detected and recovered.

In general, the parent-child relations may form rooted tree, where each node of the tree is either M-loop or L-loop. We then need to detect and recover M-loop and L-loop by a bottom-up approach. Thus, we can get Algorithm 2, in which M-loop and L-loop can be detected and recovered iteratively. In the algorithm, queues *Queue-M* and *Queue-L* are used to store M-loops and L-loops respectively, Here, 'Enqueue' is an operation to insert M-loop or L-loop at the end of *Queue-M* or *Queue-L*, respectively. 'Dequeue' is an operation to remove and return the element (v_i, v_j) at the top of *Queue-M* or *Queue-L*. Moreover, in Algorithm 2, 'RecLoop(α)' for $\alpha \in \{\mathrm{M-loop}, \mathrm{L-loop}\}$ denotes an algorithm for recovering the dynamic stroke information corresponding to an M-loop or an L-loop, and it is given as Algorithm 3 .

Figure 10 shows two recovery examples by the proposed algorithm. In order to examine the effectiveness of proposed algorithm, we here used characters which is hardly appeared in the normal handwritings. Here, the original character images in black lines are stored by employing a pen-tablet device. Also, green and magenta lines are paths on the corresponding connected graphs U_1 and U_2 respectively. Moreover, the arrow marks denote the drawing order, where the arrow-head direction has been chosen by employing the heuristic rule based on natural writing behavior of top-to-bottom and left-to-right. From these results, we may observe that our algorithm correctly recovers dynamic stroke information even though the character images contain complex patterns.



(a) P-composition of M-loops



(b) $e_1 \sim e_4$ are merged to e'

Figure 9. Detection of P-composition.

Algorithm 2 Stroke order of P-compositions

```
Require: connected graphs $U_k, k = 1, 2, \ldots$.
Ensure: recover stroke order of M-loops and L-loops.
  for all $U_k$ in arbitrary order do
    Find all M-loops $(v_i, v_j)$ in $U_k$ and call Enqueue($(v_i, v_j)$, Queue-M).
    Find all self-loops $(v_i, v_i)$ in $U_k$ and the neighbor $v_j$ of $v_i$, and call Enqueue($(v_i, v_j)$, Queue-L).
    while Queue-M $\neq \emptyset$ or Queue-L $\neq \emptyset$ do
      if Queue-M $\neq \emptyset$ then
        Call Algorithm 3 with input Dequeue(Queue-M), Queue-M and Queue-L.
      end if
      if Queue-L $\neq \emptyset$ then
        Call Algorithm 3 with input Dequeue(Queue-L), Queue-M and Queue-L.
      end if
    end while
  end for
```

Algorithm 3 **Stroke order of α-loop for** $\alpha \in \{\mathrm{M-loop}, \mathrm{L-loop}\}$

```
Require: α-loop $(v_i, v_j)$, Queue-M and Queue-L.
Ensure: recover stroke order of the given α-loop, and
  enqueue its parent loop structure, if any.
  Add new edge $e'$ that connects $v_r$ and $v_s$.
  if $\alpha$ =M-loop then
    Set $c(v_r, e', v_s) = c(v_r, e_r, v_i)|c(v_i, e_p, v_j)\ |c(v_j, e_q, v_i)|c(v_i, e_p, v_j)|c(v_j, e_s, v_s)$.
  else if $\alpha$ =L-loop then
    Set $c(v_r, e', v_s) = c(v_r, e_r, v_i)|c(v_i, e_p, v_j)\ |c(v_j, e_q, v_j)|c(v_j, e_p, v_i)|c(v_i, e_s, v_s)$.
  end if
  Remove $e_p, e_q, e_r, e_s, v_i$ and $v_j$.
  if $(v_r, v_s)$ is a parent M-loop of $(v_i, v_j)$ then
    Enqueue($(v_r, v_s)$, Queue-M).
  end if
  if $v_r = v_s$ and $(v_r, v_t)$ is a parent L-loop of $(v_i, v_j)$ then
    Enqueue($(v_r, v_t)$, Queue-L).
  end if
```

VI. Main Theorems

For the results in previous section, we get the following two theorems.

Theorem 1. *The algorithm correctly recovers dynamic stroke information for any multi-stroke character images defined by Definition2.*

Sketch of proof: We readily see that the global labeling step works correctly. We can prove that the local labeling step works correctly by mathematical induction on tree structure of parent-child relation of P-composition, from the leaves upwards. □

Theorem 2. *The time complexity of the algorithm is $O(n)$, where n is the number of vertices of the graph G.*

Sketch of proof: We can prove that the number m of edges of a simple planar graph with $n \geq 3$ vertices satisfies $m \leq 3n - 6$ by using the Euler's formula. From this fact, we can show that the cost of processing each crossing structures is $O(n)$. Since the algorithm process each crossing structure at once, we can prove that the time complexity of the algorithm is $O(n)$. □

VII. Concluding Remarks

In this paper, we considered the problem of recovering dynamic stroke information from multi-stroke handwritten character images. In particular, we extended the Kato and Yasuhara's work in [4], so that a class of characters whose strokes are recursively formulated can be recovered. Then we presented a formulation on such single- and multi-characters. Moreover, we developed a systematic algorithm which can recover dynamic stroke information from character images in the class of our interest. It was shown that the correctness of our algorithm is not only guaranteed mathematically, but the time complexity of our algorithm also becomes $O(n)$. We examined the performances of the developed recovery method by some examples. To conclude, the developed method is effective as well as powerful for the recovery problems of character strokes.

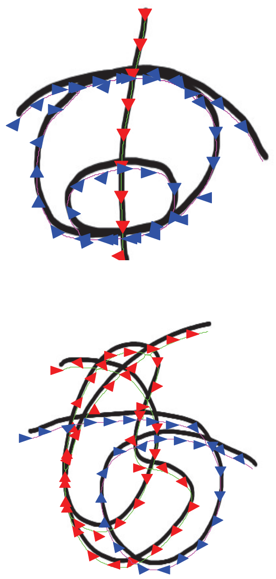


Figure 10. Recovery examples.

References

[1] R. Plamondon and S. N. Srihari, Online and off-line handwriting recognition: a comprehensive survey, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.22, no.1, pp.63-84, 2000.

[2] S. Lee and J. C. Pan, Offline Tracing and Representation of Signatures, *IEEE Trans. Systems, Man, and Cybernetics*, vol.22, no.4, pp.755–771, 1992.

[3] S. Jäger, Recovering Writing Traces in Off-Line Handwriting Recognition: Using a Global Optimization Technique, *Proc. of 13th Int. Conf. on Pattern Recognition*, pp. 931-935, Vienna, Aug.25–29, 1996.

[4] Y. Kato and M. Yasuhara, Recovery of drawing order from single-stroke handwriting images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.22, No.9, pp.938–949, 2000.

[5] H. Fujioka and T. Nagoya, Recovering Stroke Order from Multi-Stroke Character Images, *Proc. of the 2011 2nd Int. Conf. on Innovative Computing and Communication, and 2011 2nd Asia-Pacific Conf. on Information Technology and Ocean Engineering*, pp.34-37, Macao, March 5-6, 2011.

[6] J. R. Parker, *Algorithms for Image Processing and Computer Vision*, Second Edition, Wiley Publishing Inc., 2011.