Journal of Mobile Multimedia, Vol. 8, No. 2 (2012) 073–087

RECOVERING DRAWING ORDER OF SINGLE-STROKE HANDWRITTEN IMAGES USING PROBABILISTIC TABU SEARCH

TAKAYUKI NAGOYA
Humanities Center, Tottori University of Environmental Studies
1-1-1 Wakabadai-Kita, Tottori, Tottori 689-1111, Japan
nagoya@kankyo-u.ac.jp

HIROYUKI FUJIOKA
Department of System Management, Fukuoka Institute of Technology
3-3-1 Wajiro-Higashi, Higashi-ku, Fukuoka 811-0295, Japan
fujioka@fit.ac.jp

Received November 1, 2011
Revised May 9, 2012

This paper considers the problem for recovering a drawing order of static handwritten images with single stroke. Such a stroke may include the so-called double-traced lines (D-lines). The problem is analyzed and solved by employing the graph theoretic approach. Then the central issue is to obtain the smoothest path of stroke from a graph model of input handwritten images. First, the graph model is constructed from an input images by image processing techniques. Then, we locally analyze the structure of graph. In particular, the method to identify D-lines is developed by introducing the idea of 'D-line index'. The method enables us to transform any graphs with D-lines to semi-Eulerian graphs. Then, the restoration problem reduces to the problem of globally computing the maximum weight collection of perfect matchings. For solving such a problem, we propose a method using a probabilistic tabu search algorithm. The effectiveness and usefulness of the proposed method are examined by some experimental studies.

Keywords: Recovery drawing order, handwritten characters, graph theory, tabu search.
Communicated by: D. Taniar

1 Introduction

Recovering drawing order of static handwritten images is a key technology for the wide range of applications – such as off-line character recognition, character structure analysis, and design of character fonts, etc. Thus such a recovering problem have been studied extensively, and various algorithm for solving the problems have been developed. Such algorithms may be mainly classified as (i) local tracing, (ii) global tracing, and (iii) hybrid tracing methods.

A typical example of local tracing methods in (i) is a heuristic rule method [1]. Applying some heuristic rules at any points where some stokes intersect, direction of drawing order is decided one after another. Such a local tracing algorithm usually have low computational cost, but is very sensitive to noises. In order to improve the robustness on noises, the global tracing methods in (ii) has been developed [2, 3]. By employing the graph theory, the recovering problems are formulated as some combinatorial optimization ones on a graph. Then, it

has been shown that such problems reduce to the Traveling Salesman problem or Chinese Postman problem. However, this approach may lead to computational explosion as the given handwritten image becomes complex. Moreover, in order to resolve the issues of above two methods, hybrid tracing methods in (iii) using graph theoretic algorithm have recently been studied (see e.g. [4, 5, 6]). These methods consists of two steps processes as follows: First, graph structure is locally analyzed at every crossing points on given handwritten image. From such information on the graph structure, we judge the types of corresponding edge and vertex based on some template models. Then, the labeling information of stroke is obtained and we get the drawing order of handwritten images. However, this approach fairly depends on the local structures of the given image. Thus, it is not guaranteed to obtain the smoothest drawing order of the possible ones.

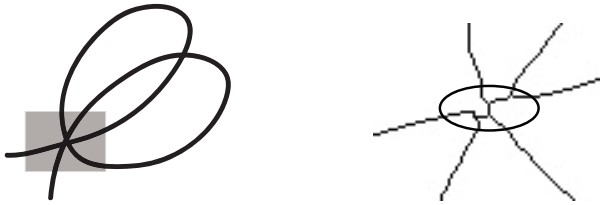
In this paper, we develop a new method for recovering drawing order of static handwritten images with single stroke. Such a stroke may include the so-called double-traced lines which is referred as 'D-lines'. The problem is analyzed and solved by employing the graph theory. Then the central issue is to obtain the smoothest path of stroke from a graph of a given handwriting image. First, an undirected graph is constructed from an input handwriting image by employing some image processing techniques. Then, the drawing order is recovered by analysing the obtained graph. In particular, the method to identify D-lines is developed by introducing the idea of 'D-line index'. The method enables us to transform any graph models including D-lines to semi-Eulerian graph models. Then, the smoothness of Euler path is formulated by the weight of collection of perfect matchings in a continuity graph. Thus, the restoration problem reduces to maximum weight perfect matching problem of graph. For solving such a problem, we propose a method using a probabilistic tabu search algorithm. The effectiveness and usefulness are examined by some experimental studies.

This paper is organized as follows. In Section 2, we briefly present the method to obtain a graph model from input handwritten image. In Section 3, we show how double-traced lines can be identified. Then, in Section 4, we develop a method for finding the smoothest drawing order of the possible ones by employing a probabilistic tabu search algorithm. We examine the performances of the proposed method by experimental studies in Section 5. Concluding remarks are given in Section 6.

2 Constructing Graph Model

In this section, we briefly present the method to obtain a graph model from input handwritten image with single stoke whose start and end points are not identical. Here, we assume that the input images are stored as binary image data, where any blurred and scratched points are removed.

Now, let I and I_s be an input handwritten image and a corresponding skeleton image, respectively. Also, let G be an undirected graph with a set of vertex $V(G) = \{v_1, v_2, \cdots, v_n\}$ and a set of edge $E(G) = \{e_1, e_2, \cdots, e_m\}$. Then, the graph G is constructed from the input I as follows: First, we obtain the skeleton I_s from input I by employing some thinning algorithm. As seen in a lot of classic problems with thinning algorithm, the skeleton I_s may be distorted due to small irregularities in the input I. We thus apply some smoothing filter to the input I before applying thinning algorithm in order to remove the artifacts. In order to obtain the skeleton I_s from the input I, we here used the Zhang-Suen's thinning algorithm



(a) Stroke with intersection at narrow point (b) R-edges and S-edges

Fig. 1. Extracting C-vertex.

together with Stentiford preprocessing (see e.g. [7, 8]).

Next, undirected graph G is constructed from the skeleton I_s. Then, our task is to extract a set of edges $E(G)$ and a set of vertices $V(G)$ from the skeleton I_s. Each edge $e_i \in E(G),\ i = 1, 2, \cdots, m$ is a segment in skeleton. Each vertex $v_i \in V(G),\ i = 1, 2, \cdots, n$ corresponds to a geometrical feature point at which an edge terminates or two or more edges are connected. Such two types of geometrical feature points are referred as 'T-vertex' and 'C-vertex' respectively in the sequel. T-vertex corresponds to a terminal of a line of the skeleton. Then, we may readily extract T-vertices since they consist of only a pixel having one 8-connected neighbor. Extracting C-vertex may however not be easy due to the thinning process (see Fig. 1). For example, let us consider the case where a stroke in input I intersects at some narrow point (see Fig. 1 (a)). Then, in the constructed skeleton I_s, the corresponding intersection point may be stretched into a set of small segments (see Fig. 1 (b)). Thus, the skeleton around C-vertex is constituted by two kinds of edge: 'Real edge (R-edge)' and 'Spurious edge (S-edge)'. Here, R-edge is an edge corresponding to real stroke in the input I. S-edge is an extra edge yielded by thinning process. Since S-edge never exists in input I, S-edge may distort the structure of handwriting image in I. We thus need to differentiate S-edges from R-edges. The identification of S-edge can be done by the so-called double threshold method (see [9] for details). Then, a cluster of the connected S-edges with associated vertices is transformed into a C-vertex. Hence, all the T-vertices and C-vertices in the skeleton I_s are stored as a set of vertex $V(G)$ and the set of R-edges is stored as a set of edge $E(G)$.

3 Identification of Double-Traced Lines

Now, suppose that a undirected graph G is modeled from an input image I by the method in Section 2. If the graph G is a semi-Eulerian, there exists a walk on G which traverses all of the edges exactly once from a T-vertex (i.e. start vertex) and another one (i.e. end vertex). Such a walk is referred as 'Euler path'. It is well known that the problem of finding an Euler path on a semi-Eulerian graph can be computed in polynomial time (see e.g. [10]). Moreover, we may often face the situations in which a line is drawn twice. Such a stroke which is traversed twice is called as 'double-traced line (D-line)'. In such cases, the graph G obtained from input I never become semi-Eulerian. We thus need to detect all the D-lines in order to construct a semi-Eulerian graph from G.

Figure 2 shows a set of stroke including D-line, where the numbers and arrow marks denote the drawing order. Moreover, the corresponding graphs are shown in Fig. 3, where Fig. 3 (a) is the graph corresponding to Fig. 2 (a) and Fig. 3 (b) is one corresponding to others

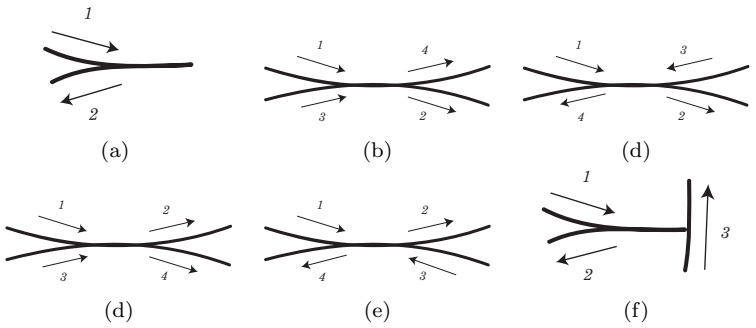


Fig. 2. Six types of possible D-lines.

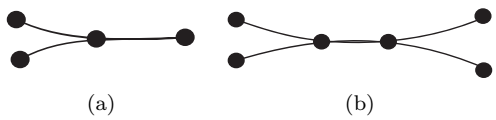


Fig. 3. Graph structures of D-lines in Fig. 4. The left and right figures illustrate the graph structures corresponding to Fig. 4 (a) and Fig. 4 (b)-(f), respectively.

in Fig. 2. We then see that only the structure of Fig. 2 (a) is different from the others in Fig. 2. However, the graph corresponding to Fig. 2 (b)-(f) is quite same. Thus, it may be difficult to differentiate the drawing order among Fig. 2 (b)-(f) from the local graph structure around D-line. For solving such a difficulty, we first identify all the D-lines and then detect the corresponding drawing order. In order to identify D-lines, we introduce an idea of 'D-line index' (see Section 3.1). The index is used to locally identify all the D-lines. Then, we have only to consider the problem to detect the corresponding drawing order of D-lines. One of the simplest approach for solving such a problem is the brute-fource method [11] which enumerate all the possible drawing orders on D-lines. Such a method however faces the combinatorial explosion problem. Hence, we may need more sophisticated method so that the writing order is detected by analyzing globally the structure of graph G . We then develop the method for transforming a graph with D-lines to a semi-Eulerian one by employing the so-called path duplication method (see Section 3.2).

3.1 D-line Index

We here present 'D-line index' for identifying D-lines. As shown in Fig. 3, all the types of D-line consist of an edge between two vertices with odd degree, where we suppose that the length of edge is relatively small. Then, the edge between two vertices with odd degree may be D-line candidate. Moreover, stroke may intersect with D-line as shown in Fig. 4. Thus, representing D-line as an edge between two vertices with odd degree may be hardly adequate. One of natural way is to represent a candidate of D-line as a path P_D of G with length more than or equal to 1. When the D-line is given as a path between two vertices $v_{D_1}, v_{D_2} \in V(G)$, we thus express the candidate path P_D as

$$P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_1}, v_{k_2}), \cdots, (v_{k_l}, v_{D_2}) \rangle \tag{1}$$

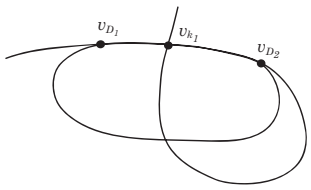


Fig. 4. D-line with $P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_1}, v_{D_2}) \rangle$.

with $v_{k_i} \in V(G), i = 1, 2, \cdots, l$. Here, (v_p, v_q) denotes an edge between two adjacent vertices v_p and v_q and two terminals v_{D_1} and v_{D_2} of P_D has odd degree.

Example 1 *In Fig. 4, stroke intersects with an edge (v_{D_1}, v_{D_2}) corresponding to D-line at vertex v_{k_1}. Hence, the candidate path P_D in (1) is expressed as $P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_l}, v_{D_2}) \rangle$.*

In order to determine whether a candidate P_D in (1) is a D-line, we introduce D-line index as follows. D-line index is based on the straightness of P_D and the smoothness between P_D and the incident edges to terminals of P_D. Letting $DLI(P_D) \in [0, 1]$ be 'D-line index' for a candidate P_D, we define $DLI(P_D)$ as

$$DLI(P_D) = \lambda ST(P_D) + (1 - \lambda) SM(P_D), \tag{2}$$

where $\lambda \in [0, 1]$ is a weighted parameter, and $ST(P_D)$ and $SM(P_D)$ denote the ratios on straightness and smoothness for P_D defined as follows.

First, the straightness ratio $ST(P_D)$ is evaluated by

$$ST(P_D) = \frac{d(v_{D_1}, v_{D_2})}{l(P_D)}, \tag{3}$$

where $d(v_{D_1}, v_{D_2})$ is Euclidean distance between two vertices v_{D_1} and v_{D_2}, and $l(P_D)$ is the length of P_D defined as

$$l(P_D) = \sum_{1 \le i \le l-1} d(v_{k_i}, v_{k_{i+1}}) + d(v_{D_1}, v_{k_1}) + d(v_{k_l}, v_{D_2}). \tag{4}$$

It then holds that $ST(P_D) \le 1$ with upper bound being achieved when P_D becomes exactly straight line. As the straightness of path P_D decreases, $ST(P_D)$ approaches to 0.

Next, the smoothness ratio $SM(P_D)$ in (2) is defined by

$$SM(P_D) = \min \left\{ \sum_{i=1,2} S\left(e_{D_1}, e_i^{v_{D_1}}\right), \sum_{i=1,2} S\left(e_{D_2}, e_i^{v_{D_2}}\right) \right\} \tag{5}$$

with

$$S(e_{D_j}, e_i^{v_{D_j}}) = \begin{cases} \frac{|\alpha|}{\pi} & \text{if } \alpha \neq 0 \\ 1 & \text{if } \alpha = 0 \end{cases}, \ i, j = 1, 2. \tag{6}$$

Here, $e_{D_1} = (v_{D_1}, v_{k_1})$ and $e_{D_2} = (v_{D_2}, v_{k_l})$, and $e_i^{v_{D_j}}$ denote an edge $(v_{D_j}, v_i^{D_j})$ for $i, j = 1, 2$, where $v_i^{D_j}$ is an adjacent vertex to v_{D_j}. $|\alpha| \in [0, \pi]$ is a difference angle between the edges

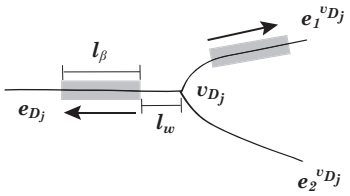


Fig. 5. Computation of angle.

e_{D_j} and $e_i^{v_{D_j}}$ around v_{D_j}. It then holds that $S(e_{D_j}, e_i^{v_{D_j}})$ approaches to 1 as $|\alpha|$ approaches to π. Also, when $\alpha = 0$, we set $S(e_{D_j}, e_i^{v_{D_j}}) = 1$. The case of $\alpha = 0$ may not usually occur, but this expression will be used for constructing a continuity graph for the case where D-line in Fig. 2 (a) includes (see Section 4.1 for detail.)

On the other hand, it is difficult to estimate α precisely since the segments around the intersection between e_{D_j} and $e_i^{v_{D_j}}$ may be distorted due to the sensitivities of thinning algorithm to noises. Thus, we estimate α from the stroke part near the vertex v_{D_j} as follows: Suppose that two edges e_{D_j} and $e_i^{v_{D_j}}$ are connected as shown in Fig. 5. From the edges e_{D_j} and $e_i^{v_{D_j}}$, we first pick out two segments $\vec{e}_{D_j}$ and $\vec{e}_i^{v_{D_j}}$ with length l_β from a point which is length l_w away from v_{D_j}. Then, by employing such segments $\vec{e}_{D_j}$ and $\vec{e}_i^{v_{D_j}}$, we compute α as

$$\alpha = \cos^{-1} \frac{\vec{e}_{D_j} \bullet \vec{e}_i^{v_{D_j}}}{l_\beta^2}, \tag{7}$$

where $\bullet$ denotes inner product. We empirically set l_β and l_w as $l_\beta = 2l_w = 2w(I)$, where $w(I) \in \mathbf{R}$ denote the average stroke width of input I defined by

$$w(I) = \frac{2a(I)}{c(I)} \tag{8}$$

where $a(I)$ denotes the area of stroke (i.e. total number of pixels in stroke area) in input I and $c(I)$ is the contour length of stroke area in I.

Remark 1 *If a candidate P_D is D-line corresponding to Fig. 2 (a), the degree for either of v_{D_j}, $j = 1, 2$ may be 1, i.e. $\deg(v_{D_j}) = 1$. Then, there exists no incident vertices for such a vertex v_{D_j} with $\deg(v_{D_j}) = 1$. Thus, if $\deg(v_{D_k}) = 1$ on P_D is satisfied for $k \in \{1, 2\}$, we have only to evaluate $SM(P_D)$ in (5) as*

$$SM(P_D) = \sum_{i=1,2} S\left(e_{D_k}, e_i^{v_{D_k}}\right). \tag{9}$$

3.2 Constructing Semi-Eulerian Graph

We are now in the position to develop a method to transform the undirected graph G with D-lines to a semi-Eulerian graph. It is known that undirected graph G is a semi-Eulerian if and only if G has exactly two vertices of odd degree. Thus, our task is to identify D-lines

by employing D-line index in Section 3.1 and transform the odd degree vertices on D-lines to even degree ones.

When there exist some D-lines on the stroke, the odd degree vertices may consist of those of D-line or terminal points of stroke (i.e. the start and end points) in G. Now, let $V_1 \subseteq V(G)$ be a set of vertex with degree one defined as

$$V_1 = \{v_i \mid \deg(v_i) = 1,\ i = 1, 2, \cdots, n\}. \tag{10}$$

Each vertex $v_i \in V_1$ must be either a terminal point of stroke or a degree one vertex of a D-line as shown in Fig. 2 (a). In other words, there exist exactly two vertices, denoted by $v_s, v_t \in V_1$, which correspond to start and end points of stroke. We then identify v_s and v_t by using DLI in (2) as follows.

Letting $V_{\text{odd}} \subseteq V(G)$ be a set of vertex with odd degree defined as

$$V_{\text{odd}} = \{v_i \in V(G) \mid \deg(v_i) \text{ is odd}\}, \tag{11}$$

we find the odd degree vertex $v_j \in V_{odd} (j \neq i)$ for each $v_i \in V_1$. Then we evaluate an index $TI(v_i),\ v_i \in V_1$ defined by

$$TI(v_i) = \max_{v_j \in V_{\text{odd}}} \{DLI(P_S(v_i, v_j))\}, \tag{12}$$

where $P_S(v_i, v_j)$ is the shortest path between v_i and v_j. Note here that the length of D-line is generally short. Thus, the vertex v_i with large $TI(v_i)$ may be a terminal of D-line. Hence, two vertices with smallest $TI(v_i)$ can be identified as the terminal vertices v_s and v_t, say the start and end points of drawing.

Next, let V_D be a set of odd degree vertices of G defined as

$$V_D = V_{\text{odd}} \backslash \{v_s, v_t\}. \tag{13}$$

We then see that V_D is a set of odd vertex corresponding to both end vertices of D-lines, hence $|V_D|$ is even. According to the fact that any D-line does not connect two vertices with degree one, all the D-lines are identified as follows. Let G_D be an edge weighted graph with a set of vertex V_D in (13) and a set of edges $E(G_D)$ defined as

$$E(G_D) = (V_D \times V_D) \backslash \{(v_i, v_j) | \deg(v_i) = \deg(v_j) = 1\}, \tag{14}$$

where $A \times A$ is the Cartesian product of A with itself. Moreover, we introduce the weight function $w : E(G_D) \to \mathbf{R}^+$ defined as

$$w(v_i, v_j) = DLI(P_S(v_i, v_j)) \tag{15}$$

for each $(v_i, v_j) \in E(G_D)$. Then, the detection of D-lines can be regarded as the maximum weight matching problem on G_D. As is well known, the maximum weight matching problem can be solved in polynomial time (see, e.g. [12, 13]). Letting M be a maximum weighted matching of G_D, we identify the shortest paths corresponding to edges of M as D-lines.

After identifying the terminal vertices of stroke and D-lines, we readily transform the graph G into a semi-Eulerian graph G_{eul} by employing the path duplication method as follows.

Supposing that a D-line is given as $P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_1}, v_{k_2}), \cdots, (v_{k_l}, v_{D_2}) \rangle$, all the edges in P_D are removed and v_{D_1} is connected with v_{D_2} by two new edges. Then, the degree of vertex $v_{k_i}, i = 1, 2, \ldots, l$ may reduced to two. Since there is no more branch of stroke at such vertex v_{k_i}, we can remove v_{k_i} by merging two edges adjacent to v_{k_i} to an edge (see Fig. 6). Hence, we get the following lemma.

Lemma 1 *G_{eul} is a semi-Eulerian graph. Furthermore, the Euler paths of G_{eul} corresponds one-to-one with possible strokes of input image.*

Proof of Lemma 1: Let G be an arbitrary graph obtained by the method in Section 2. Also, let G_{eul} be a graph obtained from G by the path duplication method. We first prove that G_{eul} is a semi-Eulerian. As is well known, a graph is a semi-Eulerian if and only if the graph is connected and has exactly two vertices of odd degree. Noting that G_{eul} is obviously connected, we here show that exactly two vertices of G_{eul} have degree one and others have even degree.

Now, letting v_i be an arbitrary vertex of G with even degree, we consider the following three cases:

(C1) v_i is also in G_{eul} and the degree of v_i is same as that of v_i in G.

(C2) v_i is also in G_{eul} and the degree of v_i is smaller than that of v_i in G by two.

(C3) v_i is not in G_{eul}.

In any cases of (C1)-(C3), v_i has even degree in G_{eul}. Let v_i be an arbitrary vertex of G with odd degree except ones corresponding to start and end points of stroke, i.e. v_s and v_t. That is, v_i is a vertex corresponding to terminal of D-line in G, which is shown in Fig. 3. By the path duplication, the degree of such vertices v_i may be increased by one. Thus, v_i has even degree in G_{eul}. Since the path duplication does not change the degree of v_s and v_t, then the degree of v_s and v_t is also one in G_{eul}. Thus, exactly two vertices (i.e. v_s and v_t) of G_{eul} have degree one and others even degree.

Next we prove that Euler paths of G_{eul} corresponds one-to-one with possible paths on G. Letting S be an arbitrary possible path of G, then S may traverse the edge corresponding to D-line twice, but other edges exactly once. However, as described above, the edge corresponding to D-line is replaced as two edges by the path duplication, and then S becomes an Euler path of G_{eul}. Thus, S corresponds to an Euler path of G_{eul}.

Conversely, letting P_E be arbitrary semi-Eulerian path of G_{eul}, P_E traverse each edge of G_{eul} exactly once. Since each D-line of G is replaced by two edges in G_{eul}, then P_E is regarded as a path of G that traverse each D-line twice as well as other edge exactly once. Hence, P_E is a stroke of G. This completes the proof of Lemma 1. □

The above method for constructing semi-Eulerian graph G_{eul} from G with D-lines is summarized in Algorithm 1 and Algorithm 2, where Algorithm 2 is the procedure of path duplication method.

Example 2 *When a handwritten image in Fig. 7 (a) is given as input image I, we get semi-Eulerian graph G_{eul} as shown in Fig. 7 (b) by Algorithm 1.*

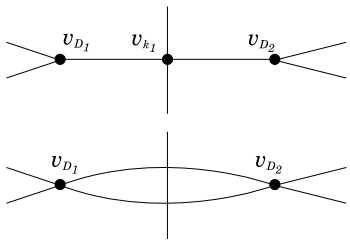


Fig. 6. The path duplication method. The upper figure is an example of a path $P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_1}, v_{D_2}) \rangle$ that represents a D-line. The lower figure is the result of the path duplication method. The vertex v_{k_1} is removed and P_D is replaced by multiple-edges between v_{D_1} and v_{D_2}.

Algorithm 1 SemiEulerian(G)

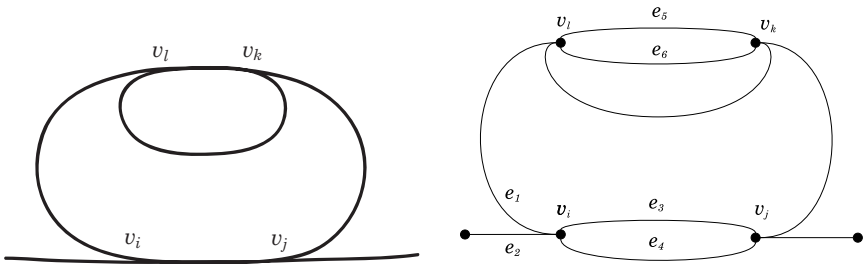
Require: an undirected graph G.
Ensure: convert G to its semi-Eulerian G_{eul}.
1: **for all** vertex $v_i \in V_1$ **do**
2: Compute $TI(v_i)$ by (2).
3: **end for**
4: Select two vertices $v_s, v_t \in V_1$ with the smallest TI.
5: Construct edge weighted graph G_D.
6: Compute the maximum weighted matching M in G_D.
7: **for all** $e \in M$ **do**
8: **PathDuplication(G, P_S)** where P_S is the shortest path corresponding to e.
9: **end for**

Algorithm 2 PathDuplication(G, P_D)

Require: an undirected graph G and a path P_D.
Ensure: dupulicate the path P_D.
1: Let $P_D = \langle (v_{D_1}, v_{k_1}), (v_{k_1}, v_{k_2}), \ldots, (v_{k_l}, v_{D_2}) \rangle$.
2: Remove the edges $(v_{D_1}, v_{k_1}), (v_{k_1}, v_{k_2}), \ldots,$ and (v_{k_l}, v_{D_2}).
3: Add new multiple-edges those connect v_{D_1} and v_{D_2}.
4: **for** $j = 1 \to l$ **do**
5: **if** $\deg(v_{k_j}) = 2$ **then**
6: Let $e_1 = (v_{k_j}, v_p), e_2 = (v_{k_j}, v_q)$ be two adjacent edges of v_{k_j} which are not in P_D.
7: Remove e_1 and e_2, and add new edge (v_p, v_q).
8: **end if**
9: **end for**

4 Recovering The Smoothest Drawing Order

Our concern is to recovery the smoothest writing order, which is normally produced by humans (see e.g. [14]), from input image I. From Lemma 1, we see that the possible strokes on handwritten character image corresponds to Euler paths of G_{eul}. Thus, our task is to find the most suitable Euler path corresponding to the smoothest writing order from G_{eul}. For achieving such a task, we first construct continuity graph from G_{eul} (Section 4.1). Then, the smoothest Euler path is approximated by employing the method using probabilistic tabu



(a) given handwritten image I (b) corresponding semi-Eulerian graph G_{eul}

Fig. 7. An example of handwritten image input I and the corresponding semi-Eulerian graph G_{eul}.

search algorithm (Section 4.2).

4.1 Constructing Continuity Graph

We construct a continuity graph G_{con} from a semi-Eulerian path G_{eul}. Let v_i be even degree vertex of G_{eul} and let $\{e_1, \ldots, e_d\}$ be a set of adjacent edges of v_i (see e.g. Fig. 7 (b)). For each v_i, we create a complete graph C_{v_i} whose vertices are $\{v_{e_1}, \ldots, v_{e_d}\}$. Then, we replace v_i with C_{v_i} and connect v_{e_i} to e_i for $i = 1, \ldots, d$. Moreover, the weight $S(e_i, e_j)$ is assigned for each edge (v_{e_i}, v_{e_j}).

Note that the graph G_{con} is no longer Eulerian path. However, by using a perfect matching of C_{v_i}, we can readily reduce the graph G_{con} to a path corresponding to an Euler path of G_{eul} as follows. Let $\mathcal{M}$ be a set of perfect matchings defined as

$$\mathcal{M} = \{M_{v_i}\}_{v_i \in V(G_{\mathrm{eul}})}, \tag{16}$$

where $M_{v_i} \in \mathcal{M}$ is a perfect matching of C_{v_i}. Then, the graph $G_{\mathrm{con}}^{\mathcal{M}}$ defined by

$$G_{\mathrm{con}}^{\mathcal{M}} = \left(V(G_{\mathrm{con}}), \left(E(G_{\mathrm{con}}) - \bigcup_{v_i \in V(G_{\mathrm{eul}})} E(C_{v_i}) \right) \cup \mathcal{M} \right) \tag{17}$$

is a path corresponding to an Euler path of G_{eul} whenever $G_{\mathrm{con}}^{\mathcal{M}}$ is connected. Thus the following Lemma 2 holds. Moreover, by Lemmas 1 and 2, we get a main theorem in Theorem 1.

Lemma 2 *For a collection of perfect matchings $\mathcal{M}$ in (16), the graph $G_{\mathrm{con}}^{\mathcal{M}}$ in (17) is a path corresponding to an Euler path of G_{eul} whenever $G_{\mathrm{con}}^{\mathcal{M}}$ is connected. Conversely, for every Euler path P_E of G_{eul}, there exists a collection of perfect matchings $\mathcal{M}$ such that $G_{\mathrm{con}}^{\mathcal{M}}$ corresponds to P_E. Furthermore, total weight of $\mathcal{M}$ indicates smoothness of corresponding Euler path.*

Theorem 1 *Corrections of perfect matchings $\mathcal{M}$ on a continuity graph G_{con} corresponds one-to-one with possible strokes of a given image. Furthermore, the total weight of $\mathcal{M}$ indicates the smoothness of corresponding stroke.*

Algorithm 3 ContinuityGraph(G_{eul})

Require: an undirected semi-Eulerian graph G_{eul}.
Ensure: construct a continuity graph G_{con} from G_{eul}.
1: Create a copy G_{con} of G_{eul}.
2: **for all** $v_i \in V(G_{\mathrm{con}})$ **do**
3: Let $\{e_1, \ldots, e_d\}$ be the set of incident edges of v_i.
4: Replace v_i by a complete graph C_{v_i} with vertices set $\{v_{e_1}, \ldots, v_{e_d}\}$ and connect v_{e_h} to e_h for each $h = 1, \ldots, d$.
5: Assign the weight $S(e_i, e_j)$ to each edge (v_{e_i}, v_{e_j}).
6: **end for**
7: **return** G_{con}.

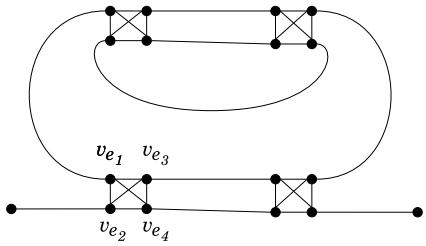


Fig. 8. Continuity graph G_{con} constructed from semi-Eulerian graph G_{eul} in Fig. 7 (b).

The above procedure for constructing continuity graph G_{con} from G_{eul} is summarized in Algorithm 3.

Example 3 *Using Algorithm 3, the semi-Eulerian graph G_{eul} in Fig. 7 (b) is converted to the continuity graph G_{con} as shown in Fig. 8.*

4.2 Probabilistic Tabu Search

By Theorem 1, we see that the problem of computing the smoothest drawing order of handwritten image reduces to the problem of computing the maximum weight collection of perfect matchings $\mathcal{M} = \{M_{v_i}\}_{v_i \in V(G_{\mathrm{eul}})}$, where M_{v_i} is a perfect matching of C_{v_i} such that $G_{\mathrm{con}}^{\mathcal{M}}$ is connected. For computing such a maximum weight collection, we here employ a probabilistic tabu search algorithm which is a meta-heuristic local search algorithm for solving combinatorial optimization problems..

For this purpose, we first compute an Euler path P_E of G_{eul}. Then, a set of perfect matchings of complete graph $\{C_{v_i}\}_{v_i \in V(G_{eul})}$ corresponding to P_E is given as $\mathcal{M}$ in (16). Thus, $\mathcal{M}$ is set as initial feasible solution of probabilistic tabu search. In the probabilistic tabu search, an optimal solution is obtained by iteratively modifying the initial feasible solution to better one. Then, a tabu list T of feasible solutions is used to avoid modifying to feasible solution that have been visited in the recent past. An initial setting of T is set as $T = \{\mathcal{M}\}$. Then, the algorithm may find better feasible solution $\mathcal{M}'$ of which the weight is larger than that of previously visited feasible solutions by repeating the following local change: First,

Algorithm 4 TabuSearch(G_{eul}, G_{con})

Require: an undirected semi-Eulerian graph G_{eul} and a continuity graph G_{con}.
Ensure: compute an approximation of the maximum collection of perfect matchings.
1: Compute an arbitrary Euler path P_E of G_{eul}.
2: Compute the collection of perfect matchings $\mathcal{M}$ corresponding to P_E.
3: Set a tabu list $T = \{\mathcal{M}\}$.
4: **for** $i = 1 \to N$ **do**
5: Choose C_{v_i} randomly from $\{C_{v_i}\}_{v_i \in V(G_{\text{eul}})}$ with probability inversely proportional to the weight of $M_{v_i} \in M$.
6: Choose two edges $e_1 = (v_1, v_2)$ and $e_2 = (v_3, v_4)$ from M_{v_i} uniformly at random.
7: Set $E_1 = \{(v_1, v_2), (v_3, v_4)\}, E_2 = \{(v_1, v_3), (v_2, v_4)\}$ and $E_3 = \{(v_1, v_4), (v_2, v_3)\}$.
8: **if** there exists $i \in \{2, 3\}$ such that $w(E_i) \geq w(E_1)$ and $\mathcal{M}' = \mathcal{M} \backslash \{M_{v_i}\} \cup \{M_{v_i} \backslash E_1 \cup E_i\}$ is a feasible solution such that $\mathcal{M}'$ is unlisted in T and $G_{\text{con}}^{\mathcal{M}'}$ is connected **then**
9: Set $T = T \cup \mathcal{M}'$.
10: Set $\mathcal{M} = \mathcal{M}'$.
11: **end if**
12: **end for**
13: **return** the best feasible solution $G_{\text{con}}^{\mathcal{M}}$ that was found so far.

Algorithm 5 StrokeRecover(G)

Require: an undirected graph G.
Ensure: recover a stroke order of G.
1: $G_{\text{eul}} \leftarrow$ **SemiEulerian**(G).
2: $G_{\text{con}} \leftarrow$ **ContinuityGraph**(G_{eul}).
3: **return TabuSearch**(G_{eul}, G_{con}).

we choose C_{v_i} randomly from $\{C_{v_i}\}_{v_i \in V(G_{\text{eul}})}$ with probability inversely proportional to the weight of $M_{v_i} \in \mathcal{M}$. Then, two edges $e_1 = (v_1, v_2)$ and $e_2 = (v_3, v_4)$ of M_{v_i} are chosen uniformly at random. In addition, we set $E_1 = \{(v_1, v_2), (v_3, v_4)\}, E_2 = \{(v_1, v_3), (v_2, v_4)\}$, and $E_3 = \{(v_1, v_4), (v_2, v_3)\}$. If there exists $i \in \{2, 3\}$ such that $w(E_i) \geq w(E_1)$ and $\mathcal{M}' = \mathcal{M} \backslash E_1 \cup E_i$ is a feasible solution unlisted in T, then we update T and M as $T = T \cup \{\mathcal{M}'\}$ and $\mathcal{M} = \mathcal{M}'$ respectively. This process is iteratively carried out for new $\mathcal{M}$ and T until a predefined time limit N is exceeded. The $\mathcal{M}$ is finally given as the best feasible solution, i.e. the desired drawing order of input handwritten image I.

The above method is summarized as Algorithm 4.

Example 4 *Using Algorithm 4, we get the graph $G_{\text{con}}^{\mathcal{M}}$ in Fig. 9 from the continuity graph G_{con} in Fig. 8.*

Hence, by using Algorithm 1 – Algorithm 4, the algorithm for recovering a drawing order of an input handwritten image I is given by Algorithm 5

5 Experimental Studies

We examine the effectiveness and usefulness of our proposed method by some experiments. Here, the algorithm is implemented in Java and the time limit N in tabu search algorithm is set as $N = n^2$, where n is the number of vertices on the graph G constructed from handwritten

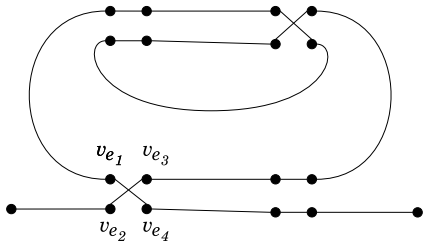


Fig. 9. The resulting graph $G^{\mathcal{M}}_{\rm con}$ obtained from continuity graph $G_{\rm con}$ in Fig. 8.

image I. Also, we set λ in (2) as $\lambda = 3/5$.

The recovered results for 8 handwritten images are illustrated in Fig. 10. Here, the original input images in black lines are stored by employing a pen-tablet device with 4 pixels width. Also, green lines are paths on the corresponding graph G. 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. Note that the handwritten images in Fig. 10 (a)-(c) have no D-lines, but have complex structures. Also, the handwritten images in Fig. 10 (d)-(h) include some D-lines. From these results, we may observe that our proposed method works quite well and can correctly recover the drawing order of handwritten character image even when D-lines are included.

6 Concluding Remarks

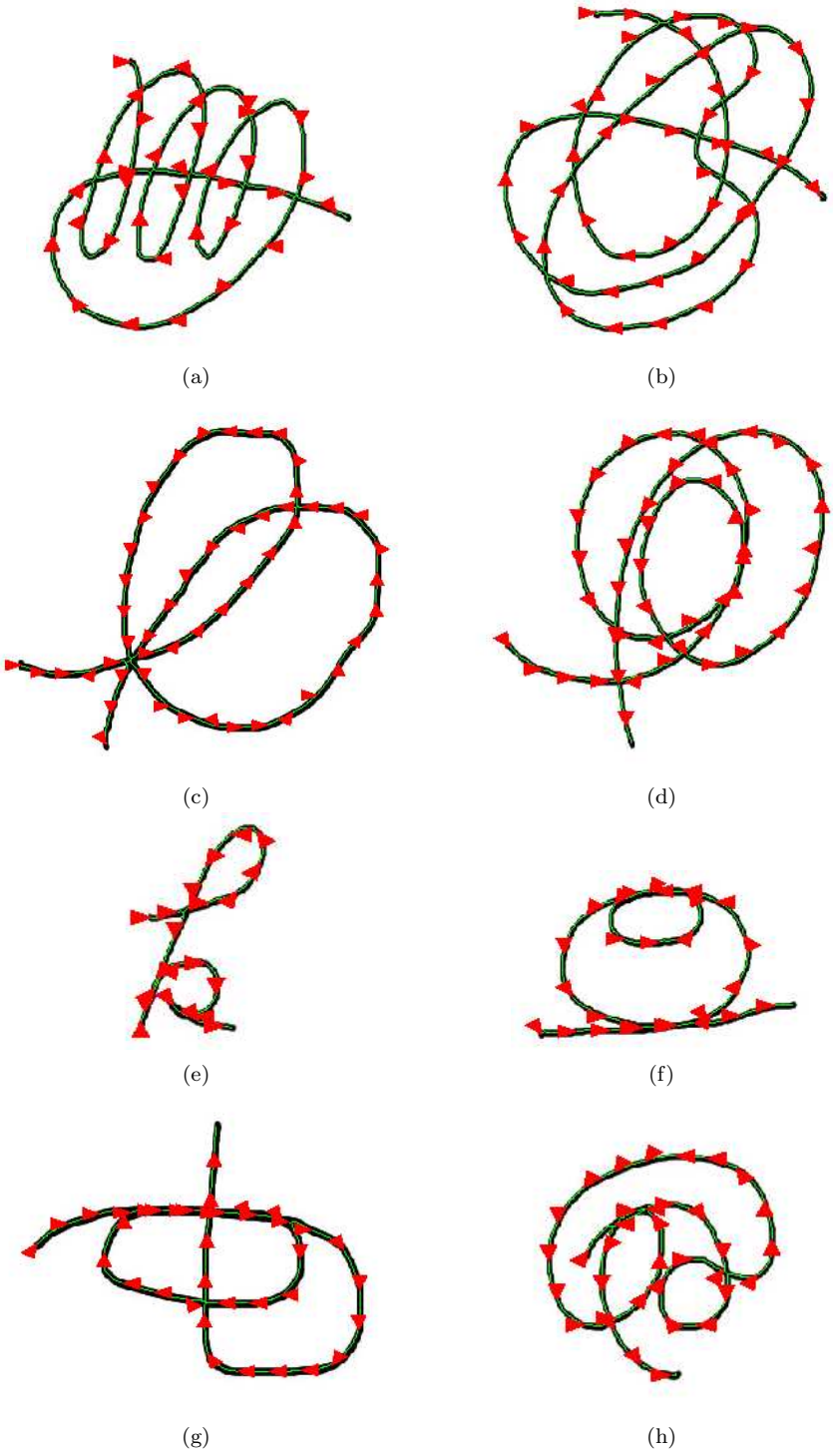
In this paper, we developed a new method for recovering a drawing order from static handwriting images with single stroke. The problem was analyzed and solved by employing the so-called graph theoretic approach. Then the central issue was to obtain the smoothest path of stroke from a graph model of input handwriting image. First, the graph model was constructed from the input handwritten image by employing thinning algorithm. We then analyze the structure of graph at each vertex. In particular, the method to identify double-traced lines (D-lines) was developed by introducing the D-line index. The method enables us to transform any graph models including D-lines to semi-Eulerian graph models. Then, the restoration problem reduced to maximum weight perfect matching problem of graph, thus a probabilistic tabu search algorithm was developed to solve the problem. The effectiveness and usefulness were experimentally demonstrated.

Acknowledgements

T. Nagoya was supported in part by the Japan Ministry of Education, Culture, Sports, Science and Technology under Grants 22700018. H. Fujioka was supported in part by a grant from Computer Science Laboratory, Fukuoka Institute of Technology.

References

1. S. Lee and J. C. Pan (1992), *Offline Tracing and Representation of Signatures*, IEEE Trans. Systems, Man, and Cybernetics, Vol.22, No.4, pp.755–771.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Fig. 10. Recovery results for 8 handwritten images.

2. T. Huang and M. Yasuhara (1995), *Recovery of Information on the Drawing Order of Single-Stroke Cursive Handwritten Characters from Their 2D Images*, IPSJ Trans., Vol.36, No.9, pp.2132–2143.
3. S. Jäger (1996), *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.
4. Y. Kato and M. Yasuhara (1999), *Recovery of Drawing Order from Scanned Images of Multi-Stroke Handwriting*, Proc. of Fifth Int. Conf. on Document Analysis and Recognition, pp.261–264, Bangalore, India.
5. Y. Kato and M. Yasuhara (2000), *Recovery of Drawing Order from Single-Stroke Handwriting Images*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.22, No.9, pp.938–949.
6. H. Fujioka and T. Nagoya (2011), *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 Conference on Information Technology and Ocean Engineering, pp.34–37, Macao.
7. R. C. Gonzalez, R. E. Woods and S. L. Eddins (2009), *Digital Image Processing Using MATLAB*, Prentice Hall.
8. J. R. Parker (2011), *Algorithms for Image Processing and Computer Vision*, Second Edition, Wiley Publishing Inc.
9. Y. Qiao and M. Yasuhara (2004), *Recovering Dynamic Information from Static Handwritten Images*, Proc. of the Ninth Int. Workshop on Frontiers in Handwriting Recognition, Washington, DC, USA.
10. A. Gibbons (1985), *Algorithmic Graph Theory*, Canbridge University Press.
11. D. Avis, A. Hertz and O. Marcotte (2005), *Graphy Theory And Combinatorial Optimization* , 3rd edtition, Springer.
12. J. Edmonds B (1965), *Paths, trees, and flowers*, Canadian J. Math., Vol. 17, pp.449–467.
13. S. Micali and V. V. Vazirani (1980), *An* $O(\sqrt{|V|}|E|)$ *algoithm for finding maximum matching in general graphs*, Proc. 21st Annual Symposium on Foundations of Computer Science, Syracuse, NY, USA.
14. Y. Wada and M. Kawato (2004), *A via-point time optimization algorithm for complex sequential trajectory formation*, Neural Networks, Vol.17, No.3, pp.353–364.