RECOVERING DRAWING ORDER FROM OFFLINE HANDWRITTEN IMAGE USING DIRECTION CONTEXT AND OPTIMAL EULER PATH

Yu Qiao, Makoto Yasuhara

University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, Japan
{qiaoyu, yas}@math-sys.is.uec.ac.jp

ABSTRACT

This paper addresses the problem to recover drawing order from single-stroked offline handwritten image. The recovery problem is formulated as to find the smoothest path to cover all the edges in the graph representation of an input image. The two main contributions are: (1) we introduce direction context to calculate the smoothness between edges; (2) the smoothest path is found by solving a new graph problem: the optimal Euler path problem. An $O(n+m+n\log(m/2))$ time algorithm is developed to find the optimal Euler path in a graph with 4-degree nodes only. The double traced lines are identified using the maximum weighted matching of general graph. Experimental results on about 13,000 static images converted from the online data in the Unipen database demonstrated the utility of our methods.

1. INTRODUCTION

It is well known that handwriting recognition can be divided into two types: offline and online, which differ in the input device and information available. The online recognition makes use of online information of pen-tip movement, and achieves higher recognition performances [1], [2], [3], [4] than the offline recognition does. Importance of the online information in the recognition task is also supported by psychological researches [5], [6].

In this study, we focus on the problem to recover the drawing order (pen-tip trajectory) from a single-stroked offline static handwriting image. This can be seen as to convert two-dimensional image I to one dimensional vector sequence $\{(x_t, y_t), t=1,2,...,n\}$, where (x_t, y_t) denotes the position of pen-tip at time t in vector form. The recovery of drawing order is seen as the inverse problem of human handwriting process. Like many inverse problems, our recovery problem is ill-posed in the sense that there are multiple drawing paths possible to result in the given image. Therefore it is necessary to introduce a guiding principle to convert the problem into the well-posed one. Here, we adopt the *minimum energy principle* that the human write usually in the smoothest way with the least energy cost. Then there are two fundamental questions: (1) how to define the smoothness, (2) how to find the path that maximizes the smoothness. In this paper, this first question is answered by introducing *direction context*; and the second question is answered by defining and solving the *optimal Euler path* problem.

The approaches toward recovering drawing order information can be roughly divided into two main categories: local tracing method and global searching method. In the local tracing method, the next path is selected at each point of junction according to the present local configuration and tracing history [4], [5], [7]. Advantages of this method come from their simplicity and low computational cost, however, the method is sensitive to noise and it is extremely difficult to design general heuristic rule(s) that can be applied to various styles of handwriting. In the global search methods, a graph model is built to represent the input image and then the problem is to find a path (single stroke) and paths (multiple strokes) within the graph [2], [8]. However, these methods may lead to computational explosion. To avoid this, some hybrid methods were proposed, which combined the local analysis and global analysis [3], [9], [10]. But they cannot ensure the global optimization.

In this paper, we proposed a novel approach to the recovery problem, which can find globally optimal solution in polynomial time. The rest of the paper is organized as follows: the method to construct the graph model is described in Section 2. Section 3 introduces the direction context. The Optimal Euler Path problem is formulated in Section 4. In Section 5, we present a method to find double traced lines using the maximum weighted matching. The experimental results are given and discussed in Section 5. Finally, we conclude the paper in Section 6.

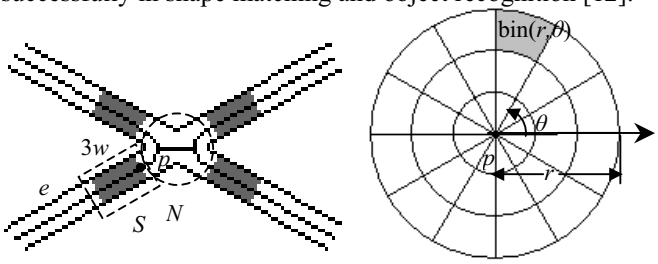
2. GRAPH MODEL

In this section, we describe the method to construct graph model G from the skeleton of an input handwriting image. We used a smoothing filter to reduce the noise such as peaks and holes in the input image and calculate the skeleton using thinning algorithm [11]. To construct G, we need to extract vertex and segment from the skeleton at first. A *segment* represents a part of the stroke in the skeleton and a *vertex* corresponds to a local geometrical point, where the

segment starts/ends or multiple segments joint. Because the skeleton resulted from thinning process may include unwanted spurious outputs, the segments detected above can be classified into two types: real segment (*r-segment*) and spurious segment (*s-segment*). The r-segment corresponds to a part of the real stroke; and the s-segment results from the thinning and does not exist in the original handwritten image. These undesirable s-segments will distort the structure of the original pattern, thus we need to identify them at first. We use the method described in our former work [9] to identify the s-segments. Then a cluster of the connected s-segments is transformed into a *node*. Terminal vertex and individual crossing vertex in the skeleton are transformed also into the nodes in G and the real segment is seen as an *edge* of G.

3. DIRECTION CONTEXT

Because the thinning algorithm is sensitive to noise, it is not reliable to calculate the smoothness directly from only the skeleton [4], [9]. In this paper, *direction context* descriptor is introduced to estimate the smoothness from the original image. The direction context uses a set of vectors to describe the distribution of pixels along the stroke in the original image. Thus it can represent the general direction of this stroke part. The direction context can be seen as a special type of shape context, which had been used successfully in shape matching and object recognition [12].



(a) Stroke area (b) Diagram of bins

Fig. 1 Direction Context

Suppose edge e is incident to node N as shown in Fig. 1a. p is a pixel on e neighbor to N. w is the stroke width of e near N. S denotes a set of black pixels in the original image I. For each black pixel p_k in I, its nearest pixel $N_m(p_k)$ in the skeleton is defined as,

$$N_m(p_k) = \arg\min_{p_s}\{Dis(p_k, p_s) \mid p_s \in Skeleton\},$$

where $Dis(p_k, p_s)$ is the Euclidean distance between two pixels p_k and p_s. We find line L in the skeleton with length $3w$ by tracing from p along e and obtain the associated stroke part S of L by,

$$S=\{p_k \mid N_m(p_k) \in L\}.$$

Calculate the set of vectors originating from point p to all the pixels in S and then summarize these vectors by a coarse histogram using the same technology in [12]:

$$h_p(r,\theta) = \#\{q \in S : (q-p) \in bin(r,\theta)\}, \quad (1)$$

where the bins are shown in Fig 1.b. The histogram is called the direction context of e at N. For two edges e_i and e_j, we calculate direction context $h_i(r, \theta)$ at first and estimate the smoothness as the angle a, which maximizes (2):

$$s(e_i, e_j) = \arg\max_{0\le\alpha\le\pi} \frac{1}{2}\sum_{r,\theta} \frac{[h_i(r,\theta) - h_j(r,\alpha+\theta)]^2}{h_i(r,\theta) + h_j(r,\alpha+\theta)} \quad (2)$$

In experiments, we calculate the direction angle a_i and a_j of S_i and S_j at p_i and p_j by using Principal Component Analysis [9] on the coordinates of pixels in S_i and S_j respectively. Note the first eigenvector of PCA corresponds to the main direction of a set of vectors. Then we initialize $\alpha=|a_i-a_j|$ and tune α to get the angle maximizing $s(e_i, e_j)$.

4. OPTIMAL EULER PATH

The recovery of drawing order can be seen as the problem of finding the smoothest path passing through each of the edges at least once. The edge-traversing problem in graph was first studied by Euler in 1736, known as the famous Königsberg bridge problem. Euler showed that an undirected graph has *Euler path*, which visits each edge exactly once iff all but two nodes are of odd degree. The two odd nodes must be a start node and/or an end node of the Euler path. In the next, we will define the optimal Euler path problem and describe the algorithm to solve it.

4.1 Continuous Cost and Optimal Euler Path

In graph G, two incident edges e_i and e_j connected to a node are referred to a *continuous pair* (e_i, e_j). The *continuous cost* $c(e_i, e_j)$ is given for each of the continuous pairs in G. In this paper, we set the continuous cost $c(e_i, e_j)=\pi-s(e_i, e_j)$. The graph model is represented by $G=(V, E, C)$, where V and E is the set of nodes and edges, respectively and C denotes the set of continuous cost. The total number of continuous cost (size of C) is $\sum_k d(N_k)(d(N_k)-1)/2$, where $d(N_k)$ is the degree of N_k. The continuous cost of path $l = e_1,e_2,..,e_n$ is defined as the sum of the continuous cost of every two adjacent edges along l. Formally,

$$f(e_1,e_2,..,e_n) = \sum_i c(e_i, e_{i+1}). \quad (3)$$

The Euler path that minimizes the continuous cost is called *optimal Euler path*.

4.2 Finding Optimal Euler Path

The direct idea of finding the optimal Euler path is to enumerate all the Euler paths and calculate the continuous cost for each of the paths to select the one with the minimum continuous cost. However, the number of the Euler paths increases exponentially with the number of the nodes. Note that the number of the Euler paths in a graph is greater than 2^n, where n is the number of the nodes of degree≥4 in the graph. Before presenting our algorithm to find the optimal Euler path, we introduce some definitions as follows:

Edge Continuity Relation (ECR) at node N is defined as a set of continuous pairs, which covers all the edges

connected to N. Formally, for node N of degree $2k$, we have

$$ECR(N)=\{(e_{i1}, e_{i2})\},\ i=1,2,\ ...,k \quad (4)$$

where (e_{i1}, e_{i2}) denotes a continuous pair. The total number of ECRs possible at N is $(2k\text{-}1)(2k\text{-}3)...1$. We define the *node continuous cost* as:

$$c(ECR(N))=\sum_i c(e_{i1}, e_{i2}) \quad (5)$$

It is not hard to see that there is a mapping from an Euler path to $ECR(N_j)$ at each node N_j. That is, given an Euler path $l=e_1,e_2,..,e_n$, if every two adjacent edges (e_i, e_{i+1}) in l is regarded as a continuous pair, then we can obtain a unique $ECR(N_j)$ at each node N_j. Then we can reduce Eq. (1) to the ECR form:

$$f(l)=\sum_j c(ECR(N_j)) \ , \ \text{for all } N_j \in V \quad (6)$$

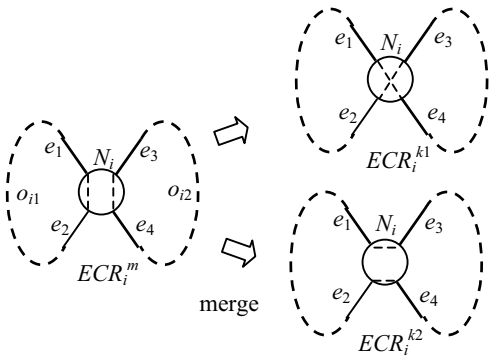


Fig. 2 Merge two circuits into one by changing ECR at node (the dash lines inside the circle represent ECR)

Assume that there is no node of degree higher than 4 in G, we have the following efficient algorithm to find the optimal Euler path.

(1) Finding all the ECRs

For node N_i connected by 4 edges e_1 - e_4. There are 3 cases of ECR_i^j (j=1,2,3): $ECR_i^1=\{(e_1, e_4), (e_2, e_3)\}$, $ECR_i^2=\{(e_1, e_3), (e_2, e_4)\}$ and $ECR_i^3=\{(e_1, e_2), (e_3, e_4)\}$. We calculate the node continuous cost for every ECR_i^j according to (5).

(2)Connecting the edges

For each node N_i, find ECR_i^m with the minimum node continuous cost. Then we connect the edges at N_i into path(s) according to ECR_i^m. If there is only one path obtained, it must be the optimal Euler Path. Otherwise, there are a path from the start to the end node and several circuits. We use o_i, i=1,2,3,…,K to denote the path and the circuits.

(3) Calculate merge cost

Take any two circuits/path o_{i1} and o_{i2} jointing at node N_i. If we change ECR_i^m at N_i to one of the other two ECRs, denoted by ECR_i^{m1} and ECR_i^{m2}, then o_{i1} and o_{i2} will be merged into one circuit/path (Fig. 2). Assume $c(ECR_i^{m1})\le c(ECR_i^{m2})$, we define the merge cost of o_{i1} and o_{i2} at node N_i as:

$$Merge(o_{i1}, o_{i2}, N_i)=f(ECR_i^{m1})-f(ECR_i^{m}), \quad (7)$$

$Merge(o_{i1}, o_{i2}, N_i)$ must be non-negative. Based on this, we define the *merge cost* of o_{i1} and o_{i2} as:

$$Merge(o_{i1}, o_{i2}) = \min_{N \in o_{i1} \cap o_{i2}} \{Merge(o_{i1}, o_{i2}, N)\}. \quad (8)$$

(4) Finding the optimal Euler path

Build circle graph $G_o=\{O, R\}$, O is a set of K vertices, each of which corresponds to a path/circle o_i found in step (2). If two circles o_i and o_j have jointing node(s), we add edge $e_{i\text{-}j}$ between them into R and weight $w(e_{i\text{-}j})$ is set as $Merge(o_i, o_j)$. It is not difficult to find that for merging the K path/circles into one path l, we need to find spanning tree H which contains all the vertex and K-1 edges in G_o. The continuous cost of path l can be written as:

$$f(l)=\sum_{i=1}^{K} f(o_i)+\sum_{e_j \in H} w(e_j). \quad (9)$$

To find optimal Euler path l with minimum continuous cost $f(l)$, we only need to minimize the second item $\sum w(e_j)$ in Eq. (9). This is the classical Minimum Spanning Tree (MST) problem. It can be solved by the Kruskal algorithm or Prim algorithm [13].

It is not hard to see that the algorithm above is self-proved optimal. The complexity of the proposed algorithm is estimated as: calculating the node continuous cost in $O(n)$, finding path/circles in $O(m)$, searching the minimum spanning tree in $O(n\log(m/2))$ if the Kruskal algorithm is used. So totally the complexity is $O(n+m+ n\log(m/2))$ where n, m are the number of nodes and edges in G_o, respectively.

5. DOUBLE TRACED LINES

Double tracing lines (*d-lines*) are common in human writing [3]. To recover the whole drawing order, we need to identify these d-lines. The d-line problem is related with odd nodes [9], [10]. A d-line normally locates between two odd nodes. There are two types of the odd node [10]: 1) *T-node* (the gray node in Fig. 3 at which an edge (*T-leg*) terminates as shown in Fig. 3a, 3b or returns back as shown in Fig. 3c, 3d), and 2) *Y-node* (the black node shown in Fig. 3 to which a double-traced edge (*Y-leg*) and its two continuous edges (Y-hands are connected). Based on this, d-line can be divided into two types: (1) *Y-T type*: the d-line between Y-node and T-node (Fig. 3c, 3d), where the d-line is a T-leg of T-node or a Y-leg of Y-node, (2) *Y-Y type*: the d-line between two Y-nodes (Fig. 3e).

The difficulty to find the d-lines comes from the fact that it is not easy to determine which edge connected to the odd node is T/Y-leg depending merely on the local structure of the node. The robust decision should be made in more global fashion. We achieve this by using maximum weighted matching algorithm [14], the details of which were described in our former work [10].

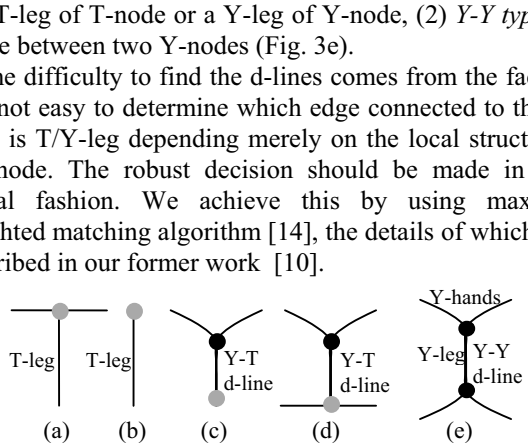


Fig. 3 Types of odd nodes and d-lines

After finding all the d-lines, for a Y-T type d-line, replace the two hands and the d-line totally by a new edge of G (Fig. 4a) and for a Y-Y type d-line, combine the d-line and the two Y-nodes into a new node of degree 4, connected by the four hands (Fig. 4b). The node generated from the Y-Y type d-line has two possible ECRs: $ECR_1=\{(e^{i1}, e^{j1}), (e^{i2}, e^{j2})\}$ and $ECR_2=\{(e^{i1}, e^{j2}), (e^{i2}, e^{j1})\}$. For each of the continuous pair, for example (e^{i1}, e^{j1}), calculate the SLALOM smoothness $s^{i1\text{-}j1}$ of path e^{i1}-l-e^{j1} [2], [9]. Let $s_m=\max\{s^{i1\text{-}j1}, s^{i2\text{-}j2}, s^{i1\text{-}j2}, s^{i2\text{-}j1}\}$, the node continuous cost is given by

$$c(ECR_1) = (s^{i1\text{-}j1}+ s^{i2\text{-}j2})/ s_m$$
$$c(ECR_2) = (s^{i1\text{-}j2}+ s^{i2\text{-}j1})/ s_m. \quad (10)$$

After merging, there will be no d-line in G, so we can recover the order of edges by finding the optimal Euler path.

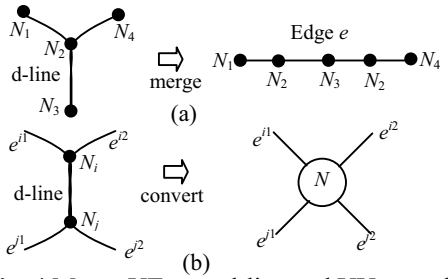


Fig. 4 Merge YT-type d-line and YY-type d-line

6. EXPERIMENTS

We have applied our methods on the two sets of single stroked handwritten images. The first set consists of 145 offline images collected by ourselves, which includes cursive English words, Chinese characters, Japanese Kana characters and several artificial patterns. The second set includes 13,306 static images converted from the online data in the Unipen database [15]. Each stroke, randomly selected from the Unipen database, is converted to a single stroke offline image by connecting the adjacent points through the straight lines. The width of the line is set as 3. For node with degree more than 4, the ECR is set as the one where all the lines traversing through node crosses each other. Totally 93.7% of all the samples are recovered successfully. Several samples are shown in Fig. 5.

We also executed experiments to compare the recognition performance of offline image and recovered path using the Unipen 1.a database [15]. The k-nearest neighbor classifier using Euclidean distance was selected as the recognition engine. Among the 15,612 samples, 30% were selected randomly for testing. The results are summarized in Table 1. We find that the recognition rates of recovered path are higher than those of offline image.

Table 1 Comparison of Recognition Results
(k is the num of neighbors)

k	1	3	5	10
Recovered Path	93.6%	93.5%	93.2%	93.6%
Offline Image	90.4%	90.0%	90.6%	90.0%

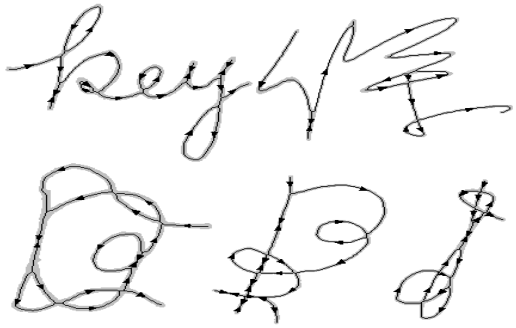


Fig. 5 Experimental Results

7. CONCLUSION

A new method has been proposed to recover the drawing order of the handwriting character from its 2D static image. We introduced the direction context to estimate smoothness and converted the recovery problem to the optimal Euler path problem. A polynomial algorithm was presented to solve optimal Euler path in graph with 4-degree nodes only. More generally, our method, due to its globally optimal nature, can efficiently convert a graph to a sequence of edges. It is well known that the string is much easier and more efficient to be dealt with than the graph. By choosing properly the method to calculate continuous cost, we can preserve the structure of the graph into the sequence.

We consider to study in future graph analysis and recognition problem by using the optimal Euler path.

8. REFERENCES

[1] R.Plamondon, S.N.Srihari, "On-Line and Off-line Handwriting Recognition: A Comprehensive Survey", *IEEE Trans. on PAMI*, Vol.22, No. 1, pp.63-84, 2000

[2] T. Huang and M. Yasuhara, "Recovery of Information on the Drawing Order of Single-Stroke Cursive Handwritten Characters from Their 2D Images," *IPSJ Trans.*, vol. 36, no. 9, pp. 2,132-2,143,Sept. 1995.

[3] Y. Kato and M. Yasuhara, "Recovery of Drawing Order from Single-Stroke Handwriting Images", *IEEE TPAMI*, vol.22, no.9, pp.938-949, 2000.

[4] D.S. Doermann and A. Rosenfeld, "Recovery of Temporal Information from Static Images of Handwriting", *Int'l J. Computer Vision*, vol. 15, pp. 143-164, 1995.

[5] R. Plamondon and C.M. Privitera, "The Segmentation of Cursive Handwriting: An Approach Based on Off-Line Re-covery of the Motor-Temporal Information", *IEEE Trans. IP*, vol. 8, no. 1, pp.80-91, 1991.

[6] Babcock, M.K., & Freyd, J.J. Perception of dynamic information in static handwritten forms. *American Journal of Psychology*, vol. 101, pp. 111-130, 1988

[7] S. Lee and J.C. Pan, "Offline Tracing and Representation of Signatures," *IEEE Trans. SMC*, vol. 22, no. 4, pp. 755-771, 1992

[8] S. Jager, "Recovering Writing Traces in Off-Line Handwriting Recognition: Using a Global Optimization technique," *Proc. 13th Int'l Conf. Pattern Recognition*, pp. 150-154, 1996.

[9] Y. Qiao and M. Yasuhara: "Recovering Dynamic Information from Static Handwritten Images", *Proc. IWFHR* Oct. 2004 Tokyo, Japan

[10] Y. Qiao, M. Nishiara and M. Yasuhara; " A Novel Approach to Recover Writing Order From Single Stroke Offline Handwritten Images", pp.227-231 *Proc. ICDAR* 2005

[11] T. Y. Zhang, Ching Y. Suen: A Fast Parallel Algorithm for Thinning Digital Patterns. *Commun. ACM* 27(3): pp. 236-239 (1984)

[12] S. Belongie, J. Malik and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE PAMI* 24(24), pp. 509-522, 2002

[13] Graham, R. L. and Hell, P. "On the History of the Minimum Spanning Tree Problem." *Ann. History Comput.* vol. 7, pp. 43-57, 1985

[14] Edmonds, J. and Johnson, E.L. Matching, Euler tours and the Chinese postman. *Math Programming* 5 (1973), 88-124.

[15] Guyon, I., Schomaker, L., Plamondon, R., Liberman, M. & Janet, S. "UNIPEN project of on-line data exchange and recognizer Benchmarks", *Proc. of ICPR*, pp. 29-33, October 1994