A Framework toward Restoration of Writing Order from Single-Stroked Handwriting Image

Yu Qiao, Student Member, IEEE, Mikihiro Nishiara, Member, IEEE, and Makoto Yasuhara, Member, IEEE

(Accepted for publication in the IEEE Transactions on Pattern Analysis and Machine Intelligence)

The authors are with Department of Information Management Science, Graduate School of Information Systems, the University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, JAPAN e-mail: {qiaoyu, mikihiro, yas}@math-sys.is.uec.ac.jp
**Abstract**

Restoration of writing order from a single-stroked handwriting image can be seen as the problem of finding the smoothest path in its graph representation. In this paper, a 3-phase approach to restore a writing order is proposed within the framework of the *Edge Continuity Relation (ECR)*. In the initial, local phase, in order to obtain possible ECRs at an even-degree node, a neural network is used for the node of degree 4, and a theoretical approach is presented for the node of degree higher than 4 by introducing certain reasonable assumptions. In the second phase, we identify double-traced lines by employing maximum weighted matching. This makes it possible to transform the problem of obtaining possible ECRs at odd-degree node to that at even-degree node. In the final, global phase, we find all the candidates of single-stroked paths by depth first search and select the best one by evaluating SLALOM smoothness. Experiments on static images converted from online data in the Unipen database show that our method achieves a restoration rate of 96.0%.

**Index Terms**

Handwriting recognition, Writing order restoration, Edge continuity relation, Temporal information, Graph matching, Euler path

1. **INTRODUCTION**

Automatic handwriting recognition has received significant attention for more than 40 years [1]. Basically, handwriting recognition can be categorized into two types, online and offline, which differ in the input device and the information available for recognition. In an online recognition system, a tablet digitizer together with a pen is used to record the coordinates of pen-tip position as a function of time, while in an offline recognition system, a scanner is used to capture 2-dimensional static image. Online recognition has found acceptance in many applications with commercial success, while offline recognition for unconstrained handwriting remains a hard problem [1], [2], [3], [4]. The success in the online recognition can largely be ascribed to the availability of temporal information of pen-tip movement. The importance of online information for recognition is supported also by psychological researches, which indicate that even humans make use of dynamic writing information for the perception of characters [5], [6].
This study is concerned with the problem of how to restore writing order from its static image. The recovery of writing order can be seen as the transformation of a 2-dimensional handwriting image into an ordered sequence of \((x_i, y_i)\), where \((x_i, y_i)\) represents the pen-tip coordinates at the \(i\)-th point sampled at equally spaced intervals. This is one of the typical inverse problems of the human handwriting process. Like many inverse problems, our problem is ill-posed in the sense that the unique solution cannot be obtained without constraints. A usual approach to the inverse problem is to introduce constraints or principles which allow the ill-posed problem to be converted to a well-posed one. By limiting our studies to single-stroked line images produced by cursive handwriting, we employ the minimum energy cost criterion as a guiding principle based on the observation that handwriting is usually generated in a smooth way with the least energy cost.

Previous studies to restore temporal information can be divided into two categories: local line trace and global graph search. In the former, the next tracing path is selected at each of the branch points based on current configuration and tracing history. Fujimoto et al. [7] calculated direction codes to represent handwritten characters in the way they are written and used elastic matching for recognition. Lee and Pan [8] traced the skeleton of offline signatures by a set of heuristic rules. Doermann and Rosenfeld [9], [10] described a stroke recovery platform based on local, regional, and temporal clues, and reported detailed information which can be used for recovery. Boccignone et al. [11] recovered the most likely trajectories according to good continuity criteria by taking into account direction, length and width of the stroke. Govindaraju and Srihari [12] presented an approach for separating handwritten text from interfering strokes based on Gestalt segmentation and grouping principles. Liu, Huang and Suen [13],[14] proposed a stroke segmentation method for Chinese characters by use of polygonal approximation. Plamondon and Privitera [6] developed a scanning method to find a natural course for strokes by calculating curvature of contour.
In global graph search, usually a graph is constructed to describe the topological structure of an input image and the problem is regarded as finding the smoothest path which covers all the edges in the graph. Both Huang, Yasuhsara [2] and Jäger [15], [16] used a graph model to represent skeleton images and searched for an Euler or a Hamilton path that minimized a certain cost function. A similar approach was adopted by Lallican et al. [17], in which they used their Tabou algorithm to search for a Hamiltonian cycle. In [3], Bunke et al. used a directed graph with attributes to describe skeleton components and searched for a drawing path based on specific likelihood criteria.

The advantages of the local line tracing method come from its simplicity and low computational cost, however it is difficult to design general heuristic rules that can be applied to various handwriting styles. Though the global analysis on the graph model makes it possible to overcome the limitation of the local tracing method, it may lead to combinational explosion and require huge computational cost. Some approaches have tried to combine these two paradigms together. Kato and Yasuhsara [4], [18] presented a method that combined both the local and the global processing in a 2-phase framework. In [19], Qiao and Yasuhsara identified continuous segments within a probability framework and evaluated a goodness of each of the possible traces by a global smoothness calculation. However, the precise estimation of the probability of continuous segments is not easy.

In this paper, a new concept named Edge Continuity Relation (ECR) is introduced to restore the writing order of single-stroked handwriting. Our approach to obtain ECRs is presented in three phases. In the first phase, we obtain the possible ECRs at even-degree node. This is achieved through a new perspective which distinguishes node type between crossing and touching. We apply a neural network to nodes of degree 4 to identify its possible type. A series of theorems are proved for node of degree higher than 4 under two assumptions. In the second phase, a weighted matching of general graphs is applied to identify double-traced
line(s) connected to the odd-degree node. Identification of all the double-traced lines makes it possible to transform the problem of obtaining the ECR at an odd-degree node to that at an even-degree node. In the final phase, a global tracing algorithm is presented to determine ECRs left unspecified in the previous phases as well as to find all the candidate paths from a start node to an end node. The smoothness is evaluated for each of the candidate paths based on the SLALOM method [20] which is equivalent to Spline approximation, and the smoothest path is selected as the final result.

This study not only proposes a systematic approach to be applied in practice, but also defines the restoration problem for writing order of single-stroked handwriting in a clearer, more general form and restates it within the framework of the ECR. Our purpose is to understand the degree to which this problem can be addressed using less heuristic knowledge and at as low a computational cost as possible. The description of the problem within the framework of the ECR is helpful not only to present it in a simple and convenient form but to deepen our insight into the intrinsic nature of the problem as well. This paper discusses these points in depth and presents a novel approach to the restoration problem for single-stroked handwriting. Theoretically, the proposed method has no limitation as to the number of lines traversing through a node and can deal with touching junctions. The method is also efficient in identifying double-traced lines and dealing with the odd-degree nodes of degree 5 or higher. A portion of this study has been published in an earlier conference paper [21].

The rest of the paper is organized as follows. Section 2 describes briefly the construction of a graph model and node classification. The concept of the ECR is introduced in Section 3 and the methods to obtain candidate ECRs at even-degree and odd-degree nodes are presented in Section 4 and Section 5, respectively. Section 6 describes graph division and global tracing algorithms. Experimental results are summarized in Section 7. Our conclusions and some thoughts for future studies are given in Section 8 and in Section 9, respectively.
2 GRAPH MODEL

In this section, we explain how to construct a graph model from the skeleton of an input handwriting image. Before applying a thinning algorithm to obtain the skeleton image, we use the smoothing filter in [22] to reduce various types of noise, such as small peaks and holes in the input image. The graph model, denoted by \( G=(M, E, R) \), is used to represent both the topological structure and the positional information of the input image, where \( M \) and \( E \) are sets of nodes and edges, respectively, and \( R \) contains the location information of nodes and edges.

2.1 Analysis on Skeleton Image

In order to construct graph model \( G \), it is necessary to extract vertices and segments from the skeleton image. Each of the pixels in the skeleton belongs to one of three types: terminal pixel, connection pixel or junction pixel depending on whether it is adjacent to one, two, or more pixels, respectively. A vertex is composed of a single terminal pixel or a cluster of junction pixel(s). We identify the terminal pixels and the junction pixels by the method proposed in [14]. A sequence of connecting pixels between two vertices is referred to as a segment.

The thinning procedure iteratively deletes unnecessary contour pixels until a skeleton of 1-pixel width remains without altering the topological structure of the input image [23]. It works well for strokes with smooth contours, but in noisy or stroke junction areas where the contours are eroded or overlapped, the skeleton may include artifact lines [1], [4], [19], [23].

A real segment (r-segment) corresponds to a part of real stroke, while a spurious segment (s-segment) is an unwanted artifact resulting from thinning, which never exists in the original stroke. In order to construct the graph model, we need to discriminate the s-segments from the r-segments. We use the method from our previous work [19], which makes use of segment length and stroke width. Examples of the r/s-segments are shown in Fig. 1a.
Although the skeleton obtained by thinning preserves the topological structure of an input image, it could be contaminated by noise. To overcome this shortcoming, the geometrical feature, i.e. directional angle, is calculated from its original image. Also, for areas where more than two lines join together, our analysis relies mainly on the topological information. It should be noted that although the identification of r/s-segments is largely dependent on the thinning method used, some erroneous identifications will be corrected by the later analyses such as the large node analysis in Section 4 and the double-traced line identification in Section 5. It might safely be said that problems caused by the thinning process are relieved to some extent.

2.2 Node

After all the s-segments are identified in the skeleton image, each of the r-segments is regarded as an edge of \( G \). A set of the connected s-segments together with the associated vertices, or a single vertex, is seen as a node of \( G \). We use \( N=(V_N, S_N, R_N) \) to denote a node graph model (Fig. 1b, 1c), where \( V_N \) and \( S_N \) denotes a set of the vertices and the s-segments of \( N \), respectively, and the positional information of the vertices and the s-segments is retained in set \( R_N \). The node graph model is introduced for the purpose of further analyses on its interior structure, as the configuration of the s-segments contains useful clues to infer the continuity relations of handwriting among the edges connected to the node. Details will be given in
Sections 3 and 4.

The degree of node $N$ is the number of the edges connected to it and denoted by $d(N)$. A vertex in $N$ connected by both r-segment(s) and s-segment(s) is called an exterior vertex of $N$, while a vertex connected by s-segment(s) only is referred to as an interior vertex of $N$ (Fig. 1b, 1c).

A node can be classified into Cycle-structured node (C-node) (Fig. 1b) and Tree-structured node (T-node) (Fig. 1c) depending on its interior structure. The former is composed of a graph of cycle structure, while the latter is of tree structure.

A stroke is a handwriting path from pen-down to pen-up. A line represents a part of stroke that may contain one or more consecutive edge(s) of $G$. A restored handwriting path from start node to end node is called a trace. A node corresponds to the end of edge, while a vertex to the end of segment. Hence a node includes one or more vertices.

3. EDGE CONTINUITY RELATION

3.1  Introduction of Edge Continuity Relation at node

Restoration of the writing order from a single-stroked handwriting image can be seen as the problem of finding the smoothest path that passes through all the edges in $G$ at least once. The exhaustive search of all possible paths is not practical, as this will lead to combinatorial explosion - the number of such the paths is larger than $2^m$, where $m$ is the number of nodes of degree 4 or higher (the proof of this is available in the Supplemental Material).

In order to reduce complexity, our study is based on the framework of Edge Continuity Relation (ECR). The ECR is defined as a set of continuous pairs which cover all the edges connected to a node. By "continuous pair", we mean a pair of two edges connected to the node which pass continuously along the stroke (Fig. 1a). Consider node $N$ connected by $n$ edges $e_1, e_2, \ldots, e_n$, $ECR(N)$ at $N$ is:

$$ECR(N) = \{(e_i, e_j) | i = 1, \ldots, K, and s_i, t_i \in \{0, 1, \ldots, n\}\}$$  (1)
where \((e_1, e_0)\) denotes a continuous pair, \(K\) is the number of continuous pairs at \(N\) and \(e_0 = NULL\). If there is edge \(e_1\) that has no continuous pair at \(N\) (called a terminal edge), we introduce a special continuous pair \((e_1, e_0)\).

An analysis is made to obtain all the possible ECRs at each node based on its local structure as shown in Fig. 2. We will transform the problem of obtaining the ECRs at odd-degree node to that at even-degree node by identifying the double-traced and terminal lines. For a C-node of even degree, it will be decomposed into multiple T-nodes. A T-node will be divided into two classes: the nodes of degree 4 and the nodes of degree higher than 4. A neural network is used for the former class to distinguish between crossing and touching, while a theoretical approach is applied to the latter class by introducing certain simple, reasonable assumptions. Details will be described in Section 4 and Section 5.

### 3.2 Multiplicity

The *multiplicity* of interior s-segment \(s\) in node \(N\) of degree \(2n\) \((n \geq 2)\), denoted by \(\rho(s)\), is the number of times \(s\) is passed along the whole trace. Note that multiplicity does not make sense...
when considering ordinary line images, but does for handwriting images. Given 
\[ ECR(N) = \{(e_s, e_i)\} \] for each \( i = 1, 2, \ldots, n \), we find interior path \( L_{e_s-e_i} \) between the two exterior vertices, to each of which \( e_s \) or \( e_i \) is connected. Since node graph \( N \) is limited to a small local area, the interior path must be almost straight, so \( L_{e_s-e_i} \) can be found as the shortest path between \( e_s \) and \( e_i \). In this paper, we assume that there exists only one path with the shortest length.

Then multiplicity \( \rho(s) \) can be calculated by:

\[
\rho(s) = \sum_{i=1}^{n} g(s, L_{e_s-e_i}) \quad i = 1, 2, \ldots, n, \tag{2}
\]

where \( g(s, L_{e_s-e_i}) = \begin{cases} 
1 & \text{if } s \in L_{e_s-e_i}, \\
0 & \text{otherwise}.
\end{cases} \)

Note that \( 1 \leq \rho(s) \leq n \) holds. Examples are shown in Fig. 1b, c.

4 EDGE CONTINUITY RELATION AT EVEN-DEGREE NODE

In this section, we show how to obtain the possible ECRs by exploring the interior structure of even-degree nodes. The analysis is executed separately for nodes with degree 4 and those with degree higher than 4.

4.1 Node Type - Crossing and Touching

Even-degree nodes can be classified into two types: crossing and touching. A crossing node (Fig. 3a) is named either \( T_c \)-node or \( C_c \)-node depending on whether it is of tree or circle structure. All the lines that traverse through a crossing node cross exactly once. A pair of lines
that traverse through a node without crossing each other is called a *touching pair*. A touching node (Fig. 3b), classified as either a $T_h$-node or a $C_h$-node, has at least one touching pair.

We introduce the *Crossing Node Traversing Rule (CNTR)* as follows:

**CNTR:** *For crossing node $N$ of degree $2n$, the edge continuous to $e_k$ ($k=1,2,\ldots,n$) is determined uniquely as the $n$-th edge $e_{k+n}$ counting from $e_k$ either clockwise or counter-clockwise.*

The $ECR$ at crossing node $N$, denoted by $ECR_a(N)$, is represented by,

$$ECR_a(N) = \{(e_k, e_{k+n})| k=1,2,\ldots,n \},$$

where $(e_k, e_{k+n})$ is called a *CNTR pair*. We have the following theorem for the CNTR.

**Theorem 1:** *CNTR holds if and only if $N$ is a crossing node.*

Due to page limitations, all proofs from this point onward are provided into the Supplemental Material.

We have made extensive experimental studies to estimate two statistics: (1) frequency of nodes with degree 4, and (2) frequency of crossing nodes out of all such nodes. We have found that 95.8% among all the nodes of degree $\geq 4$ have degree 4, and also that 95.1% among all the nodes with degree four are of crossing type. Taking advantage of this a priori knowledge, we will analyze how to obtain $ECRs$ separately for the 4-degree case and for nodes of degree higher than 4.

### 4.2 ECR at Node of Degree 4

Consider node $N$ connected by 4 edges $e_1,\ldots,e_4$ as shown in Fig. 4a. There are three cases for $ECRs$: $\{(e_1, e_3), (e_2, e_4)\}$, $\{(e_1, e_4), (e_2, e_3)\}$, and $\{(e_1, e_2), (e_3, e_4)\}$. The first is of crossing type, named *type 1*, and the other two cases are of touching type, named *type 2* and *type 3*. 
We need to determine the type of the ECR at \( N \). Our experiments have indicated that the determination of continuous pairs based solely on the local configuration around the node is hopeless. Here instead of obtaining the ECR in the direct manner, a screening function will be prepared to examine whether or not the touching type is possible. If the touching type is not possible, the node is labeled as crossing type (\( X\)-type), otherwise as questioned type (\( Q\)-type).

In order to devise such a screening function, we train a 3-layered neural network with 6 inputs, a hidden layer composed of 22 units, and 1 output. The neural network is applied because of its ability to learn complex nonlinear discriminative function from noisy samples \([24]\). When the neural network is presented with six features (the four tangential directions of \( e_1 \) through \( e_4 \), the directional angle of the vector between the two end vertices of \( s \), and the length of \( s \)), a response, either X-type or Q-type, is obtained at its output. If the node is identified as X-type, the ECR at that node is determined as type 1. Otherwise, all the three types (type 1-3) are reserved as possible, and the final determination of the type will be delayed for further analysis.

The tangential direction \( \alpha \) of edge \( e \) is calculated depending not on the skeleton image but on its corresponding original image as shown Fig. 4b, as the thinning algorithm is very sensitive to noise. For each black pixel \( p_k \) in the original image, we define its nearest pixel \( N_m(p_k) \) in the skeleton as:

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

(4)

where \( \text{Dis}(p_k, p_s) \) is the Euclidean distance between two pixels \( p_k \) and \( p_s \). Let \( p \) represent the
pixel on $e$ adjacent to $N$. We find line $B$ with length $2w$ ($w$ is the stroke width) by tracing along $e$ from $p$, and obtain the associated stroke part $A$ of $B$ by:

$$A = \{p_k \mid N_m(p_k) \in B\}. \tag{5}$$

Then we apply principal component analysis (PCA) to the set of coordinates of the black pixels in $A$. The first principal vector of the PCA corresponds to the direction of the maximum-variance and gives the best linear summary of these pixels [25]. Therefore $\alpha$ can be obtained as the direction of the first principal vector.

We have collected a total of 200,652 node samples from offline images obtained by converting the online data from the Unipen database [26]. Among them, 18,500 degree-4 node samples were used for training the neural network. The results of this test are summarized in Table 1. 91.1% out of the 173,159 crossing nodes were identified correctly as X-type, and 90.3% out of the 8,993 touching nodes were labeled as Q-type. This means that 12.9% among all the test samples were labeled as Q-type. The screening error rate was only 0.47% ($=873/(173,159+8,993)$) among all the nodes examined.

<table>
<thead>
<tr>
<th>Table 1 X-type/Q-type identification result of the neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossing nodes</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>Identified as X-type</td>
</tr>
<tr>
<td>Identified as Q-type</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

4.3 ECR at Node of Degree Higher Than 4

Although the neural network works well for the nodes of degree 4, it is difficult to adopt the learning based method to obtain the ECR at nodes of degree higher than 4. This is due not only to their complicated structures but to the difficulty of collecting enough training samples as well. Moreover, the skeleton around nodes of high degree is always highly deformed. This makes it difficult to extract reliable geometrical features such as directional angle. Since many thinning algorithms can preserve the topological structure of the original image [23], we can
use the topological information to infer the possible ECRs.

### 4.3.1 Assumptions and Pseudo Multiplicity

Generally, without any constraint, the total number of possible ECRs at node $N$ of degree $2n$ ($n>2$) is $(2n-1)(2n-3)\ldots3$. In the following, we introduce two reasonable assumptions to simplify the problem of determining the ECRs.

**Assumption 1:** *At most one pair of touching lines traverses through a node.*

This assumption is due to the smoothness of human handwriting: touching lines usually occur with higher curvature than crossing lines. Under this assumption, the number of the possible ECRs at $N$ can be reduced greatly to $n+1$.

**Theorem 2:** Under Assumption 1, there exists no line between two touching lines traversing through a node.

There are three possible configurations of the touching pair depending on the interior paths of $N$ (Fig. 5): (a) there exists no common segment or vertex though which two touching lines $L_{A'B'}$ and $L_{B'A'}$ pass, (b) a common s-segment $\sigma$ exists, or (c) a common vertex $\nu$ exists.

**Assumption 2:** *Two touching lines have no common segment or vertex through which both of them pass, that is, only case (a) in Fig. 5 is acceptable.*

This is because the common segment of the two smooth touching lines will be a long one, and the long segment will usually be identified not as a s-segment, but as a r-segment.
We found experimentally that 95.5% of the nodes satisfy the two assumptions above. Most of the nodes not satisfying the two assumptions are so deformed or noisy that it is difficult even for humans to determine their ECRs correctly.

In order to obtain the possible ECRs at nodes of degree higher than 4, instead of determining the ECRs directly depending on the local analysis outside the nodes, we begin with examining the existence of the touching lines by exploring the interior structure of the node. Multiplicity $\rho(s)$ can be used to analyze the node structure and thus enables us to infer the possible location of the touching lines traversing through the node. However, multiplicity cannot be calculated unless the ECR is known. Instead, we introduce the pseudo-multiplicity $\mu(s)$ of interior s-segment $s$ of $N$. $\mu(s)$ is defined as the multiplicity of $s$ under the assumption that $N$ is of crossing type, thus $\mu(s)$ is available at any time it is necessary.

4.3.2 Edge Continuity Relation at T-Node

- Calculation of Pseudo-Multiplicity at T-Node

In the case of a T-node, there exists a simple way to calculate the pseudo-multiplicity of the interior s-segment, which is guaranteed by the following theorem.

**Theorem 3:** The pseudo-multiplicity $\mu(s)$ of s-segment $s$ contained in T-node $N$ of degree $2n$ ($n \geq 3$) is given by,

$$\mu(s) = \min(n_1, n_2),$$

where $s$ separates $N$ into 2 disjoint sub-nodes $N_1$ and $N_2$, each of which has $n_1$ and $n_2$ edges connected ($n_1 + n_2 = 2n$), respectively.

**Corollary:** The following is satisfied for s-segment $s$ contained in the T-node of degree $2n$.

$$2 \leq \mu(s) \leq n.$$  \hfill (7)

- MS-Segment and MS-Vertex

The edge that separates a graph into two disjoint sub-graphs is called a *bridge*. The vertex that, if removed, separates a graph into multiple disjoint sub-graphs is called a *cut vertex*. Note that
all the edges and vertices of a tree are bridges and cut-vertices, respectively. Consider T-node $N$ of degree $2n$, a spanning segment is defined as an interior bridge s-segment, denoted by $\sigma_p$, that has the largest pseudo-multiplicity $p$ where $p=\max\{\mu(s_i)\}$. $\Omega=\sigma_n$ is called a Maximum Spanning segment (MS-segment). A Maximum Spanning vertex (MS-vertex), denoted by $\Phi$, is defined as a cut vertex where $\Sigma(\mu(s_i))=2n$ ($s_i$ represents the $i$-th segment incident to $\Phi$). Two examples are given in Fig. 6a (MS- vertex) and Fig. 6b (MS- segment).

**Theorem 4:** There exists either exactly one MS-segment $\Omega$ or exactly one MS-vertex $\Phi$ in a T-node of even degree.

![Image of MS-vertex and MS-segment](image)

**Procedure of Obtaining ECR at T-Node**

From Theorem 4, a T-node of even degree must be identified as either one of the following two cases: (1) the T-node with exact one MS-segment, or (2) the T-node with no MS-segment but one MS-vertex. Under Assumptions 1 and 2, we have the following theorem:

**Theorem 5:**

1. If $N$ contains exactly one MS-vertex $\Phi$, $N$ is a $T_x$-node.

2. Suppose that $N$ contains exactly one MS-segment $\Omega$.
   
   (a) if a touching pair traverses through $N$, $N$ is a $T_x$-node. One of the touching lines passes through one of the end vertices of $\Omega$ and another touching line passes through another end vertex of $\Omega$.

   (b) otherwise, $N$ is a $T_z$-node.
Based on this theorem, the possible ECRs at T-node $N$ of even degree can be obtained by executing the following.

1. If $N$ has an MS-vertex, $ECR(N) = ECR_x(N)$, which is given by Eq. (3).

2. If $N$ has an MS-segment, there are 2 possible ECRs. These cannot be resolved without exploring the exterior paths of $N$. The ECR will be finally determined by using the Global Tracing Algorithm described in Section 6.

4.3.3 Edge Continuity Relation at C-Node

* C-Node Decomposition

In order to obtain the ECR at C-node $N$, we will decompose $N$ into multiple tree-structured nodes called $V$-nodes. Assume that $N$ is composed of a set of connected $s$-segments $s_i, i=1,2,\ldots,I$ with pseudo-multiplicity $\mu(s_i) (0 \leq \mu(s) \leq n)$, where $n$ is the number of lines traversing through $N$. The $V$-node is a tree-structured sub-graph of the C-node graph, which is composed of a vertex or a cluster of connected $s$-segment(s) of multiplicity 2 or higher. A virtual real segment is defined as an $s$-segment of multiplicity 1. Examples are shown in Fig. 7.

To decompose C-node $N$ into the multiple V-nodes, it is necessary to find the virtual real segment(s). The problem is that the multiplicity of an $s$-segment is unknown before the writing order is restored and only pseudo-multiplicity $\mu(s)$ is available. Fortunately, we have Theorem 6 below which guarantees to find the virtual real segments based only on $\mu(s)$.

**Theorem 6:** Under Assumptions 1 and 2, for C-node $N$,

- **Case 1:** $N$ contains no $s$-segment of pseudo multiplicity 0.
  
  If $\mu(s)=1$, then $\rho(s)=1$.

  Such $s$ that satisfies $\mu(s)=1$ is a virtual real segment.

- **Case 2:** $N$ contains $s$-segment(s) of $s$ that satisfy $\mu(s)=0$.

  $N$ is a touching node and one of the touching lines passes through $s$. 

16
According to Theorem 6, the procedure to obtain the possible ECRs at C-node $N$ is given as follows.

1. If there is no s-segment $s$ of $\mu(s)=0$ interior $N$, label each of the s-segments $s$ of $\mu(s)=1$ as virtual real segment $s_v$, decompose $N$ into multiple different V-nodes by replacing each $s_v$ by a real segment, and apply the method described in Section 4.3.2 to obtain the possible ECRs at each of the V-nodes. The V-node of degree 4 is regarded as of crossing type.

2. If there exists s-segment $s$ of $\mu(s)=0$, the node is of touching type and the touching lines must pass through $s$. We preserve the touching ECR(s), whose associated touching lines pass through $s$, as the possible ECR(s).

5 EDGE CONTINUITY RELATION AT ODD-DEGREE NODE

An odd-degree node is connected by either a double-traced edge or a terminal edge. The odd-degree node can be classified into three types according to its shape: y-node, t-node, and i-node. The node of degree one is called an i-node and the edge connected to it is called an i-leg. Both the t-node and the y-node are of degree 3 or higher. The t-node is connected by a
terminal edge (t-leg), which has no continuous pair (Fig. 8b, d), while the y-node is connected by a double-traced edge (y-leg) and its two continuous edges are called y-hands (Fig. 8c, d, and e). A double-traced line, referred to as a d-line, may include one (Fig. 8c, d) or more edges (Fig. 8e), which always lies between the two odd-degree nodes. There are three kinds of d-lines: (1) y-i d-line that spans between a y-node and an i-node (Fig. 8c), (2) y-t d-line between a y-node and a t-node (Fig. 8d), and (3) y-y d-line between two y-nodes (Fig. 8e).

In order to obtain the ECR at an odd-degree node, it is necessary to identify a y-leg and a t-leg. Their identification is related directly to that of the d-line. In order to identify the d-line, most researchers have used local smoothness or curvature calculations [6], [8]. Their methods were sensitive to noise, especially for the odd-degree nodes of degree higher than 3. The references [4], [18], [19] took account of the topological information and classified the d-lines according to the degree and the type of the nodes. However, these methods relied heavily on the local tracing algorithm to find the d-lines. Instead, we propose a method to find the d-lines by applying maximum weighted matching (MWM) of general graph, which makes efficient use of not only the local smoothness but the global and topological information as well.

5.1 Search for Double-Traced or Terminal Edge by General Graph Matching

Graph Matching is a classical problem in graph theory. A maximum matching [27] tries to find as many independent edges as possible in a graph. Two edges are independent if they do not share a node. In the MWM, the goal is to find the maximum matching that has the highest/lowest total weight. Gabow [28] proposed a blossom-shrinking based algorithm
whose time complexity is $O(n(m+n\log n))$, where $n$ and $m$ are the number of nodes and edges, respectively. In our experiments, we used the implementation of Gabow’s algorithm in [29]. MWM can be used to solve the Chinese Postman Problem (CPP) [16], [30] that aims at finding the shortest path covering all the edges in a graph. Note the following method of d-line identification is similar to the technique to solve the CPP.

The difficulty lies in the fact that it is hard to determine which edge connected to the odd-degree node is the y-leg or the t-leg based solely on its local structure. A robust decision should be made in more global fashion. We can find all the candidate d-lines as the shortest paths between each pair of the two odd-degree nodes, as the d-line is usually straight. Since all the odd-degree nodes except for the ones at the start and the end are connected by exactly one d-line, the d-line must be independent in the sense that no two d-lines share the same odd-degree node. This enables us to apply the MWM to identify the d-lines among all the candidates by defining the tracing cost (weight) for each of the candidate d-lines.

(1) **Construction of graph $G_m$ for matching**

Construct graph $G_m=(O, C)$, where $O$ is a set of the odd-degree nodes in $G$ and $C$ is a set of the candidate d-lines, which will be found as follows. For every two odd-degree nodes, we calculate the shortest path $l$ between them, if $l$ satisfies conditions (a), (b), and (c) below, it is regarded as a candidate d-line. It is noted that these 3 conditions stem from heuristic knowledge of handwriting in general and are useful for simplifying $G_m$.

(a) Path $l$ must be straight. $\theta=\text{Dis}(l)/\text{Len}(l)$ ($\theta \leq 1.0$) is used to evaluate the straightness of $l$, where $\text{Dis}(l)$ is the Euclidean distance between its two end nodes and $\text{Len}(l)$ is the length of $l$. The nearer $\theta$ is to 1.0, the straighter $l$ is.

(b) The two end nodes of $l$ cannot simultaneously be i-nodes.

(c) Path $l$ must not contradict the possible ECRs obtained at the nodes lying along it.

(2) **Discrimination between y-node, t-node, and i-node**
We introduce the continuous cost function \( Ctg(e_i, e_j) \), which represents the smoothness between two contiguous edges \( e_i \) and \( e_j \). In our experiments, it is calculated as the directional difference:

\[
Ctg(e_i, e_j) = |\pi - |a_i - a_j||, \tag{8}
\]

where \( a_i \) and \( a_j \) are the tangential directions of \( e_i \) and \( e_j \), respectively, which can be calculated by the method described in Section 4.2. It is noted here that one can consider more factors such as those discussed in [9], [11] to define more precise continuous cost functions, however, more computation is required as well.

Take candidate d-line \( l_k \) in \( C \) together with its two end nodes \( N_k^i (i=1,2) \). For simplicity, \( Ctg(l_k, e_k^i) \) has the same meaning as \( Ctg(e_k, e_k^i) \) in the context of calculating the cost, where \( e_k \) is an end edge of \( l_k \) connected to \( N_k^i \). The t/y/i type of \( N_k^i \) is identified as:

(a) The node of degree 1 is labeled as i-node.
(b) If \( l_k \) includes a bridge edge of \( G \) and \( N_k^i \) is of degree 3 or higher, \( N_k^i \) must be a y-node due to the single stroke constraint.
(c) Suppose that \( N_k^i \) is of degree 3. Let \( e_k^{i1} \) and \( e_k^{i2} \) denote the two edges (not \( l_k \)) connected to \( N_k^i \). If \((Ctg(l_k, e_k^{i1})+Ctg(l_k, e_k^{i2}))/2<Ctg(e_k^{i1}, e_k^{i1}), N_k^i \) is a y-node, \( l_k \) is its y-leg, and \( e_k^{i1} \) and \( e_k^{i2} \) are its two associated hands, otherwise, \( N_k^i \) is a t-node and \( l_k \) is its t-leg.
(d) Suppose that \( N_k^i \) is of degree higher than 3. Among all the \( s \) edges \( e_k^{ij} (j=1,...,s, e_k^{ij} \neq l_k) \), find the best two continuous edges \( e_k^{im} \) and \( e_k^{in} \) \((m, n=1,2,...,s m\neq n) \) by minimizing \( Ctg(l_k, e_k^{im})+Ctg(l_k, e_k^{in}) \). If \( l_k \) forms the minimizing pairs of both \( e_k^{im} \) and \( e_k^{in} \), \( N_k^i \) is identified as a y-node, \( l_k \) is its y-leg, and \( e_k^{im} \) and \( e_k^{in} \) are its two associated y-hands, otherwise \( N_k^i \) is identified as a t-node and \( l_k \) is its t-leg.

(3) Calculation of cost of edge/line in \( G_m \)

(a) Calculate node continuous cost as follows: If \( N_k^i \) is a t/i-node, the node cost is set as a constant. If \( N_k^i \) is a y-node, the node continuous cost is defined as,
\[ \text{Cost}(l_k, N_k^i) = \text{Cost}(l_k, e_k^{im}) + \text{Cost}(l_k, e_k^{in}), \quad (9) \]

where \( e_k^{im} \) and \( e_k^{in} \) are the two associated hands of \( l_k \).

(b) If both the two end nodes of \( l_k \) are labeled as \( \overline{t}/i \)-nodes in step (2), \( l_k \) is removed from \( G_m \).

(c) The cost of \( l_k \) is calculated by,

\[ \text{Cost}(l_k) = \text{Cost}(l_k, N_k^1) + \text{Cost}(l_k, N_k^2). \quad (10) \]

(4) Matching

We employ Gabow’s matching algorithm [28] on \( G_m \) to obtain the maximum number of independent d-lines with the minimum total cost: \( \min \{ \Sigma_i \text{Cost}(l_i) \} \). If there are two isolated nodes to which no matching edge is connected, they must be either a start or an end node. Otherwise, we select the odd-degree node with the highest continuous cost in \( G_m \) as a start node and remove it together with its connected lines from \( G_m \), then employ the matching algorithm again. Each of the matching edges corresponds to a double-traced line. We need to identify terminal edge \( e_t \) connected to start/end node \( N_s \).

If \( N_s \) is an \( i \)-node, the only edge connected to it must be the terminal edge. Or if \( N_s \) is of degree 3 or higher, \( e_t \) is identified by:

\[ e_t = \arg \max_{e_i} \{ \text{Ctg}(e_i) \}, \quad (11) \]

where \( \text{Ctg}(e_i) = \min \{ \text{Ctg}(e_i, e_j) \mid i \neq j \} \), and \( e_i \) and \( e_j \) represent edges connected with \( N_i \).

(5) Checkup

The d-lines obtained from procedure (1)-(4) may include overlapped ones. There are two kinds of overlapping: intra-overlapping (Fig. 9a) and inter-overlapping (Fig. 9b). Since only the intra-overlapped d-line occurs in normal handwriting, the inter-overlapped d-lines should be removed from \( C \), if they exist, and matching procedure (4) is executed again.

An example of the identification procedure for d-lines is illustrated in Fig. 10.
Figure 9 Overlapped d-lines. M-N and P-Q represent the two d-lines. For the intra-overlapped case (a), d-line P-Q is located completely inside of d-line M-N, while for the inter-overlapped case (b), P-Q is located partially inside of M-N. Only case (a) is assumed acceptable since case (b) rarely occurs in normal handwriting.

Fig.10 Search for d-lines by matching. Each of the capital letters in (a) denotes a node. The graph model of input image (a) is shown in (b), where the number attached to each of the lines represents its cost. In (c), the thick lines represent the identified d-lines at step (4). Since B-D and C-E are inter-overlapped, we remove them and apply the matching algorithm again. Finally three d-lines BE, CD, and IH are identified and nodes A and F are found as the start and end nodes as shown in (d).

5.2 Edge Continuity Relation at Odd-Degree Node
After identifying the d-lines connected to the odd-degree nodes, it becomes easy to obtain the ECR at the odd-degree nodes. In fact, since the t-node and the y-node become even degree by removing the t-leg or duplicating the y-leg as identified, the problem can be transformed to that of even-degree node as explained in Section 4.

6 GLOBAL TRACE AND SMOOTHNESS EVALUATION
In this section, we perform a global analysis to find the smoothest path within the graph model based on the possible ECRs obtained in Section 4 and the d-lines identified in Section 5 by executing two steps: (1) Graph Reduction, and (2) Global Tracing and Evaluation.
6.1 Graph Reduction

The continuous pairs in the possible ECRs can be divided into a *determinate group* and an *indeterminate group*. The former is composed of the ECRs, all of whose continuous pairs have already been determined uniquely in Sections 4 and 5, and the latter is composed of the ECRs, one of whose continuous edges is left undetermined. We reduce G to a simpler model $G_r$, called a *reduced graph model*, by applying 3 kinds of operators to the determinate continuous pairs: *Merge*, *Separate*, and *Replace*. Similar operations had been adopted in [4].

![Figure 11 Separate and Replace Operator](image)

(a) Separate t-node  
(b) Replace y-i d-line  
(c) Replace y-y d-line

(1) **Merge:** If $(e_i, e_j)$ is identified as a determinate continuous pair, merge $e_i$ and $e_j$ into a new edge $e_{ij}$, add $e_{ij}$ to $G$, and remove $e_i$ and $e_j$ from $G$.

(2) **Separate:** If $e_i$ is identified as a t-leg connected to a t-node, separate $e_i$ from the t-node, add to $G$ a new terminal node $N'$ to which $e_i$ is connected (Fig. 11a).

(3) **Replace:** (a) For a y-i/y-t d-line, replace the two hands of the y-node and the d-line by a new edge $e$ (Fig. 11b).

(b) Replace a y-y d-line together with the two y-nodes by a new node $N$ of degree 4 to which the four hands are connected (Fig. 11c). The d-line is seen as an interior s-segment of $N$. $N$ has two possible ECRs: $ECR_1=\{(e^{1l}, e^{1l}), (e^{2l}, e^{2l})\}$ and $ECR_2=\{(e^{1l}, e^{2l}), (e^{2l}, e^{1l})\}$.

After applying the 3 operators above to all the determinate continuous pairs found, there will be no d-line or t-node left in $G_r$. The only two odd-degree nodes left are the start and the end nodes of degree 1. All the other even-degree nodes have indeterminate ECRs.
6.2 Global Tracing and Smoothness Evaluation

After reducing $G$ to $G_r$, we can employ a global tracing and evaluation algorithm to find the smoothest single trace as a final result. Because there is no d-line left, all the edges in $G_r$ must be traced exactly once, thus the problem has become to find the smoothest Euler path in $G_r$.

An Euler path [30] of a graph is a path which visits all the edges exactly once. The simplest way is to enumerate all the candidate Euler paths in $G_r$ and to select the smoothest one by evaluating the smoothness for each of them. However, this brute-force approach will still lead to great computational complexity. In order to significantly reduce the complexity, the principle of divide and conquer is applied. Specifically, $G_r$ is divided into multiple independent sub-graphs so that the search for the smoothest Euler path can be executed separately for each of the sub-graphs. Suppose $G_r$ can be divided into $K$ sub-graphs $G_k$ ($k=1,\ldots,K$), each of which includes $n_k$ Euler paths. Without dividing, we have $\prod_{k=1}^{K} n_k$ Euler paths, for each of which the smoothness must be evaluated. If dividing is applied, we have only $\sum_{k=1}^{K} n_k$ Euler paths.

$G_r$ is divided by cutting bridge(s) and 2-cut set(s), where a 2-cut set is defined as a set of two edges whose removal disconnects the graph. Let $N_S$ denote the start node and $N_E$ denote the end node of $G_r$. We have the following theorem:

Theorem 7: (1) If a bridge is cut to divide $G_r$ into two disjoint sub-graphs $G_1$ and $G_2$, then $N_S$ and $N_E$ must belong separately to either $G_1$ or $G_2$.

(2) If the two edges in the 2-cut set are cut to divide $G_r$ into two disjoint sub-graphs $G_1$ and $G_2$, then both $N_S$ and $N_E$ belong to the same sub-graph, either $G_1$ or $G_2$.

Note that according to (1), each of the Euler paths of $G_r$ is produced by concatenating the respective Euler paths of $G_1$ and $G_2$, and therefore the smoothest Euler path of $G_r$ is constructed by concatenating the respective two smoothest Euler paths found in $G_1$ and $G_2$. 
Moreover, according to (2), if $N_S$ and $N_E$ belong to $G_1$, the smoothest Euler path of $G_r$ contains
the smoothest Euler path of $G_2$. Note that there are efficient algorithms of linear order to find
all the bridges and 2-cut sets based on depth first search [31].

The global tracing and evaluating algorithm includes three procedures: (1) main-procedure
($GLOBAL\_TRACE$), (2) depth first search and smoothness evaluation procedure
($DFS\_EVALUATE$), (3) tracing procedure ($TRACE$). In the main procedure, we divide the
graph recursively into disjoint sub-graphs by breaking bridges or 2-cut sets. If the sub-graph
cannot be divided any more, $DFS\_EVALUATE$ is called to find the smoothest Euler path.
Procedure $DFS\_EVALUATE$ includes two steps: tracing and evaluating the smoothness. In the
tracing step, we find all the Euler paths of each sub-graph based on depth first search, which
is implemented by a separate procedure, $TRACE$. In the evaluating step, we select the
smoothest path among all the candidates by maximizing the SLALOM [20] smoothness. The
details of the SLALOM smoothness calculation were described in our earlier work [2], [19].

**Main PROCEDURE $GLOBAL\_TRACE(G)$**

BEGIN
1. IF there exists a bridge $e$ in $G$ THEN
   1.1 cut $e$ to divide $G$ into two sub-graphs $G_1$ and $G_2$
   1.2 set $path_1 = GLOBAL\_TRACE(G_1)$
   1.3 set $path_2 = GLOBAL\_TRACE(G_2)$
   1.4 return $path_1 + path_2$ [concatenate $path_1$ and $path_2$]
2. ELSE IF there exists a 2-cut set ($e_1$, $e_2$) of $G$ THEN
   2.1 cut both $e_1$ and $e_2$ to divide $G$ into two sub-graphs $G_1$ and $G_2$
      [suppose the start/end nodes of $G$ belong to $G_1$]
   2.2 set $path = GLOBAL\_TRACE(G_2)$
   2.3 set $G_1 = G_1 \cup path$ [add $path$ into $G_1$]
   2.4 return $GLOBAL\_TRACE(G_1)$
3. ELSE
   return $DFS\_EVALUATE(G)$
END IF
END PROCEDURE
SUB PROCEDURE DFS_EVALUATE(G)
BEGIN
1. Find start node $N_s$ and end node $N_d$ and their associated start/end edges $e_s$ and $e_d$, respectively.
2. Initialize $F_g(M)=0$ for each node $M$ with undetermined continuous relations. $[F_g(M)=0$ means that $M$ has not been traced, and $F_g(M)=k$ represents the $k$-th possible ECR is currently selected, here $k=1,...,K_m$ and $K_m$ is the total number of possible ECRs at $M$]
3. set $path\_list = \Phi$ and $path = \Phi$ [\Phi represents empty set]
4. $TRACE(G, N_s, e_s, path, 1)$
5. Calculate the SLALOM smoothness for all the paths in $path\_list$
6. return the smoothest $path$ in $path\_list$
END PROCEDURE

SUB PROCEDURE TRACE(G, S, e, path, i)
BEGIN
1. $path[i] = e$ [set the $i$-th edge in $path$ as $e$]
2. Trace $e$ from node $S$ to obtain another end node $M$ of $e$
3. IF $M=N_e$ THEN
   3.1 if all the edges in $G$ are traced, then put $path[1...i]$ into $path\_list$
   3.2 return
4. ELSE IF $F_g(M)\neq0$ THEN
   4.1 set $e =$ continuous pair edge of $e$ at node $M$ under the $F_g(M)$-th ECR
   4.2 $TRACE(G, M, e, path, i+1)$
5. ELSE IF $F_g(M)=0$ THEN
   5.1 FOR $k=1: K_m$
      5.1.1 set $F_g(M) = k$
      5.1.2 set $e =$ continuous pair edge of $e$ at node $M$ under the $k$-th ECR
      5.1.3 $TRACE(G, M, e, path, i+1)$
   END FOR
   5.2 set $F_g(M)=0$
END IF
6. return
END PROCEDURE

7 EXPERIMENTS
7.1 Restoration Results

Fig. 12 Examples of T/C-node and d-line: The thin line represents s-segment and the number attached to it denotes pseudo-multiplicity. The thick line represents r-segment and the thick gray line represents d-line.

To examine the performance of the proposed method, we prepared two different sets of test images. The first set includes 187 single-stroked offline images collected by ourselves, which contains letters, cursive English words, cursive Chinese characters, and artificial shapes. These images cover most of the possible combinations of the touching and crossing nodes as well as the double-traced lines. For each of these images, we specified its writing order manually. The second set includes 708,811 static images obtained by converting online data of the Unipen database [26]. The Unipen database contains a huge number of samples with wide variety of handwriting styles. Each of the strokes in the database, the online data between pen-down and pen-up, was converted to a single-stroked offline image by connecting the sequential points. The width of the stroke was set as 3 pixels. It should be noted that although some samples in the Unipen are very noisy (Fig. 14a, 14g), we applied no online smoothing filter. Examples of the node analysis and the d-line identification are shown in Fig. 12. After restoring a trace for each of the offline images, we compared the restored trace with the original online one by calculating the Dynamic Time Wrapping (DTW) distance [32] between them. A total of 178 images in the first set and 680,255 images in the second set were restored successfully and the correct restoration rate was 96.0% in total. Several successful examples are shown in Fig. 13. The average time required was 135ms/sample. Of this, 82ms(60.7%)}
was consumed by the thinning process. We changed the stroke width to 4 and then to 5, and executed experiments for a small database (henceforth named SM-database) that contained 10,014 strokes selected randomly from the Unipen database. The restoration rates were 95.6% and 95.5%, respectively.

![Figure 13 Examples restored successfully. 4 images in the first row are from the first set and the others are from the second set.](image)

We also compared our current method with the Basic Tracing Algorithm (BTA) approach [4], the Probability approach [19], and the Optimal Euler Path approach [33] based on the SM-database above. In [4], the d-lines were detected by graph labeling and the BTA was used to find the trace. In [19], the ECR at a node was analyzed in a probability framework based on the directional angles between edges. The Optimal Euler Path (OEP) [33] is defined as the
trace which passes over each of the edges at least once and minimizes the sum of a continuous cost. The continuous cost was defined for each of the two contiguous edges to represent the smoothness. The writing order is recovered by finding the OEP [33]. The experiments were executed on a PC with P4-3G-CPU and 1G memory. The restoration results and time cost are shown in Table 2. The current approach has the highest performance among all those we tested, while the time cost is comparable.

Table 2 Comparison of restoration rates and time cost

<table>
<thead>
<tr>
<th>Method</th>
<th>Restoration rates</th>
<th>Time Cost (ms/sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Trace Algorithm [4]</td>
<td>91.6%</td>
<td>118</td>
</tr>
<tr>
<td>Probability Framework [19]</td>
<td>91.4%</td>
<td>188</td>
</tr>
<tr>
<td>Optimal Euler Path [33]</td>
<td>93.7%</td>
<td>122</td>
</tr>
<tr>
<td>Current Approach</td>
<td>96.0%</td>
<td>135</td>
</tr>
</tbody>
</table>

7.2 Recognition Results

In order to compare the recognition rates obtained for the 3 cases of the data: online trace, static 2D image, and restored trace, we have conducted experiments on the Unipen 1a digit database [26]. Each of the online samples was normalized in size by 120×80 and then we generated its static image according to method explained in Section 7.1. In order to be fair in all three cases, for the online and restored data, 50 points were sampled at equal intervals along the trace and a vector of the (x, y)-coordinates of the sampled points was used as an input feature. The static image was divided into 100 blocks of size 12×8 and the number of black pixels in each block was used as an element of the feature vector. The k-nearest neighbor classifier was selected as the recognition engine because of its known good performance [34].

Among the 15,612 samples in total, 4,683 samples were selected randomly for the test. The experimental results are shown in Table 3. No significant difference is observed in the recognition performances between the online case and the restored trace case, while compared
with these two cases, significantly lower performance can be observed for the case of the static image.

For some restored samples, although their traces are restored incorrectly, we had correct recognition results. This is because some of the redundant lines (i.e., the overlapped lines explained in Section 7.3.3b) happen to be removed during restoration, and this results in a cleaner input and thus improves recognition. In our experiments, more than 10% of the samples that had been misclassified in the online recognition experiments were correctly classified in the restored recognition ones. Similar phenomena had been observed and discussed in [35].

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Path</td>
<td>96.00%</td>
<td>95.98%</td>
<td>96.22%</td>
<td>95.79%</td>
</tr>
<tr>
<td>Restored Trace</td>
<td>95.04%</td>
<td>94.99%</td>
<td>94.23%</td>
<td>94.68%</td>
</tr>
<tr>
<td>Static Image</td>
<td>90.36%</td>
<td>89.96%</td>
<td>90.56%</td>
<td>90.02%</td>
</tr>
<tr>
<td>Corrected*</td>
<td>19.78%</td>
<td>17.55%</td>
<td>15.81%</td>
<td>14.2%</td>
</tr>
</tbody>
</table>

* number of samples misclassified in online experiment but recognized in restored experiment ×100%

number of samples misclassified in online experiment

### 7.3 Error Analysis

The errors caused at each of the 3 phases in the proposed approach will result in the wrong restoration of the writing order. We have collected 325 samples restored incorrectly for a more detailed error analysis. The errors may originate mainly from the following 7 sources:

1. Noisy or deformed image: Some images are so deformed or noisy that the skeleton images cannot reflect their structures well (Fig. 14a),

2. Misidentification of the r/s-segment (Fig. 14b,c),

3. Missing node:
Figure 14 Examples failed to restore.

(a) Two i-nodes coincide each other, leading to vanishing of the node (Fig. 14d),

(b) When either the start or end of the stroke is overlapped with other parts, the start node or the end node disappears after thinning (Fig. 14e),

(4) Analysis error of the ECR (Fig. 14f,g),

(5) Identification error of the d-line (Fig. 14h),

(6) Misclassification between the t-node and the y-node (Fig. 14i),

(7) Unsuccessful global tracing or improper evaluation of the smoothness. The smoothest path may not always be expected to restore the true trace (Fig. 14k).

Some examples for the 7 cases above are shown in Fig. 14. The error distribution over the 7 cases is summarized in Table 4.

Table 4 Distribution of error in terms of its sources

<table>
<thead>
<tr>
<th>Error Type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>10.1%</td>
<td>11.0%</td>
<td>25.6%</td>
<td>14.0%</td>
<td>16.2%</td>
<td>15.5%</td>
<td>7.6%</td>
</tr>
</tbody>
</table>

8 CONCLUSION

This paper presented an approach to address the problem of restoring the writing order from a single-stroked handwriting image based on its graph representation. First, a new concept of Edge Continuity Relation (ECR) was introduced, and a framework to obtain the ECR at an
even-degree node was presented based on exploring its interior structure. Secondly, we proposed to apply maximum weighted matching of general graphs to find double-traced lines. Finally, we introduced a global division and trace algorithm to find the writing path from the start node to the end node.

One of our primary goals throughout this work has been to depend on as little heuristic knowledge as possible. The application of weighted matching of general graphs has enabled us to find all the double-traced lines connected to odd-degree nodes, leading us to solve the odd-degree node problem by transforming it to the one for even-degree nodes. A global optimization has been developed to select the final result by evaluating the total smoothness for each of the candidate paths found.

Our method achieved a restoration rate of 96.0% on the 708,811 static images converted from the online data of the Unipen database. Experiments showed that restored paths achieve recognition rates comparable to online handwriting data.

9 COMMENTS ON FUTURE STUDIES

We are extending our proposed method to restore the writing order of multi-stroked handwriting. One of the ideas toward this direction is to introduce new concepts: *positive stroke* and *negative stroke*. The positive or negative stroke is defined as a pen-tip trajectory during which the pen-tip moves with or without touching the surface of paper, respectively. Conceptually, the negative stroke exists between two consecutive positive strokes. It can be considered that the multi-stroked handwriting image can be converted to a single-stroked one by connecting two consecutive positive strokes with a negative stroke. If all the negative strokes are found properly, the problem of the multi-stroked case can be reduced to that of the single-stroked one, allowing us to apply the proposed method in this paper. Thus the problem is how to find the negative stroke. If we can define properly the weight of the negative stroke that reflects the cost to produce it, negative strokes can be identified together with
double-traced lines by using the maximum weighted matching.

We argue also that for the restoration problem of a multi-stroke image, heuristic knowledge at the higher level, such as a model of character/word, may be necessary to reduce the complexity of searching for the negative stroke. The heuristics vary from language to language, for example, constructing rules for Chinese characters is very different than for English characters. All of these problems remain topics for future study.

ACKNOWLEDGMENTS

The authors would like to thank the associate editor and the reviewers for their constructive and insightful comments. They would also like to thank Dr. Yoshiharu Kato for his comments and discussions on the research.

Reference


[29] MATHPROG, Solver for the Maximum Weight Matching Problem,


