A Method of Generating Feature Graph for Handwritten Character Recognition of Japanese Historical Documents

Masaki Hayashi[1], Shuichi Nishida[1], Mitsuru Nakata[2], Qi-Wei Ge[2] and Makoto Yoshimura[2]
[1]Graduate School of Education, Yamaguchi University, 1677-1 Yoshida, Yamaguchi-shi 753-8513, Japan
[2]Faculty of Education, Yamaguchi University, 1677-1 Yoshida, Yamaguchi-shi 753-8513, Japan
Tel: +81-83-933-5402 Fax: +81-83-933-5304
E-mail : {k020kn, m017kn, mnakata, gqw, y_makoto}@yamaguchi-u.ac.jp

Abstract: In this paper, we propose a method of generating feature graphs for recognition of handwritten characters in Japanese historical documents. The feature graph represents structure of a handwritten character. In our method, the feature graph is generated as follows: (1) a rough graph is generated by thinning, clustering and other processes; (2) connections between vertices are retrieved with edge trace; (3) the curve vertices, which express curved strokes of a character, are added into the graph; (4) relative locations between vertices are calculated.

1. Introduction

Recently, researches on application of information technology in the field of humanities are actively done. And a labor-saving and an automation of identifying the age of historical documents are the one of the important problems.

We aim at the realization of age identifying system for Japanese historical documents (including sentences of only **kana**s, which are Japanese syllabic character, or mixed writing of kanas and **kanji**s) [1].

The procedures of age identification are supposed as follows: (1) to recognize "*obsolete kanas*" in images of a historical document and specify these original kanji characters. Note that a kana is a character made from a kanji, and the original kanji character of a kana is called "*mother character*" in the following discussion; (2) The age of a historical document is identified by comparing appearance frequency of obsolete kana in the document and already-known historical documents. We have proposed the character recognition model which based on graph theory for obsolete kana recognition of procedure (1).

This model uses the graph which expresses the shape of a kana character called **feature graph**. That is, we firstly build a database that preserves feature graphs of a character, which is obtained from a lot of known historical documents. In the database, feature graphs are stored with the information such as a character image and the mother character. Next, we make a feature graph of characters written in a document that is recognition target. And then, we recognize handwritten characters in the target document by comparing them with ones in the database.

About the character recognition for historical documents, there were some researches by using neural networks [2]-[4] under the premise that the characters have been clipped manually from the whole images of the documents. Our model aims at recognizing characters without such premise by applying subgraph identification theory.

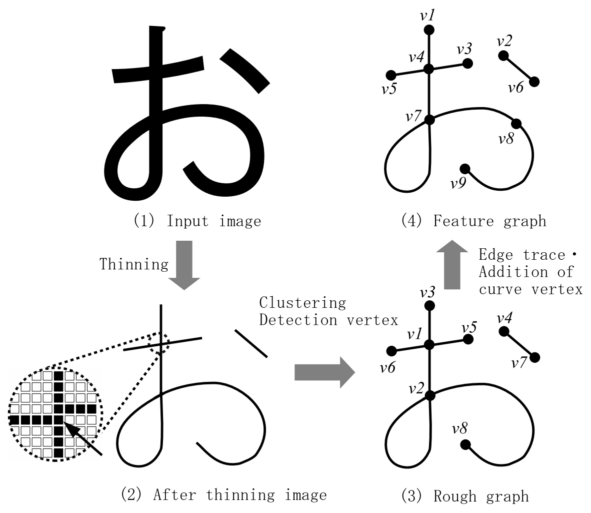


Figure 1. Flow of feature graph making

In this paper, we propose a method of generating the feature graph of one character from a monochrome binary image. In our method, a rough graph is firstly made by the preprocesses, thinning, clustering and other processes. After that, the feature graph is refined by edge trace and addition of the curve vertices that express a curve, and computing relative location between vertices. We will explain the structure and the making method of the feature graph in section 2, and state the way of thinning and clustering in section 3. After that, the process of edge trace and the addition of curve vertices are described in section 4.

2. Structure of feature graph and outline of generating method

A feature graph, which expresses the shape of a character that is a monochrome binary image, is a graph where end points and intersections, etc. of the character image are considered as vertices. Figure 1 shows the flow in making a feature graph. All pixels of Figure 1-(1) are horizontally numbered from the upper left corner of the image toward the lower right corner $(0, 1, \cdots)$. These numbers are called **pixel numbers**. It is similar about the other images (for example, image of Figure 1-(2)). In the following, only a black pixel is called a **point** and it is distinguished from a white pixel. The feature graph shown in Figure 1-(4) consists of the vertex set $V = \{v1, v2, \dots, v9\}$ and the edge set $E = \{(v1, v4), (v2, v6), \dots, (v8, v9)\}$. Here, $v8$ is a vertex expressing the curve from $v7$ to $v9$, and it is called **curve ver-**

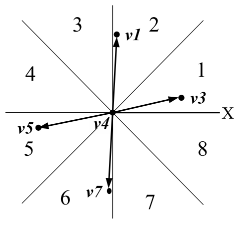


Figure 2. Relative location of adjacent vertices

tex. The making method of feature graph is as following 4 steps.

Step 1 (Thinning): A thick line like Figure 1-(1) is converted into a line with one pixel width (Figure 1-(2)). A circle of the broken line in the figure is a part of enlarged image in the thinned image, and black rectangles show points. Here, each point has eight adjoined points (called **adjacent point**) in maximum. For example, the point indicated by the arrow in the circle has four adjacent points, and "the number of adjacent points" is 4. There are the algorithm of Hilditch [5] and Classical Thinning Algorithm [6] (CTA at the following) as thinning algorithm often used. The former might leave the isolated black pixels to the result of thinning, and such the black pixels obstruct making the feature graph. So CTA is adopted in our method.

Step 2 (Clustering and vertices detection): There might be several points whose number of adjacent points is more than 3 in the narrow area of the thinned image. Because such points complicate a feature graph more than the necessity, these points are integrated into one point (this step is called clustering). After that, points whose number of adjacent points is not 2 are chosen as vertex, and temporary vertex labels are decided (Figure 1-(3)).

Step 3 (Edge trace and Addition of curve vertex): For the image of Figure 1-(3), connections between vertices are examined by tracing the adjacent point from each vertex to compose edges. Furthermore, points on an edge corresponding to a curve and whose number of adjacent points is 2 are chosen as curve vertices (they have a temporary vertex label such as other vertices).

Step 4 (Relabeling and calculating relative location): A temporary vertex labels which are decided in Steps 2 and 3 is relabeled in ascending order of the pixel number (Figure 1-(4)). Next, relative location between vertices at edge both ends is examined. For example, four edges are connected to vertex $v4$ in Figure 1-(4) and these are adjacent to the vertex $v3$, $v1$, $v5$, and $v7$. In this case, a relative locations of $v3$, $v1$, $v5$, and $v7$ to $v4$ are denoted by 1, 2, 5, and 6 as shown in Figure 2 respectively. These numerical values from 1 to 8 show eight areas divided from X-$axis$ ($X \geq 0$) with 45° individually and they show the directions of adjacent vertices ($v3, v1, v5, v7$) from a certain vertex ($v4$). Relative locations between vertex $v4$ and vertices $v3, v1, v5$ and $v7$ are written by $L_{v4} = (v4 : (1, v3), (2, v1), (5, v5), (6, v7))$.

We explain some terms before explaining the details of Steps 2-4 in the following section.

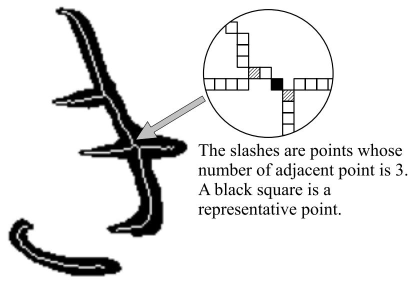

The slashes are points whose number of adjacent point is 3. A black square is a representative point.

Figure 3. The result of thinning process

Eight neighbor points: Every pixel has eight adjacent pixels in the surrounding. These pixels are called 8 neighbor points or 3×3 neighbor points. Moreover, the 16 pixels that is adjacent outside of 3×3's neighbor points are called 5×5 neighbor points, and so on.

Start point, integrated points, representative point: The point that becomes the starting point of the clustering process is called start point. The points integrated by clustering are called integrated points, and the point that remains after clustering is called representative point.

3. Clustering and vertices detection

In Figure 3, the character written in black heavy line is a kana "き" and the white thin line is a result of the thinning. And the circle in the figure is a closeup of the intersection. As shown in the circle, the intersection, which should be expressed by only one point, is expressed by two points (shown as slash rectangles) with 3 adjacent points. Clustering process integrates these points into the single point shown with a black rectangle. The details of clustering process to the image in Figure 4-(1) are shown below. In the figure, a black rectangle shows a point (black pixel) whose number of adjacent points is 3 or more and the alphabet shows the pixel number. Moreover, straight lines show lines which consists of points whose number of adjacent points is 2 or less.

(i) Before clustering process, the clustering area with a central focus on each start point candidate is decided. Here, start point candidates are points whose number of adjacent points is 3 or more in a thinned image, and some of these candidates are chosen actually as the start points of clustering process. Although the width of the clustering area should be equal a stroke width of a target character of recognition, the width of the stroke is unsteady. So, we adopt the value called **threshold** (ew) for a gauge of stroke width. It is an average value of distance to the nearest white pixel from a point in the thinning result in Figure 3. The clustering area is a square $2ew$ on a side such as square A, D and H as in Figure 4-(1). In addition, although thresholds might have a different value in each start point candidate, it is assumed that all thresholds are same here.

(ii) To create "*the start point candidate list*" as Table 1. The table shows other start point candidates included in the clustering area of a start point candidate. Note that the point d isn't included in the list because it has no integrated points.

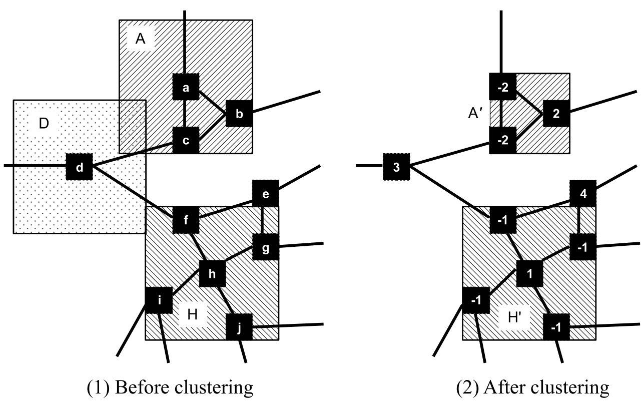


Figure 4. Clustering Process

Table 1. Start point candidate list

pixel number	integrated points	number of integrated points
a	b,c	2
b	a,c	2
c	a,b	2
e	g	1
f	h	1
g	e,h	2
h	f,g,i,j	4
i	h	1
j	h	1

Table 2. A part of connection relationships in Fig.4

pixel number	label	PN	VL
a	-2	$b, c, \cdots$	$3, \cdots$
b	2	$a, c, \cdots$	$3, \cdots$
c	-2	a, b, d	$3, \cdots$
d	3	$c, f, \cdots$	$2, 1, \cdots$

(iii) A representative point is derived in this step. First, the start point, which is a point with the most integrated points (such as h), is selected from Table 1. Then the minimum rectangle (e.g. A', H') that contains all the start point candidates in the clustering area with a central focus on the point is drawn. The diagonal intersection (if the intersection is a white pixel, the nearest black pixel) of the rectangle is adopted as a representative point. In addition, the representative point is labeled positive value $n \geq 1$, and integrated points in the rectangle which has representative point n are labeled negative value n' ($\mid n' \mid = n$).

(iv) First, lines corresponding to the start point and its integrated points (e.g. h, f, g, i, j) are deleted from Table 1, and the integrated points of the start point are deleted from integrated points of other start point candidate (so, the integrated point g of e is deleted). And then, start point candidates (such as e) whose the number of integrated points is 0 are also deleted from the table.

(v) (iii) and (iv) are repeated until the start point candidate list is empty. When the list is empty, points which have no label are labeled in ascending order of pixel number (such as 3 and 4 in Figure 4-(2)). In other words, a vertex is a point which has a positive value label.

4. Edge trace, addition of curve vertex and calculating relative locations

4.1 Edge trace

Edge trace is a process which traces a black pixel from a certain labeled point to the other labeled point and reveals connection relationships between vertices. A connection relationship of certain one point is shown as a pair of "*pixel number list*" (PN) and "*vertex label list*" (VL) of the point. The pixel number list of a point consists of pixel numbers of other points which are connected from the point, and the vertex label list is composed of label of vertices which are connected form the point. For example, pixel number list and vertex label list of point d (with label 3) in Figure 4 are shown as $(c, f, \cdots)$ and $(2, 1, \cdots)$ respectively. When similar pairs (such as shown in Table 2) are obtained for all points with a positive or negative label, connection relationships between all vertices are expressed. Note that some points with a label are omitted in Figure 4 (e.g. the end point in the left line of point d).

4.2 Addition of curve vertex

By thinning, clustering and edge trace processes, the rough graph shown in Figure 1-(3) is generated and connection relationships between vertices become clear. Next, curve vertices which express the curve line like edge $(v2, v8)$ of Figure 1-(3) is added into the graph. Although similar algorithms for obtaining vertices which express a curve edge have already been proposed [7], [8], a few differences of writing might cause a big difference of a graph in these algorithms.

In our method, curve vertices are added into the rough graph according to the shape of consecutive points between two vertices, which have connection mutually (e.g. $(v1, v2)$, $(v2, v8)$ or $(v2, v2)$ in Figure 1-(3)). A series of consecutive points is called **path**.

Concretely, a path is firstly classified into five types: self-loop, straight-line, spiral-curve, S-curve and C-curve. S-curve is a undulating curve such alphabet "S" and C-curve is a curve looks like alphabet "C" such as shown in Figure 5. Then, **convex vertex** , **S dividing vertex** and **spiral dividing vertex** are added into C-curve, S-curve, and spiral-curve respectively. The convex vertex is a kind of curve vertices corresponding to the most convex part on C-curve. S dividing vertex is a kind of curve vertex which divides S-curve into two or more C-curves, and it corresponds to inflection point of a S-curve in many cases. Moreover, the spiral dividing vertex divides into two or more C-curves. A spiral curve can be divided into two or more C-curves by the intersection with the straight line that connects both of end points. Figure 6 shows a flow chart of curve vertices addition to a certain path.

4.3 Relabeling and calculating relative location

All vertices and edges (or connection relationships) in a feature graph are obtained by the processes mentioned above. In following processes (step 4 of section 2), the vertices are relabeled in ascending order of its pixel number and connection

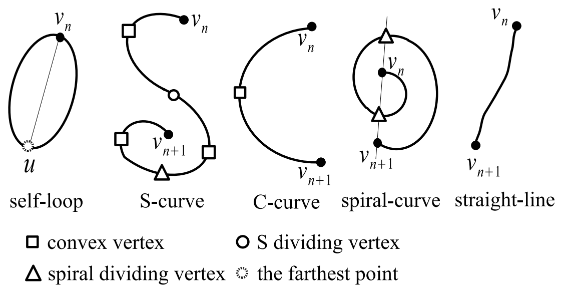
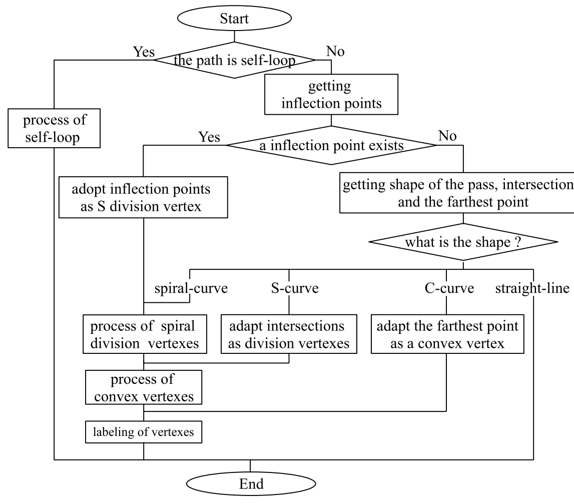


Figure 5. Five kinds of curves



Start
the path is self-loop
Yes
No
process of self-loop
getting inflection points
a inflection point exists
adopt inflection points as S division vertex
getting shape of the pass, intersections and the farthest point
what is the shape ?
spiral-curve
S-curve
C-curve
straight-line
process of spiral division vertexes
adapt intersections as division vertexes
adapt the farthest point as a convex vertex
process of convex vertexes
labeling of vertexes
End

Figure 6. A flow chart of curve vertex addition

relationships shown in Table 2 are updated. Moreover, relative locations of the vertices (such as Figure 2) are derived.

The relative location from vertex v_i to v_j is expressed as $\lfloor 1 + \theta/45 \rfloor$, where θ ($0° \leq \theta < 360°$) is the angle made by the vector $\overrightarrow{v_{ij}}$ and X-$axis$ ($X \geq 0$). However, if $v_i = v_j$ (it means that the path from v_i to v_j is a self-loop), the farthest point from v_i on the path is adopted instead of v_j (the point is shown as u hereafter). Relative locations of v_i are stored into list L_{vi} such as described in section 2.

4.4 An example of a feature graph

We have implemented a Java program to generate feature graphs according to the above-mentioned procedures. Figure 7 shows the feature graph of the Kana "ぬ" which appears in the reference [9] (the white line in the figure is a result of thinning). Connection relationships in the vertices are shown as the list set $\{L_{vi} | i = 1, \cdots, 14\}$= $\{(v1 : (6, v6)), \cdots, (v11 : (2, v7), (21, v11), (21, v11), (8, v13)), \cdots, (v14 : (1, v13))\}$.

The area (A) of the image has been integrated into $v10$ by clustering. And path (B) is expressed as straight line $v5$-$v10$, although the path waves a little. The curve (C) is expressed by two curve vertex ($v4$ and $v7$). The loop of (D) is expressed as a self-loop drawn as a oval of the broken line in the figure. The self-loop is expressed in the list $L_{v11} = (v11 : \cdots, (21, v11), (21, v11), \cdots)$. Here, $21 (= 5 + 16)$ means

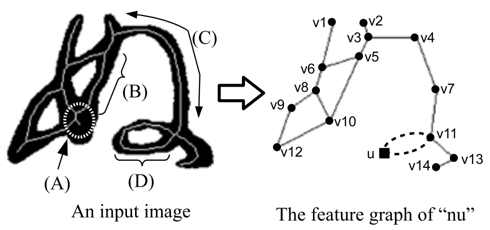


Figure 7. The feature graph of Kana "ぬ"

that relative location between $v11$ and the farthest point u from $v11$. Therefore, it is expressed that the loop has a direction 5 from $v11$. As shown as Figure7, our method can generate a feature graph expresses appropriately the structure of the character.

5. Conclusion

We have proposed a method of generating feature graphs which express structure of handwritten characters. In our method, firstly a rough graph is created by thinning, clustering, edge trace, etc. Then, curve vertices are added into the graph to express curve edges, and relative locations between vertices are obtained. An example feature graph generated by our method has been given to show the effectiveness of our method. As a future problem, we need to do computational experiments for various characters to qualitatively evaluate our method .

References

[1] Ri-Nan Cui, et al., "On Modeling Handwritten Characters for Identifying Ages of Japanese Historical Documents Based on Graph Theory", *IEICE Technical Report*, vol.107, no.203, pp.13-18, 2007.

[2] Shoji Yamada, et al., "Outline of Historical Character Recognition Project", *Journal of Information Processing Society of Japan*, vol.43,no.9, pp.950-955, 2002.

[3] Yuji Izumi, et al., "A Study for Character Recognition of Ancient Documents using Neural Network", *IPSJ SIG Notes*, vol.2000-CH-45, pp.9-15, 2000.

[4] Shinji Hioki, et al., "Recognition of Hand Writing Historical Japanese Characters", *IPSJ SIG Notes*, vol.98-CH-37,pp.35-42, 1998.

[5] C.Judith Hilditch, "Linear Skeletons from Square Cupboards", *Machine Intelligence*, no.4, pp.403-420, 1969.

[6] Theo Pavlidis, "Algorithms for graphics and image processing", *Computer Science Press*, pp.199-201, Rockville, 1981.

[7] C. Lee, B. Wu, "A Chinese-Character-Stroke-Extraction Algorithm Based on Contour Information", *Pattern Recognition*, vol.31, no.6, pp.651-663, 1998.

[8] C. H. Teh, R. T. CHIN, "On the Detection of Dominant Points on Digital Curves", *IEEE Trans. Pattern Anal. and Machine Intell.*, vol.11, issue.8, pp.859-872, 1989.

[9] Kouichi Nakano, *Guidance of obsolete kana*, Musashino Shoin, 1978.