Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2014, Article ID 357380, 11 pages
http://dx.doi.org/10.1155/2014/357380

Research Article

Development of a System to Assist Automatic Translation of Hand-Drawn Maps into Tactile Graphics and Its Usability Evaluation

Jianjun Chen and Noboru Takagi

Department of Intelligent Systems Design Engineering, Toyama Prefectural University, 5180 Kurokawa Imizu, Toyama 939-0398, Japan

Correspondence should be addressed to Jianjun Chen; jianjun0221.happy@163.com

Received 7 April 2014; Accepted 30 June 2014; Published 23 July 2014

Academic Editor: Erich Peter Klement

Tactile graphics are images that use raised surfaces so that a visually impaired person can feel them. Tactile maps are used by blind and partially sighted people when navigating around an environment, and they are also used prior to a visit for orientation purposes. Since the ability to read tactile graphics deeply depends on individuals, providing tactile graphics individually is needed. This implies that producing tactile graphics should be as simple as possible. Based on this background, we are developing a system for automating production of tactile maps from hand-drawn figures. In this paper, we first present a pattern recognition method for hand-drawn maps. The usability of our system is then evaluated by comparing it with the two different methods to produce tactile graphics.

1. Introduction

Producing tactile maps is an important effort to bring blind people to more self-supported life. There have been many studies of computer-aided systems in order to assist the production of tactile graphics [1–6]. Tactile map automated creation system (TMACS) [2, 3], for example, has been developed to produce tactile maps automatically. It is a web application and produces the digital file for a tactile map from the information about two places: departure place and destination. TMACS assumes that users can be blind, and so it produces the tactile map automatically from the map database if a user only provides a departure place and destination to the system. However, tactile maps produced by TMACS are sometimes difficult to read for the blind because it is possible to include unnecessary information in the tactile maps. Further, Geospatial Information Authority of Japan has also developed a tactile map production system [4]. This system assumes that users are sighted people, and it is totally provided as a GUI application. Operating this GUI application is not easy for users who are not familiar with computers.

Based on the background above, we are now developing a system for automating production of tactile maps. In the tactile map production method using our system, a sighted user first draws manually a hand-drawn map using a pencil and paper, and the map is then converted to a digital image using an image scanner or a digital camera. Finally, by using our system, the digital image is recognized and translated into digital files which are available to produce the tactile maps. Our system chooses the Edel [7] and scalable vector graphics (SVG) [8] documents as the output file formats. Here, Edel is a software system to create digital files available to produce tactile graphics by Braille embossers, and this system is used widely in Japanese blind schools. The advantages for our system are as follows.

(1) Sighted users can draw maps by using pencils and papers; no computer operation is needed when drawing maps.

(2) Sighted users can easily add necessary information and remove unnecessary information. This implies that providing tactile maps individually is not time consuming.

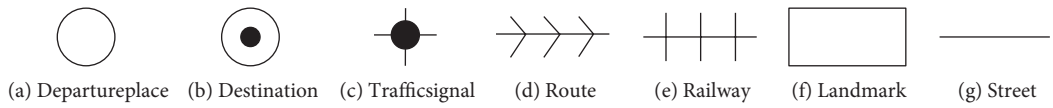


Figure 1: Object types and symbols.

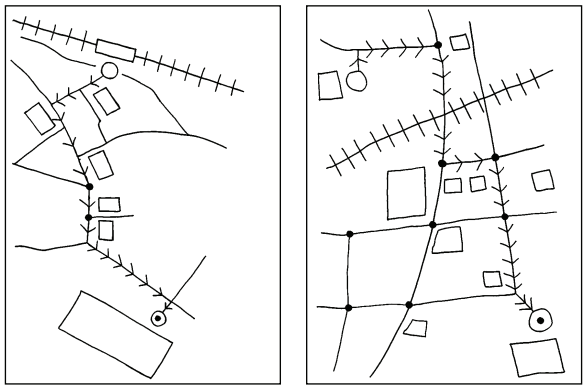


Figure 2: Examples for hand-drawn maps.

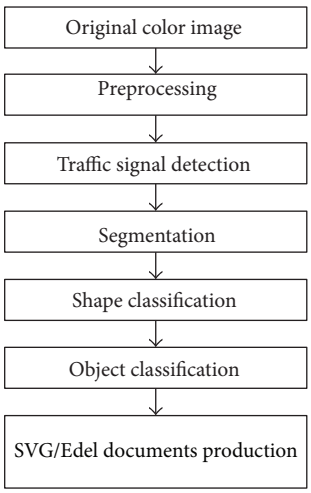


Figure 3: Outline for hand-drawn map translation.

It is important to evaluate the usability for the method using our system. The evaluation was done by comparing with the methods using different software systems.

The remainder of this paper is constructed as follows. Section 2 outlines our method and shows some of the basic procedures. Object classification is explained in Section 3; fuzzy inference is applied to realize the classification methods. The production of SVG and Edel documents is described in Section 4. The classification accuracy for our method is shown in Section 5. In Section 6, we discuss the usability evaluation for our system and conclude the study in Section 7.

2. Outline for Our Method and Basic Procedures

In order to facilitate hand-drawn map recognition, we assume the following conditions for drawing maps.

(1) A hand-drawn map consists of the following object types: departure place, destination, traffic signal, route, railway, landmark, and street. The symbols for objects are summarized in Figure 1.

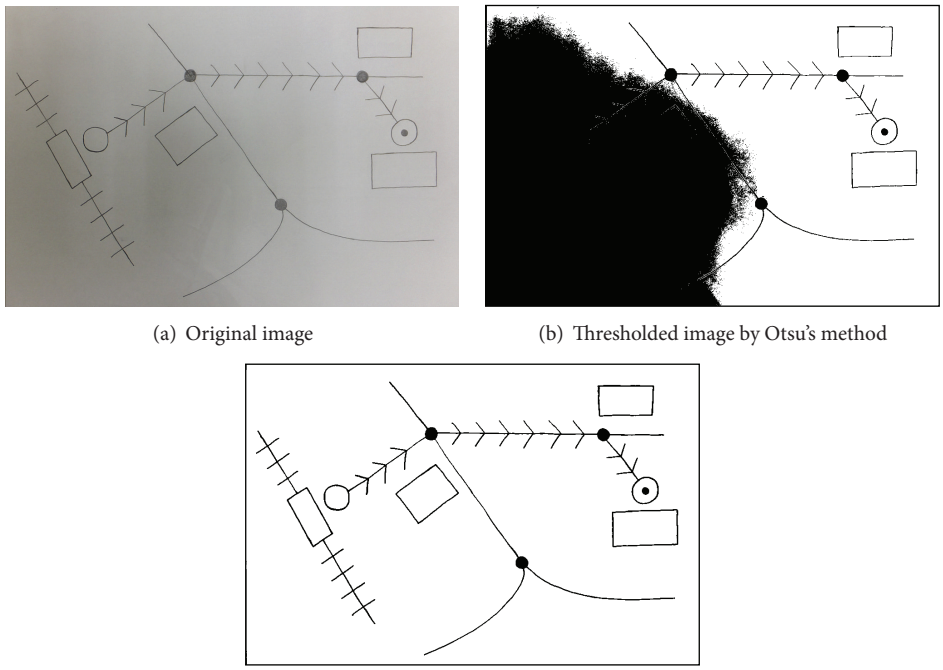
(2) A hand-drawn map is a line drawing, except for traffic signals and the bullet for the destination symbol.

(3) A landmark object is represented by a polygon and forms a connected component. Further, it does not connect to any other objects.

Figure 2 shows examples for hand-drawn maps. Inputting the digital image for a hand-drawn map to our system, it outputs two digital files for tactile maps: the SVG and Edel documents. The procedure for hand-drawn map translation is outlined in Figure 3.

In our tactile map production method, a map is first drawn manually using a pencil and paper. Then, the hand-drawn map is captured by an image scanner or a digital camera. Finally, our system translates the digital image for the hand-drawn map into SVG and Edel documents, producing the tactile maps.

There are various new technologies applied in hand-drawn sketch recognition. For example, Broelemann et al. [9] developed a method for automatic street graph construction of hand-drawn maps, but this method is restricted to detection of streets. The sketch recognition methods introduced



(a) Original image

(b) Thresholded image by Otsu's method

(c) Thresholded top-hat image

Figure 4: Using the top-hat transformation for shading correction, the top-hat transformation corrects the effects of nonuniform illumination, and, as a result, the fine binary image is created using Otsu's method.

in [10–14] cannot be used to recognize the hand-drawn map due to their limitations. So, a new method is proposed to recognize the hand-drawn map in this paper.

In the remainder of this section, preprocessing, traffic signal detection, segmentation, and shape classification are explained.

2.1. Preprocessing and Traffic Signal Detection. The preprocessing consists of the following three procedures.

(1) An original color image is converted to a grayscale image I by the formula

$$I = 0.2989 \cdot R + 0.5866 \cdot G + 0.1145 \cdot B, \quad (1)$$

where R, G, and B are the intensities for red, green, and blue of the original image.

(2) The two noise reduction methods are applied to the grayscale image.

(a) The median filter with 3×3 window is first applied to eliminate salt-and-pepper noise;

(b) a grayscale morphological top-hat transformation [15] is applied to correct the effects of nonuniform illumination. Here, a disk structuring element of radius 3 pixels is used for the top-hat transformation. The effect for a top-hat transformation is explained in Figure 4.

(3) The grayscale image is converted to a binary image using Otsu's method [16].

Mathematical morphology is applied to extract traffic signals from the binary image. First, an erosion operator is performed to the binary image for three times, and a dilation operator is then applied to the eroded image for three times. The structuring element is a disk of radius 3 pixels. The erosion operation eliminates thinner line segments, but it does not remove traffic signals completely. The dilation operation then detects all the traffic signals. By this detection, the bullet for the destination symbol is also extracted.

2.2. Segmentation and Shape Classification. After detecting traffic signals, we segment objects into fragments. The four procedures are introduced to divide objects: (1) thinning, (2) short branch removal, (3) feature point detection, and (4) dividing.

(1) Hilditch's thinning algorithm [17] is first applied to the binary image, and the skeleton image is obtained.

(2) Short branch removal is as follows. Every short branch of the skeleton is removed if its length is shorter than a threshold value.

(3) Feature point detection is as follows. Intersections and corner points are detected in this stage. A 3×3 sliding window is applied to the skeleton image in order to detect intersections. The method for detecting corner points is explained below.

(4) Dividing is as follows. By removing all intersections and corner points in the skeleton, the skeleton is divided into fragments. It is characteristic that every

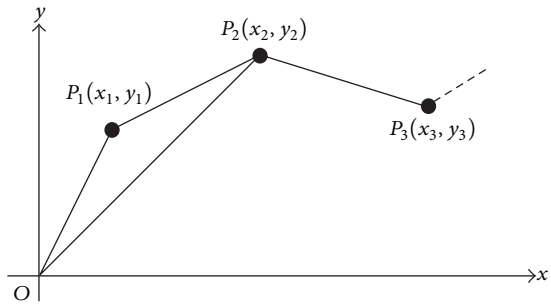


Figure 5: Piecewise linear approximation.

fragment has two endpoints or no endpoint. Fragments are called elements in this paper.

Corner Point Detection. There have been many studies for detecting corner points in digital images [18–20]. We introduce a new method to precisely detect corner points for hand-drawn figures. This method first finds a piecewise linear approximation (PL approximation, for short) for a digital curve; this approximation is detected by Wall and Danielsson's algorithm [21]. Let O, P_1, and P_2 be the origin and the two different points on the xy plane (see Figure 5). If the ratio of the area for the triangle OP_1P_2 to the length of the line segment $\overline{OP_2}$ exceeds a threshold value, then the point P_1 is chosen as a point of the PL approximation; if not, the next point P_2 is examined. Let us consider a digital curve, denoted by a sequence of the points $C = P_0, P_1, \ldots, P_{n-1}$. The Wall and Danielsson's algorithm is then presented below.

Step 1. Set $i \leftarrow 1$, and choose the point P_0 as the first starting point for the PL approximation.

Step 2. Let O be the point P_{i-1}, and set $e_i \leftarrow 0$.

Step 3. Set $i \leftarrow i + 1$, and calculate e_i and ℓ_i. Consider

$$\begin{aligned} e_i &\longleftarrow e_{i-1} + \Delta e_i, \\ \ell_i &\longleftarrow \sqrt{x_i^2 + y_i^2}, \end{aligned} \tag{2}$$

where Δe_i is the value $x_{i-1}y_i - x_iy_{i-1}$ which is the singed area for the parallelogram spanned by the two vectors $\overrightarrow{OP_{i-1}}$ and $\overrightarrow{OP_i}$.

Step 4. If $|e_i| \leq t \cdot \ell_i$ holds, then go to Step 3; if not, choose P_{i-1} as a point of the PL approximation, and then go to Step 2. Here, t is a threshold value.

The above algorithm cannot detect corner points correctly. However, if a corner point exists, it must be close to a point of the PL approximation. So, we improve the Wall and Danielsson's algorithm by exchanging Step 4 with the following Step $4'$ so that the algorithm is able to precisely detect corner points of a hand-drawn digital curve.

So, we introduce two procedures explained below, and our corner detection procedure is the one that is obtained by exchanging Step 4 with the following Step $4'$.

Step $4'$. If $|e_i| \leq t \cdot \ell_i$ holds, then go to Step 3; if not, the point P_{i-1} is detected as a point of the PL approximation, and perform the following process to test whether a corner point exists near the point P_{i-1}.

(1) Conduct the following procedure 1 whose input is P_{i-1}. If this procedure returns a point, choose this point as a corner point of the digital curve C, and then go to Step 2.

(2) Conduct the following procedure 2 whose input is P_{i-1}. If this procedure returns a point, choose this point as a corner point of the digital curve C, and then go to Step 2.

Procedure 1 aims to detect a sharp corner point. This procedure is performed in the following steps.

Step 1. Let P_{i-s} $(s > 1)$ be a point of the digital curve C satisfying the following two conditions:

(1) P_{i-s} has been chosen as a point of the PL approximation;

(2) P_{i-1} is the next PL approximation point to P_{i-s}.

Step 2. For every point P_{i-s+k} of the digital curve C between P_{i-s} and P_{i-1} $(k = 1, 2, \ldots, s-1)$, calculate the distance, denoted by ℓ_k, between P_{i-s} and P_{i-s+k}. Then, let L be the sequence $\ell_1, \ell_2, \ldots, \ell_{s-1}$.

Step 3. For the sequence L, if the distances for L monotonously increase in the middle and the monotonously decrease to the end, that is, the inequalities $\ell_1 < \ell_2 < \cdots < \ell_q > \ell_{q+1} > \cdots > \ell_{s-1}$ hold, then the point P_{i-s+q} is determined as a corner point of the digital curve C.

Next, the procedure 2 is explained; it is motivated by the method for the literature [22]. Remember that the point P_{i-1} has been chosen as a point of the PL approximation.

Step 1. For every P_{i-k} $(k = 1, 2, \ldots, 21)$, calculate the sharpness $s(P_{i-k})$; the procedure to calculate sharpness is explained below.

Step 2. For the sequence of the sharpnesses $s(P_{i-1}), s(P_{i-2}), \ldots, s(P_{i-21})$, calculate the simple moving averages $\mathrm{smv}(P_{i-1}), \ldots, \mathrm{smv}(P_{i-21})$.

Step 3. If the maximum value of $\mathrm{smv}(P_{i-1}), \ldots, \mathrm{smv}(P_{i-21})$ exceeds a threshold value, then the point P_{i-k} which gives the maximum value is determined as a corner point of the digital curve C; if not, we decide that no corner point exists near the point P_{i-1}.

The calculation for the sharpness $s(P)$ for a point P is as follows. We first detect two subsequences of points, denoted by $F = P_{k_1}, \ldots, P_{k_n}$ and $B = P_{t_1}, \ldots, P_{t_m}$, in the following way (see Figure 6). We visit the points forward from P and select

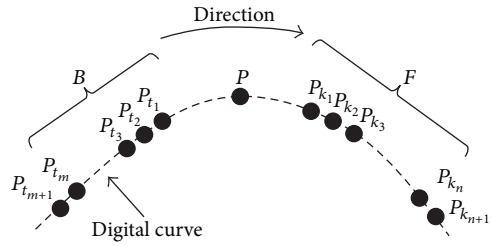


Figure 6: Sharpness.

a point P_{k_j}, the distance between P_{k_j} and P is more than or equal to 2 pixels and less than or equal to 10 pixels; P_{k_1} is the first point satisfying this condition. We select points until the condition breaks; P_{k_n} is the last point satisfying the condition. The subsequence B is obtained similarly, while visiting the points backward from P. The sharpness $s(P)$ is then defined as the following formula:

$$s(P) = \frac{1}{nm} \sum_{r=1}^{n} \sum_{s=1}^{m} \left(\text{the angle of } \angle P_{k_r} P P_{t_s} \right). \tag{3}$$

Next, shape classification is explained. Every element is classified into one of the four shapes: straight line, circle, circular arc, and curve. This classification is done by the method of least squares. That is, we minimize the sum of the squared Euclidean distances between the points of the element and the corresponding points on the model. The element is then classified as the shape for the model if the sum is smaller than a threshold value. A curve is expressed by a piecewise cubic Bézier curve; we will describe cubic Bézier curves in Section 4.

3. Object Classification

In this section, object classification is described. First of all, a landmark object is classified by the following simple procedure: if an object is isolated and closed and has more than 3 corners, the object is classified as a landmark. Except for landmark classification, fuzzy inference is applied to design the classification methods. The fuzzy inference systems introduced in this paper are constructed by Mamdani's fuzzy inference method [23], but the minimum operator is exchanged with the product operator. To detect route and railway objects, it is needed to find arrows and crosses. So, elements are first grouped into a single cluster if they are connected to the same intersection. Every cluster is then examined if it is an arrow or a cross. Note that an element can belong to different two clusters when the two endpoints of the element connect to the different intersections.

3.1. Arrow Classification. An arrow consists of four or three straight line elements (see Figures 7(a) and 7(b)). We apply two fuzzy inference systems to calculate the similarity for arrow. The following description is the outline of the first fuzzy inference system.

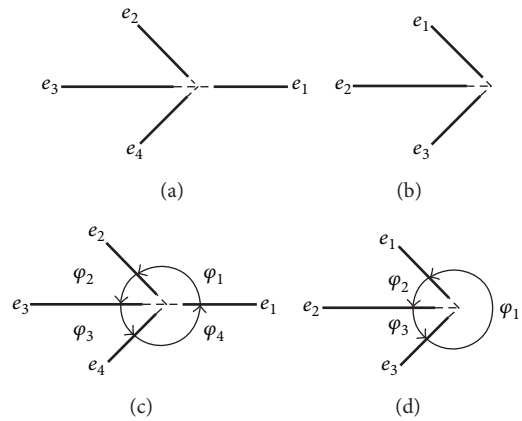


Figure 7: Elements for arrows.

(1) We first choose a cluster, E, which includes four straight line elements, denoted by e_1, e_2, e_3, and e_4. The four angles, φ_1, φ_2, φ_3, and φ_4 are then measured (see Figure 7(c)).

(2) Next, the following three attribute values are calculated:

  (1) the standard deviation of φ_1, φ_2, φ_3, and φ_4; it is denoted by x_1.

  (2) the average value of $|\varphi_1 - \varphi_4|$ and $|\varphi_2 - \varphi_3|$; it is denoted by x_2.

  (3) the difference between the two lengths of the elements e_2 and e_4; it is denoted by x_3.

(3) The fuzzy inference system is applied to x_1, x_2, and x_3. The fuzzy if-then rules are denoted below, and the membership functions are shown in Figure 8. We then have a value, s, of $[0, 1]$ from the system; s implies the similarity for E.

Rule 1. If x_1 is *large*, x_2 is *small*, and x_3 is *small*, then it is *plausible* that E is an arrow.

Rule 2. If x_1 is *small*, then it is *implausible* that E is an arrow.

Rule 3. If x_2 is *large*, then it is *implausible* that E is an arrow.

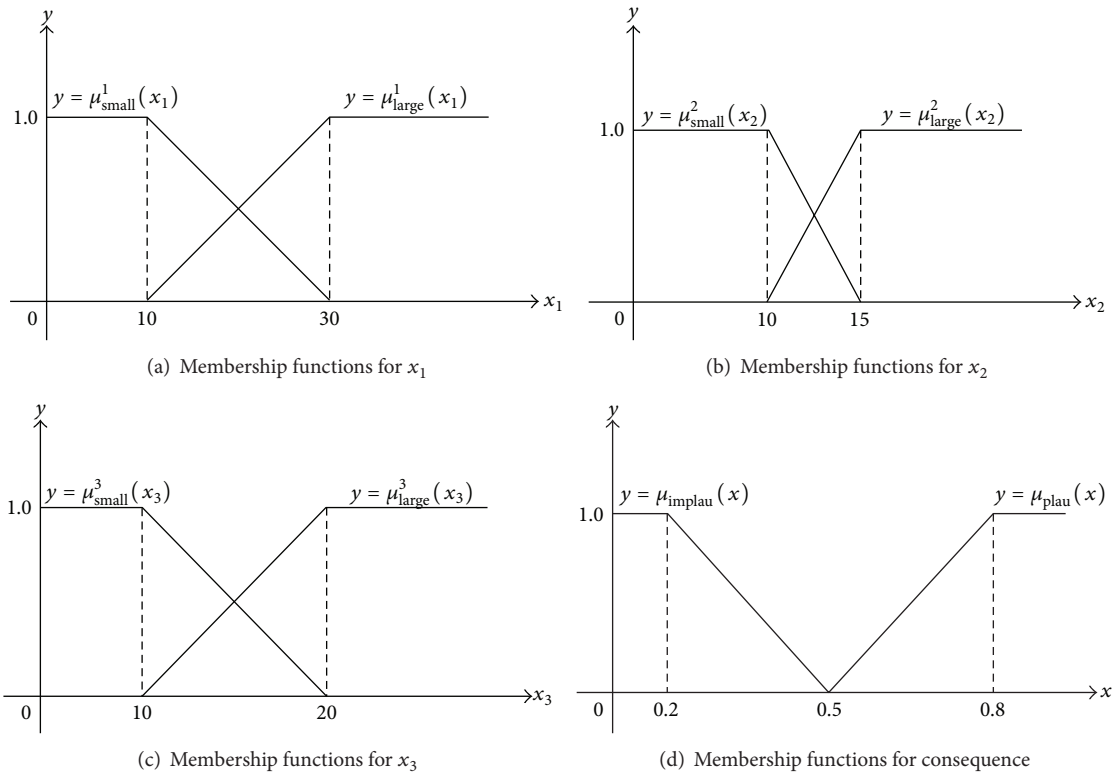
Rule 4. If x_3 is *large*, then it is *implausible* that E is an arrow.

The second fuzzy inference system is applied to a cluster, E, which includes three straight line elements. This system has the following three input attributes:

(1) the standard deviation for φ_1, φ_2, and φ_3 (see Figure 7(d)); it is denoted by x_1;

(2) the difference between φ_2 and φ_3; it is denoted by x_2;

(3) the difference between the two lengths of the elements e_1 and e_3; it is denoted by x_3.



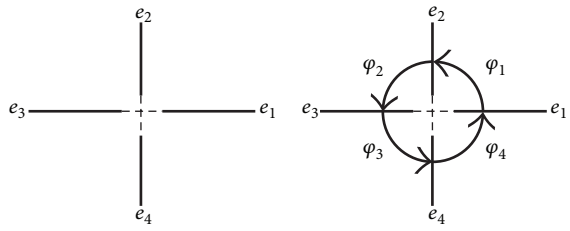
(a) Membership functions for x_1

(b) Membership functions for x_2

(c) Membership functions for x_3

(d) Membership functions for consequence

FIGURE 8: Membership functions: μ^i_{small} and μ^i_{large} ($i = 1, 2, 3$) are membership functions for the fuzzy sets *small* and *large* in Rule j ($j = 1, 2, 3$); μ_{plau} and μ_{implau} are membership functions for the fuzzy sets *plausible* and *implausible* in the consequence.



(a) (b)

FIGURE 9: Elements for cross.

The following rules are the fuzzy if-then rules. The membership functions are omitted.

Rule 1. If x_1 is *large*, x_2 is *small*, and x_3 is *small*, then it is *plausible* that E is an arrow.

Rule 2. If x_1 is *small*, then it is *implausible* that E is an arrow.

Rule 3. If x_2 is *large*, then it is *implausible* that E is an arrow.

Rule 4. If x_3 is *large*, then it is *implausible* that E is an arrow.

3.2. Cross Classification. A cross consists of four straight line elements (see Figure 9(a)). A cluster, E, is applied to the fuzzy inference system for cross classification if E includes four straight line elements, e_1, e_2, e_3, and e_4. This fuzzy inference system requires the following two input attributes:

(1) the standard deviation of φ_1, φ_2, φ_3, and φ_4 (see Figure 9(b)); it is denoted by x_1.

(2) the standard deviation for $\varphi_1 + \varphi_2$, $\varphi_2 + \varphi_3$, $\varphi_3 + \varphi_4$, and $\varphi_1 + \varphi_4$; it is denoted by x_2.

The fuzzy if-then rules are given below, while the membership functions are omitted.

Rule 1. If x_1 is *small* and x_2 is *small*, then it is *plausible* that E is a cross.

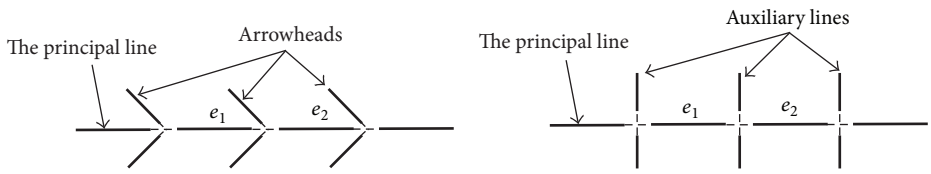
Rule 2. If x_1 is *large*, then it is *implausible* that E is a cross.

Rule 3. If x_2 is *large*, then it is *implausible* that E is a cross.

If the similarity for a cluster obtained by the cross classification is larger than a threshold value, the cluster is classified as a cross. Similarly, if the similarity from the arrow classification exceeds a threshold value, the cluster is classified as an arrow.

3.3. Route and Railway Classification. Route and railway classifications are conducted by the following way.

(1) We first detect a sequence, denoted by Q, of clusters such that every two adjacent clusters include a common straight line element (see Figure 10).



(a) Route with three arrows: the elements e_1 and e_2 are common to the left and the right arrows

(b) Railway with three crosses: the elements e_1 and e_2 are common to the left and the right crosses

Figure 10: Route and railway symbols.

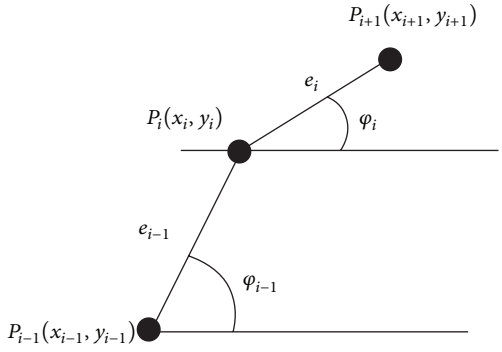


Figure 11: Curvature for digital curve.

(2) If the sequence Q consists of arrows, then the route classification is executed to calculate the similarity, denoted by s_1, for the sequence Q. If s_1 is larger than a threshold value, the sequence Q is classified as a route.

(3) If the sequence Q consists of crosses, then the railway classification is executed and we obtain the similarity, denoted by s_2, for the sequence Q. If s_2 exceeds a threshold value, the sequence Q is classified as a railway.

Fuzzy inference systems are applied to compute the two similarities. To calculate the attribute values for the fuzzy inference systems, we detect the principal line and the arrowhead (or the auxiliary lines) for the sequence Q as follows. If two straight line elements, e_1 and e_2, satisfy the following two geometric characteristics, it is plausible that e_1 and e_2 are part of the same straight line.

(1) The elements e_1 and e_2 are connected at an intersection, P.

(2) The curvature at the intersection is small.

Let C be a digital curve, and let P_i be a point on C. A curvature κ of C at P_i is then defined as the subtraction $|\varphi_i - \varphi_{i-1}|$ (see Figure 11); that is,

$$\kappa = \left|\varphi_i - \varphi_{i-1}\right|, \tag{4}$$

where φ_i is the measure of an angle which is formed between the x-axis and the line segment from P_i to P_{i+1}, and φ_{i-1} is also the measure of an angle which is formed between the x-axis and the line segment from P_{i-1} to P_i [24]. Two adjacent elements e_{i-1} and e_i are merged into a single straight line segment if the curvature between e_{i-1} and e_i is less than a threshold value. After that, we extract the principal line and the arrowheads (or the auxiliary lines) for the sequence Q.

After extracting the principle line and the arrowhead (or the auxiliary lines) for the sequence Q, two fuzzy inference systems are applied to classify the sequence Q as a route or railway. The first fuzzy inference system is for route classification. It has the following two input attributes:

(1) the standard deviation for the lengths of elements in the principal line; it is denoted by x_1;

(2) the standard deviation for lengths of elements in the arrowheads; it is denoted by x_2.

The fuzzy if-then rules for the first fuzzy inference system are denoted below, but the membership functions are omitted.

Rule 1. If x_1 is *small* and x_2 is *small*, then it is *plausible* that Q is a route.

Rule 2. If x_1 is *large*, then it is *implausible* that Q is a route.

Rule 3. If x_2 is *large*, then it is *implausible* that Q is a route.

The second fuzzy inference system is for railway classification, and the description for the second system is omitted because it is similar to the first system.

4. SVG and Edel Documents Production

In our method, a figure consists of elements whose shapes are as follows: straight line, circle, circular arc, and curve. As described in Section 2.2, the shape for an element is determined by the method of least squares. If an element has been classified as a curve, then this element is expressed by a piecewise cubic Bézier curve; the piecewise cubic Bézier curve is detected by an algorithm based on the method of least squares [25].

A Bézier curve $Q(t)$ of degree n is defined as

$$Q(t) = \sum_{i=0}^{n} V_i B_i^n(t) \quad (t \in [0,1]), \tag{5}$$

where the V_is are the control points and the B_i^ns are the Bernstein polynomials:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (i = 0, 1, \ldots, n). \tag{6}$$

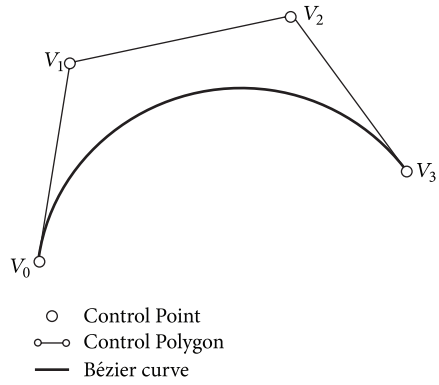


Figure 12: A single cubic Bézier segment.

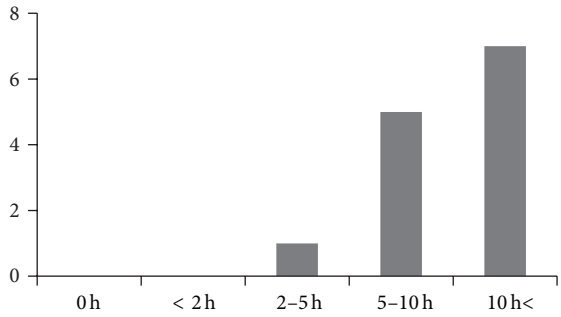


Figure 13: Experience in computers: the vertical axis indicates the number of participants.

Equation (5) is called a cubic Bézier curve when $n = 3$. Figure 12 shows an example of cubic Bézier curves; the four points V_0, V_1, V_2, and V_3 are the control points, the thick curve is the cubic Bézier curve, and the polygon expresses the one constructed by the four control points.

SVG has the formats to express the four shapes, and, therefore, we can create an SVG document for a hand-drawn map directly once we have detected the shapes for all the elements of the map. On the other hand, an Edel document is a collection of embossed dots. So, it is easy to create an Edel document once we have detected the shapes for all the elements of a hand-drawn map.

5. Experimental Results

This section describes the accuracy for our classification system. Five participants drew 15 maps using pencils and papers. These maps were then captured by using an image scanner; the resolution of the scanner was set to 100 dpi. These digital images were saved as 24-bits bitmap images. The sizes of the images are in $1,169 \times 850$ pixels. We have measured the accuracy for our classification system using precision, recall, and f-measure.

In the 15 maps, there are 50 traffic signals, 15 destinations, 15 departure places, and 40 landmarks. The classification accuracy for our system is summarized in Table 1. We can conclude that our system can produce the Edel and SVG documents almost correctly from hand-drawn maps.

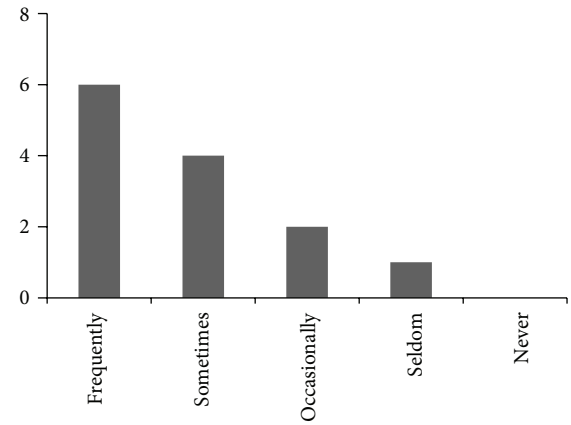


Figure 14: Experience in PPT: the vertical axis indicates the number of participants.

Table 1: Experimental results are as follows: N is the number of all objects for the symbol, T is the number of objects correctly classified as the symbol, C is the number of objects classified as the symbol, P is precision, R is recall, and F is F-measure.

Symbol	N	T	C	P (%)	R (%)	F (%)
Traffic signal	50	50	52	96.2	100	98.1
Destination	15	15	15	100	100	100
Departure place	15	14	14	100	93.3	96.5
Landmark	40	38	38	100	95.0	97.4
Cross	130	125	125	100	96.2	98.1
Arrow	122	121	121	100	99.2	99.6
Route	15	14	16	87.5	93.3	90.3
Railway	28	28	28	100	100	100

6. Usability Evaluation for Our System

6.1. Method. We have studied usability evaluation for our system. Thirteen participants, 12 males and 1 female, aged 20, participated in this investigation; all of them are the third-year university students. To evaluate the usability for our system, we selected two common methods for production of tactile graphics; one is the method to use the software system Edel which assists us to draw diagram images for tactile graphics produced by Braille embossers, and another one is the method using swell papers. In the second method, a diagram image is transferred to a swell paper by a printer and so forth. A swell paper has been coated with thermally foamed microcapsules that respond to irradiation and cause the dark image on the paper to swell, creating the tactile graphic. Microsoft PowerPoint (PPT) was selected to draw a diagram image on a swell paper, because this software system is commonly used in the universities in Japan.

Before starting the experiment, we asked the following questions Q1, Q2, and Q3 to all the participants, and the results for the questions Q1 and Q3 are summarized in Figures 13 and 14. We omit to show the result for the question Q2 because all the participants have no experience in using Edel. All the participants are familiar with using computers and have much experience in using PTT.

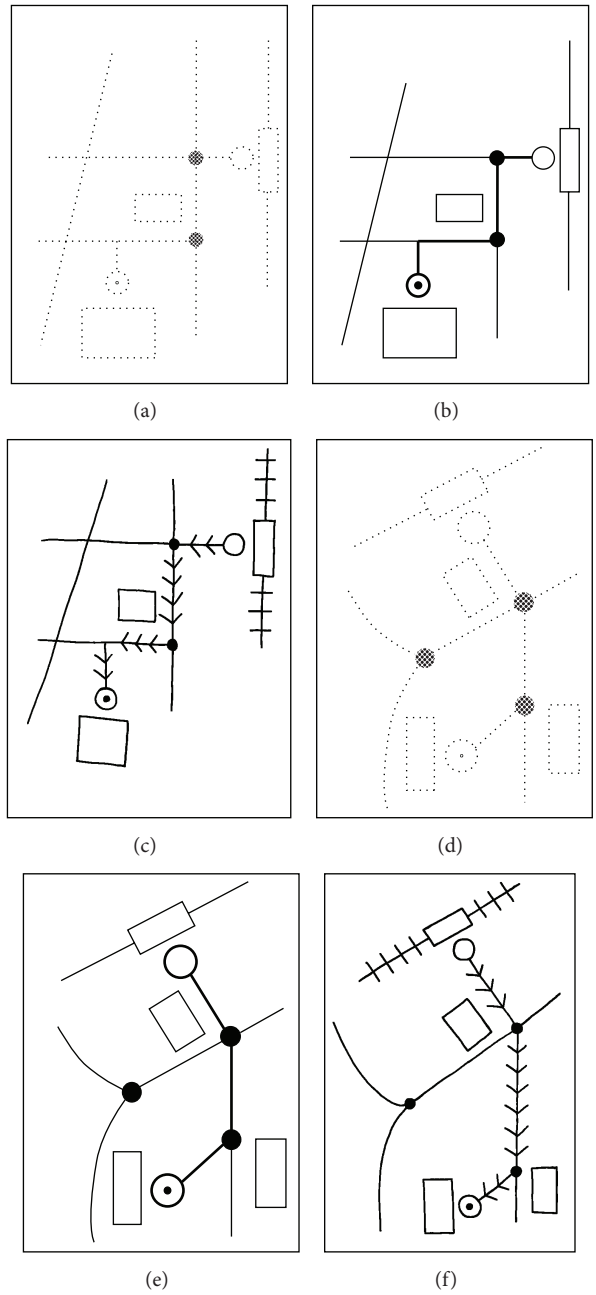


Figure 15: Maps for Sessions 1 and 2: (a) Edel map for Session 1, (b) PPT map for Session 1, (c) hand-drawn map for Session 1, (d) Edel map for Session 2, (e) PPT map for Session 2, and (f) hand-drawn map for Session 2.

Q1 : How long do you use computers every week?

Q2 : Have you ever created figures by using Edel?

Q3 : Have you ever created figures by using PPT?

We had two sessions in this investigation. For the first session, a participant created the map diagram shown in Figures 15(a)–15(c); the digital files were produced by using Edel, PPT, and our system. Note that, in our system, a user first draws a map manually using a pencil and paper, then converts the hand-drawn map to a bitmap image using an image scanner, and finally the bitmap image is transformed into the two digital files, the Edel and SVG documents. After conducting all tasks, the participant was asked to answer the following two questions Q4 and Q5.

Q4: Were you able to easily create the digital file(s) for the tactile map?

Q5: Do you want to use this software system to create the digital file for a tactile map?

After the first session, the participant had 5 minutes for training the software systems by himself/herself. The second session followed this learning session. In the second session,

the participant conducted similar tasks to the first session, but the maps were those shown in Figures 15(d)–15(f). After creating all digital files, the participant was asked to answer two questions Q4 and Q5 again and was also asked to answer the following question Q6.

Q6: Were you able to draw the map as you like?

The orders of the software systems were determined randomly in both the first and the second sessions.

6.2. Results. We applied two-way ANOVA to the answers for the 2 questions Q4 and Q5. The first factor, denoted by Factor 1, is the difference in systems, and the second factor, denoted by Factor 2, is the difference in learning. Factor 1 includes three levels, Edel, PPT, and our system, and Factor 2 includes two levels, before learning and after learning. The results from the two-way ANOVA for Q4 shows that there was no significant interactions between Factor 1 and Factor 2 ($F(2, 72) = 0.92$, $P = 0.40$). Furthermore, there was a significant difference for Factor 1 ($F(2, 72) = 19.26$, $P < 0.01$), while no significant difference existed for Factor 2 ($F(1, 72) = 1.58$, $P = 0.21$). The results for Q5 are similar to those for Q4. That is, there was no significant interaction between Factor 1 and Factor 2 ($F(2, 72) = 0.64$, $P = 0.53$), there was a significant difference for Factor 1 ($F(2, 72) = 14.00$, $P < 0.01$), and there was no significant difference for Factor 2 ($F(1, 72) = 1.32$, $P = 0.25$).

We then conducted a multiple comparison test for the answers of the questions Q4 and Q5; we selected Tukey's honestly significant difference (HSD) criterion as the multiple comparison test. For Q4, there was a significant difference between Edel and our system ($P < 0.01$) and there was also a significant difference between PPT and our system ($P < 0.01$); however, there was no significant difference between Edel and PPT (see Figure 16). Furthermore, for Q5, there was a significant difference between Edel and our system ($P < 0.01$), and there was also a significant difference between PPT and our system ($P < 0.05$); however, we observed a marginally significant difference between Edel and PPT ($P < 0.1$) (see Figure 17).

Lastly, we applied a one-way ANOVA to the answers for the question Q6; the factor of this one-way ANOVA is the difference between systems. The results are shown in Figure 18, and the results for the one-way ANOVA show that there was a marginally significant difference ($F(2, 36) = 2.59$, $P = 0.089$). We then applied Tukey's HSD, and we observed a marginally significant difference between Edel and our system.

As a result of the discussion above, we can conclude that users feel that our method is easier to produce tactile maps than the conventional two methods. Furthermore, map images produced by our system are visually as fine as map images produced by Edel and PPT.

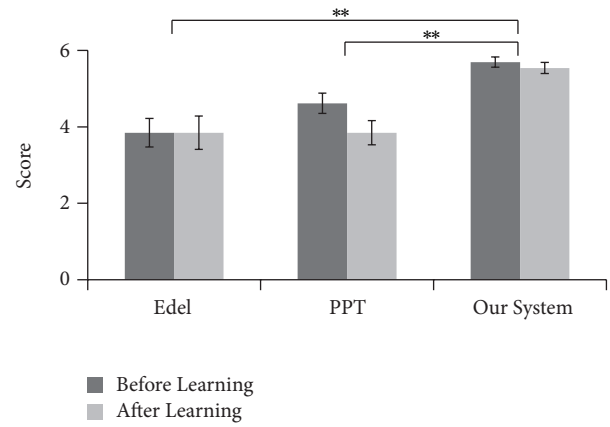


Figure 16: Results for Q4: the average and standard error for each system ($^{**}P < 0.01$).

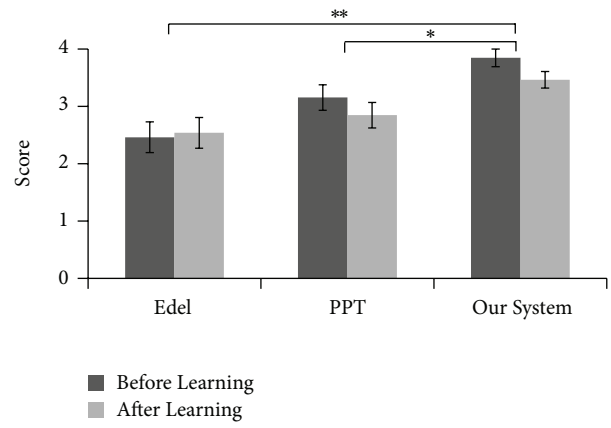


Figure 17: Results for Q5: the average and standard error for each system ($^{**}P < 0.01$, $^{*}P < 0.05$).

7. Conclusions

In this paper, a computer-aided system for automating production of tactile graphics from hand-drawn maps has been developed. The system includes the following four main procedures: (1) preprocessing, (2) segmentation, (3) pattern recognition based on fuzzy inference, and (4) SVG and Edel documents production. The preprocessing is applied to an initial hand-drawn map in order to remove noise, and then a binary image is obtained. In the segmentation procedure, the binary image is first skeletonized, and then the skeleton is segmented into elements by eliminating intersections and corner points. Then, a pattern recognition method is applied to recognize symbols. Lastly, SVG and Edel documents are created to save the recognition results. The usability for our system was evaluated by comparing our method with the two conventional systems for creating tactile maps. The results show that our system is easier to create tactile maps than the other two systems.

In our fuzzy inference systems, triangular and trapezoidal membership functions are applied to define the fuzzy sets due to their simplicity. The shapes of membership functions influence the accuracy of recognition results. How to choose an optimal membership function is our future work. In this study, we do not assume a map includes character strings. However, adding character strings to a map is very important

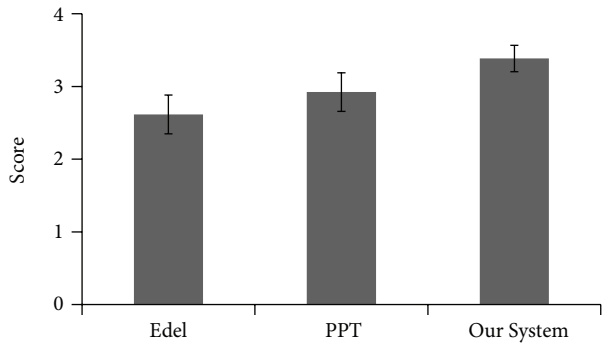


Figure 18: Results for Q6: the average and standard error for each system.

because it increases the comprehension for the map. In the present work, the types of symbols used in hand-drawn maps are restricted. Developing a system for automating translation of hand-drawn maps with character string and various symbols is one of our future works.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the JSPS Grant-in-Aid for Scientific Research (C) no. 24501198. The authors thank Dr. Keisuke Ido for giving much advice to the experiments of usability evaluation.

References

[1] J. A. Miele, S. Landau, and D. Gilden, "Talking TMAP: automated generation of audio-tactile maps using Smith-Kettlewell's TMAP software," *The British Journal of Visual Impairment*, vol. 24, no. 2, pp. 93–100, 2006.

[2] T. Watanabe, T. Yamaguchi, K. Watanabe et al., "Development and evaluation of a tactile map automated creation system accessible to blind persons," *IEICE Transactions on Information and Systems*, vol. J94-D, no. 10, pp. 1652–1663, 2011 (Japanese).

[3] TMACS, http://tmacs.eng.niigata-u.ac.jp/tmacs-dev.

[4] http://tenpuchizu.gsi.go.jp/shokuchizu/.

[5] S. E. Krufka and K. E. Barner, "Automatic production of tactile graphics from scalable vector graphics," in *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '05)*, pp. 166–172, October 2005.

[6] R. E. Ladner, M. Y. Ivory, R. Rao et al., "Automating tactile graphics translation," in *Proceedings of the 7th International ACM SIGACCESS Conference on Computer and Accessibility*, pp. 150–157, New York, NY, USA, October 2005.

[7] Edel, http://www7a.biglobe.ne.jp/ EDEL-plus.

[8] J. D. Eisenberg, *SVG Essentials*, O'REILLY, 2002.

[9] K. Broelemann, X. Jiang, and A. Schwering, "Automatic street graph construction in sketch maps," in *Graph-Based Representations in Pattern Recognition*, X. Jiang, M. Ferrer, and A. Torsello, Eds., vol. 6658 of *Lecture Notes in Computer Science*, pp. 275–284, Springer, Berlin, Germany, 2011.

[10] L. B. Kara and T. F. Stahovich, "An image-based, trainable symbol recognizer for hand-drawn sketches," *Computers & Graphics*, vol. 29, no. 4, pp. 501–517, 2005.

[11] B. Edwards and V. Chandran, "Machine recognition of hand-drawn circuit diagrams," in *Proceedings of the IEEE Interntional Conference on Acoustics, Speech, and Signal Processing*, vol. 6, pp. 3618–3621, June 2000.

[12] P. Sala, "A recognition system for symbols of electronic components in hand-written circuit diagrams," CSC 2515-Machine Learning Project Report, 2004.

[13] Y. Qiao and M. Yasuhara, "Recovering dynamic information from static handwritten images," in *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition*, pp. 118–123, jpn, October 2004.

[14] M. Notowidigdo and R. C. Miller, "Off-line sketch interpretation," in *Proceedings of the AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural*, pp. 120–126, Arlington, Va, USA, April 2004.

[15] R. J. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Education, Upper Saddle River, NJ, USA, 3rd edition, 2007.

[16] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[17] C. J. Hilditch, "Linear skeletons from square cupboards," in *Machine Intelligence, Vol. IV*, B. Meltzer and D. Michie, Eds., pp. 403–420, Elsevier, New York, NY, USA, 1969.

[18] S. Hermann and R. Klette, "Global curvature estimation for corner detection," Communication and Information Technology Research Technical Report 171, 2005.

[19] N. Nain, V. Laxmi, B. Bhadviya, and A. Gopal, "Corner detection using difference chain code as curvature," in *Proceedings of the 3rd IEEE International Conference on Signal Image Technologies and Internet Based Systems (SITIS '07)*, pp. 821–825, Shanghai, China, December 2007.

[20] B. Kerautret, J. Lachaud, and B. Naegel, "Curvature based corner detector for discrete, noisy and multi-scale contours," *International Journal of Shape Modeling*, vol. 14, no. 2, pp. 127–145, 2008.

[21] K. Wall and P. Danielsson, "A fast sequential method for polygonal approximation of digitized curves," *Computer Vision, Graphics, & Image Processing*, vol. 28, no. 2, pp. 220–227, 1984.

[22] D. Chetverikov, "A simple and efficient algorithm for detection of high curvature points in planar curves," in *Computer Analysis of Images and Patterns*, vol. 2756 of *Lecture Notes in Computer Science*, pp. 746–753, Springer, Berlin, Germany, 2003.

[23] T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems and Its Applications*, Academic Press, Boston, Mass, USA, 2nd edition, 1992.

[24] N. Ono and R. Takiyama, "On calculations of curvature of sampled curves," Technical Report of IEICE IE93-74, 1993 (Japanese).

[25] P. J. Schneider, "An algorithm for automatically fitting digitized curves," in *Graphics Gems*, A. S. Glassner, Ed., pp. 612–625, Academic Press, 1990.