On Consideration of A Pattern Recognition Method for Mathematical Graphs with Broken Lines

Noboru Takagi
Toyama Prefectural University
Kosugi, Toyama, Japan
takagi@pu-toyama.ac.jp

Abstract

A computer-aided system for transformation of mathematical figures into tactile graphics is useful for visually impaired students when they learn mathematics and science. To develop such a system, research on mathematical figure recognition techniques is needed. There are so many mathematical figures in which graphs are drawn using broken lines. Under this situation, this paper discusses a method of extracting and classifying broken lines from a mathematical figure.

1 Introduction

Graphs are frequently used to present functions and equations in mathematics and science. Since most of these graphs are in visual form, they cannot be utilized by visually impaired users. Through tactile graphics, pictures can be understood by the visually impaired. Although tactile graphics are useful for visually impaired students when they learn mathematics and science, in 80% of Japanese schools for the visually impaired, there is no department to produce tactile teaching material [10]. Teachers produce most tactile graphics using less intelligent computer-aided systems. So, a better computer-aided system of making tactile graphics is needed.

R. E. Ladner[12] and his research group studied a system that enables us to automate tactile graphics translation. However, in this system, Photoshop, for example, is assumed to be applied for manipulating pure graphics. So, graph recognition techniques will be a key technology to develop more intelligent computer-aided system of making tactile graphics. This paper focuses on mathematical graphs and discusses a method of recognizing mathematical graphs. This is because structure of mathematical graphs is more logical then that of the other graphics. This fact enables us to develop a mathematical graph recognition technique. There is some research on graph recognition. Aso et al. [9] studied a graph recognition method, and in their method, graphs must satisfy many assumptions. For example, a graph has to be drawn inside a rectangular area that is specified by the x-axis and the y-axis. The graph recognition methods introduced in [3, 6, 11] must also satisfy assumptions about graphs. But, many mathematical figures do not satisfy all of the assumptions. Therefore, to develop a computer-aided system for transformation of mathematical figures into tactile graphics, it is necessary to study mathematical figure recognition techniques.

We are developing such a computer-aided system [14], which will transform mathematical figures into tactile graphics. This paper describes some of the mathematical figure recognition techniques which were introduced for developing our system; techniques that are related to classification of broken lines.

Mathematical figures that this paper focuses on have the following characteristics and Fig.1 shows some of such examples.

1. Character strings and mathematical formulae may be distributed in and around the graph.
2. A character string or a mathematical formula my not lie on the correct orientation (i.e., on the horizontal orientation).
3. Graphs can be drawn by using several types of broken lines.

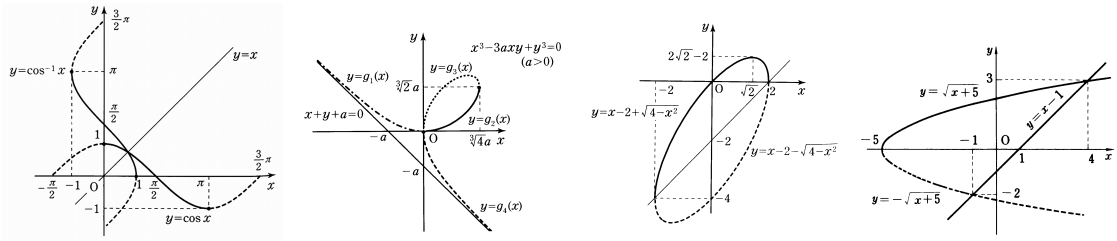


Figure 1: Examples of Mathematical Figures

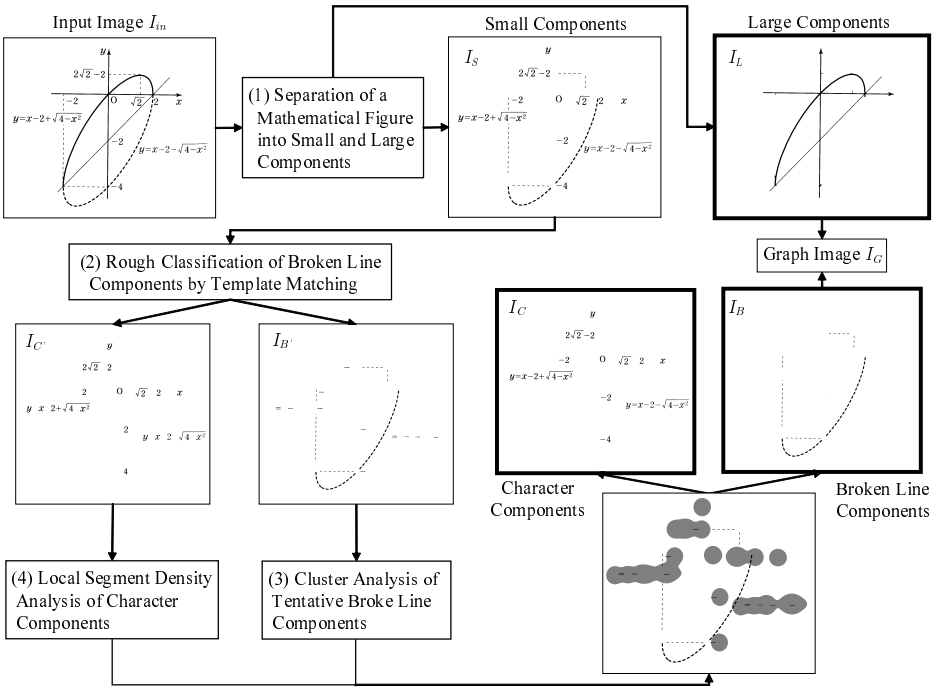


Figure 2: Outline of Separation

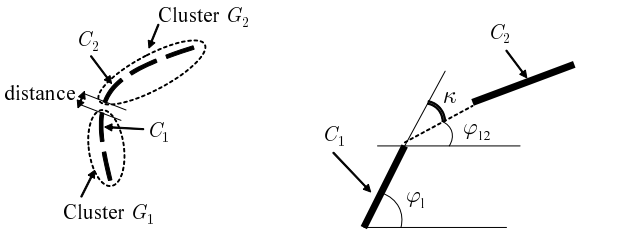
2 Outline of Separation of Mathematical Figures into Character and Graph Regions

In our method [14], all of the connected components in a mathematical figure are first divided into three groups; large components, character components, and broken line components. The outline of this method is shown in Fig.2.

We first apply the preprocessing to an input image I_{in}; a binarization, a noise reduction, and a labeling process. After the labeling process, we have all the connected components of I_{in}, that is, $C_1, \ldots, C_t$. We then classify large components as graph components, and separate them from small components. In the following, let I_L be an image consisting of large components, while I_S an image of small components. Since broken line components are small, they are classified as small components. Given an image I_S, the following procedure is the outline for dividing small components into broken line and character components.

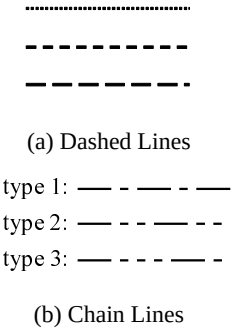
Input: An Image I_S

Output: Images I_B and I_C



(a) Distance between Two Clusters (b) Curvature at Connected Point

Figure 3: Distance and Curvature of Two Clusters



(a) Dashed Lines

(b) Chain Lines

Figure 4: Broken Lines

Step 1: Classify every component of I_S as a rectangular or a non-rectangular component by a template matching. Then, let $I_{B'}$ and $I_{C'}$ be images of rectangular and non-rectangular components.

Step 2: Let B' be the set of components in $I_{B'}$. Then, apply a hierarchical clustering to B' in order for each cluster to consist of components of the same broken line.

Step 3: Calculate the local segment densities of $I_{C'}$, and then roughly divide $I_{C'}$ into some character areas.

Step 4: For each cluster G, if α% or more of G is covered by a character area, then classify all components of G as character components.

Step 5: Remove every component, which was classified as a character component, from $I_{B'}$, and let the remaining image be I_B. Let I_C be the image given by combing $I_{C'}$ and the character components. Output I_B and I_C.

The following is the procedure of the hierarchical clustering in Step 2.

Input: A set of components $\{C_1,\ldots,C_t\}$ of I_B

Output: A set of clusters Γ

Step 1: Calculate the distance between every pair of components C_i and C_j $(i \neq j)$. Let D be the set of all distances, and set $G_i \leftarrow \{C_i\}$ for every $i = 1,\ldots,t$, and $\Gamma \leftarrow \{G_1,\ldots,G_t\}$.

Step 2: Select the minimum distance from D, say $\hat{d}$. If $\hat{d} \geq \theta_1$, then output Γ and stop the procedure, otherwise set $D \leftarrow D - \{\hat{d}\}$ and go to the next step.

Step 3: Let C_i and C_j be the two components which give the distance $\hat{d}$. If C_i and C_j are members of two different clusters, say G_p and G_q, then go to the next step, otherwise go to Step 2.

Step 4: Compute the curvature κ between the two clusters G_p and G_q. If $\kappa \leq \theta_2$, then set $\Gamma \leftarrow (\Gamma - \{G_p, G_q\}) \cup \{G_p \cup G_q\}$. Go to Step 2.

Fig.3 illustrates how the distance and the curvature between two clusters G_1 and G_2 are obtained. This clustering was designed for extracting clusters so that each cluster might be formed of the same broken line components. However, since this clustering is not sophisticated enough, the accuracy of its result is not acceptable.

3 Broken Line Classification

Fig. 4 shows examples of broken lines which this paper focuses on. Each of them is called a dashed line, a chain line of type 1, 2, and 3, respectively. In this section, we describe methods that classify a cluster of the previous section into a dashed line or a chain line.

3.1 Dashed Line and Chain Line Classifications

A dashed line consists of one kind of components, while a chain line is composed by two kinds of components. Therefore, components of a dashed line are grouped into one homogeneous set. Similarly, components of a chain line are grouped into two homogeneous sets. If we could evaluate the optimal number of groups for a cluster of the previous section, then it makes us enable to classify the cluster into a dashed line or a chain line. To evaluate the optimal number of homogeneous groups (i.e., clusters), we measure it by two cluster validities, $v_{DB}(\cdot)$ and $v_D(\cdot)$ [7], whose definitions are given below. Given a set of clusters $\Gamma = \{G_1, \ldots, G_k\}$,

$$v_{DB}(\Gamma, k) = \left(\frac{1}{k}\right) \sum_{i=1}^{k} \left[\max_{j(j \neq i)} \left\{ \frac{\alpha_i + \alpha_j}{\|\overline{v}_i - \overline{v}_j\|} \right\} \right]$$

where for $i = 1, \ldots, k$,

$$\overline{v}_i = \sum_{x \in G_i} \frac{x}{|G_i|} \text{ and } \alpha_i = \sum_{x \in G_i} \frac{\|x - \overline{v}_i\|}{|G_i|}$$

$$v_D(\Gamma, k) = \min_{1 \leq i \leq k} \left[\min_{1 \leq j \leq k, j \neq i} \left\{ \frac{\hat{\delta}(G_i, G_j)}{\max_{1 \leq t \leq k} \{\Delta(G_t)\}} \right\} \right]$$

where for any clusters S and T,

$$\Delta(S) = \max_{x,y \in S} \{\delta(x,y)\} \text{ and } \hat{\delta}(S,T) = \min_{x \in S, y \in T} \{\delta(x,y)\}.$$

Here, $\delta(x,y)$ is the distance between x and y.

These two cluster validities do not work correctly when $k = 1$. So, we introduce fuzzy inference to avoid this drawback. Note that in the following, a component is expressed by the two characteristics; the number of pixels and the length of its long-side.

Dashed Line Classification

Input: A cluster G of broken line components

Output: If G is a dashed line, then Yes, otherwise No.

Step 1: For every $k = 2, 3, 4, 5$, apply k-means clustering method to G, and let Γ_k be the result when the number of clusters was k. Calculate $v_{DB}(\Gamma_k, k)$ and $v_D(\Gamma_k, k)$.

Step 2: Based on the values $v_{DB}(\Gamma_k, k)$ and $v_D(\Gamma_k, k)$, apply the fuzzy inference. If G was classified as a dashed line, then return Yes, otherwise return No.

The fuzzy inference in Step 2 is the max-product-centroid fuzzy inference, and its fuzzy rules are shown below. Because of the lack of the space, the definition of membership functions is omitted.

Rule 1: If $v_{DB}(\Gamma_2,2)$ is *Large*, $|\min\{v_D(\Gamma_k,k) : k = 3,4,5)\} - v_D(\Gamma_2,2)|$ is *Large*, and (N^+/N^-) is *Large*, then G is a dashed line.

Rule 2: If $v_{DB}(\Gamma_2,2)$ is *Small*, $|\min\{v_D(\Gamma_k,k) : k = 3,4,5\} - |v_D(\Gamma_2,2)$ is *Small*, and (N^+/N^-) is *Small*, then G is not a dashed line.

In the description above, $N^+ = \max\{n(C_1),\ldots,n(C_t)\}$ and $N^- = \min\{n(C_1),\ldots,n(C_t)\}$, where $n(C_i)$ is the number of pixels of a component $C_i \in G = \{C_1,\ldots,C_t\}$.

We discuss chain line classification, which distinguishes a cluster of chain line components. If a cluster was classified as a chine line, then the chain line classification also gives its type. The following is the procedure.

Chain Line Classification

Input: A cluster G of broken line components

Output: The type of G, if G was classified as a chain line, otherwise No.

Step 1: Apply k-means clustering method to G by setting $k = 2$, and then divide G into two groups. Assign the label a to elements of one group, and also assign the label b to elements of the other group. Then, a sequence of labels corresponding to G is obtained.

Step 2: Calculate the similarity $S_p(G)$ $(p = 1,2,3)$ between the sequence and the template of a type p chain line. Here, the template of type 1 chain line is ababab$\cdots$. Similarly, those of types 2 and 3 are abbabbabb$\cdots$ and abbbabbb$\cdots$, respectively.

Step 3: If the similarity $S_p(G)$ was equal to the number of elements of G, then classify G as a type p chain line and return p, otherwise return No.

For the sequence given by Step 1 and the template of a type p chain line, the similarity $S_p(G)$ is defined as the number of continuously corresponding labels. Lastly, the procedure of the broken line classification is described below.

Broken Line Classification

Input: A set of clusters $\Gamma = \{G_1,\ldots,G_t\}$

Output: A set of dashed line clusters Δ, sets of type p chain line clusters X_p $(p = 1,2,3)$, and a set of clusters Φ in which each of them contains only one element.

Step 1: Set $\Delta \leftarrow \emptyset$, $X_p \leftarrow \emptyset$ $(p = 1,2,3)$, and $\Phi \leftarrow \emptyset$.

Step 2: If Γ is empty, then output Δ, X_p $(p = 1,2,3)$, and Φ, and stop the procedure.

Step 3: Select G_i from Γ, and set $\Gamma \leftarrow \Gamma - \{G_i\}$. If G_i includes only one component, then set $\Phi \leftarrow \Phi \cup \{G\}$, and go to Step 2.

Step 4: Apply the dashed line classification to G_i. If G_i was classified as a dashed line, then set $\Delta \leftarrow \Delta \cup \{G_i\}$ and go to Step 2.

Step 5: Apply the chain line classification to G_i. If G_i was classified as a type p chain line, then set $X_p \leftarrow X_p \cup \{G_i\}$ and go to Step 2.

Step 6: Divide G into three groups G_ℓ, G_p, and G_r in the following way. Suppose $S_p(G)$ is the largest among the 3 similarities $S_1(G)$, $S_2(G)$, and $S_3(G)$. G_p includes all the elements that gave the similarity $S_p(G)$, G_ℓ and G_r include elements whose locations are the left-side and the right-side of G_p, respectively.

Step7: G_p is classified as a type p chain line, and set $X_p \leftarrow X_p \cup \{G_p\}$. Also set $\Gamma \leftarrow \Gamma \cup \{G_\ell, G_r\}$. Go to Step 2.

3.2 Merging Clusters

Since the broken line classification divides a cluster into several groups until every cluster is classified into one of the 4 types of broken lines. Therefore, we need a merging process that merges clusters of the same broken line into a single cluster. Our merging process consists of two parts;

1. merging based on fuzzy inference, and
2. merging based on the spline interpolation.

The following is an outline of our merging process.

Input: A set of clusters Γ before merging

Output: A set of clusters Γ after merging

Step 1: Select a pair of clusters (G_i, G_j) which is not tested yet. If there exists no such pair, then output Γ and stop the procedure.

Step 2: If the types of G_i and G_j are different, then go to Step 1.

Step 3: Apply (G_i, G_j) to the merging process based on fuzzy inference. If this pair was merged into a single cluster G, then go to Step 5.

Step 4: Apply (G_i, G_j) to the merging process based on the spline interpolation. If this pair was not merged into a single cluster G, then go to Step 1.

Step5: Update Γ by setting $\Gamma \leftarrow (\Gamma - \{G_i, G_j\}) \cup \{G\}$, and go to Step 1.

If two clusters G_i and G_j had the characteristics,

1. the number of pixels of every component in G_i and G_j is close to each other,
2. the length of the long-side of every component in G_i and G_j is also close to each other, and
3. the distance between the two broken lines corresponding to G_i and G_j is short,

then it is plausible that these two clusters G_i and G_j are part of the same broken line. The merging process based on this idea is realized by fuzzy inference. The characteristics that are applied to fuzzy inference are the following.

Average number of pixels: When a cluster G corresponds to a dashed line, we have one average ($a_1(G)$), while in the case where a cluster corresponds to a chain line, we have two averages; one ($a_{11}(G)$) is for short segments, and the other one ($a_{12}(G)$) is for long segments.

Average length of long-sides: Similarly, we have one average ($a_2(G)$) when a cluster G corresponds to a dashed line, but there are two averages ($a_{21}(G)$ and $a_{22}(G)$) for the cluster corresponding to a chain line.

Distance: This is the shortest distance ($a_3(G_i, G_j)$) between the two broken lines corresponding to the two clusters G_i and G_j.

Curvature: This is the curvature ($a_4(G_i, G_j)$) at the connected point of the two broken lines corresponding to the two clusters G_i and G_j.

In the case where both of two clusters G_i, G_j are dashed lines, the following fuzzy rules are applied.

Rule 1: $|a_1(G_i) - a_1(G_j)|$ is *Small*, $|a_2(G_i) - a_2(G_j)|$ is *Small*, $a_3(G_i, G_j)$ is *Short*, and $a_4(G_i, G_j)$ is *Small*, then these two clusters are merged into one.

Rule 2: $|a_1(G_i) - a_1(G_j)|$ is *Large*, $|a_2(G_i) - a_2(G_j)|$ is *Large*, $a_3(G_i, G_j)$ is *Long*, and $a_4(G_i, G_j)$ is *Large*, then these two clusters are not merged into one.

Next, in the case where both of two clusters G_i, and G_j are chain lines, the following fuzzy rules are applied.

Rule 1: $|a_{11}(G_i) - a_{11}(G_j)|$ is *Small*, $|a_{12}(G_i) - a_{12}(G_j)|$ is *Small*, $|a_{21}(G_i) - a_{21}(G_j)|$ is *Small*, $|a_{22}(G_i) - a_{22}(G_j)|$ is *Small*, $a_3(G_i, G_j)$ is *Short*, and $a_4(G_i, G_j)$ is *Small*, then these two clusters are merged into one.

Rule 2: $|a_{11}(G_i) - a_{11}(G_j)|$ is *Large*, $|a_{12}(G_i) - a_{12}(G_j)|$ is *Large*, $|a_{21}(G_i) - a_{21}(G_j)|$ is *Large*, $|a_{22}(G_i) - a_{22}(G_j)|$ is *Large*, $a_3(G_i, G_j)$ is *Long*, and $a_4(G_i, G_j)$ is *Long*, then these two clusters are not merged into one.

Broken line classification processes the rectangular components of an image I_B. Non-rectangular broken line components are classified as character components, and any broken line component which connects to the x-axis, for example, will be classified as a large component. It means that broken line components which were classified as character components or large components may exist between two clusters. Based on this idea, we introduce merging based on the spline interpolation below. First, we calculate a cubic spline function between the two broken lines corresponding to clusters G_i and G_j. Then, the gap between these two broken lines is interpolated by this spline function. If 50% or more of this interpolated route was occupied by character components and large components, then these two clusters G_i and G_j are merged into a single cluster.

4 Experimental Results

We selected 19 mathematical figures with broken lines from mathematics and science text books. These printed figures were scanned by an image scanner, whose resolution was set at 600 dpi. The electric images were saved in a bitmap format. The font size of characters in the figures is almost 8pt; it was checked by visual observation. The size of bitmap images is in almost $1,500 \times 1,500$ pixels. The total number of broken lines that are included in the 19 figures is 90. Fig.5 shows examples of the mathematical figures.

The 19 images were first applied to the separation method from Section 2. Fig.6 shows the images of character components which were separated from graphic components. Then, 138 clusters of broken line components were extracted by the hierarchical clustering in Step 2 of the separation method. After that these 138 clusters were inputted to the broken line classification from Section 3, we then had 147 dotted lines and chain lines.

The correct numbers: dotted lines (106), chain lines (18), others (23)

The classification results: dotted lines (100), chain lines (14), others (33)

No incorrect classification existed when a cluster was classified as a dotted line or a chain line. Since 6 dotted line clusters and 4 chain line clusters included only one or two components, these clusters were not classified as a broken line.

Next, we applied the merging process from Section 3.2 to the 124 broken line clusters (i.e., the 106 dotted line clusters and the 18 chain line clusters). Then, these 124 broken line clusters were merged into 97 broken line clusters. 70 clusters out of the 97 clusters form completed broken lines, and each of the remaining clusters forms a part of a broken line.

We often measure the effectiveness of a system by the precision and the recall. The precision is defined as the ratio of the number of broken lines that were correctly classified to the number of classified broken lines. The recall is defined as the ratio of the number of broken lines that were correctly classified to the number of correct broken lines. The precision (p) and the recall (r) of our method is therefore $p = 70/97 = 0.72$ and $r = 70/90 = 0.78$.

Fig.7 shows the results of the merging process. We did not obtain the correct merging results for the two broken lines in the graph of Fig.5-(4). The dotted line clusters (b) and (c) in Fig.7-(4) must be merged into a single cluster, and also the dotted line clusters (d) and (f) must be merged into one. The reason why they were not merged is that spline function could not find the correct route between the two clusters.

5 Conclusion

In this paper, we discussed pattern classification of mathematical figures. Especially, we proposed a method for extracting and classifying broken lines drawn in mathematical figures. Our method creates good classification results, however, it includes some drawbacks. So, these drawbacks will be improved in the near future.

References

[1] C. J. Hilditch, “Line Skeletons from Square Cupboards”, in *Machine Intelligence IV*, B. Meltzer and D. Michie, Eds., University Press, Edinburgh, pp.403-420, 1969.

[2] K. Wall and Per-Erik Danielson, “A Fast Sequential Method for Polygonal approximation of Digitized Curves”, CVGIP, vol.28, pp.220-227, 1984

[3] S. Shimada, S. Kakumoto and M. Ejiri, “A Recognition Algorithm of Dashed and Chained Lines for Automatic Inputting of Drawings”, *IEICE Transactions on Information and Systems*, Vol.J69-D, No.5, 1986, pp.759-770.

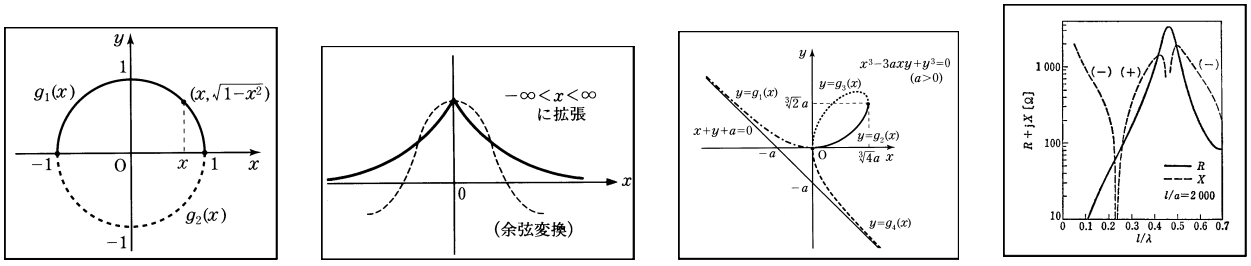
[4] N. Ono and R. Takiyama, “On Calculations of Curvature of Sampled Curves”, *IEICE Technical Report*, Vol.IE93-74, 1993, pp.7-14.

[5] O. Shiku, A. Nakamura and H. Kuroda, “A Method for Character String Extraction Using Local and Global Segment Densities”, *Journal of Information Processing*, Vol.40, No.8, 1999, pp.3230-3238.

[6] N. Yokokura and T. Watanabe, “Recognition of Business Bar-graphs Using Layout Structure Knowledge”, *Journal of Information Processing*, Vol.40, No.7, 1999, pp.2954-2966.

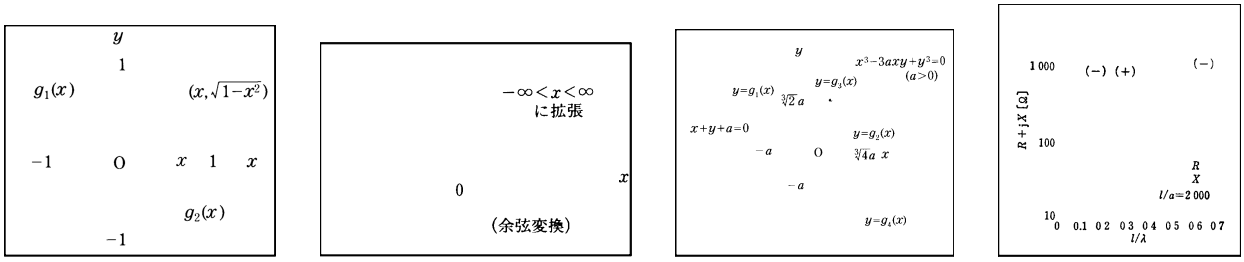
[7] J. C. Bezdek *et. al*, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers, 1999.

[8] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, 2nd Edition, A Wiley-Interscience Publication, 2001.



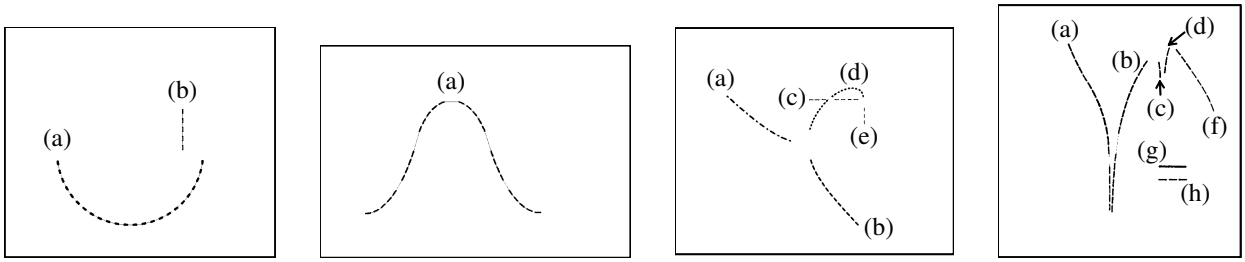
(1) (2) (3) (4)

Figure 5: Example of Figures



(1) (2) (3) (4)

Figure 6: Character Components



(1) (2) (3) (4)

Figure 7: Broken Lines

[9] T. Fuda, S. Omachi and H. Aso, “Recognition of Line Graph Images in Documents by Tracing Connected Components”, *IEICE Transactions on Information and Systems*, Vol.J86-D-II, No.6, 2003, pp.825-835.

[10] S. Oouchi, M. Sawada, T. Kaneko and K. Chida, “A Survey on Making and Using Tactile Educational Materials in Schools for the Blind”, *Bulletin of National Institute of Special Needs Education*, Vol. 31, 2004, pp.113-125.

[11] M. Yo, T. Kawahara and R. Fukuda, “Handwriting Graphics Input System for High School Mathematics”, *IEICE Technical Report*, PRMU2003-291, 2004, pp.43-48.

[12] R. E. Ladner et al, “Automating Tactile Graphics Translation”, *Proc. of the 7th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, pp.150-157, 2005.

[13] N. Takagi “Mathematical Figure Recognition for Automating Production of Tactile Graphics”, *Proc. of Int. Conf. on Systems, Man, and Cybernetics*, pp.4796-4801, 2009.

[14] N. Takagi, “Pattern Recognition in Computer-Aided Systems for Transformation of Mathematical Figures into Tactile Graphics”, *International Journal of Intelligent Computing in Medical Sciences and Image Processing* (to be published)