Systems and Computers in Japan, Vol. 38, No. 14, 2007
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J86-D-II, No. 6, June 2003, pp. 825–835

Recognition of Line Graph Images in Documents by Tracing Connected Components

Toshimichi Fuda,* Shinichiro Omachi, and Hirotomo Aso

Graduate School of Engineering, Tohoku University, Sendai, 980-8579 Japan

SUMMARY

Up to the present there has been a great deal of research related to recognition and comprehension of document images. A large amount of this research focused on text although a significant amount of information in graph images within document images is also included. If it is possible to recognize and comprehend a graph image, electronic documents can be used more efficiently. Research on recognition methods focusing on graph images within document images dealt with solid line graph images (bar graphs and markers). This paper proposes a recognition method of graphs focusing on line graph images which do not include markers comprised by solid lines, dotted lines, broken lines, and dash-dot lines. This paper also shows the effectiveness of this through experimentation. © 2007 Wiley Periodicals, Inc. Syst Comp Jpn, 38(14): 103–114, 2007; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.10615

Key words: graph recognition; document recognition; OCR; connected components.

1. Introduction

Up to the present there has been a great deal of research related to recognition and comprehension of document images and many results have been obtained [1]. A large amount of this research targeted text although a significant amount of information in graph images within document images is also included. If it is possible to recognize a graph image as individual data and not only as an image, electronic documents can be reutilized more efficiently thereby making it possible to freely change and output the layout and style depending on the different objectives of users.

For this reason, research is being performed with the objective of recognizing graphs within document images. Yokokura and Watanabe proposed a method focusing on bar graphs that recognizes bar graphs with various different expression formats based on structure rules between graph elements [2]. Lee and colleagues recognized line graph images by focusing on solid line graph images having markers and using template marking to recognize a marker that forms a line and then compare the location of this marker to a label axis [3]. There are many instances, however, of a marker not being displayed on the line graph drawn on paper or a line graph other than a solid line. These graphs are outside the intended recognition method of Lee and colleagues.

This paper proposes a method to recognize line graphs without markers. The proposed method initially divides a graph image into regions (regions outside the axes) such as titles, axis name, and scale and regions where lines exist (regions outside the axes). Information such as scale is obtained from the regions outside the axes and coordinate values of each of the line graphs are obtained from the regions outside the axes. Finally, these are integrated to convert the graph information into numerical information.

*Now affiliated with NTT Data, Inc.

In order to make it easy to distinguish lines in a graph without any indicated markers, the lines are displayed using different patterns such as solid lines, dotted lines, and dash-dot lines. Because of this, line type patterns of lines must be distinguished in order to recognize these. Research into methods which recognize various types of lines within images include a method [4] that focuses on broken lines and chain lines linking line segments comprising these lines and a method [5] that focuses on dotted lines and utilizes the fact that the dots repeat at a uniform period to extract collections of dots. Although this is the case, line graph images are usually a mixture of solid lines, dotted lines, broken lines, and dash-dot lines. Besides this, line graph images also have complex intersections. Because this method uniformly handles these various different types of lines, we propose a method that expresses line type patterns as a characteristic of the number of black pixels of the connected components and the distance between the connected components. In addition, this method traces lines using the features of line graphs and then recognizes the lines corresponding to each line type pattern. In the following, Section 2 describes the graph image recognition method proposed in this paper and Section 3 describes the recognition method of lines by tracing connected components. The proposed methods are evaluated in Section 4, and Section 5 is the conclusion.

2. Proposed Method

This section initially describes a graph to be recognized by the proposed method and then describes a summary of the proposed method. Table 1 and Fig. 1 show definitions of terms used in this paper.

2.1. Graph to be recognized

The line graph image to be recognized in this paper satisfies the following conditions. This image is not limited to lines. Line graphs comprised by curved lines can also be recognized. For sake of clarity, all line graph images will be described here. We conclude with clipping of graph images from document images and gradient corrections.

(1) Marker is not shown.

(2) Graph axes surrounded by rectangular frame that contains x-axes and y-axes.

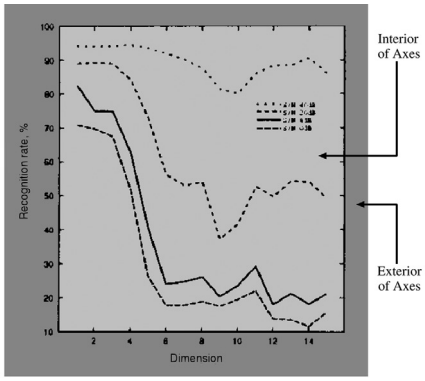
(3) Lines comprising the graph are nonreturnable in the direction of z-axes.

(4) Legend region collected in one area.

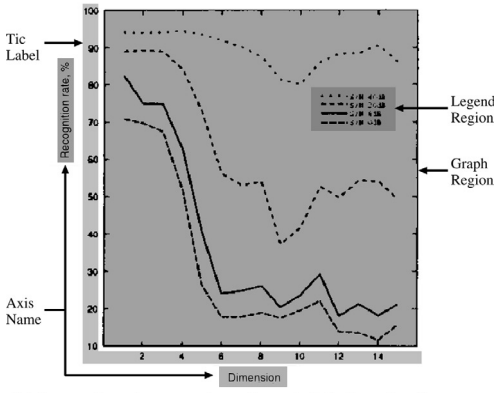
(5) Line type patterns are solid lines, dotted lines, broken lines, and dash-dot lines. Can also be multiple types of dotted line patterns, broken line patterns, and dash-dot line patterns.

Table 1. Technical terms used in this paper

Term	Definition
Region inside the axes	Region surrounded by rectangular frame that contains x-axes and y-axes. Region that contains graph region and legend region.
Region outside the axes	Region that excludes region inside the axes from entire graph image. Title, axis name, and scale of graph exist in this region.
Legend region	Partial region where legend exists in region inside the axes.
Graph region	Region that excludes legend region from region inside the axes. Only connected region where lines are formed exist.
Line axes pattern	Patterns of lines where lines are formed. Examples include solid lines, dotted lines, broken lines, and dash-dot lines.
Scale amount	Alphanumeric that represents the scale of a graph.
Scale line	Line used as a standard for coordinates indicating scale amount.



(a) Interior of axes and exterior of axes



(b) Legend region, graph region, tic label, and axis name

Fig. 1. Definition of regions and technical terms.

(6) Line type pattern and line have a one-to-one correspondence.

(7) Scale lines have equal intervals (logarithmic scale outside range).

2.2. Outline of proposed method

Figure 2 shows the flow of the line graph recognition method proposed in this paper. Numerical data of each point of the line graph is obtained for the final objective of the proposed method. Thereupon, the input image must be divided into a graph region and a character region. A division process of three regions is included in Fig. 2. These are processes to divide all input images into graph regions and character regions. Each process will be described below.

2.2.1. Preprocess

In a preprocess, images which were input are thinned beforehand [6] and then the axes detected. The axes are represented by solid lines and black pixels are connected lengthwise in the horizontal or vertical direction. Because of this, the preprocess detects the axes by scanning the entire image in the vertical direction or the horizontal direction and finding the number of accumulations of black pixels.

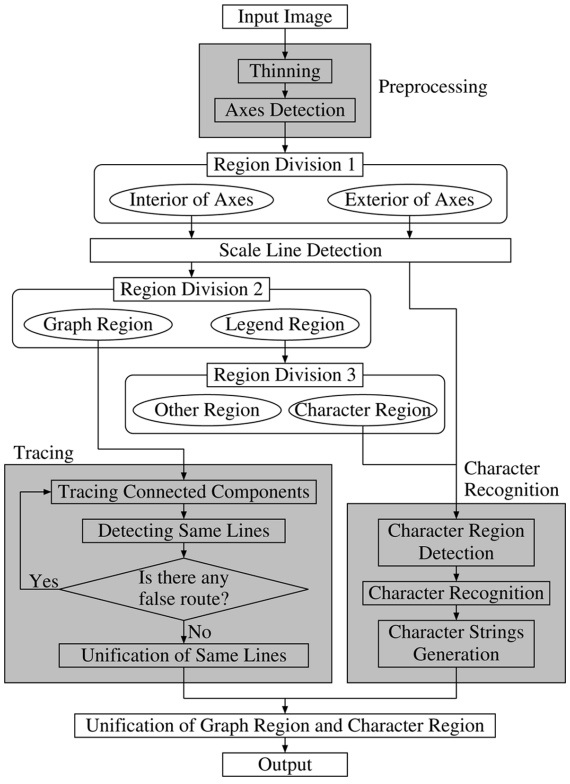


Fig. 2. Flowchart of the proposed method.

2.2.2. Dividing regions

A region division process executes three times to divide an image into a graph region and a character region.

Region division 1: Divides the input image into two regions: a region inside the axes and a region outside the axes. Divides this into three regions of axes, inside the axes, and outside the axes based on the axes detected by the axis detection process described in Section 2.2.1.

Region division 2: Divides the region inside the axes after the scale line detection process (described above) into two regions of a graph region and a legend region. If we take the number of accumulations of the number of connected components in the direction of the y-axis in the region inside the axes when a legend region exists in the region inside the axes, the number of connected component accumulations in the x-coordinates where the legend region exists will become larger compared to other x-coordinates. We extract the legend region from the region inside the axes using this characteristic.

Region division 3: Divides a legend region into two regions of a character region and another region. A character region for showing line names and a line pattern corresponding to this exists in the legend region. Extracting the character region divides these two regions.

2.2.3. Detecting scale lines

At first, we extract connected components (scale line candidates) from the region inside the axes. The connected components (scale line candidates) make contact with the axes and are comprised by a comparatively small number of pixels. Thereafter, we detect scale lines from among the detected scale line candidates by detecting connected components lined up in equal intervals from among the scale line candidates.

2.2.4. Tracing process

In a tracing process lines are extracted by tracing connected components against the graph region extracted by Region Division 2 and then coordinates which correspond to the lines are output. This process will be described in detail in Section 3.

2.2.5. Processing for character region

This processing cuts a connected component per character from a character region divided by Region Division 1 to 3 and then executes a character recognition process for each. Thereafter, the process considers the connection of the characters and generates a character string. The

processing for a character region utilizes an already existing method.

2.2.6. Integrating graph region and character region

Final output data is produced by integrating line information of graph regions obtained by tracing and recognition results obtained by character recognition processing for character regions. In other words, output data is produced by means of using scale amount (recognition results of a character region) to reread position information of a graph region.

3. Recognizing Lines Based on Connected Components

In order to recognize line graphs without markers, several types of lines must be recognized using line type patterns existing inside a graph region and then these are distinguished. It is not possible, however, to know the line type patterns existing inside a graph region in advance. The first step in this method is to detect the lines represented by solid lines and then detect the lines outside the solid lines by means of tracing the connected components. Each process will be described below.

3.1. Detecting solid lines

Detecting lines comprised by solid lines is done by tracing thinned solid lines. The connected components represented by solid lines are connected components sufficiently longer than other connected components inside a graph region. A sufficiently large amount of black pixels comprise these connected components. Thereupon, from among the connected components inside the graph region, the connected components, with a number of pixels greater than n_{max}, are extracted as connected components represented by solid lines. The value of n_{max} is determined by prior experimentation.

3.2. Tracing connected components

Next, we will show an extraction method of line type patterns comprised by connected components with comparatively small dotted lines and dash-dot lines. This method traces connected components and extracts line type patterns from the characteristics of the connected components on the trace route. In the following section we will describe two processes. One process normalizes the tracing distance and the other process divides the tracing direction. These processes are introduced in order to trace each of the connected components on a precise trace route.

3.2.1. Normalizing tracing distance

This is used as a standard to trace the distance between connected components when tracing connected components. Assuming this standard is simply the Euclidean distance between connected components, even when the connected components belonging to different line type patterns are in close proximity, there is a chance that tracing toward these types of connected components in close proximity will be incorrect. Thereupon, a tracing standard must be provided so as to avoid incorrect trace routes.

The distance between connected components comprising either line type patterns of dotted lines or dash-dot lines has a certain fixed value. If this certain fixed value can be extracted, the range in which all connected components to be traced can be judged. Therefore, the connected components are traced on a precise trace route by means of detecting the distance of all traces from the connected components of the origin trace and then executing a trace based on that distance. The method to detect the distance of all traces is as follows.

Step 1: Extract all connected components located within distance r_{max} from connected component i of the origin trace. r_{max} is the maximum distance that can be traced when tracing connected components, is determined by prior experimentation, and presented in advance.

Step 2: Calculate minimum distances r_{minj} toward other connected components inside graph region for each connected component j extracted in step 1.

Step 3: Detect greatest number of minimum distances (frequency of appearance in a certain allowable range occurs often) from among all minimum distances r_{minj} calculated in step 2. This is the distance of all traces r_std_i from connected components i.

Apply the processes in steps 1 to 3 above to all connected components i inside the graph regions and detect traces r_std_i which should be traced from each of the connected components. Thereafter, normalize distance $r_{i,j}$ between connected components using r_std_i. The equation below is the standard when tracing this normalized distance $r_{\_newi,j}$ between connected components.

$$r\_new_{i,j} = \frac{r_{i,j}}{r\_std_i} \tag{1}$$

3.2.2. Dividing tracing direction

The direction of tracing from connected components is subdivided into four categories. Direction 1 is a region where connected components exist which are traced from the connected components of the origin trace toward the upper direction. In the same manner, direction 2 is a region where connected components exist which are traced from the connected components of the origin trace toward the left direction (x-direction reducing direction), direction 3 is a

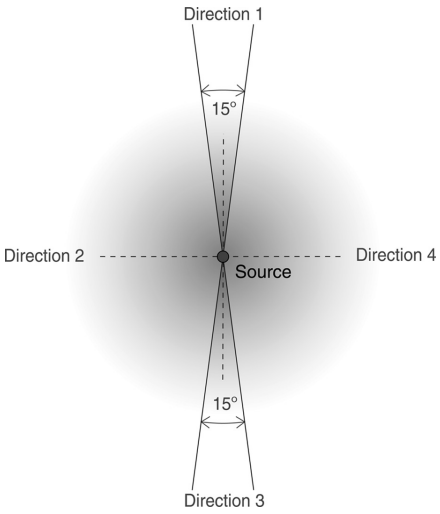


Fig. 3. Directions for tracing connected components.

region where connected components exist which are traced from the connected components of the origin trace toward the lower direction, and direction 4 is a region where connected components exist which are traced from the connected components of the origin trace toward the right direction (x-direction increasing direction). The regions in the upper and lower directions (direction 1, direction 3) are indicated by region of 15°. This is a region that functions to absorb quantization errors of input images.

3.2.3. Fusion of tracing distance and tracing direction

The normalized tracing distance and the divided tracing direction are fused and divided into subregions to select a trace route. Figure 4 shows an outline of this subregion. When all conditions shown below are satisfied for the surrounding connected components seen from the connected components of the origin trace located at the center of Fig. 4, those connected components will be the trace target and the trace will execute from the origin trace toward the trace target.

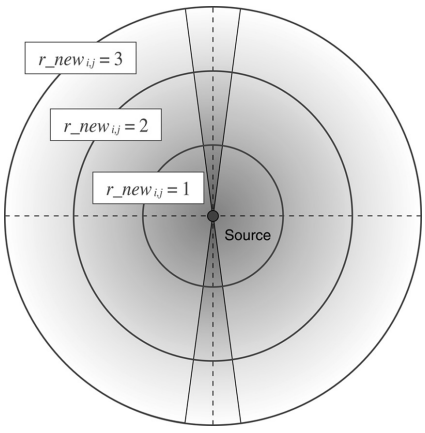


Fig. 4. Fusion of distance and directions for tracing.

Condition 1: The connected components existing in the region where the normalized tracing distance $r\_new_{i,j}$ is the smallest exist in two or less tracing directions from the origin trace.

Condition 2: When assuming the connected components which are the smallest $r\_new_{i,j}$ with respect to each of the tracing directions are the trace target, the connected components which are the trace target can be specified as one.

Condition 3: The connected components of the origin trace from connected components of the destination trace are also a trace target. In other words, the trace target is in both directions for two connected components.

Figure 5 shows examples of three types of tracing conditions. Connected components and the connected components which are the trace targets within these connected components are shown. For (a) in the figure, two connected components exist in the region of $r\_new_{i,j} \leqq 1$ and the distance of other connected components is greater than 1. Because the two connected components existing in the region of $r\_new_{i,j} \leqq 1$ exist in regions with different directions, they are the trace targets. Furthermore, for (b), three connected components exist in the region of $r\_new_{i,j} \leqq 1$. Since only one connected component exists in tracing direction 4, this connected component is the trace target. However, since two connected components exist in the same region in tracing direction 2, it is not possible to judge which of the connected components from among these two connected components is the trace target and as a result there is no trace target. Even further, since condition 1 is not satisfied for (c), there is no connected component that is a trace target.

If only the connected components which satisfy conditions 1 to 3 (trace targets) are trace targets and the trace executes from the origin trace, the trace route to be created will be a trace route cut at the portion where the line type

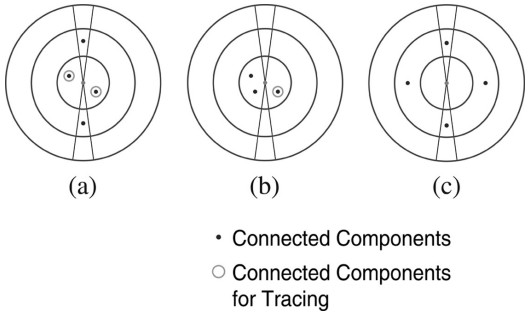


Fig. 5. Examples of tracing of connected components.

patterns are blocked by points or solid lines intersecting at different line type patterns. In other words, a trace route will be formed that prevents wrong traces as much as possible.

In addition, we detect and eliminate incorrectly traced partial trace routes from within the created trace route. The detection of incorrectly traced trace routes is done by extracting line type patterns which have the trace routes based on those line type patterns.

The trace routes at the moment when the connected components are traced have the wrong trace routes removed as much as possible. Because of this, when only one trace route is seen, the correct partial trace route will be longer than the incorrectly traced partial trace route. Namely, when extracting patterns of connected components existing on the trace route, the correct line type pattern can be extracted through the use of a majority decision process.

3.3. Extracting line type patterns of trace route

The properties between the connected components on trace routes are defined as a three-dimensional vector (n_i, n_j, $r_{i,j}$) in order to extract the line type patterns of the trace routes. Here, n_i is the number of black pixels comprising each connected component i on the trace route and $r_{i,j}$ is the Euclidean distance between connected components i and the connected components j (see Fig. 6).

Polling is used as a method to extract line type patterns which have trace routes from features between the connected components on trace routes [7, 8]. This method is as follows.

Step 1: Performs polling of features (n_i, n_j, $r_{i,j}$) between connected components i, j on trace route k in the three-dimensional feature space shown in Fig. 3 within a super ellipse determined by center (n_i, n_j, $r_{i,j}$) and radius (δ_1, δ_2, δ_3) (this is considered to be an error). When the line type patterns are dash-dot lines at this time, the polling is performed such that $n_i \leqq n_j$ to obtain correspondence. In other words, when performing polling of features on a trace route ($n_i > n_j$), n_i and n_j will reverse and the features will be polled.

Step 2: Extracts points where the highest number of polling is obtained within feature space as line type patterns ($n_{pat\,k_1}$, $n_{pat\,k_2}$, $r_{pat\,k_{1,2}}$) which have trace route k.

Step 3: Judges whether the partial trace route not polled at the point of the line type pattern extracted in step 2 is an incorrect partial trace route and then cuts this partial trace route.

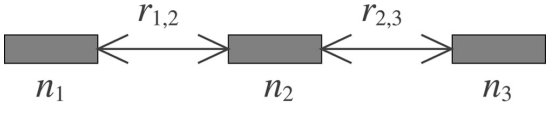


Fig. 6. Features of connected components.

The radius (δ_1, δ_2, δ_3) during pattern polling is determined from quantization errors in graph images. The following equations are defined with δ_1, δ_2, δ_3 being errors of the number of connected component pixels of origin trace n_i, the number of connected component pixels of destination trace n_j, and the distance between the connected components $r_{i,j}$, respectively, and using constants a, b, c:

$$\delta_1 = a + \frac{n_i}{b} \tag{2}$$

$$\delta_2 = a + \frac{n_j}{b} \tag{3}$$

$$\delta_3 = a + \frac{r_{i,j}}{c} \tag{4}$$

The values of a, b, c are determined by prior experimentation.

3.4. Integrating and interpolating trace routes

The trace route created in Section 3.2 is a broken trace route cut at the intersection point between overlapping solid lines or other line type patterns. The integration of the trace routes is performed by means of using line type patterns of each trace route to integrate the broken trace routes. Thereafter, interpolation between the integrated trace routes is done to restore the shape of the lines.

3.4.1. Integrating trace routes

Similarly to Section 3.3, the linear patterns extracted on each trace route in the three-dimensional feature space as polled as shown in Fig. 7 and the polling results are used to judge whether other trace routes have the same linear type patterns.

Step 1: Performs polling of line type patterns extracted at each trace route in the three-dimensional feature space shown in Fig. 7 at all points of super ellipse of radius (δ_1, δ_2, δ_3) centered on (n_{pat_1}, n_{pat_2}, $n_{pat_{1,2}}$).

Step 2: Judges and extracts line type patterns where points with the highest number of polling obtained in feature space exist within a graph region.

Step 3: Judges whether the trace routes polled at the points of the line type patterns extracted in step 2 trace routes that have the same line type patterns and then integrates these. In other words, makes the trace routes into trace routes belonging to identical lines.

Step 4: Clears polling of integrated trace routes from the feature space.

Step 5: Repeats step 2 to step 4 until the points polled in the feature space disappear or in other words until the trace routes not integrated yet do not exist anymore.

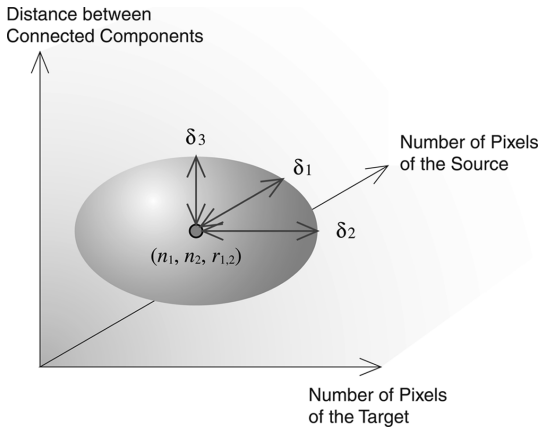


Fig. 7. Feature space for extracting line pattern.

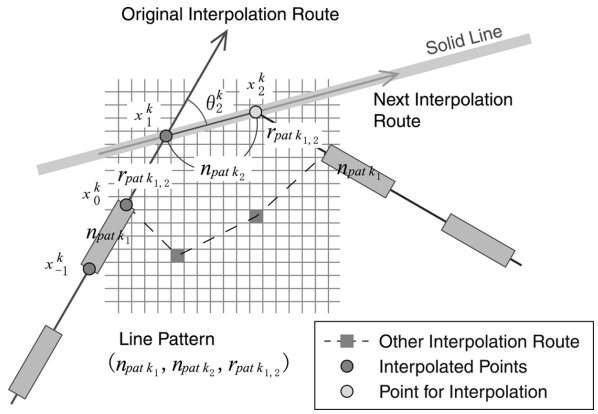


Fig. 8. Interpolation between tracing routes.

The radius (δ_1, δ_2, δ_3) of the line type pattern polling is determined by Eqs. (2), (3), and (4) in the same manner as Section 3.3. In addition, there is not only one vote of a poll for one trace route while polling. The polling is only performed for a number in proportion to the number of connected components existing on each trace route. The reason for this is because it is possible to consider the reliability of the trace routes to be in proportion to the length of the created trace routes.

The processing above makes it possible to detect all line type patterns existing in a graph region and that number from all trace route line type patterns.

3.4.2. Interpolating trace routes of identical line type patterns

This section describes a method to interpolate between trace routes and integrate each trace route for trace routes which have identical line type patterns.

Line type patterns in which each trace route is found by means of the processing of Section 3.4.1 and lines belonging to each trace route as well as their line type patterns are also known. Therefore, even if the trace routes are divided up, the distance up to the position where the next connected component should exist can be estimated. Thereupon, DP matching is used to search for the most optimum route based on the line type patterns of the trace routes and achieve interpolation between trace routes. The following equation defines the evaluation function used at this time. Pruning is performed using a beam search in the DP matching process.

$$P = \max_{k} \sum_{i=1}^{N} f(x_{i-1}^{k}, x_{i}^{k})/N \tag{5}$$

$$f(x_{i-1}^{k}, x_{i}^{k}) = \begin{cases} 2 + \cos\theta_{i}^{k} & (\text{when } x_{i}^{k} \text{ is black pixel}) \\ -2 + \cos\theta_{i}^{k} & (\text{when } x_{i}^{k} \text{ is white pixel}) \end{cases}$$

$$\text{Restriction conditions} : \|x_{i}^{k} - x_{i-1}^{k}\| = m_{i}^{k}$$

x_i^k is the i-th interpolation point on interpolation route k, N is the total number of interpolation points required for the interpolation, θ_i^k is the angle formed by the origin interpolation route and the following interpolation route, and m_i^k follows the equation shown below:

$$m_{i}^{k} = \begin{cases} \text{when} \quad n_{pat\,k_1} \ (\|x_{i-3}^{k} - x_{i-2}^{k}\| = n_{pat\,k_2}) \\ \text{when} \quad n_{pat\,k_2} \ (\|x_{i-3}^{k} - x_{i-2}^{k}\| = n_{pat\,k_1}) \\ \text{when} \quad r_{pat\,k_{1,2}} \ (\|x_{i-3}^{k} - x_{i-2}^{k}\| = r_{pat\,k_{1,2}}) \end{cases}$$

Figure 8 shows a conceptual diagram of this interpolation. $f(x_{i-1}^k, x_i^k)$ is a function for performing an interpolation in preference to points which are black pixels. In addition, because $\cos\theta_i^k$ is included in the function, an interpolation route that stops abrupt changes can be selected.

4. Recognition Experiments

4.1. Experiment conditions

We performed experiments to confirm that the line type pattern extraction process (3.3), the trace route integration process (3.4.1), and the trace route interpolation process (3.4.2) proposed in this paper execute effectively. Table 2 shows information images used as input for evaluations. All input images were captured from each document at 400 dpi using a scanner. We applied the proposed method to each image and used gnuplot to plot numerical information (coordinate information) of the obtained line graphs. These were output images.

Table 2. Information about input images

Input image	Image format	Image size	Original reference document
1	ppm	496×502	Ref. 9
2	ppm	1984×1502	Ref. 10
3	ppm	1192×1040	Ref. 11
4	ppm	1224×816	Ref. 12
5	ppm	1208×1170	Ref. 13

Before the experiments, we used 10 line graph images (images separate from those used for the evaluations) for prior experimentation and then established values of n_{max}, r_{max}, and a to c of Eqs. (2) to (4) as shown below.

$$n_{max} = Cols/12$$

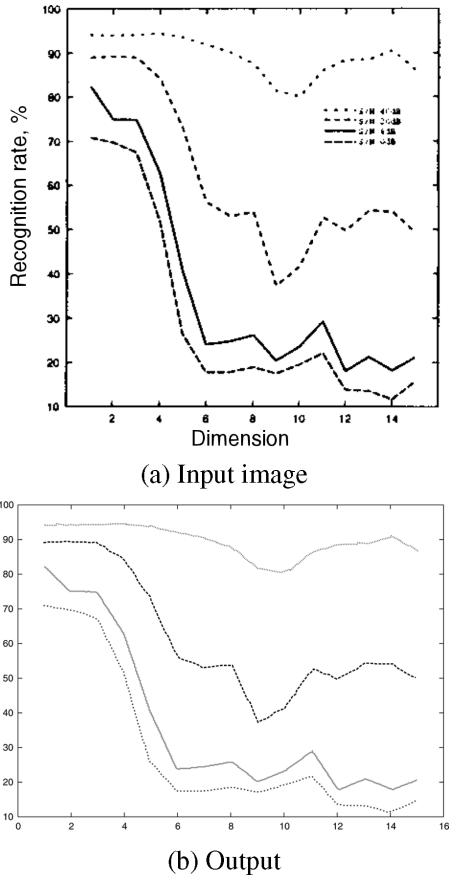
$$r_{max} = Cols/15$$

$$a = 2$$

$$b = 10$$

$$c = 20$$

Cols is the number of pixels in the longitudinal direction of the input image. Therefore, n_{max} and r_{max} are not dependent on the resolution of the image. The values of a to c are values for 400 dpi. For other resolutions the values must be suitably selected in proportion to the resolution.



(a) Input image

(b) Output

Fig. 9. Experimental result 1.

4.2. Experimental results and observations

4.2.1. Evaluating line type pattern extraction process

At first, the image of Fig. 9(a) was input in order to verify that it was possible to extract line type patterns within a graph using the proposed connected component trace method and a method to express line type patterns and then correctly recognize the lines. This was done when four different line patterns existed in a graph region and these lines did not intersect. Figure 9(b) is an output image. From this image it is understood that each of the line patterns can be correctly recognized and are output as patterns with four different lines. Furthermore, from this image it is understood that the line type pattern extraction process is executing effectively.

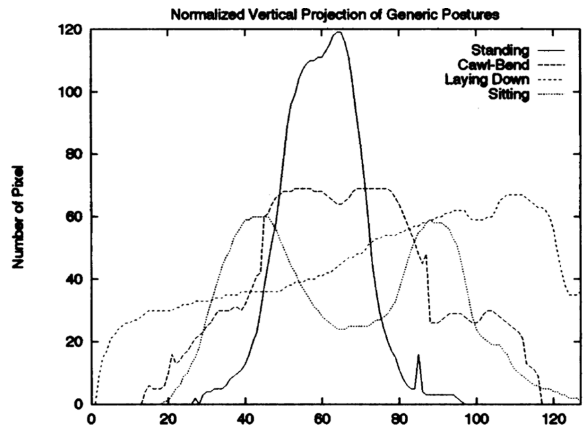
4.2.2. Evaluating integration process of routes

Next, we performed experiments to confirm that the identical lines can be identified by integrating routes of traced lines. Figure 10(a) is a line graph image that has intersecting dotted lines with different line type patterns. As described in Section 3.4, because lines obtained by tracing connected components are cut at points where lines intersect, the same types of lines must integrate. Because of this, the integration process of line routes is required to correctly recognize the lines of Fig. 10(a).

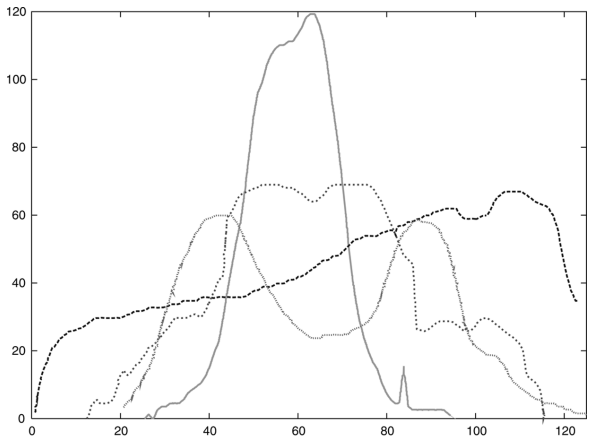
If we look at the output results [Fig. 10(b)], it is understood that the lines existing in the graph region are recognized correctly and the integration process of trace routes of identical line type patterns is executing effectively. Although disappearance of peaks and valleys in one portion of the graph is seen, we can consider this to be due to abrupt changes to the graph and a short distance between nearby connected components resulting in the connected components which should be integrated jumping over and incorrect integration between connected components.

4.2.3. Evaluating interpolation process of routes

Next, we performed experiments to confirm that the identical lines can be identified by interpolating the routes of the divided lines. Figure 11(a) shows a line graph in

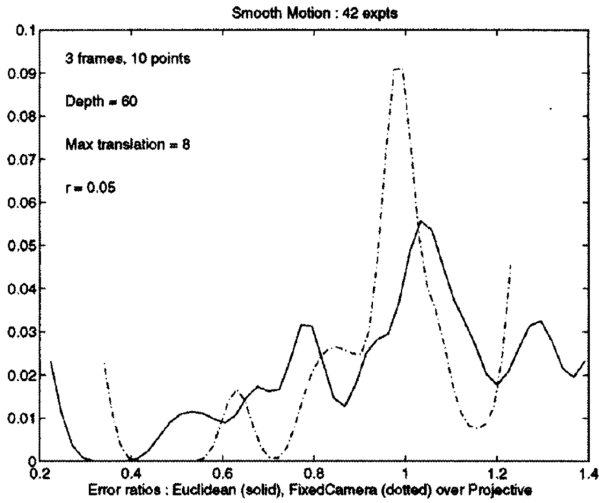


(a) Input image

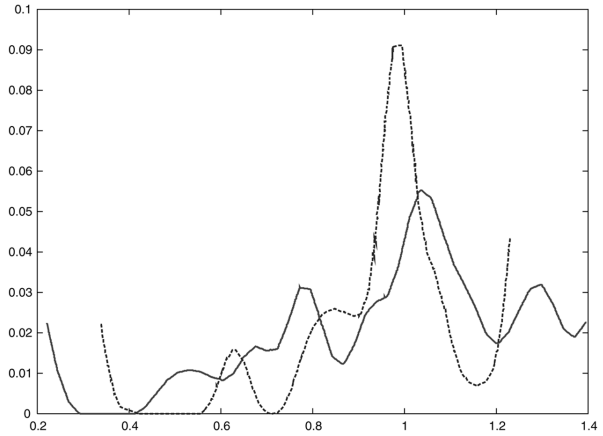


(b) Output

Fig. 10. Experimental result 2.



(a) Input image



(b) Output

Fig. 11. Experimental result 3.

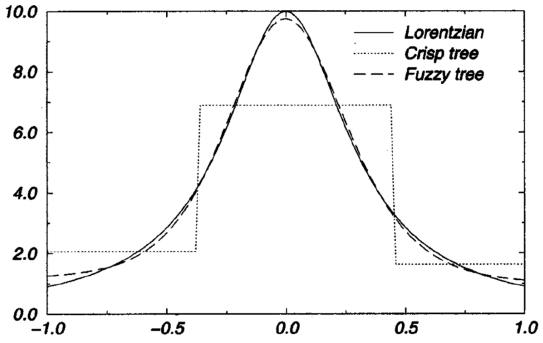
which the lines from the graph region are divided by the axis. Since divided trace routes are generated for this case, these trace routes must be correctly integrated based on the line type patterns and the area above the axis between the trace routes interpolated. If we look at the results in Fig. 11(b), it is understood that the trace routes divided by the axis are correctly integrated and the lines above the axis between the trace routes correctly interpolated.

Figure 12(a) shows a line graph image cut off by lines whose major portion is comprised by solid lines. This major portion of these lines (line type pattern corresponding to a fuzzy tree in a legend region) is comprised by comparatively large connected components. For this case as well, it is understood that integration and interpolation of identical line type pattern trace routes execute effectively and that it is possible to accurately interpolate between trace routes with large intervals between them.

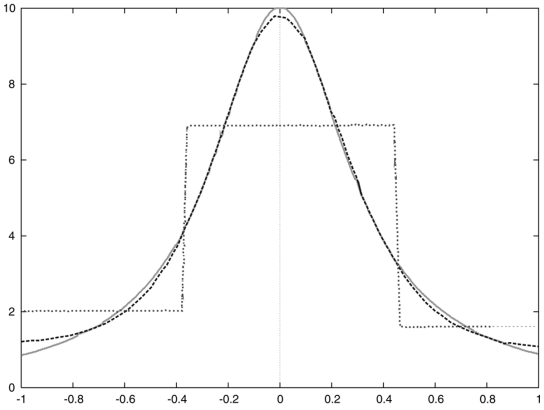
4.2.4. Observations

From the experimental results above, it is understood that lines existing in graph regions are correctly recognized by the proposed method and that each essential technology of the proposed method such as tracing of connected components, extraction of line type patterns on trace routes, and integration and interpolation of identical line type pattern trace routes are executed effectively. These results show that the proposed method is effective to recognize line graph images without markers.

This method, however, cannot always correctly recognize lines when, for example, lines are in close proximity to line type patterns which have the above-mentioned intervals or when lines are overlapping other lines. Figure 13 shows the latter example. If we look at the output, it is understood that one part is missing. The reason for this is because there are overlapping dotted lines and dash-dot lines inside the graph region resulting in a route being

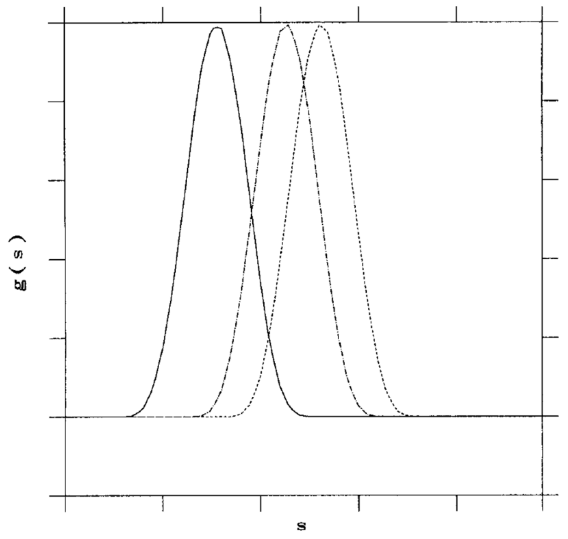


(a) Input image

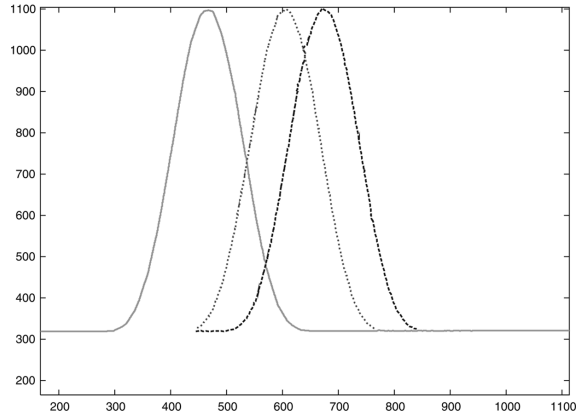


(b) Output

Fig. 12. Experimental result 4.



(a) Input image



(b) Output

Fig. 13. Experimental result 5.

handled as a trace route that does not belong to any lines because the line type pattern that a route has is different from the line type pattern other trace routes have. It appears that knowledge related to the graph must be utilized in order to solve this.

4.2.5. Processing time

Table 3 shows the calculation time required to recognize each input image. The computer used is an Intel Pentium III 800 MHz. From the table it is understood that when the input image is the image in Fig. 9(a) that does not require interpolation processing of identical line type pattern trace routes (experiment 1), the calculation time will be much shorter compared to when the input image is a line graph that requires other interpolation processing. In other words, it is understood that almost all the calculation time required to recognize line graphs is required for the interpolation processing of identical line type pattern trace routes. In addition, by comparing the processing times of Fig. 11 (experiment 3) and Fig. 12 (experiment 4), it is also understood that the calculation time required for recognition becomes much longer as the intervals between the trace routes which have identical line type patterns requiring interpolation become larger. In this method we consider all possibilities for interpolation routes between trace routes where interpolation is performed. Because of this, it is thought that the number of trace routes forming trace route candidates will increase and the calculation time required for the interpolation between trace routes which have identical line type patterns will become longer as the distance between the trace routes requiring interpolation becomes larger. However, if the interpolation is performed using

Table 3. Recognition time

Experiment	1	2	3	4	5
Computation time, s	1.8	629.0	61.3	1643.9	29.1

simple straight lines in order to avoid this, there is a chance that shapes different from the original graphs might be formed. In order to shorten the calculation time, knowledge related to the graph must be utilized for better efficiency.

5. Conclusion

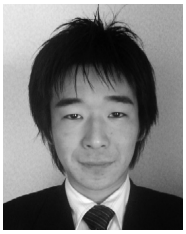
This paper proposed a recognition method for line graphs targeting line graph images without markers. We confirmed that according to the proposed method line graphs can be recognized correctly when dotted lines and solid lines are overlapping, when multiple line type patterns exist inside a graph region, or when multiple line type patterns intersect and mix in complicated patterns.

We could not recognize line type patterns of dash-dash-dot lines in this method, however. Future topics will be examinations of recognition methods which consider line type patterns of dash-dash-dot lines. Other future topics will be examinations into situations when this method cannot be applied when, for example, lines are in close proximity to line type patterns which have the above-mentioned intervals or when lines are overlapping other lines.

REFERENCES

1. Nagy G. Twenty years of document image analysis in PAMI. IEEE Trans Pattern Anal Mach Intell 2000;22:38–62.
2. Yokokura N, Watanabe T. Recognition of bar graphs using layout structure knowledge. Inf Process Soc Japan 1999;40:2954–2966.
3. Lee MH, Babaguchi N, Kitahashi T. Symbolization and presentation of graph images for intelligent communication of document images. Proc ACCV'95, Vol. 3, p 680–684.
4. Shimada S, Sumimoto S, Ejiri M. Recognition algorithms of broken lines and dotted lines for automatic drawing input. Trans IEICE 1986;J69-D:759–770.
5. Shima Y, Shinjo H, Marukawa K, Nakajima K. One method of dotted line extraction from polled images. J IEICE 2002;D-12-60.
6. Hilditch CJ. Linear skeleton from square cupboards. In: Machine intelligence, Vol. 6, p 403–733. Edinburgh University Press; 1969.
7. Duda RO, Hart PE. Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 1972;15:11–15.
8. O'Gorman F, Clowes M. Finding picture edges through collinearity of feature points. IEEE Trans Comput 1976;25:449–456.
9. Nakanishi I, Yoshida Y. Effects of several compound note characteristics during numerical character recognition. J IEICE Information Systems 2, D-14-5, 1997.
10. Haritaoglu I, Harwood D, Davis LS. W^4: Real-time surveillance of people and their activities. IEEE Trans Pattern Anal Mach Intell 2000;22:809–830.
11. Oliensis J, Govindu V. An experimental study of projective structure from motion. IEEE Trans Pattern Anal Mach Intell 1999;21:665–671.
12. Suarez A, Lutsko JF. Globally optimal fuzzy decision trees for classification and regression. IEEE Trans Pattern Anal Mach Intell 1999;21:1297–1311.
13. Chalmond B, Girard SC. Nonlinear modeling of scattered multivariate data and its application to shape change. IEEE Trans Pattern Anal Mach Intell 1999;21:422–432.

AUTHORS



Toshimichi Fuda received his BSEE degree from Tohoku University in 2000, completed the first half of the Ph.D. program in 2002, and joined NTT Data Institute of Management Consulting, Inc. His university research concerned document image recognition.

AUTHORS (continued) (from left to right)

Shinichiro Omachi (member) received his BSEE and D.Eng. degrees from Tohoku University in 1988 and 1993 and became a research associate at the Information Processing Education Center. He has been with the Graduate School of Engineering since 1999. He was a guest professor at Brown University from 2000 to 2001. His research interests are pattern recognition, computer vision, parallel processing, and development of character recognition systems. He is a member of IEEE, Information Processing Society of Japan, Japanese Society for Artificial Intelligence, and Pattern Recognition Society.

Hirotomo Aso (fellow) received his BSEE and D.Eng. degrees from Tohoku University in 1968 and 1974. He was appointed a professor at the School of Engineering in 1991, and is now a professor at the Graduate School of Engineering. His research interests are learning automaton, cell construction automaton, parallel processing theory, systemic algorithm design theory, character recognition, voice recognition, and neural networks. He received an Achievement Award from IEICE in 1991. He is a member of IEEE, ACM, EATCS, Information Processing Society of Japan, and Japanese Society for Artificial Intelligence.