数学グラフからの成分抽出方法について

 Extraction Method of Graph Elements in Mathematical Graphs

〇陳 建軍 1, 長屋 竜治 1, 高木 昇 2
〇Jianjun Chen 1, Ryoji Nagaya 1, Noboru Takagi 2

1富山県立大学大学院工学研究科知能デザイン工学専攻
2富山県立大学工学部知能デザイン工学科

Graduation School of Engineering, Toyama Prefectural University
Department of Intelligent Systems Design Engineering, Toyama Prefectural University

Abstract: Tactile graphics are images that use raised surface so that a visually impaired person can feel them. Tactile graphics are necessary to visually impaired students when they study mathematics and science. Since producing tactile graphics are not simple task, an intelligent computer-aided system for assisting the production of tactile graphics is needed. Mathematical graph recognition from printed materials plays an important role to develop such a system. So, this paper focuses on part of a method of mathematical graph recognition. A mathematical graph includes graph elements such as the rays representing the x-axis and the y-axis and the segments or the curves representing functions or equations. Graph elements are often drawn so that they are overlapped each other. So, this paper discusses a method for extracting graph elements from an element which is drawn by overlapping more than one graph element. The effectiveness of our method is evaluated by a computer experiment.
translate a bitmap image of mathematical graph into SVG.

This paper is organized as follows. Section 2 describes the outline of our method. Section 3 introduces the method of dividing large elements and Section 4 presents the technique of merging primitive elements. A brief description of curve fitting method is given in Section 5, and experiment results are demonstrated and discussed in Section 6. Section 7 is the conclusion.

2. Outline of Our Method

Figure 1 shows examples of mathematical graphs that we focus on. The characteristics of these mathematical graphs are summarized as follows.

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

A mathematical graph includes the following three elements, (1) solid line graph elements, (2) broken line graph elements, and (3) elements from character strings and mathematical formulas. This paper focuses on solid line graph elements.

A solid line graph element is often formed by overlapping two or more graph elements such as rays of the x-axis and the y-axis and the segments or the curves representing functions or equations. In order to translate bitmap images of solid line graph elements into SVG, we need to separate graph elements from an element that is drawn so that two or more graph elements are overlapped each other. The following is the outline of our method for extracting graph elements from an overlapped element (See Figure 2).

(1) Separating large elements from the original graph (Figure 2 (a))

(2) Dividing a large element into primitive elements (which will be explained in the next section) by removing intersections (Figure 2 (b))

(3) Merging primitive elements in order to form a graph element (Figure 2 (c))

In the steps (1) and (2), we focus on large elements of the original graph. This is because the size of a solid line graph element is often much larger than the size of an element from broken lines, character strings, and mathematical formulas. Procedures for dealing with broken line graph elements and elements from character strings and mathematical formulas are discussed in [10] and [11].

We first apply the following three processes to an input image: binarization, noise reduction, and labeling. After the labeling process, we have all connected components in the input image. We then separate large elements from the connected components. This is done by the following way. First, for every connected component $C$, a rectangle $R$ that circumscribes $C$ is determined. If the length of the long side of $R$ exceeds threshold value $w \times \ell$, then $C$ is classified as a large element. Here, $\ell$ is the length of the long side of the input image, and $w$ is a weight of $[0,1]$, which is determined by an experiment.

3. Dividing Large Elements
From Section II, elements in a mathematical graph are divided into two parts, large elements and small elements. The basic idea of extracting graph elements from a large element is that dividing a large element into small fragments (which we will call primitive elements), and then merging primitive elements in order to form a graph element.

We introduce the following four procedures for dividing large elements, (1) thinning, (2) removing short branches, (3) finding intersections and (4) dividing.

(1) Thinning: We first apply a thinning procedure to large elements, and then we have skeletons of those elements. Note that the widths of segments and curves constructing each skeleton are one pixel.

(2) Removing Short Branches: A skeleton given from (1) often includes many undesirable short branches, and therefore every branch whose length is less than a threshold is removed from the skeleton.

(3) Finding Intersections: For every point \( p \) in a skeleton, using \( 3 \times 3 \) spatial filters we identify whether \( p \) is an intersection point in the skeleton. Since the width of a segment/curve of an original graph element is more than one pixel, if an intersection exists in the original graph element, then the intersection occupies a certain area in the input image. This makes the intersection include more than one intersection points in the skeleton of that element (See Figure 3). Therefore, in order to group intersection points located in the same intersection, we apply an agglomerative hierarchical clustering to the set of intersection points. The linkage criterion of the hierarchical clustering is the nearest neighborhood. The process of the hierarchical clustering is terminated if the shortest distance between clusters exceeds a threshold. As the result, each cluster corresponds to one of the intersections.

(4) Dividing: By removing all intersection points existing in a skeleton, the skeleton is divided into small fragments. Note that it is a characteristic that every fragment has two endpoints, and these two endpoints are adjacent to some of the intersection points. For a fragment \( e \), if intersection points adjacent to the two endpoints of \( e \) are members of the same cluster, then we remove \( e \) from the fragments. This is because such a fragment is part of the intersection in which the intersection points associated with \( e \) are included. Furthermore, if the size (the number of pixels) of a fragment is less than a threshold, then the fragment is also removed from the fragments. The remaining fragments will be called primitive elements.

After performing the procedure above, we get all the primitive elements of a large element, and also have clusters where each of them includes intersection points located at the same intersection.

4. Merging Primitive Elements

We apply fuzzy inference in order to merge primitive elements so that the merging result will form a graph element. If two primitive elements \( e_1 \) and \( e_2 \) satisfy the following three geometric characteristics, then it is plausible that \( e_1 \) and \( e_2 \) are parts of the same graph element.

(1) Primitive element \( e_1 \) and \( e_2 \) have been connected at one of the intersections before applying the division process from Section 3.

(2) Let \( C \) be a curve which is a part of a skeleton from Section 3. If \( C \) includes primitive elements \( e_1 \) and \( e_2 \) as a part of it, then a curvature of \( C \) at a point \( p \) is low, where \( p \) is the intersection point that divides primitive elements \( e_1 \) and \( e_2 \).

(3) The widths of the original graph elements corresponding to \( e_1 \) and \( e_2 \) are close.

There are several methods [12] for measuring curvatures of a curve in digital images. In this paper, a curvature is defined as a finite difference of measures of two angles as follows. Let \( C \) be a digital curve, and let \( p_i = (x_i, y_i) \) be a point on \( C \). Then, a curvature \( \kappa \) of curve \( C \) at point \( p_i \) is defined as subtraction \( \phi_i - \phi_{i-1} \) (See Figure 4), i.e.,

\[
\kappa = \phi_i - \phi_{i-1}
\]  

(1.1)

where \( \phi \) is a measure of an angle which is formed by the x-axis and the line which two endpoints \( p_i \) and \( p_{i+1} \) determine, and \( \phi_{i-1} \) is also a measure of an angle consisting of the x-axis and the line determined by \( p_{i-1} \) and \( p_i \). Note that this definition is based on the original definition of curvatures: a
curvature is the rate of change of the angle through which the tangent to a curve turns in moving along the curve.

The following description explains the merging process. Let \( P = \{p_1, p_2, \ldots, p_n\} \) be a cluster of intersection points from Section 3, and let \( E(P) = \{e_1, e_2, \ldots, e_t\} \) be a set of the primitive elements satisfying the condition where one of the two endpoints of a primitive element in \( E(P) \) is adjacent to an intersection point of \( P \). After applying the following procedures (1)-(3) to set \( E(P) \), we will have graph elements consisting of primitive elements in \( E(P) \). Note that in the following procedures, a fuzzy inference system is executed. An input to the fuzzy inference system is a pair of two primitive elements \( e_1 \) and \( e_2 \) in \( E(P) \), and this system is designed so that an output value means that the higher the value, the more plausible the fact that primitive elements \( e_1 \) and \( e_2 \) are parts of the same graph element. The fuzzy inference system will be explained in detail after the description about merging below.

(1) Let \( e_1 \) and \( e_2 \) be two primitive elements of \( E(P) \).

Apply the pair \( (e_1, e_2) \) to the fuzzy inference system, and then let \( \mu \) be the value taken from the fuzzy inference system. After executing this process for every pair of two primitive elements in \( E(P) \), we have a sequence \( v_1 v_2 \ldots v_k \) (where \( k = |C_e| \)) of output values \( v_1, v_2, \ldots, v_k \) from the fuzzy inference system.

(2) Sort the values of sequence \( v_1 v_2 \ldots v_k \) in ascending order, and let \( v(1), v(2), \ldots, v(k) \) be the sequence sorted, where \((1), (2), \ldots, (k)\) is a permutation of sequence 1, 2, ..., \( k \).

(3) By checking the values in sequence \( v(1), v(2), \ldots, v(k) \) one by one from head \( v(1) \) to tail \( v(k) \), execute the following procedure: Let \( e_1 \) and \( e_2 \) be the two primitive elements associated with the current value \( v(i) \) of sequence \( v(1), v(2), \ldots, v(k) \). If value \( v(i) \) exceeded a threshold, or neither \( e_1 \) nor \( e_2 \) has not been merged yet, then merge primitive elements \( e_1 \) and \( e_2 \) into one element.

By applying the procedure above to primitive elements from Section 3, we have all of the graph elements. The fuzzy inference system has the following two input attributes, curvature \((x_1)\) and width \((x_2)\).

Curvature: Let \( p_1 \) and \( p_2 \) be the two endpoints which give the distance between two primitive elements \( e_1 \) and \( e_2 \). Here, the distance is the shortest in the set of distances between any pair of two endpoints of \( e_1 \) and \( e_2 \). Starting from \( p_1 \), let \( p_1' \) be the first corner of primitive element \( e_1 \). Here, the corner detection method introduced by Wall and Danielsson [13] is applied. Then, the value of \( x_1 \) is a curvature calculated from three adjacent points \( p_1', p_1, \) and \( p_2 \) by the equation (1.1).

Width: If the widths of the original graph elements corresponding to primitive elements \( e_1 \) and \( e_2 \) are close, then \( e_1 \) and \( e_2 \) can be parts of the same graph element. So, we measure the widths of the original graph elements corresponding to \( e_1 \) and \( e_2 \), then the value of \( x_2 \) is the absolute value of the subtraction between the two widths.

The following shows the fuzzy if-then rules of the merging procedure.

Rule 1: If \( x_1 \) is low and \( x_2 \) is small, then it is plausible that two primitive elements \( e_1 \) and \( e_2 \) are merged into one element (i.e., it is plausible that the two primitive element are parts of the same graph element).

Rule 2: If \( x_1 \) is high, then it is not plausible that two primitive elements \( e_1 \) and \( e_2 \) are merged into one element.

Rule 3: If \( x_2 \) is large, then it is not plausible that two primitive elements \( e_1 \) and \( e_2 \) are merged into one element.

The fuzzy if-then rules include 4 fuzzy sets (low, high, small, and large) in the antecedents and 2 fuzzy sets in the consequents. Figure 5 (a) and Figure 5 (b) show graphs of membership functions \( \mu_{\text{low}}, \mu_{\text{high}}, \mu_{\text{small}}, \) and \( \mu_{\text{large}} \), which corresponds to fuzzy sets low, high, small, and large, respectively. Figure 5 (c) shows graphs of membership function \( \mu_{\text{pos}} \) and \( \mu_{\text{neg}} \), which corresponds to the 2 fuzzy sets in the consequents. Here, \( \mu_{\text{pos}} \) is a membership function of the fuzzy set representing a positive conclusion, while \( \mu_{\text{neg}} \) is for negative. The process of fuzzy inference in our system follows Mamdani’s fuzzy inference method, but the minimum operator was exchanged with the product operator.

5. Curve Fitting with Parametric Spline

A bitmap image is composed of a fixed set of
dots, while a vector image is composed of a fixed set of shapes. In order to translate a bitmap image into SVG which provides a vector image, we need to find a function or an equation which approximately expresses the shape of a graph element represented by a bitmap image. Curve fitting is one of the best ways for finding such a function/equation that has the best fit to a series of points taken from a bitmap image of a graph element. In this paper, splines will be applied to realize curve fitting. Although there are many kinds of splines, we focus on parametric splines [14].

Our curve fitting process is applied to the skeleton of a graph element obtained from Section 4. For a set of knots \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\), a parametric cubic spline can be determined by \(x = X(t)\) and \(y = Y(t)\), where \(t\) is a parameter of \([1, n]\). We can get the expression of \(x\) and \(y\) respectively by using natural cubic spline interpolation. For example, a natural cubic spline can be determined only when there is a function \(x = f(t)\) (or \(y = f(t)\)) such that all knots \((x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)\) satisfy relations \(x_i = f(t_i)\) (or \(y_i = f(t_i)\)) for \(i = 1, 2, \ldots, n\), here \(t_i = i\). After the curve fitting process, every skeleton can be represented approximately by a parametric spline. Knots are selected from a set of points consisting of a skeleton by choosing every 10 points from a terminal point.

6. Experimental Results

We selected 38 mathematical graphs from mathematics and science textbooks. Then, these graphs were captured by using an image scanner, whose resolution was set at 600 dpi. These electric images were saved in the bitmap format. The size of images is in almost \(1,500 \times 1,500\) pixels.

We input these bitmap images to our system. Every input image was first applied to the separation method from Section 2. Then, the method from Section 3 divided the large elements, which were extracted by the separation method, into primitive elements.

6.1. Evaluation of the Merging Process

We performed experiments to confirm the fact that graph elements are extracted correctly by the merging method. However, the merging method cannot always correctly merge primitive elements which are parts of same graph element. Figure 7 shows a failed example of merging results. In Figure 7 (a), it draws the tangent to a curve at a given point, graph elements for the tangent and the curve meet at a point, and in the region close to this point the same element expresses both of the tangent and the curve. Therefore, in this case, the original element is divided into the following three fragments: the fragment corresponding to the tangent (Primitive element (1) of Figure 7 (d)), the fragment corresponding to the curve (Primitive element (2) and (3) of Figure 7 Figure (d)), and the fragment corresponding to both of the tangent and the curve (Primitive element (4) of Figure 7 (d)). In many cases, the width of the fragment corresponding to both of the tangent and the curve is thicker than that of the fragment corresponding to the tangent or the curve. Since

![Figure 6 Successful Results of Figure 1](image)

Furthermore, from Figure 6 it showed that the merging process is executed effectively.

6.2 Evaluation of the Curve Fitting Process

We performed experiments to confirm the fact that graph elements are expressed approximately by parametric splines. After executing the merging process, each graph element was applied to determine its parametric spline. If we checked the fitting results, it showed that the process of curve fitting is executed effectively.

6.3 Observation

From the experimental results above, graph elements are correctly extracted by the proposed merging method. However, the merging method cannot always correctly merge primitive elements which are parts of same graph element. Figure 7 shows a failed example of merging results. In Figure 7 (a), it draws the tangent to a curve at a given point, graph elements for the tangent and the curve meet at a point, and in the region close to this point the same element expresses both of the tangent and the curve. Therefore, in this case, the original element is divided into the following three fragments: the fragment corresponding to the tangent (Primitive element (1) of Figure 7 (d)), the fragment corresponding to the curve (Primitive element (2) and (3) of Figure 7 Figure (d)), and the fragment corresponding to both of the tangent and the curve (Primitive element (4) of Figure 7 (d)). In many cases, the width of the fragment corresponding to both of the tangent and the curve is thicker than that of the fragment corresponding to the tangent or the curve. Since
these widths are different enough to take a negative consequence from the fuzzy inference system, the primitive elements corresponding to these fragments were not able to be merged into one graph element.

Figure 7 Failed Example

7. Conclusion

We are now developing a system for automating transformation of mathematical graphs into tactile graphics. To develop this system, mathematical graph recognition is needed. So, this paper discussed part of mathematical graph recognition techniques, especially a method for extracting solid line graphs. Our method can almost correctly extract graph elements from bitmap images, but some of them cannot be obtained correctly. To determine shapes of graph elements represented by bitmap images, we applied parametric splines. The parametric splines interpolation process proposed in this paper execute effectively. However, we need to examine the comprehensibility of tactile graphics produced from parametric splines.

References

連絡先:
富山県立大学工学部知能デザイン工学科
高木 昇
E-mail: takagi@pu-toyama.ac.jp