Study of Graph-to-data Transformation

Nian-you CHEN¹,², Xiao-hong XIAO¹, You-ping LUO², Wei GUAN¹ and Min YAN¹

¹Huanggang Normal University, Huanggang, China, 438000
²Huanggang Polytechnic College, Huanggang, China, 438002

*Corresponding author

Keywords: Graph-to-data, Pixel, Binarization, Refining, Curve coordinate extraction.

Abstract. The purpose of graph-to-data transformation is to convert graphs into data and equations. Graphs drawn manually or obtained through recorders or cameras can be scanned into a computer directly or transmitted and stored within this computer through data line. Then the following treatment proceeds: loading the original graph(BMP)→denoising→smoothing and sharpening→gradation→binarization→refining→extracting curve skeleton→capturing curve coordinates. Finally the coordinates needed for digital analysis are gathered, which will be processed by specialized data-to-image transformation software to reconstruct the original graphs, for the purpose of accurate restoration and optimization of graphs.

Introduction

There are a lot of graphs drawn by old instrument (recorders) in literatures, which can only be scanned into computers, and cannot be simulated directly on computers. Such treatment not only damages the original information of the graphs, but also is not in accordance with the requirements of present mapping. If the coordinates of all the points of the original graphs can be obtained, they can be further provided to data-to-image software to restore the original graphs so that the real message the graphs reflect can be acquired for our scientific research. The key to realize the above goal is graph-to-data transformation technology.

Right now, more and more scholars put great emphasis on the research of graph-to-data transformation technology to develop reliable software to realize image optimization and reprocessing. There are the following advantages of graph-to-data transformation technology:

(1) It can convert graphs into data conveniently and accurately, and the resulted data can be processed by data processing software (like Excel, Origin) to reproduce bitmap curves which can be processed at one’s own will.

(2) It can change big images into small images conveniently and accurately so that they can easily be used in electronic scientific papers. When the image is very big, it will take a lot of storage space and transmission time, so it often needs to be compressed. A commonly used method is to directly zoom out the original image, but it will result in very thin lines which can be hardly examined sometimes.

(3) It can optimize and refine graph images. For example, images manually drawn or some coarse images can be converted into standard digital images which are more convenient for later processing.

In this paper, the original graph image is first processed through a series of mage processing methods and techniques, and then the curve information of the image is extracted, that’s, the curve coordinates. Finally, the coordinates are processed by computer to acquire the original image information. Here we only focus on graph-to-data transformation. The quality of final reconstructed image will not discussed in this paper.

Methodology

The original graph image is preprocessed (loading, denoising, smoothing, contrast stretching,
binarization, refining, etc.) so that the computer can conveniently recognize and process the information so as to extract the curve coordinates of the image at last[1].

The processing flow is shown as Figure 1. The main part includes two steps:

![Flowchart](image)

**Figure 1. The processing flow of graph-to-data transformation.**

Step 1: preprocess the original graph image so that the computer can recognize the information. The detailed processing is as follows:

1. Loading the original graph image → denoising → smoothing and sharpening → contrast stretching → gradation → wave clipping → thresholding → binarization → corrosion → swelling → open operation → close operation → refining → extracting curve skeleton

Most image processing steps are handled using existing algorithms. Here, we put more emphasis on image refining and skeleton coordinate extraction, and will discuss these two aspects in section iii in detail.

Step 2: design a fast image traversal method to access all pixels of the preprocessed image to find the points whose RGB=(0, 0, 0), that’s dark points, as a result, the whole curve coordinates are extracted).
Image Refining and Skeleton Coordinate Extraction

First, input the original graph image to be processed into a computer through either scanners or USB interface connected to digital cameras, DV, or mobile phones. Then store the input image on a fixed location for later processing and analysis.

After denoising, smoothing, sharpening, contrast stretching and binarization, the subsequent processing mainly focuses on refining and skeleton coordinate extraction. This paper discusses these two aspects emphatically.

Image Refining

Image refining is to delete some points from the original image while retaining its initial shape [2], specifically, retaining the skeleton of the original image.

**Basic Image Refining Algorithm.** For the sake of processing convenience, we only use the value of 0 and 255 of the palette of 256-level gray image.

Whether a point can be deleted or not is based on the judgment of its 8 adjacent points. The following are some example, shown as Figure 2.

![Figure 2. Point deletion mechanism.](image)

In Figure 2, (1) cannot be deleted as it is an internal point. What we want is the skeleton of the original image, if the internal point is deleted, the skeleton will be hollowed out; (2) cannot be deleted too for the same reason; (3) can be deleted because it is not inside the skeleton; (4) cannot be deleted because the original contiguous part will be disconnected after deletion; (5) can be deleted as it is not inside the skeleton; (6) cannot be deleted for it is the endpoint of the line. If it was deleted, the whole line would be deleted and nothing would be left.

To sum up, we have the following judgment during deletion: (1) the internal points cannot be deleted; (2) the isolated points cannot be deleted; (3) the endpoints of lines cannot be deleted; (4) If point P is a boundary point, and the connected component doesn’t increase after deleting P, then it can be deleted.

Based on the above judgment, we can first make a table containing 256 elements. The value of each element is either 1 or 0. Suppose the white point is 1, and the dark point is 0. The table look-up method is: the value of the upper left point corresponds to the first digit (least-significant digit) of a 8-digit number, sequentially, the right above point is the second digit, the upper right point is the third digit, the left adjacent point is the fourth digit, the right adjacent point is the fifth digit, the lower left point is the sixth digit, the right below point is the seventh digit, and the lower right point is the eighth digit. For example, in Figure 2, (1) corresponds to item 0 in the following table, and the value is 0; (2) corresponds to 37, and it is also 0; (3) corresponds to 173, and the value is 1; (4) corresponds to 231, the value is 0; (5) corresponds to 237, the value is 1; (6) corresponds to 254, the value is 0.

static int erasetable[256] = {
    0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 
    0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 
    0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 
    0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
}


With this table, the algorithm becomes very simple. Scan the image row by row every time. For each point (not including boundary points), compute its corresponding index in the table. If it is 0, then it is kept; otherwise, it is deleted. If no point is deleted during a scan, then the scan stops, and the remaining points are skeleton points. If there are points deleted during a scan, then a new scan begins. This process repeats again and again until no points will be deleted.

**Improved Image Refining Algorithm.** Actually, the above algorithm has some drawbacks. For example, there is a black rectangle, as shown in Figure 3. What we want after refining is a horizontal straight line located at the center of this black rectangle. The actual result is indeed a horizontal straight line, but it is the downmost line, not the central line.

![Figure 3. Black rectangular.](image)

![Figure 4. The result after refining.](image)

Why the result is like this? The reason is: during the up-to-down and left-to-right scanning, the first black point scanned is the upper left point in the black rectangular. Through table look-up, we find it can be deleted. The next point is its right adjacent point, and it can also be deleted. As this process goes on, the whole line is deleted. As all lines except for the lowest line are similar, they are all deleted. For the lowest line, as it is the last line left, and the first left point and right point are endpoints, they can't be deleted. Therefore, the lowest line is kept, but it is not our expected result.

To solve the above problem, during horizontal scanning, we first judge whether the left and right neighboring points are black points. If they are both black points, they will not be deleted. If one black point is deleted, then skip its right neighboring point, and process the next point. After this treatment, the final result is a short segment of vertical line (as shown in Figure 4), which still doesn't meet our expectation. The reason is: the first point to be deleted is the upper left point in the black rectangular; the second point to be deleted is the upper right point; the third point to be deleted is the leftmost point of the second line, and the fourth point to be deleted is the rightmost point of the second line. After the first turn of processing, the image width reduces 2 pixels. As this process goes on, at last only the middle column is kept for we just carry out a horizontal refining.

Based on the above analysis, we first perform a horizontal refining, and then follow a vertical refining. The detailed sequence of points to be deleted is: the leftmost point of the first line; the rightmost point of the first line; the leftmost point of the second line; the rightmost point of the second line; ...; the leftmost point of the last line; the rightmost point of the last line. The above sequence is for the vertical refining. Then the vertical refining follows. The detailed deletion sequence is: the top point of the second column (as the first column has been deleted during the horizontal refining); the bottom point of the second column; the top point of the third column; the bottom point of the third column; ...; the top point of the last but one column; the bottom point of the last but one column. As this process repeats, the rectangular seems to be peeled off, and the actual result is a central line, which meets our expectation.
**Skeleton Coordinate Extraction**

Skeleton can be regarded as the central axis of an image. For example, the skeleton of a rectangular is its central axis along its longer direction; the skeleton of a square is its central point; the skeleton of a circle is its centre; the skeleton of a straight line is itself, and so does the skeleton of an isolated point [3]. There are many methods to extract the skeleton of a curve, here we adopt curve tracking calibration method in this paper.

After refining, the image is still in a bitmap form. To acquire the required data, the curve should be tracked and vectorized, that is, sample the curve at fixed intervals, use vector line to replace the curve, and write down its coordinate sequence.

Use computer software to automatically track the target curve. The start tracking point and direction is determined by man-machine interaction. As the absolute checkerboard distance between two adjacent points in a line will not be larger than 2, once the initial starting tracking point and direction is determined, the eight-neighborhood of the initial point can be automatically searched within a ±2 area along the initial direction. If the target pixel is found, it is taken as a point of the tracked line. Then set this pixel as present start tracking point, take the chain-code value between the initial starting point and the newly found pixel as present tracking direction, and repeat the above searching procedure. During the whole process, one thing that should be remembered is to guarantee that the newly searched pixel should not be the one already tracked. If the present tracking point is an endpoint, then the tracking ends.

Based on the above automatic tracking, the curve can be further vectorized. Before tracking, the attributes of each pixel such as color, layer-number, line width, and line type are provided. When tracking the curve, starting from the initial tracking point, record the coordinates of a pixel of the curve every several pixels (a sampling step). After the tracking finishes, a software system writes the attributes of pixels and the coordinates of character pixels into a list-structure vector file. To reproduce the original image, the hardware system reads the attributes of each pixel and the coordinates of those character pixels, using straight lines to connect adjacent character pixels in sequence, and finally obtain an approximate curve path.

**System Design**

The program (Digitizer.exe) is an application software based on image processing[4][5][6], which can recognize image, preprocess image (denoising, smoothing, sharpening, binarization, refining, etc.), and obtain the coordinates of curve points.

**Interface Design**

The software is developed using Delphi[4], and the main interface is designed as Figure 5 shows.

![Figure 5. Main interface of graph-to-data (Digitizer.exe) software.](image-url)
The Main Functions of the Graph-to-data Software are as Follows:

open image, denoising: smoothing: 
sharpening: binarization: refining: 
extracting image skeleton: skeleton coordinates: 

Software Usage

(1) First use "mspaint" to load a bitmap curve, such as the sample infrared line used in this paper. Determine the greatest value and the least value of X axis and y axis respectively (in this paper, for the sample infrared line, Xmin=1000, Xmax=4800, Ymin=12, Ymax=56). Then use "eraser" to clean other parts besides the curve, or use snag software to extract the curve part that we want.

(2) When using this software to realize graph-to-data transformation, we suggest the following processing sequences to be followed:

① smoothing and sharpening;
② contrast adjustment;
③ binarization;
④ use "rotation mode 3" to rotate the image, which is very essential for the processing;
⑤ save the file;
⑥ fill in the edit box the scope of the horizontal and vertical axis;
⑦ click "extracting skeleton coordinates" button, and open the saved black-and-white image;
⑧ open "pixels.txt" according to program prompts;
⑨ save skeleton coordinates into a .txt file according to program prompts;
⑩ start EXCEL, open the above .txt file, and make a scatter diagram.

Experimental Results

Exp1: acquire the curve coordinates of exp1.bmp through Digitizer.

Start Digitizer.exe, load graph image file exp1.bmp, select the buttons (binarization, refining, extracting skeleton coordinates) on the quick bar to preprocess exp1.bmp, and obtain the skeleton of exp1.bmp.

Take any point inside the image as the origin of coordinates, and another point n located at (x, y), moving the mouse, the coordinates of the point that the mouse is pointing at are displayed in the upper right window which is designed to display coordinates.

Exp2: obtain the curve coordinates of exp2.bmp (N- acetyl chloride –L- aspartic acid, Infrared spectrum) through Digitizer (the area selected is: Xmin=1000, Xmax=4800, Ymin=12, Ymax=56)

The original image scanned into the computer is shown as Figure 6.

Figure 6. Exp2.bmp (N- acetyl chloride –L- aspartic acid, Infrared spectrum).

Start Digitizer, load the graph image, carry out binarization and refining, select the area to extract the curve, and move the mouse to extract the coordinates of the curve, then the coordinates of the points that the mouse is pointing at can be viewed in the coordinate display window, as shown in Figure 7.
Obtain the key coordinates of the curve, and select the button “save coordinate” to save these coordinates in a file (.dat) named by current time.

Run Excel, load .DAT file, and draw a curve by Excel, which is just the same as the original curve. The curve drawn by Excel is shown as Figure 8.

Conclusions

To transform graphs to data can simplify and optimize the processing of the original image, however, few literatures focus on the research of image-to-data transformation. This paper introduces the principle, the detailed procedure of image-to-data transformation, as well as its implementation as an application program. The experimental results show that the reconstructed curve is almost the same as the original one.

References