Automatic Extraction of Data from 2-D Plots in Documents

Xiaonan Lu, James Z. Wang, Prasenjit Mitra, and C. Lee Giles
The Pennsylvania State University, University Park, PA, USA
xlu@cse.psu.edu, jwang@ist.psu.edu, pmitra@ist.psu.edu, giles@ist.psu.edu

Abstract

Two-dimensional (2-D) plots in digital documents contain important information. Often, the results of scientific experiments and performance of businesses are summarized using plots. Although 2-D plots are easily understood by human users, current search engines rarely utilize the information contained in the plots to enhance the results returned in response to queries posed by end-users. We propose an automated algorithm for extracting information from line curves in 2-D plots. The extracted information can be stored in a database and indexed to answer end-user queries and enhance search results. We have collected 2-D plot images from a variety of resources and tested our extraction algorithms. Experimental evaluation has demonstrated that our method can produce results suitable for real world use.

1. Introduction

Two-dimensional (2-D) plots in digital documents, like line plots, histograms, scatter plots, etc., contain important information. Often, the results of scientific experiments, performance reports of businesses, and other quantitative information is summarized and visualized using two-dimensional plots. Although 2-D plots are easily understood by human users, current search engines rarely utilize the information contained in the plots to enhance the results returned in response to queries posed by end-users.

There exist semi-automatic tools for extracting data from 2-D plots. These tools may provide interactive interfaces for users to input requests and update results of automated extraction. For instance, users can select a data point or a line and the tools will generate a list of corresponding data values. The semi-automatic tools assist users to process individual plots. However, the amount of human interaction necessary for every single plot makes it unusable for processing a large volume of figures in a digital library.

Users can analyze the data easily with the aid of scalable automated data extraction tools. The data generated by these tools may serve as raw data for a variety of analyses, e.g., curve fitting of the raw data, analysis of published models of scientific phenomena, interpretation, and predictions.

The goal of this work is to extract data values from line curves within 2-D plots automatically. Specifically, for every single line curve in a plot, data points representing the line curve are extracted and stored in a two-dimensional data array. Automatic data extraction from 2-D plots is challenging due to diversity among figures, various imaging conditions, and the presence of noise. Furthermore, to extract a line in a figure accurately, the tool must determine the continuation of each line beyond cross-over points (i.e., points where two lines intersect).

Recognition of special types of graphics, especially engineering graphs, has attracted a lot of attention. Terrades et al. [12] presented a random transform based method to represent symbol graphics. Henderson et al. [5] proposed a structural modeling approach with a nondeterministic agent system to interpret scanned CAD drawings. Blostein et al. [1] surveyed a selection of techniques for diagram recognition. Lank et al. [6] proposed an interactive diagram recognition method. Luo et al. [8] proposed a framework for engineering drawings recognition using a case-based approach. Lu et al. [7] developed a figure categorization system. Zhou et al. [13] proposed a hough transform based technique for chart recognition. Based on our survey of the field, the problem of extracting data from lines in 2-D plots has not been addressed.

2. The Method

We show our system architecture in Figure 1 and describe the algorithm in this section. Let us first define a solid line curve for the purposes of our work. A *solid line curve* is defined as a special type of line composite with the following requirements:

- All components in a single curve are connected.
- For any pair of connected line segments in a curve, the left end point of a line segment should be connected with the right end point of the other line segment.
- Any line segment should belong to at most one curve except the intersection line segments which are shared

among several curves.

- A curve is as smooth as possible. At any intersection point or area, a curve continues in the direction that minimizes the change of derivative at the intersection.
- A curve has the maximum possible length as long as it fulfills the above requirements.

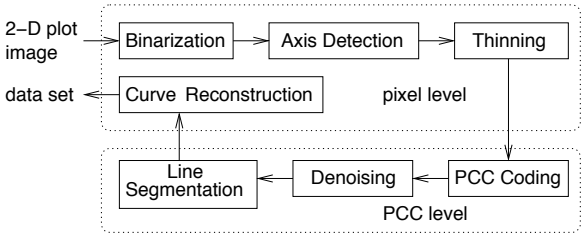
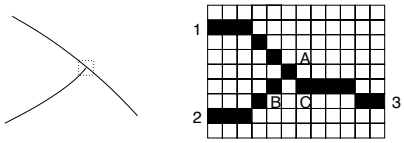
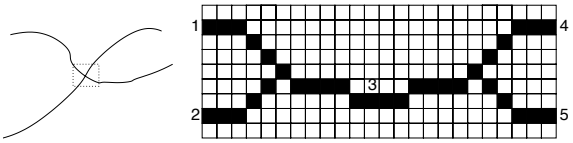


Figure 1. The data extraction process.



(a) connectivity between line segments



(b) intersection between two curves

Figure 2. Connectivity and intersection of curves.

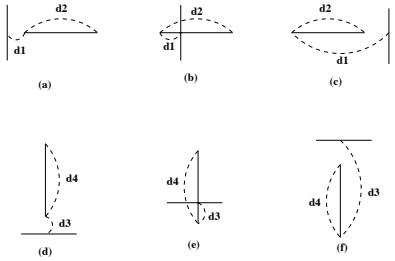
Based on the second requirement, we determine that the line segments 1 and 2 in Figure 2(a) are not on the same curve, because the two right end points are connected. According to the forth requirement, line segments 1 and 5 in Figure 2(b) should belong to the same curve, while line segments 2 and 4 should belong to the other curve. Because of the first requirement, and the fact that an intersection is a common part of involved curves, the intersection segment should be included in both curves (1-3-5 and 2-3-4) to maintain connectivity.

At the begining of the system, 2-D plot images are binarized, since we do not need any color information to detect lines. We use a global thresholding method [11] because the background is usually uniform and distinct in intensity range from the foreground. From the intensity histogram, we choose the top 10% of the points as the foreground and the rest of the points as the background.

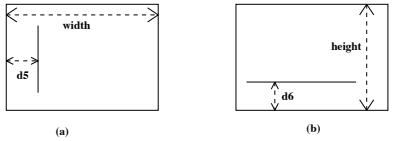
2.1. Axis Detection

First, candidate axis lines in 2-D plots are detected by a customized Hough transform algorithm. In the second step, axis lines are selected by sifting through candidate lines using a set of rules specifying orientation, relative position and length of lines. The algorithm only considers horizontal and vertical lines as candidates for the axes. The customized Hough transform share the basic ideas with the original Hough transform: parameter space construction and voting-based technique. While the original Hough transform [2] detects potential lines in any direction, the customized Hough transform only detects potential lines in horizontal and vertical directions. To tolerate slight skewness of images, lines in near horizontal and vertical directions are also detected. Compared with the original Hough transform, advantages of the customized Hough transform include that it reduces the need to check directions of potential lines in later steps; the computation and memory cost decrease significantly.

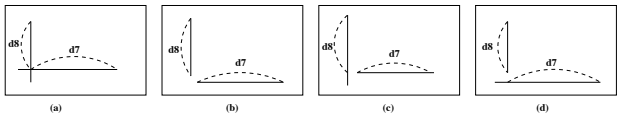
Axis lines are selected by sifting through candidate lines using a set of rules. The rules are designed to specify three categories of features of axis lines: relative position between two lines, relative position of lines in the figure, and relative length of lines. These rules represent visual characteristics of axis lines in 2-D plots.



(1) relative position between candidate axis lines



(2) relative position of candidate axis lines



(3) relative length of candidate axis lines

Figure 3. Detection of axes.

Relative position between a pair of candidate lines: As shown in Figure 3(1), two pairs of features, $\{d1, d2\}$ and $\{d3, d4\}$, define the relative position between two lines. $d1$ represents the horizontal difference between the vertical line and the left end point of the horizontal line; $d2$ represents the length of the horizontal line. $d3$ represents the vertical difference between the horizontal line and the lower end point of the vertical line; $d4$ represents the length

of the vertical line.

Relative position of a candidate line: As shown in Figure 3(2), for a vertical line, $d5$ is the distance between the vertical edge of the figure and the vertical line, and $\frac{d5}{width}$ defines its relative position within a 2-D plot. Similarly, for a horizontal line, $d6$ is the vertical distance between the horizontal line and the bottom of the 2-D plot, and $\frac{d6}{height}$ defines its relative position within a 2-D plot.

Relative length of a candidate line: As shown in Figure 3(3), for a horizontal line, $\frac{d7}{width}$ represents its length relative to the width of the 2-D plot. Similarly, for a vertical line, $\frac{d8}{height}$ represents its length relative to the length of the 2-D plot.

After axis lines in 2-D plots have been detected, the upper right region bounded by axis lines are determined as the data point region.

2.2. Thinning, Chain Coding, Denoising

We adopt the image thinning algorithm proposed by O'Gorman [9], which peels the boundaries of foreground regions, one layer of foreground pixels at a time, until the regions have been reduced to thin lines.

After the data region of 2-D plot images have been thinned, objects of interest in the image (thin lines) are represented by the chain coding instead of the raw image format. Instead of storing the "ON" and "OFF" values for every pixel, the chain coding lists the "ON"-valued pixels along lines and stores the direction from every pixel to the next pixel in its neighborhood.

We choose the primitive chain code (PCC) [10], an extension of the popular Freeman [3] chain code. The primitive chain code is designed to preserve connection, branching, and junction topology. PCC codes contain pixel chain codes and feature codes, representing pixel connections and junction or end-point features.

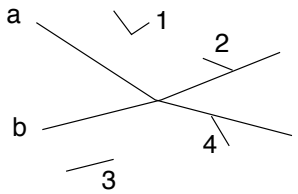


Figure 4. Various types of noise on PCC representation level.

Isolated pixels and spurious pixels branching from lines correspond to different combination of chain codes. As shown in Figure 4, Four noise segments, numbered 1, 2, 3, and 4, belong to different types on PCC level. Noise 1 and 3, isolated pixels, correspond to short "End" to "End" line segments, while noise 2 and 4, spurious lines branching from longer lines, are short "End" to "Feature" line segments. An effective noise reduction method is to set threshold length values for different types of line segments and filter out segments shorter than the corresponding threshold length.

2.3. Curve Identification

Connectivity between line segments: Two line segments are connected if one end point from each of them resides in each other's neighborhood. Two pixels are neighbors if both X-dimension and Y-dimension distances between them are equal or less than one. However, there are cases that two end points must be treated as connected even though they are not in each other's neighborhood. Figure 2(a) shows the cross-over area of a 2-D plot, where there are three line segments, numbered 1, 2, and 3. Three special points are labeled as A, B, and C. A is the right end point of line segment 1, B is the right end point of line segment 2, and C is the left end point of line segment 3. A is connected to both B and C. Even though B and C are not in each other's neighborhood, they should be treated as connected since they both are connected to a common point. Figure 2(a) illustrates the case that two end points are connected via another "bridge" point. It is also possible that two end points are connected via multiple hops of "bridge" points.

Intersection between curves: An intersection between multiple curves may appear as a single point or a collection of points (constituting a single line segment or multiple line segments). In Figure 2(b), we see that two curves intersect. The pixel-level illustration of the intersection area shows that the intersection appears as a line segment, namely line segment 3.

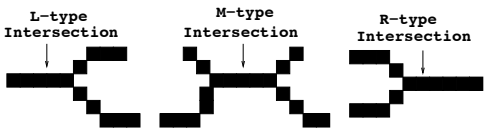


Figure 5. Three categories of intersection segments.

Based on manual check of about five hundred 2-D plots from the CiteSeer [4] digital library and from the Internet, it is apparent that even though intersections are common in plots, it is not common to have more than two curves intersect at the exact same point. Thus, in our work, we assume that any intersection segment is shared by two curves, which is reasonable and simplifies the problem. The handling of more general case is beyond the scope of this work. Based on this assumption, we have designed our algorithm to identify three types of intersection segments, as illustrated in Figure 5, namely L-type, R-type and M-type intersection segments.

Curve identification: The process of constructing curves from line segments is illustrated in the following.

1. Line segments are sorted from left to right based on the left end point. If there is a tie among multiple line segments, they can be ordered in any way.
2. Start a new curve from the first line segment in the sorted list which has valid visiting ticket.
3. Check all right connections of the current line segment.
4. If no right connections with valid visiting ticket can be found, end the current curve and go to step 2. Otherwise, choose the right connection which has the minimal change in derivative; visit it and set it as the current line segment; then go to step3.

There are two issues about the intersection segment. First, since an intersection segment is shared by two curves, it will be included in both curves. To address this, all categories of intersection segments are detected first and two visiting tickets are assigned that allow them being visited twice during curve identification. Second, for any M-type intersection segment, the selection of the line segment to the right of the M-type intersection segment should be dependent on the line segment to the left of the intersection which has been visited earlier.

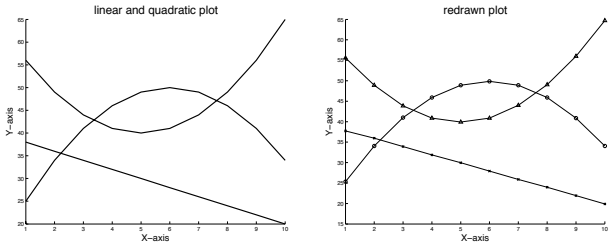
3. Experiments

We used three datasets from diverse sources: the CiteSeer [4] scientific literature digital library, the Internet, and a third set generated using MATLAB plotting tools.

MATLAB plots: We use MATLAB plotting tools to generate a set of 2-D plots. Comparisons can be made between the original data and the extracted data from 2-D plots. The data set contains random linear and quadratic plots and combinations of them. We use "L", "Q", "LL", "LQ", "QQ", "LLQ", and "LQQ" to represent seven categories of plots containing a single linear curve, a single quadratic plot, two linear curves, one linear curve and one quadratic curve, two quadratic curves, two linear curves and one quadratic curve, one linear curve and two quadratic curves respectively.

Table 1. Correspondence between original MATLAB curves and extracted curves.

Category	# Curves	# Matched	Match Rate
L	15	15	100%
Q	12	12	100%
LL	30	30	100%
LQ	24	20	83%
QQ	24	22	92%
LLQ	36	27	75%
LQQ	36	27	75%
Average	177	153	86%



(a) (b)

Figure 6. An original 2-D plot and a redrawn plot based on the extracted data sets.

Figure 6 shows an example plot with the redrawn plot based on the extracted data. We show the match results in Table 1. For any plot, the experimental system first compares the number of curves in the original plot and the number of extracted curves. If they do not match, the plot is regarded as a failure.

For all the plots where the number of original curves and extracted curves match, the correctness of extracted data are measured. In order to make the performance result consistent, the X dimension (independent variable) and Y dimension (dependent variable) values are normalized. The range of X dimension values is 0-10, and the range of Y dimension values is 0-100. For every sampled X dimension value, the distance between Y dimension value of the original curve and that of the extracted curve are recorded. Finally, the mean squared error is calculated for every pair of matched curve. In Table 2, MSE between original curves and extracted curves are illustrated for each category of plots.

Table 2. Mean squared error between original MATLAB plots and redrawn plots.

Category	# Matched	MSE
L	15	0.0662
Q	12	0.1408
LL	30	1.1086
LQ	20	0.1609
QQ	22	0.4122
LLQ	27	0.6215
LQQ	27	4.7180
Average	153	1.2575

There are many factors which affect the correctness of data extraction: image preprocessing, thinning, curve identificaiton, etc. Among them, the curve tracing at cross-points is the most influential factor. We see a higher MSE for data extraction in "LQQ" plots since there are more cross-points in those plots wihch means higher chance of incorrect tracing.

Plots from CiteSeer: We randomly selected about 2000 documents from the CiteSeer database randomly and extracted the 2-D plot figures from the documents manually. The experimental system redrew a plot based on extracted data set for every 2-D plot. Then, we manually compare every original plot with the redrawn plot and record the matched plots. If all curves in an original plot match all curves in the redrawn plot, then we treat the plot as a success case. Otherwise, the plot is regarded as a failure case. The degree of match between original 2-D plots and redrawn 2-D plots is illustrated in Table 3.

Table 3. Correspondence between original 2-D plots from the CiteSeer and redrawn plots.

# Plots	# Matched	Match Rate
77	48	62%

Plots from the Internet: Since Google image search engine is based on keyword search, we have tried combinations of keywords. Keywords have been tried include "curve", "plot", "2-D plot", "curve plot", etc, among which the "curve plot" search gives back the maximum number 2-D plot images.

Table 4. Correspondence between original 2-D plots from the Internet and redrawn plots.

# Plots	# Matched	Match Rate
40	29	72.5%

The test process for Internet plots is similar to that of the CiteSeer plots. In Table 4, the degree of match between the original plots and redrawn plots are illustrated.

We investigated the experimental results about CiteSeer plots and Internet plots, and found that higher error rate occurs in CiteSeer plots due to various reasons: scanned documents, skewness of document, poor image quality, etc. resolving which is beyond the scope of this work. In summary, since the tracing of a curve relies on connectivity, the method has difficulty in tolerating broken curves. Besides, random noises in poor quality images present another challenge, because thinned lines from noises may accidentally be traced as curves. For the second issue, post analysis of extracted curves may alleviate the problem in many cases.

4. Conclusion and Future Work

We have proposed a method for automatic extraction of lines from 2-D plots, a widely-used type of graphics in documents. An experimental system has been developed, and extensive experiments have been conducted. The experimental results demonstrate that the proposed method can be applied to real world use. In the future, we plan to work on making this algorithm more robust to noisy figures and multiple intersecting lines, and extend this work to perform automatic analysis of more categories of graphics within documents.

5. Acknowledgements

This work was supported in part by the US National Science Foundation under grants 0535656, 0347148, 0454052, and 0202007, Microsoft Research, and the Internet Archive.

References

[1] D. Blostein, E. Lank, and R. Zanibbi. Treatment of diagrams in document image analysis. In *Proceedings of the International Conference on Theory and Application of Diagrams*, pages 330–344, 2000.

[2] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of ACM*, 15(1):11–15, 1972.

[3] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, March 1974.

[4] C. L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the ACM Conference on Digital Libraries*, pages 89–98, 1998.

[5] T. Henderson and L. Swaminatha. Symbolic pruning in a structural approach to engineering drawing analysis. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 180–184, 2003.

[6] E. Lank. A retargetable framework for interactive diagram recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 185–189, 2003.

[7] X. Lu, P. Mitra, J. Z. Wang, and C. L. Giles. Automatic categorization of figures in scientific documents. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, pages 129–138, 2006.

[8] Y. Luo and L. Wenyin. Engineering drawings recognition using a case-based approach. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 190–194, 2003.

[9] L. O'Gorman. $k \times k$ thinning. *Computer Vision, Graphics, and Image Processing*, 51(2):195–215, August 1990.

[10] M. Seul, L. O'Gorman, and M. J. Sammon. *Practical Algorithms for Image Analysis*. Cambridge University Press, 2000.

[11] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.

[12] O. R. Terrades and E. Valveny. Radon transform for lineal symbol representation. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 195–199, 2003.

[13] Y. P. Zhou and C. L. Tan. Hough technique for bar charts detection and recognition indocument images. In *Proceedings of the International Conference on Image Processing*, pages 605–608, 2000.