TJHSST Computer Systems Lab Senior Research Project
Reverse Engineering Graphs: Obtaining Data Points from Scatterplots
2008-2009

Maya Wei

April 1, 2009

Abstract

Various programs exist to take data points and use them to render a graph. However, once the data are put into visual form, there is a loss of numerical information if the original data cannot be obtained. This project seeks to take data from a graph; in essence, the purpose is to reverse engineer a given graph. This will provide for a set of data points which can be used for various other numerical purposes, not simply the graph form in which they are presented.

Keywords: image analysis, scatter plots, statistical graphs

1 Introduction

1.1 Scope of Study

A scatterplot is a visual representation of bivariate data. However, once the data are in this visual form, there is minimal further testing one can do. If only given a graph, there is no way to perform statistical tests on dots laid along an image. By being able to reverse-engineer a graph to be able to look at a graph and be able to calculate the what point the coordinate represents the computers capabilities could simulate the human mind with more efficiency.

In trying to read a graph, various image analysis techniques will be used: edge analysis, and from that, shape recognition – or at the very least, shape differentiation. The intention is to develop a method which will

be capable of successfully reading a point's location to its axes regardless of point shape: various graph-creating programs (OpenOffice Calculate, Microsoft Excel) use different formatting with different colored backgrounds, different guidelines, and different shapes of points.

The results expected are an accurate recreation of the points which were utilized to create the graph. With these results, it would be possible to represent the data in other graphs, make statistical calculations, etc.

2 Background and review of current literature and research

The field of image analysis and computer vision is highly advanced at this point in time. There have been many papers written on shape identification, image recognition, and graphic rendering. While there is no project similar to what is being done here with graphs and data points, undoubtedly somebody has sought to do it before. Image analysis is the focus of a generally 2D surface; it deals with issues such as connectivity.

Much progress has been made in the field of document image analysis; the result is what we can see on programs such as Google Books. By first being able to separate specific letters through pixel analysis – finding "connected pixels" in order to read letters and ultimately words. Using image analysis, it is thereby possible to digitally recontruct a hard copy of a book; the copy would be searchable, and would recognize images within the hard copy as well. Reading off of the graph is similar in that it finds connected points and attempts to identify them from there.

3 Procedures and Methodology

This project utilizes Java. In the program, there is utilization of an edge recognition method, a method to determine connectivity, and a simple recognition method, all of which shall be explicated below.

The input data is to be found in the form of a graph in a png file. The image being used has been generated by OpenOffice Calc, displaying a graph with points at (1, 1), (2, 2), (3, 3), (4, 4), and (5, 5). The graph is minimalist; it has no background color, no guidelines, nothing that could possible throw off the basic read. Various graph inputs – square points, triangle points, circle points – will be tested to as to determine the accuracy of the program at later dates.

3.1 Edge Detection

Currently, the program has a very primitive form of edge detection. It takes the color of the point at (0, 0) and makes the assumption that the points color is also the background color. It then looks at each individual point

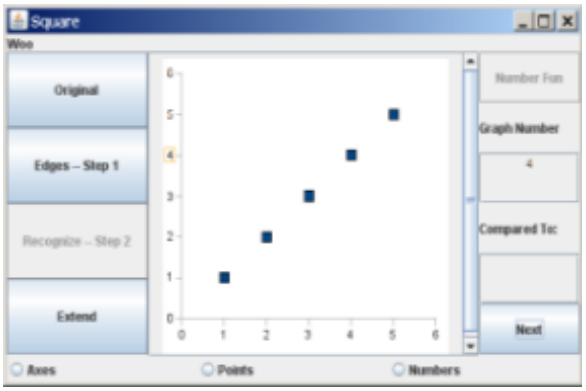

Figure 1: Standard graph used throughout. The 3rd Quarter GUI is also displayed, incorporating the number analysis columns on the right.

(x, y) and the points neighbors (x+1, y), (x 1, y), (x, y + 1), (x, y 1), and also searches the diagonal neighbors. If any of the points neighbors are the same color as (0, 0), then that means that (x, y) is touching an edge. Thus, if the edge check does not return that anything is touching an edge, the point is unimportant; otherwise, the point is stored accordingly.

Through doing this, we can obtain "edges" to be highlighted. All these edges are stored within an ArrayList, edgeList, which is used in future methods.

3.2 Connected Points and Recognition

Having found the edges, a method was written to group "connected points" together. Given a point (x, y), "connected points" are defined as the immediate neighbors – (x+1, y), (x-1, y), (x, y+1), (x, y-1). Pixel points are stored in int[]; the connected pixel points are stored with in ArrayList¡int[]¿. Connected points uses an iterative method to search the list of edges for the its neighbors, removing that point from "available edges" once it's found.

Given that the "connected points" are now together, an effort is made to differentiate the points themselves. Do the connected points form an axis, a number along the scale, or a graph point? Currently, the axis is recognized as the longest collection of "connected points"; those connected points located to the right of the x axis are points, whereas to the left of x and below the y, the connected points are numbers. The axis is somewhat burdened by having tickmarks, which are connected to the axis but don't have a use in identification. The program has been written to ignore these axis tickmarks using the longest straight line.

With the graph points having been determined, it is simple enough to find the middle of the point (because a point is connected to itself on all sides, we can at least be ensured that the average will be in the center of such a point). This middle point allows for determining where the graph point is in relation to the axis. On the GUI, there is an "extend" button so as to display this relation.

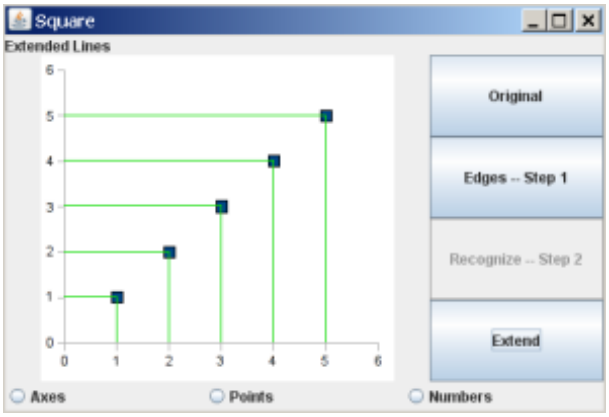

Figure 2: Extend; green lines extend from the centers of points.

3.3 Image Identification

The second portion of the program was made in intent to analyze the specific number values on the scale of the axes. Seeing as the computer cannot immediately by itself recognize a system of pixels to represent a quantitative figure, it is necessary to make a form of image analysis in itself.

The system works as follows: given the scale, the program will find what the numerical value of the code or at least, what the program projects it to be based off a certain base system. In order to compensate for error, the numerical differences between each tick mark on the scale will be calculate. The ultimate goal is to merely find the scale (pixels per one unit), not necessarily to correctly identify all the numbers along the scale.

4 Analysis

All analysis is done visually; the GUI will correctly display whatever is highlighted. It is the simplest way to determine whether or not the code is working; if the incorrect items are highlighted, then it is cause for going back and changing the code.

The GUI itself focuses on demonstrating what can be done with the project. It can display the edges, the "extended" image, and can also identify the axes / numbers / graph points. Radio buttons allow for the viewing of the aforementioned objects.

When another graph, created by Microsoft Excel, was used as a new input, the results yielded for the current code proved to be fairly consistent with what was desired; the identify and extend methods worked effectively. However, there was a slight error in the orders in which the numbers along the scale were read. Because the method places the subimages of numbers into the array according x-coordinates and then y-coordinates, it is generally assumed that the generated graphs all have the same minimum x-value.

The methods for which to compare numbers are fairly primitive in nature; they are one-to-one pixel comparisons. Currently, the method is unable to shift itself; this way, if there were a horizontal line at, say, $y = 4$, but the original image has a horizontal line at $y = 3$, there is zero match. This will be change in future versions of the project.

5 Expected Results

The desired result of this program is to find data points from a given graph. The returned result will be an array returning the locations of the points.

While application of this field is slim, the various image analysis techniques could be utilized. One could also expand upon the concept to be able to read bar graphs, pie charts, line graphs, etc.

6 Bibliography

Faure and Vincent. Document Image Analysis For Active I
International Conference Proceeding Series; Vol. 259

Igathinathine, Pordesimo, Columbus, Batchelor, and Methu
identification and particles size distribution from bas
using ImageJ. Computer and Electronics and Agricu