Chart Detection and Recognition in Graphics Intensive Business Documents

by

Jeremy Svendsen
B.Sc., University of Victoria, 2007
M.A.Sc., University of Victoria, 2009

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

© Jeremy Svendsen, 2015
University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

Chart Detection and Recognition in Graphics Intensive Business Documents

by

Jeremy Svendsen
B.Sc., University of Victoria, 2007
M.A.Sc., University of Victoria, 2009

Supervisory Committee

Dr. A. Branzan-Albu, Supervisor
(Department of Electrical and Computer Engineering)

Dr. P. Agathoklis, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. G. Tzanetakis, Outside Member
(Department of Computer Science)

Supervisory Committee

Dr. A. Branzan-Albu, Supervisor
(Department of Electrical and Computer Engineering)

Dr. P. Agathoklis, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. G. Tzanetakis, Outside Member
(Department of Computer Science)

ABSTRACT

Document image analysis involves the recognition and understanding of document images using computer vision techniques. The research described in this thesis relates to the recognition of graphical elements of a document image. More specifically, an approach for recognizing various types of charts as well as their components is presented. This research has many potential applications. For example, a user could redraw a chart in a different style or convert the chart to a table, without possessing the original information that was used to create the chart. Another application is the ability to find information, which is only presented in the chart, using a search engine.

A complete solution to chart image recognition and understanding is presented. The proposed algorithm extracts enough information such that the chart can be recreated. The method is a syntactic approach which uses mathematical grammars to recognize and classify every component of a chart. There are two grammars presented in this thesis, one which analyzes 2D and 3D pie charts and the other which analyzes 2D and 3D bar charts, as well as line charts. The pie chart grammar isolates each slice and its properties whereas the bar and line chart grammar recognizes the bars, indices, gridlines and polylines.

The method is evaluated in two ways. A qualitative approach redraws the chart for the user, and a semi-automated quantitative approach provides a complete analysis of the accuracy of the proposed method. The qualitative analysis allows the user to see exactly what has been classified correctly. The quantitative analysis gives more detailed information about the strengths and weaknesses of the proposed method. The results of the evaluation process show that the accuracy of the proposed methods for chart recognition is very high.

Contents

Supervisory Committee ii

Abstract iii

Table of Contents v

List of Tables viii

List of Figures x

Acknowledgements xviii

Dedication xix

1 Introduction 1

1.1 Motivation 1
1.2 A Brief History of Charts 2
1.3 Overview of the Graphics Recognition System 4
1.4 The Proposed System and Contributions 7
1.5 Outline of the Thesis 7

2 Related Work 9

2.1 Preprocessing 9
2.2 Segmentation 10
2.3 Primitive Detection 11
2.3.1 Text Detection 11
2.3.2 Curve and Line Detection 12
2.4 Page Classification 13
2.5 Table Classification 14
2.6 Chart Detection 14

2.6.1 Rule Based Approaches . . . 15
2.6.2 Learning Based Approaches . . . 16
2.6.3 Chart Detection Validation . . . 17
2.7 Chart Information Extraction . . . 18
2.7.1 Text/Graphics Association . . . 18

3 Background 20
3.1 Fundamental Properties of Charts . . . 20
3.2 Graphical Primitives of Charts . . . 21

4 Introduction to Formal Language Theory and Mathematical Grammars 25
4.1 Context-Free and Chomsky Normal Form . . . 28

5 Primitive Detection, Segmentation and Chart Detection 30
5.1 Segmentation . . . 30
5.1.1 Document Segmentation via Oblique Cuts . . . 30
5.2 Primitive Detection . . . 33
5.2.1 Solid Region Detection . . . 33
5.2.2 Text Detection . . . 34
5.2.3 Curve Detection . . . 35
5.3 Chart Detection . . . 40
5.3.1 Ellipse Detection . . . 40
5.3.2 Axis Detection . . . 41

6 Chart Grammars 44
6.1 Pie Chart Curve Classification . . . 44
6.1.1 The Classifications and Terminals . . . 45
6.1.2 The Nonterminals . . . 47
6.1.3 Substitution Rules . . . 50
6.1.4 Pie Chart Example Derivation . . . 55
6.1.5 Arc Labels and Pie Slice Colour . . . 60
6.2 Axis Charts Grammar . . . 60
6.2.1 Preprocessing . . . 63
6.2.2 Axis Chart Terminals . . . 66
6.2.3 The Start Nonterminal and the Initial Substitution Rule . . . 69

6.2.4 Substitution Rules for the Vertical Bars and Horizontal Indices 70
6.2.5 Substitution Rules for the Horizontal Bars and Vertical Indices 76
6.2.6 Grid Substitution Rules 80
6.2.7 The Bar Structure: B_x and B_y 104
6.2.8 Line Chart Polyline 107

7 Experimental Results 108
7.1 Experimental Database 108
7.2 Graphical User Interface 109
7.2.1 Report Window 109
7.2.2 Information Window 111
7.2.3 Redrawing the Pie Charts 112
7.2.4 Redrawing the Axis Charts 113
7.3 The Semi-Automatic Evaluation System 114
7.4 Precision, Recall, and F-Score 114
7.5 Experimental Evaluation for Recognition of Pie Charts 115
7.5.1 Slice Evaluation 115
7.5.2 Evaluation of the Curve Classification 117
7.6 Experimental Evaluation for Recognition of Axis Charts 118
7.6.1 Descriptions of the Chart Errors 118
7.6.2 The Results 122
7.6.3 Aggregating the Results 126

8 Conclusions 129

A Third Party Code 132
A.1 Tesseract 132
A.2 The Gabor Transform 132
A.3 OpenCV 133

B Input File 134

C Published Works 135

Glossary 137

Bibliography 140

List of Tables

Table 3.1 List of pie chart primitives 23
Table 3.2 List of bar/line chart primitives 24

Table 5.2 A list of the terminals used for curve detection. 37
Table 5.1 A list of the nonterminals used for curve detection. 38

Table 6.1 Testing A 47
Table 6.2 A list of the nonterminals used for pie chart curve classification. 49
Table 6.3 The angles between the curves. The first column indicates the initial curves and each row indicates the angles for each possible nonterminal relative to the initial curve. Note that many curves never intersect. The angles are in degrees. 51
Table 6.4 Process flow example. 57
Table 6.5 The terminals for the vertical and horizontal indices. 66
Table 6.6 The terminals for the axis and the gridlines. 67
Table 6.7 A list of the axis chart terminals for the bar structures and the nonterminals which are associated with them. 68
Table 6.8 The nonterminals which appear in the initial substitution rule. . 69
Table 6.9 A list of the axis chart nonterminals on the vertical subset. . . . 70
Table 6.10 A list of the axis chart nonterminals on the horizontal subset. . 76
Table 6.11 A list of the axis chart nonterminals for the grid structure. . . . 82
Table 6.12 A list of the axis chart nonterminals for the 3D bar structures. . 104

Table 7.1 The number of charts evaluated. 109
Table 7.2 Some of the information displayed in the information window. . 112
Table 7.3 The number of 2D and 3D charts evaluated. 115
Table 7.4 Results for the Pie Slices 117
Table 7.5 Results for the curves of pie charts. 117
Table 7.6 A list of the chart components evaluated. 118

Table 7.7 Results for 2D vertical bar charts. 122
Table 7.8 Results for 2D horizontal bar charts. 123
Table 7.9 Results for 3D vertical bar charts. 124
Table 7.10 Results for 3D horizontal bar charts 124
Table 7.11 Results for line charts. 125
Table 7.12 Results for 2D bar charts. 126
Table 7.13 Results for 3D bar charts. 127
Table 7.14 Results for vertical bar charts. 127
Table 7.15 Results for horizontal bar charts 128
Table 7.16 Results for all bar charts 128

List of Figures

1.1 The first time series chart. It was first published in 1765 by Joseph Priestley. This image is in the public domain. 2
1.2 The first pie chart ever created. Made by William Playfair in 1801. This image is in the public domain. 3
1.3 The first bar chart ever created. Made by William Playfair in 1786. This image is in the public domain. 3
1.4 Charles Joseph Minard's map which included pie charts. This image is in the public domain. 4
1.5 Nicole Oresme bar chart. This image is in the public domain. . . 5
1.6 The sequence of steps for the analysis of charts in a digital document. 6

3.1 a) An example of a 2D pie chart. b) An example of a 3D pie chart. 21
3.2 The two different orientations for a 3D bar chart. a) Vertical bar chart. b) Horizontal bar chart. 21
3.3 An example of a chart which contains multiple polylines. 22
3.4 Labeled primitives of a 3D pie chart. 22
3.5 Labeled primitives of a 3D bar chart. 23
3.6 Labeled primitives of a line chart. 24

4.1 An example of a parse tree. 27

5.1 Bar chart with horizontal axis labels which can be only segmented using an oblique white space search. 31
5.2 Iterative segmentation process. Iteration outputs are colour coded (0-red; 1-blue; 2-green; 3-brown). 33
5.3 Segmentation ranges for blocks with w $\leq$ h (left) and w $\geq$ h (right). 33
5.4 The primitives of a 3D bar chart. The numerical data is detected with the Gabor filter. It is best if this figure is viewed in colour. . 34

5.5 The curve on the left is not completely thinned for 8-connected adjacency as a pixel can still be removed without breaking the adjacency. The curve on the right is completely thinned as no additional pixels can be removed without breaking the adjacency. 35
5.6 a) The output of the PCC algorithm, the entire skeleton is considered one curve. b) The output from the proposed algorithm, the skeleton is split into 3 different curves. The black, red, and green colours show the different curves. The pixel at the intersection point is common to all three curves. 36
5.7 Examples of the 5 different types of connected points. The seed point is in green. a) No connections (P), b) One connection, curve end (E), c) Two connections, curve middle (C), d) Three connections, fork (F), e) Four connections, cross (C). It is best if this figure is viewed in colour. 37
5.8 This figure is used to help describe the rules. 39
5.9 The process flow for detecting pie charts. 41
5.10 The steps used to find the edge map for the same chart seen in figure 5.11. a) The binarized version of the pie chart. b) The binarized image after the morphological close operation. c) The edge map calculated using the Canny edge detection on b). The colours are inverted for the purpose of printing the document. . . 42
5.11 An example of an ellipse fitted over a pie chart. It is best if this figure is viewed in colour. 42
5.12 An example of the vertical and horizontal projection profile for a chart. The original chart is seen in the top right. The horizontal projection profile is visible in the left image. Each width of each row of pixels indicates the number of non-white pixels with the same y value. The vertical projection is seen at the bottom. The height of each column of pixels indicates the value of that column. The profiles are aligned to the original chart. 43
5.13 An example of a bounding box placed where the axis was detected. In this case the x-axis was along the bottom and the y-axis was along the left side. It is best if this figure is viewed in colour. . . . 43

6.1 Part of a chart which contains gridlines which are behind the chart bars. 45
6.2 An example of a pie slice and a pie chart. Classification is colour-coded as follows: Black: radius; Blue: top edge (in 3D), edge (in 2D), Red: bottom edge (in 3D, not used in 2D), Turquoise: side curves (in 3D, not used in 2D), Green: label connection. 46
6.3 Three connecting curves. 46
6.4 Left: The curves and their corresponding terminals. Right: A list of the terminal symbols and their names. Note that the pattern terminal and the label terminal are not included in the lefthand figure because they are not curves. It is best if this figure is viewed in colour. 47
6.5 Three connecting curves. 48
6.6 The curves and their associated nonterminals for a 3D pie slice. The directions are shown by arrows. Note that the rightmost R_0, the top R_1, and the Z_0 nonterminals correspond to an adjacent pie slice. They show that a curve belonging to two slices is associated with two nonterminals. 48
6.7 The location of the rules for T_0. 52
6.8 The location of the rules for T_1. 53
6.9 The location of the rules for B_0. 53
6.10 The location of the rules for B_1. 53
6.11 The location of the rules for Z_0 and Z_1. 54
6.12 The order in which the curves are classified and linked to the pie slice. It is best if this figure is viewed in colour. 56
6.13 This tree presents the order in which the substitution rules are applied. The colour's of the nonterminal nodes are the same colour as the curves they classify in figure 6.12. It is best if this figure is viewed in colour. 56
6.14 A zoomed in section of a pie chart. The red arrow above the label connection curve indicates the direction the algorithm searches for the pie slice label. The pie slice label is within the blue bounding box. The short red line indicates the imaginary line which is followed to find the text block. It is best if this figure is viewed in colour. 60

6.15 The primitives of a 3D bar chart. It is best if this figure is viewed in colour. 61
6.16 A bar chart with textual primitives that fulfill a dual role (horizontal index and vertical bar text). The word "Finland" is only a horizontal index because there is no bar above it. 62
6.17 A zoomed-in view of two bars occluding and the horizontal axis. This figure is from the same chart as figure 6.19. This figure should be viewed in colour. 63
6.18 The process for detecting the curves and lines. 64
6.19 The chart without the bars but with the axis intact. 64
6.20 The preprocessing steps for finding the polyline curves. It is assumed that there are N distinct hues in the chart image. 65
6.21 An example of a chart with a non-white background. 66
6.22 This parse tree represents the start of the axis chart grammar. The V, H, G, and P initiate four disjoint subsets which classify different components. 69
6.23 A visual representation of the substitution rules for the V nonterminal. 71
6.24 A visual representation of the substitution rules for the B_V nonterminal. 72
6.25 This chart is used to show the order in which the bars are classified. 73
6.26 A visual representation of the substitution rules for the B_{VL} and B_{VR} nonterminals. The B_{VL} nonterminal indicates search to the left and the B_{VR} nonterminal indicates a search to the right. . . . 73
6.27 The chart analyzed in the example derivation. 74
6.28 This is the subset for the T_{HI} nonterminal. This tree detects the horizontal indices. These rules are continued in figure 6.29. 75
6.29 This is the subset for the T_{HIR} and T_{HIL} nonterminals. These trees are associated with the trees in figure 6.28. 75
6.30 A visual representation of the substitution rules for the H nonterminal. 77
6.31 These trees show the possible subsets for the horizontal bar charts. 77
6.32 This chart is used in the example derivation. 78
6.33 This tree detects the remaining bars of a horizontal bar chart. . . 79
6.34 This chart is used in the derivation example. 80

6.35 This is the subset for the T_{VI} rule. This tree detects the vertical indices. These rules are continued in figure 6.36. 80
6.36 This is the subset for the T_{VIU} and T_{VI} nonterminals. These trees are associated with the trees in figure 6.35. 81
6.37 The two different Z-Axis gridlines can be seen here. Chart a) contains several gridlines in the yz plane. Chart b) contains gridlines in the xz plane. 81
6.38 An example of a curve intersection. 81
6.39 This is the top level of the subset for the axis and the gridlines. . 83
6.40 The directions for the X_0 and X_1 nonterminals. This is relative to the initial curve. 84
6.41 These parse trees show the rules for detecting the x-axis curves. . 85
6.42 These parse trees show the rules for detecting the XZ gridlines. . 85
6.43 The angles for the vertical gridline substitution rules. The brown arrow indicates the initial curve. In the left two images, the initial curve is represented by the X_0 nonterminal, and in the right two images, the initial curve is represented by the X_1 nonterminal. The red, blue and green arcs represent the angle for the same nonterminal as the initial curve segment, the vertical gridlines, and the XZ gridline respectively. 86
6.44 The six different configurations for the x-axis substitution rules. Each configuration corresponds to a different substitution rule. . . 87
6.45 The directions for the G_{V0} and G_{V1} nonterminals. This is relative to the initial curve. 87
6.46 These parse trees show the rules for detecting the vertical gridlines. 88
6.47 The angles for the substitution rules. The brown arrow indicates the initial curve. For the left figure, the initial curve is represented by the G_{V0} nonterminal and in the right figure, the initial curve is represented by the G_{V1} nonterminal. The blue, green, and red arcs are for the horizontal gridline going right (G_{H1}), the vertical gridline (G_{Vi}), and the horizontal gridline going left (G_{H0}) respectively. 88
6.48 The directions for the G_{V0} and G_{V1} nonterminals. 89
6.49 These parse trees show the rules for detecting the z gridlines along the xz plane. 92

6.50 The angles for the substitution rules. The brown arrow indicates the initial curve. The initial curve is also represented by the G_{XZ} nonterminal. The blue, green, and red arcs are for the horizontal gridline going right (G_{H1}), the vertical gridline (G_{V1}), and the horizontal gridline going left (G_{H0}) respectively. 92
6.51 The directions for the G_{XZ} nonterminal. 93
6.52 The directions for the Y_0 and Y_1 nonterminals. This is relative to the initial curve. 94
6.53 These parse trees show the rules for detecting the y-axis curves. . 95
6.54 These parse trees show the rules for detecting the YZ gridlines from the y-axis. 95
6.55 The angles for the substitution rules. The brown arrow indicates the initial curve. The left two images the initial curve is the Y_0 nonterminal curve and the right two images the initial curve is the Y_1 nonterminal curve. The red, blue, and green arcs represent the angle for same nonterminal as the initial curve segment, the horizontal gridlines, and the YZ gridline. 96
6.56 The six different configurations for the y-axis substitution rules. Each configuration corresponds to a different substitution rule. . . 97
6.57 The directions for the G_{V0} and G_{V1} nonterminals. This is relative to the initial curve. 98
6.58 These parse trees show the rules for detecting the horizontal gridlines. 99
6.59 The angles for the substitution rules. The brown arrow indicates the initial curve. In the left figure, the initial curve is the G_{H0} nonterminal curve and in the right figure, the initial curve is the G_{H1} nonterminal curve. The blue, green, and red arcs are for the vertical gridline going down (G_{V0}), the continuation of the horizontal gridline (G_{Hi}), and the vertical gridline going up (G_{V1}). 99
6.60 The directions for the G_{H0} and G_{H1} nonterminals. 100
6.61 These parse trees show the rules for detecting the z gridlines along the zy-plane. 102
6.62 The angles for the substitution rules. The brown arrow indicates the initial curve. The blue, green, and red arc is for the vertical gridline going down (G_{V0}), the horizontal gridline (G_{H1}), and the vertical gridline going up (G_{V1}) respectively. 102

6.63 The directions for the G_{YZ} nonterminal. 103
6.64 The two possible orientations for the bars. The left chart has the side of the bars to the left and the right chart has the side of the bars to the right. 105

7.1 An example of what a segmented bar chart would look like if it was displayed by the GUI. a) The original chart image. b) Chart image showing the results of the segmentation. Blue rectangles indicate classified blocks, red rectangles indicate unclassified blocks, and green rectangles indicate charts. There are no red bounding boxes in this image. 110
7.2 The ellipse which was detected in the pie chart classification stage. 111
7.3 A bounding box showing the location and size of the detected axis. 111
7.4 An example of a redrawn pie slice (a) and a redrawn chart (d). Figures a) and c) are the original images for figures b) and d) respectively. The colours indicate the classification. Black curves are radii curves, blue curves are top edge curves, red curves are bottom edge curves, green curves are label connection curves, and turquoise curves are side curves. It is best if this figure is viewed in colour. 113
7.5 a) The original chart image. b) The detected and classified components drawn to screen. The curve colours indicate the classification. The green curves are the axis and the black curves are the gridlines. 113
7.6 The first part of a split slice (left). The second part of a split slice (middle). Two slices merged into one (right). The original chart image is presented in figure 7.7. It is best if this figure is viewed in colour. 116
7.7 a) The original chart image for figure 7.6. b) The extracted curves. It is best if this figure is viewed in colour. 116
7.8 The left image is the original chart image and the right image is an example of a chart with a split bar. The blue bar on the left side was drawn as two distinct bars. Note that only the chart bars are shown in the right image. None of the other components were drawn to make it easier to see the split. 119

7.9 An example of a component that was merged. The bounding boxes around the vertical indices show how they were grouped together during classification. The top 5 vertical indices were detected properly. The bottom 4 vertical indices however were merged into a single block. 120
7.10 The original chart image (left). Two vertical indices were not completely detected. These indices have the values of 1.0 and 1.5. Two vertical indices are also missing. The 0.0 and 4.0 index values were not recognized. The colours of the axis and gridlines are changed intentionally. 120
7.11 The XZ gridline of this chart was added to the x-axis. The original chart image is presented on the left. The right image shows the detected x and y axis. The red bounding box surrounds the XZ gridline which was added to the x-axis. 121
7.12 Left: The original chart image and the legend associated with it. Right: The redrawn version of the same chart image. Note that part of the legend was classified as a chart bar. This is an extra classification. Only the bars and the axis were redrawn. 121
7.13 An example of the numerical data superimposed on top of the bar. 123
7.14 An example of a chart with multiple polylines. 125

8.1 An example of a chart that would not work under the current system. Licensed under the creative commons (Zemanta). 130

ACKNOWLEDGEMENTS

I would like to thank:

My fiancee Paulina Alejandra León Ruiz, my family, my friends for supporting and encouraging me during my studies, and patience.

My supervisor Dr. Alexandra Branzan-Albu for mentoring and support.

SAP Research's Academic Research Center for providing me support during this research and providing the database.

It is hardware that makes a machine fast. It's software that makes a machine slow.
Craig Bruce

DEDICATION

I dedicate this thesis to my fiancee Paulina Alejandra León Ruiz.

Chapter 1

Introduction

1.1 Motivation

The use of digital documents to present information has skyrocketed in recent years. As the speed of computation, memory storage capabilities, and display quality have increased, programs which are used for document generation have significantly improved. This has also impacted the creation of charts within documents.

As the visual quality of digital documents increase, graphical elements such as charts gain in complexity, aesthetics, and representational power. These charts require more sophisticated computer vision techniques for chart detection, recognition, and analysis. The focus of this research is on the analysis of charts contained in graphics intensive documents created by business intelligence tools such as Crystal Reports[1].

This research is motivated by the practical applications as well as the opportunity to contribute to the emerging field of graphics-intensive, digital document analysis. The primary theoretical contribution proposed in this thesis involves the syntactic modeling of pie, bar and line charts in digital business documents.

The applications of chart recognition are numerous. For example, it can allow one to convert a chart into a useful digital format, without the program which created the chart. Another example would be to change the format of the chart. A user may be interested in converting the chart to a table or a different style of chart. A third application would be to get information from the chart for the purpose of searching. A user could search the information inside the chart in addition to the text.

The remainder of this chapter is structured as follows. Section 1.2 provides some

[1] http://www.crystalreports.com/

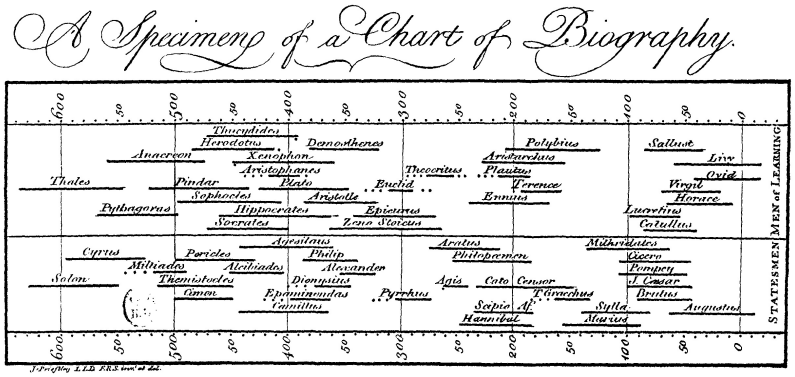


Figure 1.1: The first time series chart. It was first published in 1765 by Joseph Priestley. This image is in the public domain.

historical details about charts. Section 1.3 gives a high level overview of the proposed chart recognition system. Section 1.4 presents a short description of the purpose and contributions of this research. The chapter concludes with section 1.5 which outlines the remainder of the thesis.

1.2 A Brief History of Charts

Prior to the first bar or pie chart, a scientist named Joseph Priestley is credited with creating the first time series chart. This was first published in 1765 and is considered to be influential for the charts (see figure 1.1) which were developed later [44][2].

The inventions of the bar chart, line chart, and pie chart are attributed to William Playfair, an engineer and political economist, who is considered the founder of graphical methods for statistic[3]. The bar chart as well as the line chart first appear in his seminal work the *Commercial and Political Atlas*[41] which was published in 1786. The first pie chart appeared in his book *Statistical Breviary*[42] which was published in 1801. Figures 1.2 and 1.3 present the first pie chart and the first bar chart ever created respectively.

After Playfair introduced the pie chart, it was not used again until 1958 when the French engineer Charles Joseph Minard used it in a map[45]. The map represented the

[2]https://seeingcomplexity.wordpress.com/2011/02/03/a-short-visual-history-of-charts-and-graphs/

[3]https://en.wikipedia.org/wiki/William_Playfair

[4]http://www.jpowered.com/graphs-and-charts/pie-chart-history.htm

[5]https://en.wikipedia.org/wiki/Pie_chart

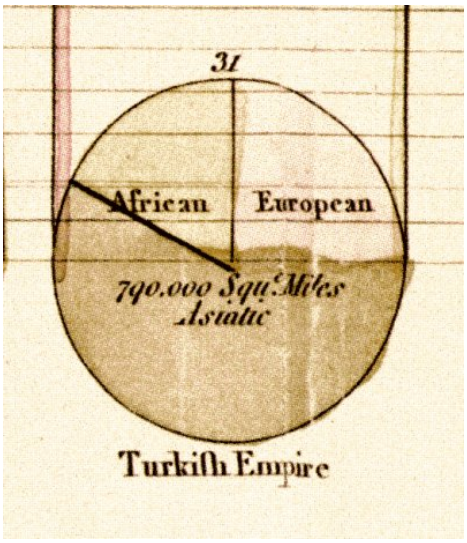


Figure 1.2: The first pie chart ever created. Made by William Playfair in 1801. This image is in the public domain.

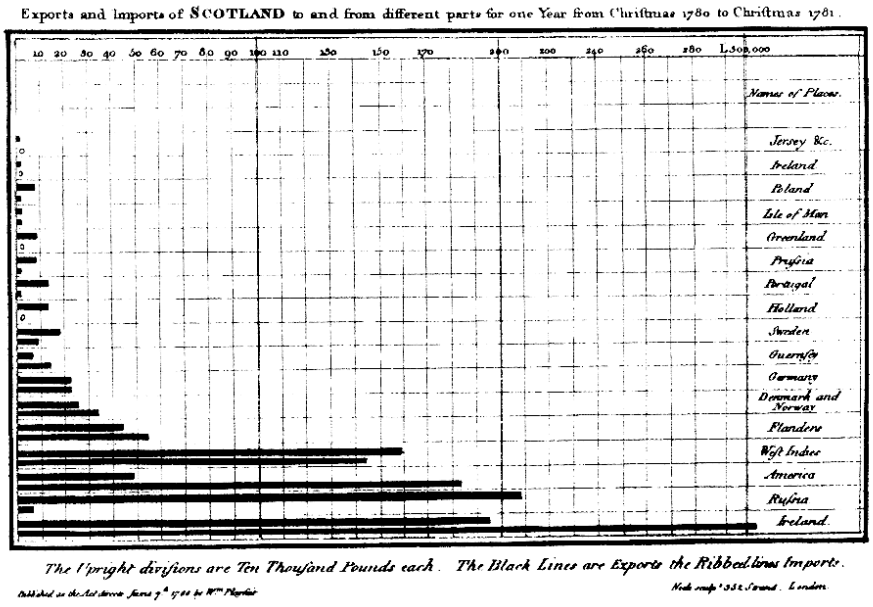


Figure 1.3: The first bar chart ever created. Made by William Playfair in 1786. This image is in the public domain.

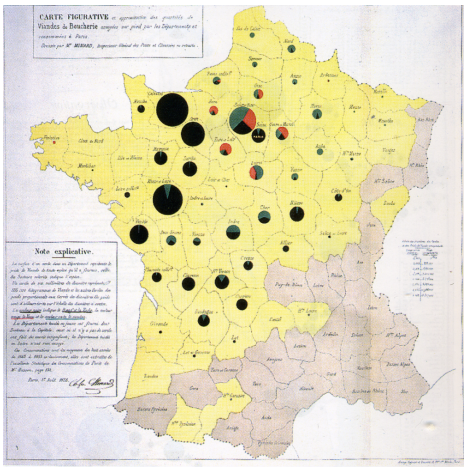


Figure 1.4: Charles Joseph Minard's map which included pie charts. This image is in the public domain.

locations in France where the cattle consumed in Paris originated. At this time, pie charts were known as "le camembert" because they resembled a wheel of camembert cheese[6]. This map can be seen in figure 1.4.

A different source[7] claims that the first bar chart was created by the Frenchman Nicole Oresme in the publication "The Latitude of Forms" in the 14th century. The chart is a scientific chart showing the relationship between velocity and time. This chart can be seen in figure 1.5.

1.3 Overview of the Graphics Recognition System

The goal of chart detection systems is to recognize and understand the charts which are present in a document image. The system starts with a document image and then determines the location as well as properties of the charts. Figure 1.6 shows the sequence of steps for the analysis of charts within a document.

The first step involves acquiring the raw images. In the proposed system, the images are created using the Crystal Reports software by SAP[8] and converted to raster images. The images in other systems are acquired using several different techniques, including scanning, digital photography, and and other digital methods. The experimental database used in this research is presented in more detail in section 7.1.

[6]http://www.nytimes.com/2012/04/22/magazine/who-made-that-pie-chart.html?_r=0
[7]http://www.jpowered.com/graphs-and-charts/bar-chart-history.htm
[8]http://www.crystalreports.com/

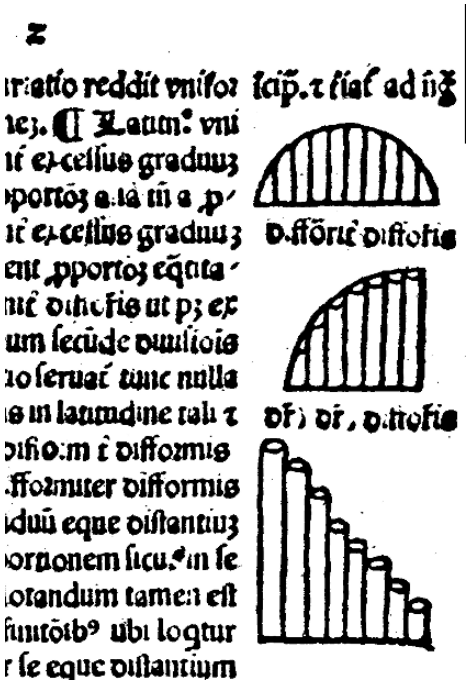


Figure 1.5: Nicole Oresme bar chart. This image is in the public domain.

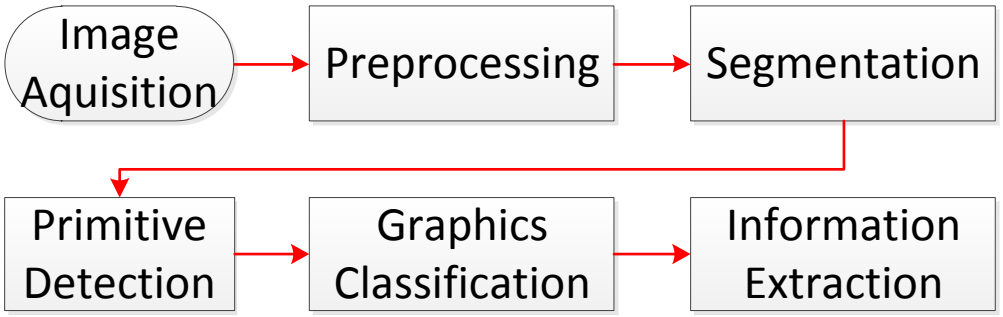


Figure 1.6: The sequence of steps for the analysis of charts in a digital document.

The second step concerns preprocessing. This step is critical for images which are obtained from a scanner or a camera, but not for digitally rendered documents. There are many noise related artifacts present in printed documents which are not seen in digitally created documents. A few preprocessing methods for removing the noise are discussed in section 2.1. Since the approach in this thesis uses digitally rendered documents there is no preprocessing step in this research. It is mentioned here since it is needed in other systems.

The third step involves segmenting the document into different regions or blocks. For example, if a page contains both a chart and a table, the segmentation process will separate the two different components. In this research, the page is segmented into blocks which can be as small as a single word. Previous approaches to segmentation are presented in section 2.2 and our approach is presented in section 5.1.

Primitive detection involves detecting small regions which cannot be logically split into further regions. This step is important for document analysis systems which focus on graphics recognition. Previous approaches to primitive detection are discussed in section 2.3 and the approach used in this research is discussed in section 5.2.

The graphics classification step involves classifying each region into a predefined class. Examples include charts, tables, images, titles, maps, and engineering drawings. Most research approaches, including this one, focus on a subset of these classes. The most common graphics in this database are charts and tables. Previous chart detection approaches are described in section 2.6 and the approaches proposed by this research are presented in section 5.3.

The final step is information extraction. This step involves detecting the properties

of previously classified blocks. An example of this would be determining the number of bars in a bar chart. Previous approaches to chart information extraction are presented in section 2.7 and the approach used in this research is presented in chapter 6.

1.4 The Proposed System and Contributions

This research focuses on extracting information from pie charts, bar charts and line charts. For example, the sizes, shape, and location of all the bars in a bar chart are found. The purpose is to extract enough information to such that one could completely redraw the chart image using only the extracted information. This is a complete solution to the problem of chart component understanding. The system proposed in this research is part of the information extraction step seen in figure 1.6. Three of the most common types of charts, namely pie, bar, and line charts[9] are analyzed.

This system is unique in that it uses a syntactic approach for extracting the information. A syntactic approach uses a mathematical grammar which defines the expected shape and structure of the object being analyzed. An introduction to formal language theory is presented in chapter 4.

Two different sets of mathematical grammars are used to extract information from charts. The first grammar defines the structure of pie charts, characterized by their elliptical shape, and is presented in section 6.1. The structure of both bar charts and line charts, which are characterized by an axis, is defined by the second grammar, which is presented in section 6.2. Both grammars are designed to work with both 2D charts and 3D charts.

1.5 Outline of the Thesis

The remainder of this thesis is organized in the following way.

Chapter 2 presents the literature review. A selection of papers which are relevant to this research are presented. This includes the areas of classification, segmentation, primitive detection, chart detection, and chart information extraction.

The problem statement is outlined in chapter 3. It starts with a discussion on the distinctive characteristics of pie, bar, and line charts. Then it discusses how these

[9] http://visual.ly/m/common-chart-types/

charts are constructed using primitives. The chart model used in this research is presented. This model is used for deconstructing the charts. The chapter finishes by introducing the mathematics of formal language theory.

Chapter 5 presents the algorithms for segmentation, primitive detection, and graphics classification.

The core of this research is presented in chapter 6. This is where we outline the methods for extracting the relevant information from pie, bar, and line charts. The methods for the pie charts are presented in section 6.1 and the methods for the bar and line charts are presented in section 6.2.

In chapter 7 we outline the evaluation method as well as the results. The chapter starts by describing the image database in section 7.1. The evaluation of our approach is a semi-automatic system where the user enters the ground truth information and the computer compares the ground truth to what the algorithm detected. The graphical user interface for this system is presented in section 7.2. Sections 7.5 and 7.6 present the results for the pie charts and the bar/line charts respectively. Both of these sections also include a discussion on the common errors.

The thesis finishes with chapter 8 which summarizes the contributions of this research.

Chapter 2

Related Work

This chapter presents some previous document analysis systems and the algorithms they use. It also discusses how these algorithms influenced some of the decisions for the methods proposed in this thesis. Sections 2.2 through 2.7 each discuss a different part of a general graphics recognition system. The order of these sections is the order which was used to describe chart analysis in section 1.3 of the introduction (figure 1.6). The majority of previous systems do not perform every step.

2.1 Preprocessing

In document analysis and recognition, preprocessing is used to remove degradation and distortions which can occur from aging and wear from paper documents. Some distortions also occur from deficiencies in the scanning or printing process. Since digitally rendered documents are not printed or scanned, they do not contain distortions. The current project deals with digitally rendered document images so preprocessing is not required.

Kasturi et al.[27] outlines the more common preprocessing techniques: binarization, noise removal, and skew estimation. Each of these problems is addressed in the following sections.

Binarization is the process of thresholding an image such that the result is a binary image. The purpose of this process is to remove the background from the image. This process is important for discoloured or degraded documents. Gatos et al. [15] discusses several different thresholding algorithms and their performance. They also propose a new method based on the Wiener filter and Sauvola's adaptive

thresholding [47] which returned a better result than all the compared algorithms when applied to their dataset of severely degraded documents.

One of the primary transformations performed is skew detection. Aghajan and Kailath [1] proposed a method for detecting text lines within images. This method is able to detect text lines and estimate their orientation.

2.2 Segmentation

Segmentation is a critical step for many chart recognition systems as it isolates the charts from the rest of the page so that further analysis can be performed. The goal of a page segmentation algorithm is to separate the different components on the page, including both text and graphics. Page segmentation algorithms follow three paradigms, namely bottom-up, top-down, and hybrid. Within the context of document image analysis, bottom-up approaches start by finding small segments, like characters and then aggregate them into larger segments, like words and paragraphs. Top-down approaches find large segments, like paragraphs, and then break up the segments to find smaller segments, like characters.

This research uses a top-down approach similar to [37]. This was chosen because it maintained the hierarchy of the page well. It is important to have the entire chart isolated from the rest of the page as well as the individual components.

The bottom-up approach by Kise et al. [29] segments the page based on the area Voronoi diagram. The area Voronoi diagram is an extension of the point Voronoi diagram. Regions are merged by removing some of the Voronoi edges based on an area ratio.

Yuan and Tan[61] propose a method for grouping blocks together. They start with the connected component analysis from [13] and propose a new method for aggregating the blocks, based on comparing the geometric mean of the block sizes to the distance that separates the two blocks.

Another bottom-up approach is the Docstrum algorithm by O'Gorman [38], which is based on the nearest-neighborhood clustering of connected components. A histogram of component size is used to split the components into two groups, characters and other larger graphics including titles and section headings. This is so characters can be grouped together to form words. The k-nearest neighbor algorithm finds the geometrically closest k connected components for each component. This is used to determine intercharacter and interword spacing.

The top-down algorithm, the recursive X-Y cut (RXYC) by Nagy et al. [37] breaks the blocks into two or more smaller blocks based on the horizontal or vertical projection profiles. If the magnitude of the valleys in the projection profile is below a predefined values then the block is split at that point. The algorithm continues until a block cannot be split either horizontal or vertically. This algorithm creates a tree structure where the root block is the entire page. It should be noted that projection profile algorithms are sensitive to the skew of the document.

The whitespace analysis algorithm by Baird [4] segments the page by looking for the background. The algorithm completely fills the white background with white rectangles until everything is segmented. After all the blocks are isolated a weighting factor is applied which favors tall skinny blocks. Favoring the horizontal blocks allows text lines to be split but not adjacent characters.

Shilman et al.[50] present a probabilistic context free grammar based system for parsing a report into logical components. They demonstrate their research by parsing document images which contain mathematical formulas.

Cheng et al. [8] propose a method for separating figures in biomedical journals where two or more figures are displayed as a single figure. They split the problem into three categories, normal figure beside normal figures, illustrations beside illustrations, and normal figures beside illustrations. A normal figure is an image and illustrations include chart images. Normal images are separated by looking for the boundary between the images. Illustrations do not contain obvious boundaries and so they applied a particle swarm optimization clustering algorithm to find the boundaries.

2.3 Primitive Detection

This section reviews text and line/curve detection approaches. This is usually a necessary step for detecting composite graphical objects which contain these primitives.

2.3.1 Text Detection

In graphics-heavy documents, text recognition involves separating text from graphics. In this research, most of the text was isolated during the segmentation step. For the text which is over graphics, an approach similar to [26] was applied.

This problem can be split separating text which is not touching graphics and separating text which is touching graphics. Text which is not touching graphics can

be detected using connected component analysis. Text which is touching graphics is more difficult to detect. These papers discuss different ways to detect text which is touching graphics.

Tombre et al. [55] demonstrates a method for isolating characters which are touching other graphics. The paper uses the assumption that characters are typically part of larger groups (ie. words or phrases). Using the characters found from an initial segmentation, it predicts the location of additional characters. This approach can be applied to text in all orientations, however they do not include titles. One limitation of this approach is that it only works on text blocks which are partially isolated.

Hoang and Tabbone [19] use the morphological distinctiveness of text and graphics to separate between them. They perform Morphological Component Analysis with two representative dictionaries, one for text and the other for graphics.

Jain and Bhattacharjee [26] were among the first researchers to use filter banks to separate text from graphics. They recognizes the distinct texture of text regions using the Gabor filter with different orientations. They partition the response images into low, medium, and high frequency areas. Text corresponds only to high frequency.

Ahmed et al.[2] recently proposed a highly accurate text-graphics separation approach using SURF features. They manage to segment text characters touching graphics by using templates of feature vectors corresponding to text that does not touch graphics. Text regions that do not touch graphics are identified using the algorithm proposed by Ahmed et al.[3].

2.3.2 Curve and Line Detection

The process of detecting lines and curves in document images is a well studied problem. It is a step used prior to chart detection in several algorithms [21, 30, 64, 32, 59].

The approach in this research is similar to the approach in [39]. This was used because it was shown to work well by other researchers [32] and because it is a syntactic approach. Since the other methods proposed in this thesis are syntactic, the underlying mathematics is the same.

Hilaire and Tombre [17] describe the curve vectorization process. The required steps include detecting the lines and curves, forming the lines and curves into vectors, and performing postprocessing steps which split, merge or delete vectors. Most vectorization systems are restricted to lines and circular curves.

In chart detection, a common method is to represent the curves via connected chains. The Directed Single-Connected Chain by Zheng et al. [63] uses run-lengths to define the curve. This breaks the curve into a series of connected lines which are perfectly vertical or horizontal. This is used by Huang and Tan [20] and liu et al. [30] in their chart recognition system.

Another chain coding algorithm, which is used by Lu et al.[32] and is similar to the approach in this thesis, is the Primitives Chain Code (PCC) by O'Gorman [39]. This is an extension of the well-known Freeman Chain Code [14]. One key difference between the PCC algorithm and the Freeman chain code is that the PCC algorithm maintains the entire branching structure of the diagram, whereas the Freeman chain code will join some curves at junction points, but not all. The algorithm proposed in this research intentionally breaks all curves at junction points.

Huang et al.[21] isolate the curves in a 3D pie chart. This is performed by creating a set of straight lines first and then using these to recover the elliptical curves. The pie chart is then flattened into a 2D pie chart by removing the perspective distortion.

Evaluating curve detection and vectorization algorithms is non-trivial. In many cases, the evaluation is qualitative or based on human observation. Wenyin and Dori [58] describe a quantitative method for evaluating curve detection algorithms. Their approach works by measuring the area overlap between the vectorized curve and the ground truth curve. They assume that both curves have a thickness and have rounded ends. These conditions are met with hand drawn curves but are not necessarily true with computer generated charts.

2.4 Page Classification

Page classification is a well studied area of document analysis and recognition. It offers the potential to assist with office automation as well as conversion of standard paper forms to their digital equivalent. A survey paper by Chen and Blostein [7] discusses the applications, what the classifiers are trying to achieve, how the classifiers are built, and a comparison between several common approaches. They recognize that one of the largest problems for comparing these classification problems is that most classifiers use a manually defined set of classes.

One application of page classification is creating digital libraries. Sarkar [46] presents a system called the Document Image Classification System (DICE) designed for the creation of digital libraries. They classify each page using low-level visual

features like intensity variations at different scales.

2.5 Table Classification

It is difficult to compare table detection systems directly as most have different definitions of what a table looks like. There is no standard definition for the characteristics of a table [62]. One of the more common approaches now is a narrowly defined table definition for a well defined document set.

A paper by Watanabe et al. [57] represents tables by a classification tree. The trees are formed based on the table-specific global and local structure. The approach focuses on forms and not tables.

A common table and form classification technique is to recognize the gridlines. One method for detecting the lines was proposed by Tang et al. [54]. This technique detects the lines using two dimensional multiresolution wavelet analysis. They tested their algorithm on Canadian bank cheques and were able to successfully find all the lines.

A recent algorithm by Kboubi et al. [28] combines many table recognition and analysis algorithms to improve the results. They combined the output of four commercial OCR softwares and found that the output was higher than any individual software program. The software they compared was Omnipage, Finereader, Sakhr, and Readiris. They found that combining methods improved the accuracy by 23.6%.

2.6 Chart Detection

There are two different paradigms to chart detection. The first paradigm uses rule based (heuristic) detection as is done by [20, 21, 30, 60, 64, 35] while the second paradigm uses a machine learning techniques which is done by [66, 65, 25, 48, 43, 32, 35].

Liu et al.[31] did a survey of chart recognition algorithms. They outline the complete process, including segmentation, chart detection, and chart interpretation. They discuss many of the common challenges chart detection and interpretation systems face.

The work presented in this thesis considers chart detection to be a preprocessing step, therefore it uses simple methods to address it. Ellipse detectors are used for

the pie chart detection, while a profile-based approach similar to [60] is used for axis charts.

2.6.1 Rule Based Approaches

Rule based approaches for chart detection use a set of predefined rules to achieve their objective. A common bottom-up approach is to detect the primitives associated with a chart and then use a chart model to classify the type of chart.

Huang and Tan [20] use the percentage of text pixels to separate charts from tables. The lines and curves are vectorized using Directed Single Connected Chains (DSCC). They use the orientation and count of perpendicular line segments to separate charts from other drawings.

Huang et al.[22] used a model-based method for recognizing the chart type. Their approach starts with an already isolated chart image and extracts the straight lines and curves. They check for the existence of the x-y axis using the spatial relationships between the lines. The straight lines and curve vectors as well as the existence of the x-y axis is entered into a weighted likelihood function. This function determines the most probable chart type from bar chart, line chart, pie chart, and high-low chart.

Liu et al.[30] use Directional Single-Connected Chains (DSCC) to vectorize pie charts. They start with an edge map of the chart and then convert these edges into a series of lines and curves. They find and repair broken curves by finding curves which end less than 8 pixels away and have ends with similar slopes.

Yokokura and Watanabe [60] provide a well articulated definition of the chart components. They detect the two axis from the image by observing the horizontal and vertical projection profile. The different chart primitives are recognized by predicting their location relative to the axes.

Zhou and Tan [64, 67] use the Modified Probabilistic Hough Transform to recognize parallel lines. The algorithm is robust enough to recognize hand drawn bar charts. The robustness of the approach was tested against skew, experimental results showed a high accuracy.

Mishchenko and Vassilieva[35] classify charts using a model based system. They start by creating an edge structure model for each of the five different classes. To classify a chart, they use a rule based approach to match it's edge map to the five edge models and determine a best fit. To eliminate non-chart graphics they perform a size estimation by performing a colour histogram. They recognize that the colour

histogram predicts the difficultly of extracting information from the chart. They evaluate their approach by comparing it to some common machine learning techniques on the same chart images.

2.6.2 Learning Based Approaches

Learning based approaches work by applying a machine learning algorithm to detect the location of the chart on the page. In some approaches the primitives are detected first and then used by the classifier.

Zhou and Tan [66, 65] classify charts using both Hidden Markov Models (HMM) and a Neural Network (NN). The HMM approach uses a high-level analysis to find information specific to each chart category. The neural network was a multi-layer feed-forward with back-propagation. Both systems were implemented in parallel and then presented in the same paper.

Huang et al. [25] uses the Diverse Density algorithm. This approach improves on [66] in that it is less dependent on the foreground/background transitions. The diverse density function is a multiple instance learning approach. A person manually classifies groups of charts and then the computer classifies the individual charts. This approach was used to recognize 2D bar, line, 2D pie, 3D pie, and doughnut charts.

A well-known paper by Rafkind et al.[45] uses a supervised machine-learning approach to classify charts and other graphics. Their approach included 5 different graphics classifications called, gel, graph, thing, mix, and model.

Demner-Fushman et al.[11] combine both text and image features to classify graphics within biomedical journals. The text features are extracted from the figure discussions. The image features were computed using the 2D Daubechies wavelet transform [16]. These features are combined using the open source YALE machine learning environment [36].

Another paper by Demner-Fushman et al. [12] expands on their previous work in [11]. This approach still uses the 2D Daubechies wavelet transform [16], however additional features are considered. They include texture features derived from Gabor Filters to capture the coarse texture of the image. The k-means algorithm is used to find the 4 dominant colours and their frequency in a 25 element feature vector. The RapidMiner SVM[1] was used to classify the figure using these image features and text features. They evaluate their approach using a search and retrieval paradigm.

[1] http://rapid-i.com

Cheng et al. [9] separate regular and graphical images in biomedical journals. They apply two different methods, an evolutionary algorithm and a particle swarm optimization to extract image features. These features are then handed to a support vector machine which classifies the images. In a later paper [10] they improve on this method by using a Multi-Layer Perceptron Neural Network classifier. They use this approach to classify diagrams, statistical figures and flow charts.

Shao and Futrelle [49] analyze the vector graphics in portable document format (pdf) files. The process involves three stages. The first stage involves extracting the graphics and text primitives from the document. Stage two determines which graphics correspond to a graphic primitive, which they call graphemes. The final stage applies the boosting-based learner LogitBoost on the graphenes to classify the figures.

2.6.3 Chart Detection Validation

Evaluating chart detection and recognition is not straightforward for two reasons. The first reason is that charts are complex and contain a large number of components. The second reason is that the recognition of the components can be imperfect in many different ways. These imperfections are discussed in greater detail in the evaluation section of this thesis 7. The evaluation approach proposed in this thesis is different from previous chart evaluation approaches, instead it is similar to the table detection evaluation system proposed by Kboubi et al. [28]. This method is very good at describing all the possible errors which can occur in a table detection system. Using a similar evaluation method in this research allowed a more complete understanding of the limitations of the chart information extraction system.

Yang et al.[59] discusses the challenges involved in the generation of ground truth data for chart recognition. Their system creates ground truth for curves and lines, text, and full charts. For curves and lines, the system draws the detected curves on the original image and then lets the user manipulate the curves until they match the ground truth. A similar system is done for text region detection in that they allow the user to manipulate the detected text regions to form the ground truth. For full charts the user is required to select the location of the feature points which represent the graphical chart components.

This work is continued in a paper by Huang et al.[24] where they discuss the difference between an automatic and a semi-automatic approach. They present an

automatic ground truth system for chart recognition. They created 2D and 3D pie charts as well as 2D and 3D bar charts. The chart images that they evaluate are intentionally degraded using the document defect models proposed by Baird [5].

2.7 Chart Information Extraction

A paper by Savva et al.[48] uses image patches to recognize the chart type and when possible, allows the user to redraw the chart in a different format. They show that their chart classification approach is an improvement over Prasad et al.[43] who used feature vectors derived from the Scale Invariant Feature Transform (SIFT) as well as the Histogram of Oriented Gradients (HOG) to classify the charts. Both of these approaches start with a chart that is already isolated.

The paper by Lu et al.[32] presents a method for finding polylines on 2D line charts. They analyze charts with multiple polylines and demonstrate how to distinguish the curves at the intersection points. They detect the axis by using a modified Hough transform which only detects vertical and horizontal lines. Then they use a set of rules to find the axis. They use the k x k Thinning Algorithm proposed by O'Gorman [40] to reduce the polyline to a thin line and then apply the Primitive Chain Code algorithm [39] to create a pixel chain code. Connected curves are separated from each other by recognizing the type of intersection.

Brouwer et al.[6] present a method for isolating data points from 2D scientific chart images. They observed scientific line chart images which contained multiple polylines with different data points. Simulated annealing was used to resolve individual data points which overlap. They reported a recall of 88.9% for diamond data points and 91.0% for triangular data points.

2.7.1 Text/Graphics Association

Huang et al.[23] proposes an approach to recognize the function of text blocks within a chart. They recognize the caption, axis titles, axis labels, legend, data value, and other values associated with the chart. This is accomplished by looking at the spatial relationship between the text and the chart components. The associations are performed using a joint probability distribution function.

Vassilieva and Fomina [56] study the problem of Optical Character Recognition (OCR) within chart images. The popular open source Tesseract OCR system [51]

achieves up to 97% accuracy on scanned document images, yet it does not exceed 3% for chart images according to their experiments. This demonstrates the importance of isolating the text regions within the chart image. The approach separates the text from the graphics by performing connected component analysis and then using a rule based system. The remainder of the text is found by assuming that the text follows a straight line.

Mishchenko and Vassilieva [34] extract numerical data from isolated chart images. They use a model-based approach to classify the type of chart using their graphical components. Next they detect the text primitives using a bottom-up approach which they proposed in [56] (see previous paragraph). The approach had an average accuracy of 90% for detecting the chart class. They showed that the text location and recognition rate was 15-20 times better than directly processing the chart with the Tesseract OCR engine.

Chapter 3

Background

This chapter outlines the application domain of this research, including the basic geometric structure and appearance for pie, bar, and line charts. A solution to the problem outlined in this chapter is presented in the subsequent chapters.

3.1 Fundamental Properties of Charts

A 2D pie chart is a circular graph which is used to display the relative proportions of numerical data. The angle represented by each slice corresponds to the numerical value that it represents. A 3D pie chart differs from it's 2D counterpart in that the viewing angle has been changed. In addition to the different viewing angle, the 3D pie chart includes a thickness. The primary benefit of a 3D pie chart over a 2D pie chart is that it is more aesthetically pleasing. Example of 2D and 3D pie charts can be seen in figures 3.1a and 3.1b respectively.

A 2D bar chart uses rectangular bars represent numerical data overtop of a rectangular grid. The length of each bar is proportional to the value it represent. Bar charts can be oriented so the bars are vertical or horizontal. Figures 3.2a and 3.2b display a vertical and a horizontal bar chart respectively.

A line chart displays one or more series of numerical data points using a curves. Similar to bar charts, the curves are overlayed on top of a rectangular grid. Line charts typically only appear as 2D charts. An example of a line chart which contains multiple polylines can be seen in figure 3.3.

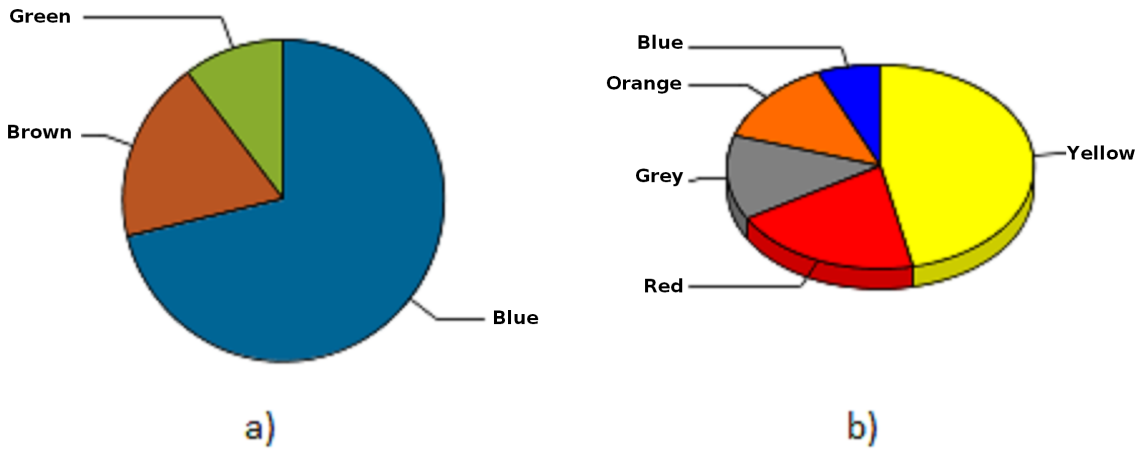


Figure 3.1: a) An example of a 2D pie chart. b) An example of a 3D pie chart.

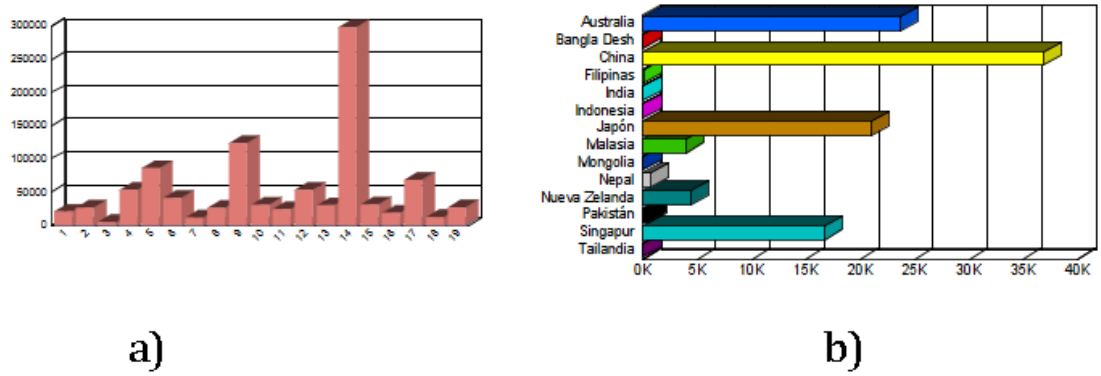


Figure 3.2: The two different orientations for a 3D bar chart. a) Vertical bar chart. b) Horizontal bar chart.

3.2 Graphical Primitives of Charts

Charts are composed of graphical primitives, which are related in geometric and semantic ways. The term primitive is used to describe the smallest components which cannot be logically broken up into smaller components. The primitives which are present in a chart depend on the chart type. There are three different primitive categories: solid regions, text, and curve/line. Primitives are characterized by properties such as colour, size, location and orientation.

Each chart type has a specific set of primitives which are characteristic of that chart type. This section presents the primitives and their classifications for pie charts

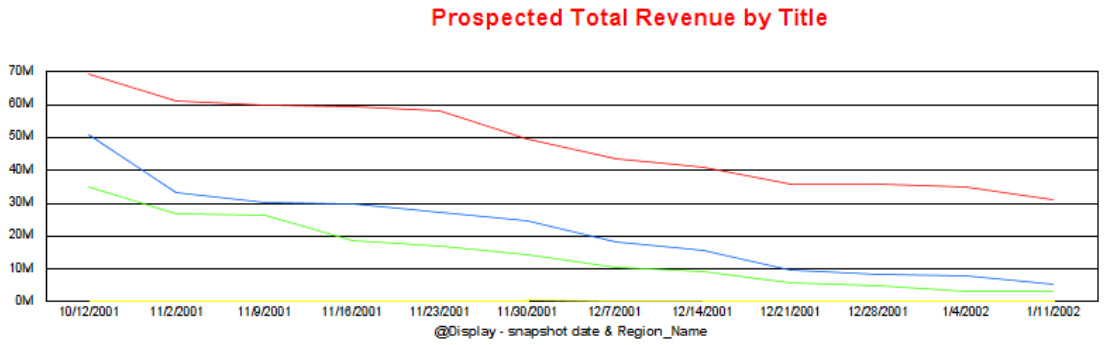


Figure 3.3: An example of a chart which contains multiple polylines.

as well as for bar/line charts.

Figure 3.4 shows a simple example of a 3D pie chart, with all its primitives labeled. The primitives are listed in table 3.1.

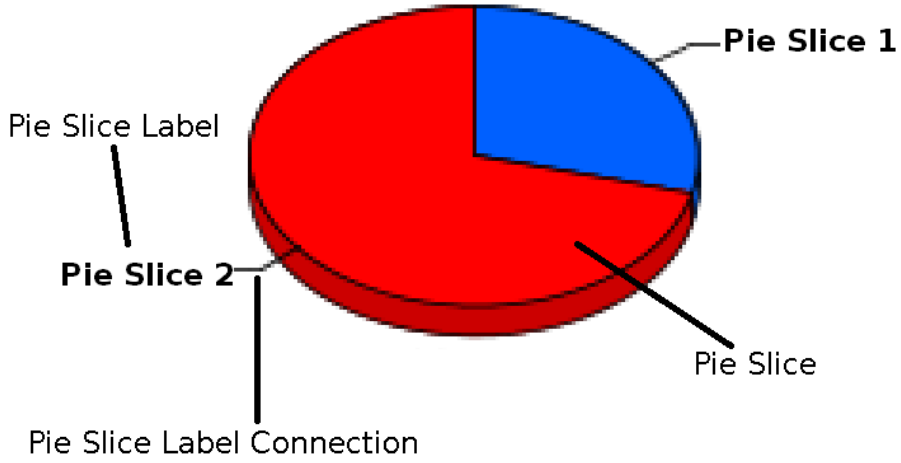


Figure 3.4: Labeled primitives of a 3D pie chart.

Figures 3.5 and 3.6 show examples of labeled primitives for bar charts and line charts respectively. Table 3.2 lists all the bar and line chart primitives. This thesis uses the same terminology as Yokokura and Watanabe [60].

Primitive Class	Primitive Type
Pie Slice	Solid Regions
Pie Slice Label	Text
Pie Slice Label Connection	Arc/Line

Table 3.1: List of pie chart primitives

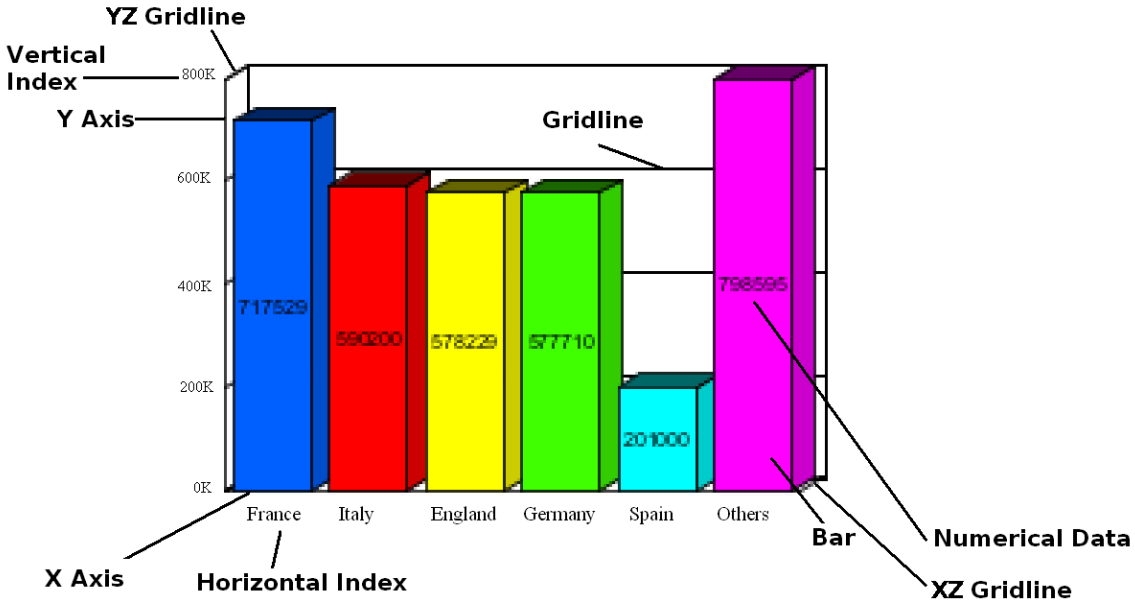


Figure 3.5: Labeled primitives of a 3D bar chart.

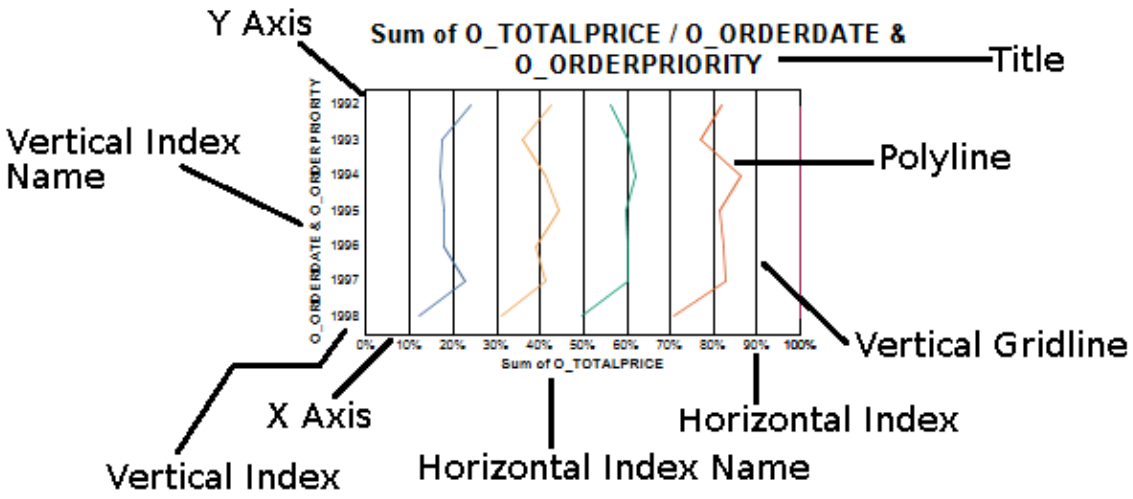


Figure 3.6: Labeled primitives of a line chart.

Primitive Classification	Primitive Category
X-Axis	Line
Y-Axis	Line
Horizontal Gridline	Line
Vertical Gridline	Line
ZX Gridline	Arc
ZY Gridline	Arc
Vertical Index	Text
Horizontal Index	Text
Vertical Index Name	Text
Horizontal Index Name	Text
Bars	Solid Regions
Numerical data	Text
Polyline	Arc
Title	Text

Table 3.2: List of bar/line chart primitives

Chapter 4

Introduction to Formal Language Theory and Mathematical Grammars

This thesis introduces new mathematical grammars and applies them to recognize chart components. This chapter provides the reader with the mathematical background necessary for understanding the following chapters.

A good introduction to formal language theory can be found in the book by Akira Maruoka [33]. In the general sense, there are two distinct categories of languages, natural and artificial. Spoken languages like English and Spanish are natural languages. An example of an artificial language is the C programming language. The chart grammars presented in this thesis are artificial languages. Artificial languages are typically simpler than natural ones. A simple example from natural language is presented in order to outline the general structure of a grammar.

Sentences in a natural language are formed by using rules to combine words together. Equation 4.1 presents some simple rules which can be used to form English sentences. These rules are applied by substituting the input term on the left side of the arrow with the terms on the right side of the arrow. Note that some lines contain multiple substitution rules. For example, there are three nouns, namely dog, cat, and hand.

$$
\begin{aligned}
&\langle \text{sentence} \rangle \rightarrow \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle \\
&\langle \text{noun phrase} \rangle \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle \\
&\langle \text{verb phrase} \rangle \rightarrow \langle \text{verb} \rangle \langle \text{noun phrase} \rangle \\
&\\
&\langle \text{noun} \rangle \rightarrow \text{dog} \qquad \langle \text{noun} \rangle \rightarrow \text{cat} \qquad \langle \text{noun} \rangle \rightarrow \text{hand} \\
&\langle \text{verb} \rangle \rightarrow \text{saw} \qquad \langle \text{verb} \rangle \rightarrow \text{saw} \\
&\langle \text{article} \rangle \rightarrow \text{a} \qquad \langle \text{article} \rangle \rightarrow \text{the}
\end{aligned}
\tag{4.1}
$$

These rules can be called *substitution rules*, production rules or simply rules. They can be combined to form sentences by starting with $\langle sentence \rangle$ and then repeatedly making substitutions. An example of how the rules can be combined to form the sentence "The dog saw the hand" can be seen in equation 4.2.

This example only makes one substitution per line and always considers the left-most element which can be substituted.

$$
\begin{aligned}
\langle \text{sentence} \rangle &\rightharpoonup \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle \\
&\rightharpoonup \langle \text{article} \rangle \langle \text{noun} \rangle \langle \text{verb phrase} \rangle \\
&\rightharpoonup \text{The} \langle \text{noun} \rangle \langle \text{verb phrase} \rangle \\
&\rightharpoonup \text{The dog} \langle \text{verb phrase} \rangle \\
&\rightharpoonup \text{The dog} \langle \text{verb} \rangle \langle \text{noun phrase} \rangle \\
&\rightharpoonup \text{The dog saw} \langle \text{noun phrase} \rangle \\
&\rightharpoonup \text{The dog saw} \langle \text{article} \rangle \langle \text{noun} \rangle \\
&\rightharpoonup \text{The dog saw the} \langle \text{noun} \rangle \\
&\rightharpoonup \text{The dog saw the hand}
\end{aligned}
\tag{4.2}
$$

A sequence of substitutions is called a *derivation*. A derivation can also be presented by a graph called a *parse tree*. The parse tree for this example can be seen in figure 4.1. In the example, the terms enclosed by brackets are called *nonterminals* and the terms without brackets are called *terminals*. Note that all the leaf nodes are terminals and all the interior nodes are nonterminals.

Definition *Terminal*: A symbol which appears as an output from one or more substitution rules which cannot be changed or replaced.

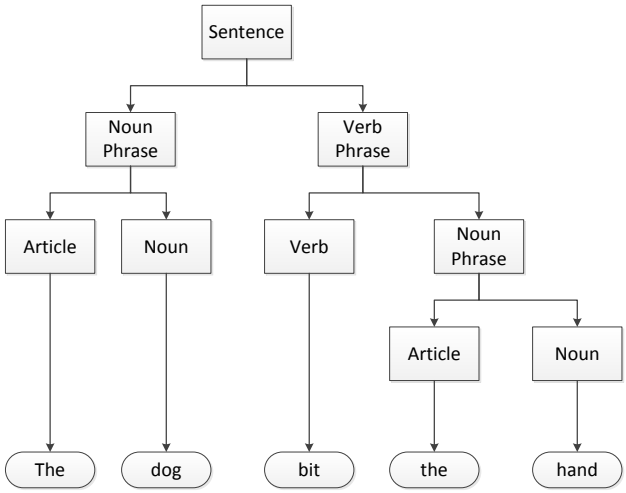


Figure 4.1: An example of a parse tree.

Definition *Nonterminal*: A symbol which is either an input or an output of a substitution rule which must be replaced using a subsequent substitution rule.

Equation 4.3 provides a second example of some substitution rules which are constructed using variables instead of English words. In this example, A, through F are nonterminals and g through n are terminals. It is typical to represent nonterminals with a uppercase letter and terminals with a lowercase letter. This convention is followed for the remainder of this thesis.

$$\begin{aligned} &A \to BC \qquad B \to DE \qquad C \to FB \\ &E \to g \qquad E \to h \qquad E \to i \\ &F \to j \qquad F \to k \\ &D \to m \qquad D \to n \end{aligned} \tag{4.3}$$

In the same way as the first example, these rules can be used to form a derivation. An example of a derivation using these substitution rules can be seen in equation 4.4. Similar to the previous derivation, only one substitution rule is used per line and the leftmost nonterminal is always the substituted one.

$$\begin{aligned} A &\rightharpoonup BC \\ &\rightharpoonup DEC \\ &\rightharpoonup mEC \\ &\rightharpoonup mgC \\ &\rightharpoonup mgFB \\ &\rightharpoonup mgkB \\ &\rightharpoonup mgkDE \\ &\rightharpoonup mgknE \\ &\rightharpoonup mgkni \end{aligned} \tag{4.4}$$

4.1 Context-Free and Chomsky Normal Form

Within a language, context is defined to be the fragments of the sequence that surrounds the symbol [33]. For example, the second line of derivation 4.2 reads $\rightharpoonup$ ⟨article⟩⟨noun⟩⟨verb phrase⟩. The context of ⟨noun⟩ is ⟨article⟩⟨verb phrase⟩ (the sentence fragments that surround it). The substitution rule used here is context-free because substituting a nonterminal does not depend on the symbols which surround it. For example, substituting ⟨noun⟩ to dog does not depend on the ⟨article⟩ or ⟨verb phrase⟩. If all the substitution rules in a language are context-free, then the language is a *context-free language*. Languages that are not context-free are context-sensitive. Most natural languages are context-sensitive. The English language is an example of a language that is context-sensitive. In English, the sentence "The big white fridge" cannot be constructed using a context-free grammar because the order of the adjectives is important.

A context-free grammar is limited to rules with a single symbol on the left side. The formal definition of a context-free grammar[33] is provided below.

Definition A context-free grammar (CFG) is a 4-tuple (V, Σ, P, S) where
(1) V is a finite set of nonterminals (also called variables)
(2) Σ is a finite set of terminals, called the alphabet, where Σ is disjoint from V
(3) P is a finite set of substitution rules, each of which has the form of

$$A \rightarrow x \tag{4.5}$$

for $A \in V$ and $x \in (V \cup \Sigma)^*$. A substitution rule is also simply called a *rule*.
(4) $S \in V$ is the *start symbol*.

An individual substitution rule can be recursive such that the same nonterminal can exist on both the right and left side of the equation. An example of a rule which has this property is $B \to BC$.

In this research the grammars are in a form called Chomsky Normal Form. This imposes several additional restrictions as to the format of the substitution rules.

A grammar is in Chomsky Normal Form if all the substitution rules are of the form $A \to BC$, $A \to d$ or $A \to \varepsilon$. Here A,B,C are nonterminals, d is a terminal, and ε is the empty string. The start nonterminal cannot exist on the right hand side of any substitution rule. That means B and C cannot be the start symbol. In addition, the start nonterminal must be the S nonterminal.

Definition A grammar is in Chomsky normal form if: (1) All rules are of the form $A \to BC$, $A \to d$ or $A \to \varepsilon$ where A, B and C are nonterminals, d is a terminal and ε is the empty string. (2) The start nonterminal cannot exist on the righthand side of any substitution rule. (3) The start nonterminal must be the S nonterminal.

One thing to note about Chomsky Normal Form is that it does not mix terminals and nonterminals on the right side. To satisfy this requirement, additional substitution rules may be required. For example, if the original substitution rule is $A \to bC$ where the A, C are nonterminals and b is a terminal. This rule is not in Chomsky Normal Form since it has both terminals and nonterminals on the right side of the equation. The rule can be rewritten as the two rules $A \to BC$ and $B \to b$. In this new format, the rules are both in Chomsky Normal Form.

Chapter 5

Primitive Detection, Segmentation and Chart Detection

This chapter describes the preprocessing required before the syntactic analysis of the charts (chapter 6). Section 5.1 describes the segmentation process. Section 5.2 describes how solid regions and text within the charts are detected. This is followed by section 5.2.3 which outlines the method used for detecting the arcs and lines. The final section, 5.3, discusses how the charts are detected and classified.

5.1 Segmentation

5.1.1 Document Segmentation via Oblique Cuts

The proposed segmentation algorithm is published in [52] and operates by searching for white space, which is interpreted as a separator between blocks. It is a top-down approach which first recognizes large structures and then recursively splits them into smaller ones. The algorithm is initialized by creating a root block, defined as a rectangular bounding box which includes every non-white pixel on the page. Next, it operates by recursive splitting along predefined directions. A hierarchy is created in the splitting process; when a block is split, the newly created blocks are children of the original block. The search for white space is performed along multiple directions (horizontal, vertical, oblique at 45 degrees, oblique at 135 degrees) in a predefined order. Since many documents have Manhattan layout, searching for white space across vertical and horizontal directions is performed prior to oblique searches. The search for white space at an arbitrary direction, described by the slope m of the line

corresponding to that direction, is formally described below.

Although many chart images can be segmented by using only vertical and horizontal directions, some chart elements such as shown in Figure 5.1 require searching for white space along oblique directions for a successful segmentation.

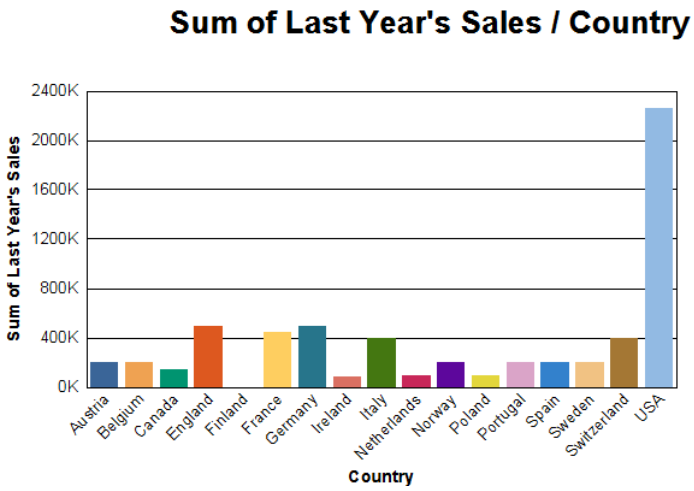


Figure 5.1: Bar chart with horizontal axis labels which can be only segmented using an oblique white space search.

In the same way as Nagy et al.[37], the segmentation process is based on the concept of projection profile, defined as the mapping of the pixel count along the search direction onto the edge of the block. Two projection profiles can be computed using equations (5.1), (5.2). It is assumed that the bottom left corner of the block being currently segmented is located at coordinate (x_0, y_0) and its rectangular bounding box is of size (w, h) (the block does not necessarily need to be rectangular). The white space search within the block depends on slope m and on the width to height ratio of the rectangular box. For $w \geq h$, start and end points for the current search are defined along the x coordinate as $x_i = x_0 - \frac{h}{m}$ and $x_f = x_0 + w + \frac{h}{m}$ and only the projection profile $P(x)$ is computed with equation (5.1). For $w < h$ start and end points for the current search are defined along the y coordinate $y_i = y_0 - wm$ and $y_f = y_0 + h + wm$ and only the projection profile $P(y)$ is computed with equation (5.2). It should be noted that searching for white space along the horizontal direction uses the lower branch of equation (5.2) with $m = 0$.

$$P(x) = \begin{cases} \sum_{j=y_0}^{y_0+h} I(x + \frac{j-y_0}{m}, j) & m \geq 1 \\ & w > h \\ \sum_{i=x}^{\frac{h}{m}+x} I(i, y_0 + m(i-x)) & 0 < m < 1 \\ & w > h \\ \sum_{j=y_0}^{y_0+h} I(x, j) & \text{vertical line} \end{cases} \tag{5.1}$$

$$P(y) = \begin{cases} \sum_{j=y}^{y+mw} I(x_0 + \frac{j-y}{m}, j) & m \geq 1 \\ & w \leq h \\ \sum_{i=x_0}^{x_0+w} I(i, y + m(i-x_0)) & 0 \leq m < 1 \\ & w \leq h \end{cases} \tag{5.2}$$

$$I(i,j) = \begin{cases} 1 & \text{if pixel at (i,j) is non-white} \\ & \text{and inside block} \\ 0 & \text{if pixel at (i,j) is white or outside block} \end{cases} \tag{5.3}$$

To determine the location of the split(s) in a block, the algorithm detects zero values in the projection profiles (either $P(x)$ or $P(y)$). This is because a value of 0 in the projection profile indicates a line of white space inside the block. The block is then split by creating two new blocks, one on each side of the white space; the split direction corresponds to the slope m used for the white space search. The segmentation is complete when after iterating through all blocks, no new blocks are created. This implies that all blocks satisfy condition (5.4). An example of a segmented chart is shown in figure 5.2.

$$\begin{array}{ll} \{x_i \leq x < x_f | P(x) \neq 0\} & w > h \\ \{y_i \leq y < y_f | P(y) \neq 0\} & w \leq h \end{array} \tag{5.4}$$

Figure 5.2 illustrates how the proposed segmentation process works. In this case, four iterations are required to completely segment the image; the output of the segmentation is a hierarchical structure, where the relationships between blocks and their children are preserved. Note that the three blocks at the top right can only be segmented by searching for white space at an oblique angle. Figure 5.3 illustrates the segmentation limits for a block with $w \leq h$ and a block with $w \geq h$.

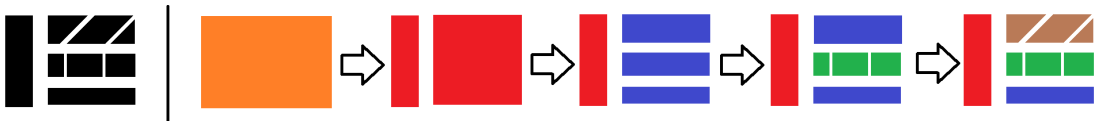


Figure 5.2: Iterative segmentation process. Iteration outputs are colour coded (0-red; 1-blue; 2-green; 3-brown).

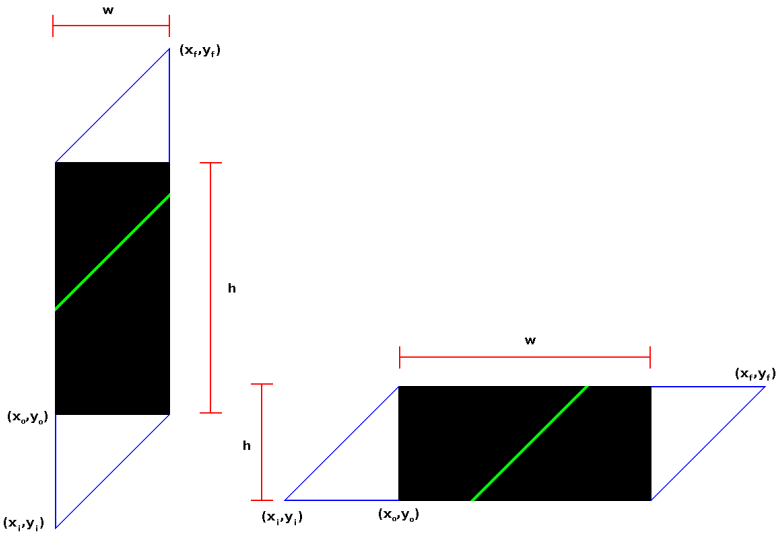


Figure 5.3: Segmentation ranges for blocks with w $\leq$ h (left) and w $\geq$ h (right).

5.2 Primitive Detection

There are three primitives used within this research, solid regions, text, and lines. The following subsections describe the process of detecting these primitives.

5.2.1 Solid Region Detection

The algorithm searches for homogeneous regions which are at least 3x3 pixels in size and have a colour other than black or white. It determines the size and shape of the solid region by using region growing. This is done by comparing each pixel to both the seed pixel and its neighbouring pixel. Comparing each pixel to the seed prevents the region from fading to either white or black and comparing to the neighbouring pixel allows region to contain slight gradients. Each solid region is a single connected component of a similar colour.

5.2.2 Text Detection

Since text is the most common primitive in a document image, many approaches look for graphics and then assume the remaining is text [55]. This is not possible for graphics heavy documents and so a different approach must be taken.

A block is considered text if two conditions are true. The first one is that the entire block is binary. That is, there are no pixels which are not either black or white. The second condition is that there are no lines detected in the block. In this approach, coloured text is not detected. The reason for this is that coloured text is rare in the analyzed database and recognizing it increased the false detection rate.

The conditions above do not recognize text that does not have a white background or is over top of other graphics. An example of this is the numerical data found in figure 5.4. The numerical data on the bars does not have a white background, and in certain cases (not seen in this example) the text can touch the edge of the bar which is a black line.

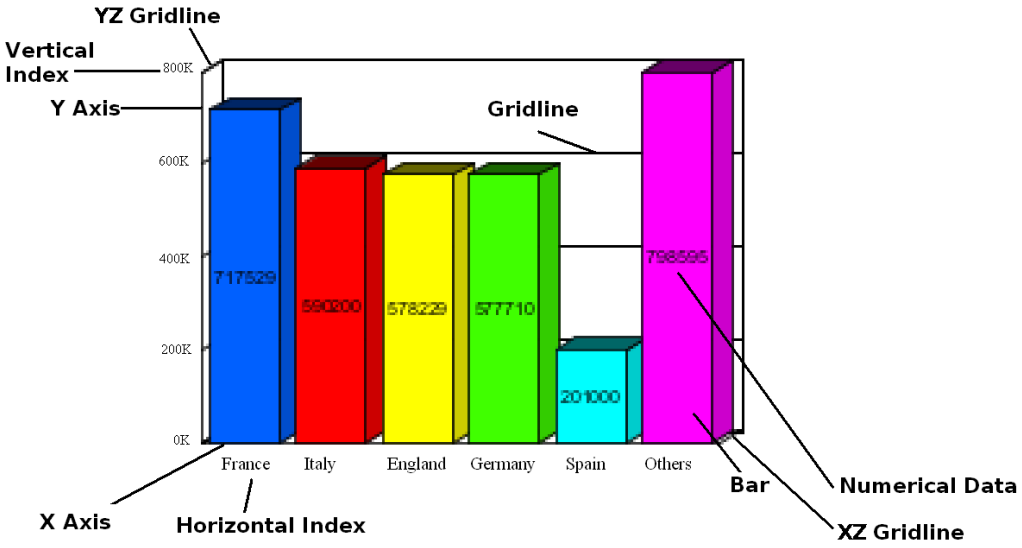


Figure 5.4: The primitives of a 3D bar chart. The numerical data is detected with the Gabor filter. It is best if this figure is viewed in colour.

This text can be detected using the Gabor filter (See Appendix A). The Gabor filter is applied to the grayscale version of the document image twice, once at 0 degrees and the other at 90 degrees. This produces two different 2D responses. The bitwise AND function is applied to the two responses to generate a new combined image.

Connected component analysis is performed on the combined image to determine the location of the text. A similar approach which also uses the Gabor filter to isolate text is proposed by Jain and Bhattacharjee [26].

5.2.3 Curve Detection

The purpose of curve detection is to represent curves as one-pixel wide sequences of adjacent pixels. The final output will be a set of curves where each curve is a series of adjacent points. This process is involves two steps, namely region thinning and linking.

Definition A curve is a continuous 8-connected chain of pixels which starts at either a single-connected pixel or an intersection, ends at either a single-connected pixel or an intersection, and does not contain an intersection within the curve.

Region thinning (also called skeletonization) transforms a region into thin, one-pixel wide lines. The output is an image which uses the minimum number of pixels required to maintain 8-connected curves. Figure 5.5 shows an example of thinning. The kxk thinning algorithm in [40] was chosen because it has been shown to work well in previous studies [32]. The *kxk* thinning approach is a generalization of the well known approach by Hilditch [18] from 3x3 to kxk, where k is greater than or equal to 3.

Figure 5.5: The curve on the left is not completely thinned for 8-connected adjacency as a pixel can still be removed without breaking the adjacency. The curve on the right is completely thinned as no additional pixels can be removed without breaking the adjacency.

The linking process involves converting the thinned image into a series of chains. This process starts by finding a single pixel which has not yet been associated with a curve. The remaining pixels of the curve are found by repeatedly going from one pixel to an adjacent pixel. This entire process is repeated until all the curves have been found.

The algorithm used in this research is a modification of the grammar found in the Primitive Chain Code (PCC) by O'Gorman [39]. The PCC algorithm is a syntactic method for detecting curves and was chosen because it worked well in previous approaches [32]. Each curve will be the result of a single derivation.

The algorithm used in this thesis is different from PCC because the curves are split at intersection points instead of connecting the curves together. The splitting of the curves is important for the curve classification step which is discussed in a later section. Figure 5.6 shows the conceptual difference between this algorithm and PCC. The reason the curves are broken at intersection points is because of the classification which done later. It is desirable for each curve to have a unique classification and without splitting the curves at the intersection points, this would not be the case.

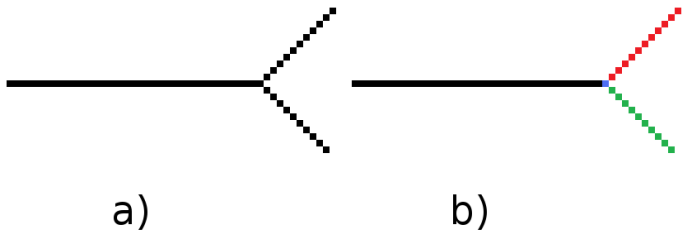


Figure 5.6: a) The output of the PCC algorithm, the entire skeleton is considered one curve. b) The output from the proposed algorithm, the skeleton is split into 3 different curves. The black, red, and green colours show the different curves. The pixel at the intersection point is common to all three curves.

Each inspected pixel will be either a point (P), end (E), curve (C), fork (F), or cross (X) pixel based on it's number of connections. Figure 5.7 presents the possible pixel connection types, which are the non-zero V8 neighbors. Due to the previous thinning step, a pixel can only have between 0 and 4 connections. Pixels with 2 connections will always be in the middle of a curve and all other pixels will be end points.

Tables 5.1 and 5.2 list the nonterminals and the terminals respectively. For each connection type there can be three different nonterminals. The first is the initial nonterminal (subscript i) which represents the first pixel detected in the curve. The second is the final nonterminal (subscript f) which represents an end pixel. The third are the nonterminals which are used for adding a pixel to the curve (subscript t). In this case, the subscript t stands for terminal. The P nonterminal does not have a subscript because it corresponds to both an initial and a final pixel. The C nonterminal does not have a subscript because it corresponds to neither an initial

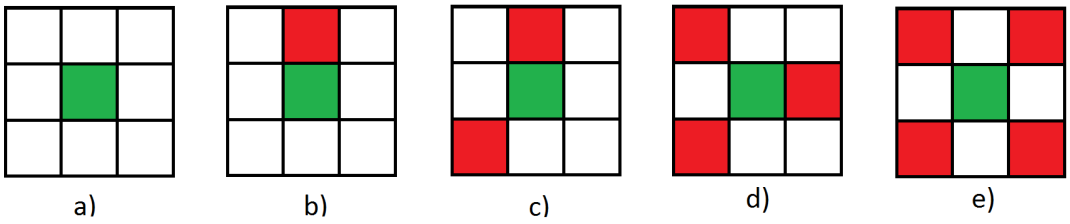


Figure 5.7: Examples of the 5 different types of connected points. The seed point is in green. a) No connections (P), b) One connection, curve end (E), c) Two connections, curve middle (C), d) Three connections, fork (F), e) Four connections, cross (C). It is best if this figure is viewed in colour.

pixel nor a final pixel. The N nonterminal can be considered a placeholder for the next pixel to be added to the curve. The N nonterminal will be encountered as an intermediate step between pixel assignments.

When a terminal is added to the string, a pixel is added to the set of points representing the curve. After a derivation there will be a single terminal for every pixel in a curve. The length of the final character string is equal to the number of pixels in the curve. Once the derivation has finished, the detection of that curve is complete. The detected pixels are ignored for the detection of subsequent curves. The process repeats until all the pixels of the image have been associated with a curve. A new derivation will occur for each remaining curve.

Table 5.2: A list of the terminals used for curve detection.

Terminal Symbol	Visual Meaning	Connections	Description
p	Point	0	Isolated point
e	End	1	Single-connected end
c	Curve	2	Pixel in the intermediate portion of the line
f	Fork	3	Pixel at an intersection of three curves
x	Cross	4	Pixel at an intersection of four curves

Table 5.1: A list of the nonterminals used for curve detection.

Nonterminal Symbol	Visual Meaning	Connections	Description
S	Start Nonterminal	NA	Start nonterminal, an initial foreground pixel has been detected.
P	Point	0	Isolated pixel designates a curve of length 1.
E_i	End	1	First pixel found on the curve, designates an end.
C_i	Curve	2	First pixel found on the curve, designates a pixel in the intermediate portion of the curve.
F_i	Fork	3	First pixel found on the curve, designates a fork.
X_i	Cross	4	First pixel found on the curve, designates a cross.
C	Curve	2	Designates the intermediate portion of the curve.
E_f	End	1	Last pixel found in a curve, designates an end.
F_f	Fork	3	Last pixel found in a curve, designates a fork.
X_f	Cross	4	Last pixel found in the curve, designates a cross.
N		NA	Searches for the next pixel.

The presence of either the S or N nonterminal in the character string indicates that the choice of the next substitution rule is based on the number of connections of the currently evaluated pixel (see figure 5.7). If the start pixel has two connections (For example, pixel number 3 in figure 5.8), the substitution rule $S \rightarrow C_i$ will be used from equation 5.5. The next rule will be the $C_i \rightarrow C_t N^2$ from equation 5.6. This is the only rule for the C_i nonterminal. If the next pixel has 3 connections (for example, pixel number 2 in figure 5.8), the substitution rule $N \rightarrow F_f$ from equation 5.7 is used.

The curve detection grammar is split into 4 groups of rules and each group is represented by an equation. Equation 5.5 lists the five substitution rules which initiate the curve detection process, one rule for each connection type. Equations 5.6, 5.7, and 5.8 have four rules, a single rule for each connection type in figure 5.7 except the trivial case of an isolated pixel.

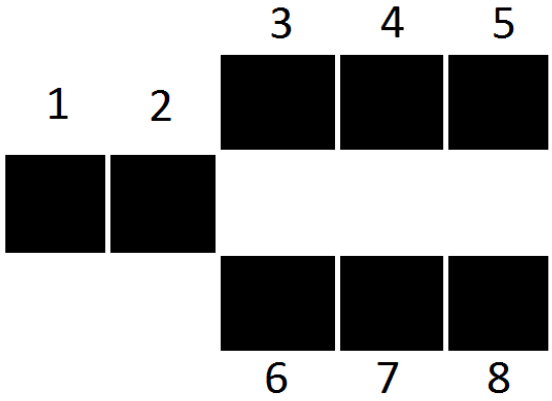


Figure 5.8: This figure is used to help describe the rules.

The nonterminal S indicates the detection of a single pixel on a curve (ie. it starts the curve detection process). It should be noted that the nonterminals E_i, F_i, and X_i are end points (see figures 5.7b, 5.7d, and 5.7e respectively) and the nonterminal C_i is an intermediate point (Figure 5.7c). The P nonterminal corresponds to a single isolated point with no connections to any other pixels, figure 5.7a.

$$S \to P \qquad S \to E_i \qquad S \to C_i \qquad S \to F_i \qquad S \to X_i \tag{5.5}$$

Equation 5.6 lists the substitution rules which are used to continue the detection of a curve. Since the curves are split at the intersection points, the fork and the cross connections are also end points. The nonterminal N indicates that the algorithm continues searching for additional pixels to add to the curve. When starting on a fork or a cross, direction for the curve following may be chosen arbitrarily. For example, if the first pixel detected is pixel number 2 in figure 5.8, the algorithm can go to either pixel 1, 3, or 6.

$$E_i \to E_t N \qquad C_i \to C_t N^2 \qquad F_i \to F_t N \qquad X_i \to X_t N \tag{5.6}$$

Equation 5.7 lists the rules for the N nonterminal. The substitution rule is chosen based on the connection type. The same connection types from figure 5.7 are possible except the isolated pixel.

$$N \to E_f \qquad N \to C_f \qquad N \to F_f \qquad N \to X_f \tag{5.7}$$

The substitution rules in equation 5.8 are applied directly after a rule from equation 5.7. Similar to the equations previous equations, the 4 rules correspond to the four connection types. It should be noted that all of these substitution rules result in a terminal except the $C_f \rightarrow C_t N$ rule. This is because an isolated end (1 connection) as well as a fork (3 connections) and cross (4 connections) are all end points. The line (2 connections) is not an end point and so the substitution rule associated with it does not yield a terminal. When a nonterminal is replaced by a terminal, the pixel which is associated with it is added to the curve.

$$E_f \rightarrow e \qquad C_f \rightarrow C_t N \qquad F_f \rightarrow f \qquad X_f \rightarrow x \tag{5.8}$$

Equation 5.9 presents the rules which contain a terminal. Similar to the previous equations, there is a single rule for each connection type. These rules correspond to adding a point to the curve.

$$P \rightarrow p \qquad E_t \rightarrow t \qquad C_t \rightarrow c \qquad F_t \rightarrow f \qquad X_t \qquad x \tag{5.9}$$

5.3 Chart Detection

This section outlines the chart detection process. This is performed by searching for the unique properties of each chart type. Pie charts are found by detecting the ellipse which is common to all pie charts. Bar and line charts are found by detecting the axis.

5.3.1 Ellipse Detection

The steps for detecting the ellipse can be seen in figure 5.9. The process starts by binarizing the image since only the shape of the graphics is important. The next step is to perform a morphological close operation which removes the connecting lines and text. The final step involves applying the Canny edge detector. These first three steps can be seen in figure 5.10.

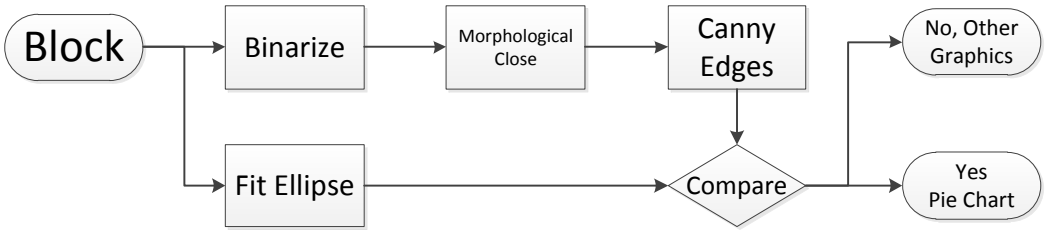


Figure 5.9: The process flow for detecting pie charts.

In addition to finding the edge map, an ellipse is fit over the original chart image using the least squares method. Figure 5.11 shows the best fit ellipse superimposed on the original chart image. These two ellipses are compared to each other and if they are similar enough the graphics are considered an ellipse.

One problem for comparing the ellipses is that both contours are one pixel wide. If one of the contours of the two ellipses were shifted by only a single pixel a direct comparison at the pixel level would show a very small overlap. To solve this problem a Gaussian blur is applied to the edge image. The algorithm iterates over all pixels in the best fit ellipse and compares the pixels to the blurred image. When a pixel in the best fit ellipse overlaps with a pixel in the blurred image, the intensity value is added to a rolling sum. The intensity of each pixel is divided by 255 to create an intensity value between 0 and 1. After the comparison is finished, if the sum is above 1% of the ellipse, the ellipse is considered to be a pie chart. It should be noted that blurring the image significantly reduces the maximum possible intensity value. It is not possible to have a sum above 10%, even with a perfect overlap and under ideal conditions. This method is described by the formula in equation 5.10 where E is best fit ellipse, $I(p_i)$ is the intensity at point p_i and V is the minimum threshold for an ellipse.

$$\sum_{p_i \in E} \left(\frac{I(p_i)}{255} \right) > V \tag{5.10}$$

5.3.2 Axis Detection

The axis of a chart is detected using the vertical and horizontal projection profiles. A projection profile is a vector which represents the mapping of the count of non-white

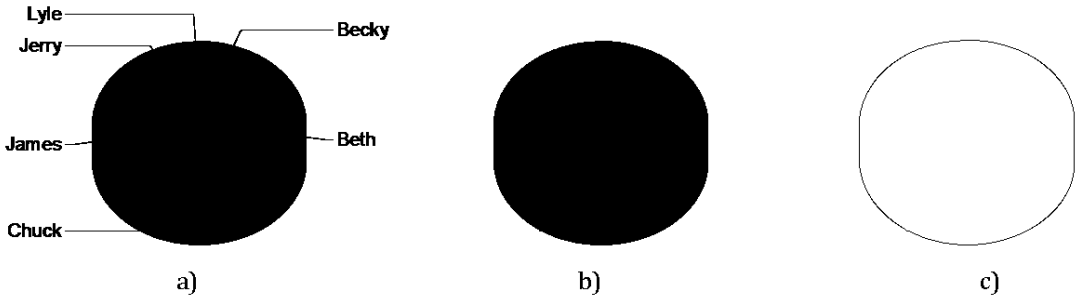


Figure 5.10: The steps used to find the edge map for the same chart seen in figure 5.11. a) The binarized version of the pie chart. b) The binarized image after the morphological close operation. c) The edge map calculated using the Canny edge detection on b). The colours are inverted for the purpose of printing the document.

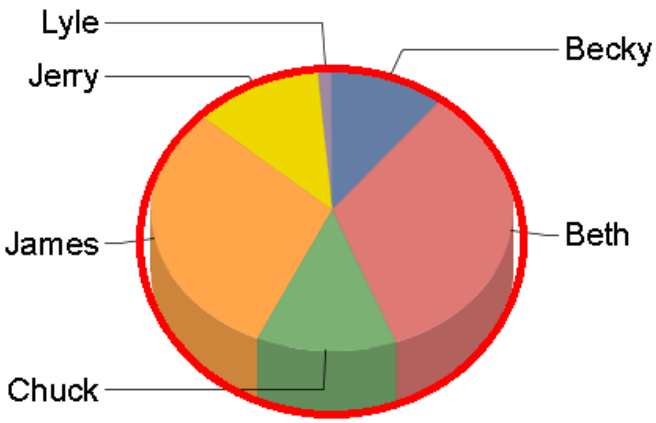


Figure 5.11: An example of an ellipse fitted over a pie chart. It is best if this figure is viewed in colour.

pixels onto a line. Long straight lines will appear as large values in the projection profile. Normally a projection profile is a single array of numbers, however they can observed visually by plotting them. Examples of a horizontal and vertical projection profiles can be seen in figure 5.12.

We look for projection profile values above 80% of the maximum possible value. The maximum possible values of the vertical and horizontal projection profiles are the height and width of the block respectively. Each value above this threshold is assumed to be the location of the either an axis or a grid line. To differentiate between the axis and grid lines, it is assumed that the y axis is the leftmost value and the x axis is the bottom value. An example of this can be observed with the horizontal

projection profile seen in figure 5.12. In this example, each grid line produces a high projection value. A bounding box representing the detected axis can be seen in figure 5.13.

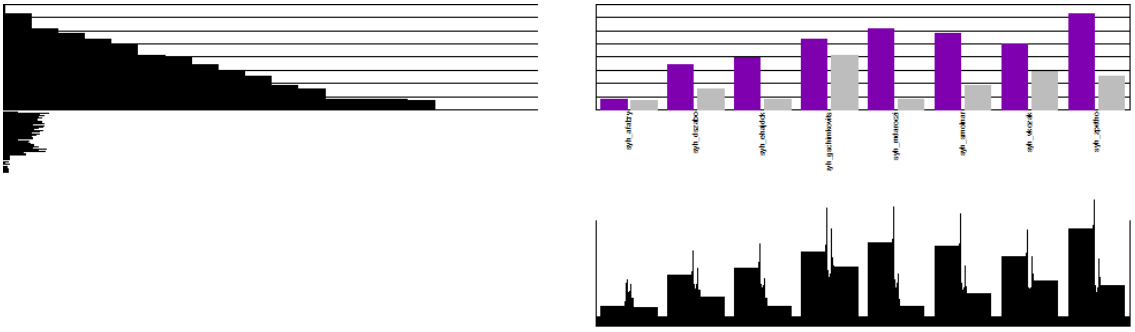


Figure 5.12: An example of the vertical and horizontal projection profile for a chart. The original chart is seen in the top right. The horizontal projection profile is visible in the left image. Each width of each row of pixels indicates the number of non-white pixels with the same y value. The vertical projection is seen at the bottom. The height of each column of pixels indicates the value of that column. The profiles are aligned to the original chart.

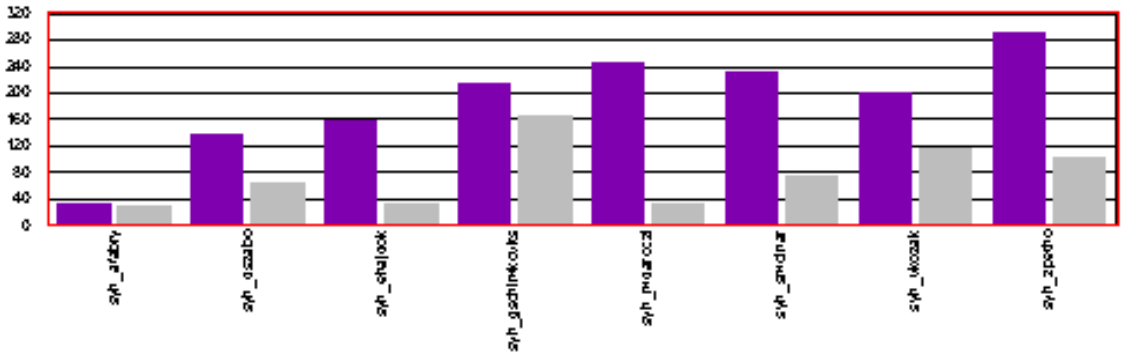


Figure 5.13: An example of a bounding box placed where the axis was detected. In this case the x-axis was along the bottom and the y-axis was along the left side. It is best if this figure is viewed in colour.

Chapter 6

Chart Grammars

The main contribution of this thesis, as mentioned in section 1.4, is the chart grammars which are a syntactic approach to recognize and classify all components of a chart. The grammars classify the previously isolated text, curves, and solid regions and determine their role within the chart. In order to maintain Chomsky normal form, rules have been added so that terminals and nonterminals are not mixed. The reason grammars are used to classify the chart components instead of a statistical method is because the charts are created using a set of rules. These same rules can be used to recognize the components. The shape and appearance of a chart is mathematically defined.

In this research, a character string will be created every time a derivation is performed. This character string will define the chart image which was analyzed. Each character in the string corresponds to a single terminal and a classified chart primitive. Some characters in the string only correspond to part of a logical chart component. For example, a horizontal gridline which is split by a vertical bar will have a terminal for each line segment. An example of a chart which contains gridlines which are separated by bars can be seen in figure 6.1.

6.1 Pie Chart Curve Classification

The objective of this grammar is to classify the extracted curves and associate them with their corresponding slice. The curves are first classified by their purpose. For example, the grammar recognizes which curves are the radii curves and which curves are on the outside edge. Second, the grammar groups the curves into distinct slices.

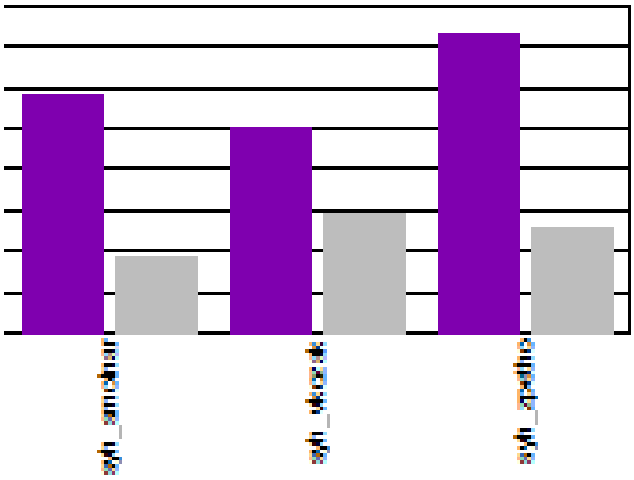


Figure 6.1: Part of a chart which contains gridlines which are behind the chart bars.

As an example, it recognizes the two radii curves for each slice. The result is enough information to redraw a single slice or the entire chart. The type and relative position of the classified curves can be seen in figure 6.2.

A pie chart exhibits some specific characteristics related to its shape and to the relative position of its components. The proposed grammar makes use of these characteristics. The rules classify the curves using the relative spatial orientation and location between two curves. They start with a single classified curve and then find the connecting curves. The relative location and orientation of the curves and classification of the initial curve allows the algorithm to classify the following curve. For example, if we know that the classification of the orange curve in figure 6.3 is a top curve, we can also classify the black and blue curves based on their angle with the orange curve.

This subsection starts by listing the classifications and their corresponding terminals in 6.1.1. It presents the nonterminals and the substitution rules in 6.1.2. Section 6.1.3 describes how to perform a derivation using the grammar.

6.1.1 The Classifications and Terminals

This subsection lists the curve classifications and their corresponding terminals. There is a one-to-one relationship between the types and the terminals. Figure 6.4 shows the locations of the classified curves and table 6.1 lists the terminals. Note that there

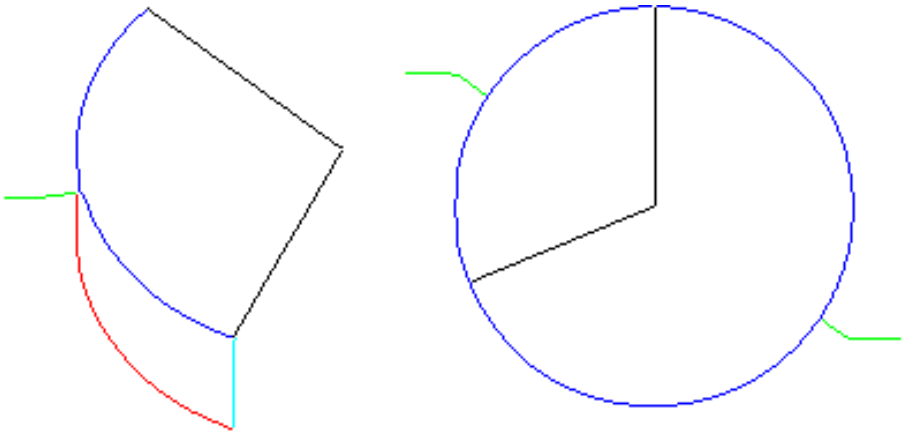


Figure 6.2: An example of a pie slice and a pie chart. Classification is colour-coded as follows: Black: radius; Blue: top edge (in 3D), edge (in 2D), Red: bottom edge (in 3D, not used in 2D), Turquoise: side curves (in 3D, not used in 2D), Green: label connection.

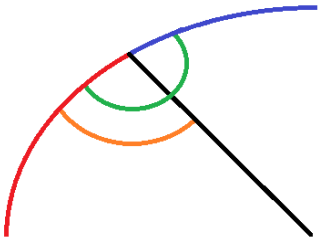


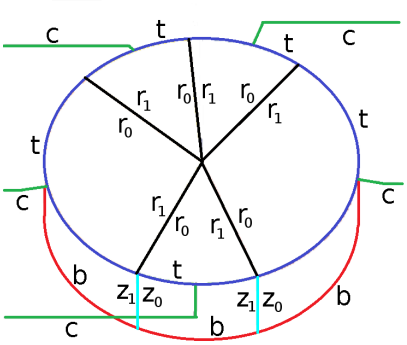
Figure 6.3: Three connecting curves.

are two terminals in table 6.1 which are not depicted in figure 6.4 because they do not correspond to a curve. These are the pie slice label and the pie slice pattern or colour. The pie slice label is text and the pattern or colour is a solid region. The difference between the t terminal and the b terminal is that the t terminal represents a curve on the top of the pie chart cylinder and b represents a curve on the bottom of the pie chart cylinder.

Each curve type is associated with a unique terminal. When a terminal is added to a string with a substitution rule, the component associated with it is classified. Each derivation will classify all the components in a single slice. Additional derivations will classify the remaining slices.

It is expected that some of the final classified curves will be a combination of two or more initial curves. An example of this is the outside curve (labeled t in figure 6.4) when there are label connections present (labeled c in figure 6.4). Initially there

will be two curves, one on either side of the label connection. The original curves prior to classification would have been split at the intersection point with the label connection (green). In this situation the curves are combined to form a single curve. The end result will be one top curve (t) per slice. The merging of two or more initial curves into a final classified curve is done when the initial curves share the same classification.



Terminal Symbol	Name	Description
s	Slice	New slice
c	Label connection	Label connecting curve
r	Radius	Radius curve
t	Top	Top edge of cylinder
b	Bottom	Bottom edge of cylinder
z	Side	Side of cylinder
p	Pattern	Pattern or colour of slice
l	Label	Text label

Figure 6.4 & Table 6.1: Left: The curves and their corresponding terminals. Right: A list of the terminal symbols and their names. Note that the pattern terminal and the label terminal are not included in the lefthand figure because they are not curves. It is best if this figure is viewed in colour.

6.1.2 The Nonterminals

In this grammar the nonterminals can have a range of roles; the most common rule is to represent a curve, as shown in figure 6.6. All curves have at least one and at most

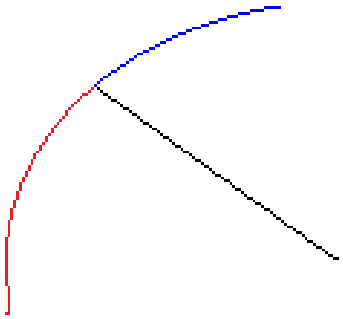


Figure 6.5: Three connecting curves.

two associated nonterminals. Two nonterminals are necessary when the curve divides two slices, or when it is followed in two directions. In the later case, the nonterminals carry directional information, as shown in figure 6.6.

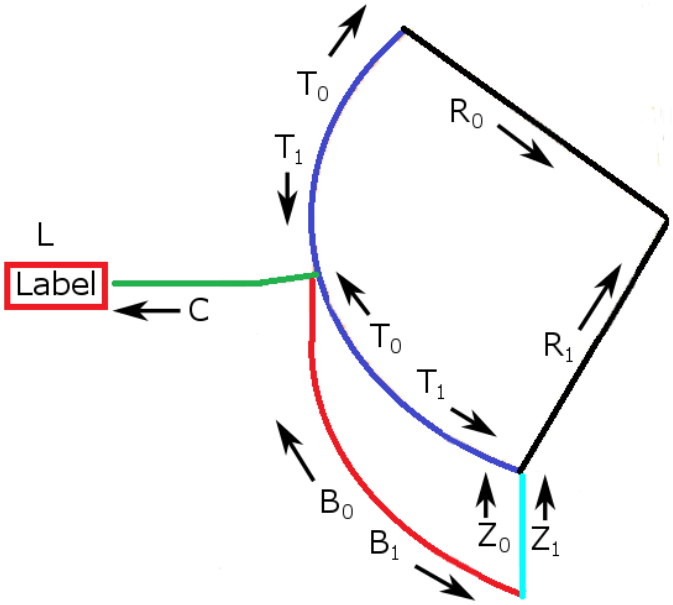


Figure 6.6: The curves and their associated nonterminals for a 3D pie slice. The directions are shown by arrows. Note that the rightmost R_0, the top R_1, and the Z_0 nonterminals correspond to an adjacent pie slice. They show that a curve belonging to two slices is associated with two nonterminals.

Table 6.2 presents some of the nonterminals and a short description of what each one represents. The S nonterminal is the start nonterminal. This starts the derivation process. The A nonterminal represents the first curve detected on the outside edge of the pie chart.

Table 6.2: A list of the nonterminals used for pie chart curve classification.

Symbol	Visual Meaning	Description
S	Start	Indicates an initial curve was found.
A	Arc	Arc on the outside edge of the circle.
T_0	Top	Follows the top edge of the cylinder in the clockwise direction.
T_1	Top	Follows the top edge of the cylinder in the counterclockwise direction.
B_0	Bottom	Follows the bottom edge of the cylinder in the clockwise direction.
B_1	Bottom	Follows the bottom edge of the cylinder in the counterclockwise direction.
C	Label Connection	The curve which connects the pie slice to the label.
R_0	Radius	Slice radius, smaller angle relative to x-axis.
R_1	Radius	Slice radius, larger angle relative to x-axis.
Z_0	Side	Pie chart side, smaller angle relative to x-axis.
Z_1	Side	Pie chart side, larger angle relative to x-axis.
L	Label	Text label
P	Pattern	Colour or texture filling out the region corresponding to the pie slice.

The T_0 and T_1 nonterminals represent the curves along the top edge of the pie chart cylinder in the clockwise and counter-clockwise directions respectively. The directions are presented in figure 6.6. The B_0 and B_1 nonterminals represent the curves along the bottom edge of the pie chart cylinder in the clockwise and counter-clockwise directions respectively.

The R_0 and R_1 nonterminals represent the radius curves. Every slice has two radius curves, one of which corresponds to the R_0 nonterminal and the other which corresponds to the R_1 nonterminal. The R_0 nonterminal represents the radius curve with the smallest angle relative to the positive x-axis and the R_1 nonterminal represents the curve with the larger angle relative to the positive x-axis.

The Z_0 and Z_1 nonterminals represent the curves along the side of the pie chart cylinder. The subscript numbering follows the same convention as the radius curves.

The curve of the Z_0 nonterminal will always connect to the curve of the R_0 nonterminal and the curve of the Z_1 nonterminal will always connect to the curve of the R_1 nonterminal.

The C nonterminal represents the curve which connects the pie slice and the pie slice label. The L nonterminal represents the pie slice label. This nonterminal is unusual in that it represents a text block instead of a curve. The P nonterminal represents the colour or texture which fills out the region corresponding to the pie slice.

The remaining nonterminals not included in table 6.2 are S_t, P_t, T_t, B_t, C_t, Z_{t0} and Z_{t1}. These nonterminals all appear in a single substitution rule which substitute to a single terminal. For example, the S_t nonterminal corresponds to the s terminal with the substitution rule $S_t \rightarrow s$. The P_t nonterminal corresponds with the p terminal and is responsible for the slice pattern or colour. The T_t nonterminal corresponds with t terminal and the top curve. The B_t nonterminal corresponds with the b terminal and the bottom curve. The C_t nonterminal corresponds to the c terminal and the label connection curve. The Z_{t0} and Z_{t1} nonterminals correspond to the z_0 and z_1 terminals respectively. These terminals will add the side curves to the pie slice object.

6.1.3 Substitution Rules

This section presents substitution rules for classifying and linking all curves and lines belonging to a pie chart. The linking is performed on a slice by slice basis, and the process continues until all slices are labeled. The grammar performs a sequential classification process; once a given curve is classified, a curve connected to it is analyzed for classification purposes. Substituting a nonterminal with another nonterminal indicates the transition between connecting curves.

To initialize the process of curve classification and linking, a curve that is on the bounding ellipse is detected. A curve is considered to be on the ellipse if all its points are on the ellipse. This can be determined by using equation 6.1. Once a curve on the ellipse has been detected the start nonterminal S is initiated.

$$\left\| \sqrt{\frac{(x-x_0)^2}{p^2} + \frac{(y-y_0)^2}{q^2}} \right\| \quad \begin{cases} < 1 & \text{Then (x,y) Inside ellipse} \\ = 1 & \text{Then (x,y) On ellipse} \\ > 1 & \text{Then (x,y) Outside ellipse} \end{cases} \tag{6.1}$$

The rules associated with the start nonterminal are presented in equation 6.2. There is only one rule for substituting the S nonterminal, which is $S \rightarrow S_t A P_t$. This rule is used to start processes associated with the S_t, A, and P_t nonterminals. There is only one substitution rule for the S_t nonterminal, which is $S_t \rightarrow s$. The s terminal indicates that a new slice data structure needs to be created. The P_t nonterminal represents the colour of the slice and is described in more detail in section 6.1.5.

$$S \rightarrow S_t A P_t \qquad S_t \rightarrow s \qquad P_t \rightarrow p \tag{6.2}$$

In order to substitute the A nonterminal the algorithm must determine if the curve is on the top or the bottom of the cylinder. For a 3D pie chart, a curve on the top of the ellipse requires the $A \rightarrow T_0 T_1$ rule and a curve on the bottom requires the $A \rightarrow B_0 B_1$ rule (equation 6.3). For a 2D pie chart, the $A \rightarrow T_0 T_1$ substitution rule will always be used. The rules in 6.3 ensure that curves are looked for in both clockwise and counterclockwise directions. It is possible to determine if the chart is 2D or 3D by analyzing the initial bounding ellipse (see section 5.3.1). If the ellipse is a perfect circle, the chart is 2D, otherwise it is 3D.

$$A \rightarrow T_0 T_1 \qquad A \rightarrow B_0 B_1 \tag{6.3}$$

Table 6.3: The angles between the curves. The first column indicates the initial curves and each row indicates the angles for each possible nonterminal relative to the initial curve. Note that many curves never intersect. The angles are in degrees.

	Top T_0	Top T_1	Bottom B_0	Bottom B_1	Label Connection C	Radius R_0	Radius R_1	Side Z_0	Side Z_1
T_0	180°	-	180°	-	270°	90°	-	-	-
T_1	-	180°	-	180°	90°	-	270°	-	-
B_0	180°	0°	180°	-	270°	90°	-	90°	-
B_1	0°	180°	-	180°	90°	-	270°	-	270°
C	-	-	-	-	180°	-	-	-	-
R_0	-	-	-	-	-	180°	-	-	-
R_1	-	-	-	-	-	-	180°	-	-
Z_0	-	90°	-	-	-	-	-	180°	-
Z_1	270°	-	-	-	-	-	-	-	180°

The set of rules for substituting the T_0 and T_1 nonterminals are presented in equations 6.4 and 6.5 while the rules corresponding to the B_0 and B_1 nonterminals are

shown in equations 6.6 and 6.7. Each rule corresponds to a different type of intersection and is determined by the angles between the previously classified curve and the connecting curves. Figures 6.7, 6.8, 6.9, 6.10 show examples of the intersections for the T_0, T_1, B_0, and B_1 nonterminals.

$$\begin{aligned} &T_0 \rightarrow T_t C T_0 \qquad T_0 \rightarrow T_t R_0 \\ &T_0 \rightarrow T_t T_0 B_0 \qquad T_0 \rightarrow T_t C T_0 B_0 \qquad T_t \rightarrow t \end{aligned} \tag{6.4}$$

$$\begin{aligned} &T_1 \rightarrow T_t C T_1 \qquad T_1 \rightarrow T_t R_1 \\ &T_1 \rightarrow T_t T_1 B_1 \qquad T_1 \rightarrow T_t C T_1 B_1 \end{aligned} \tag{6.5}$$

$$\begin{aligned} &B_0 \rightarrow B_t Z_0 \qquad B_0 \rightarrow B_t C T_0 T_1 \\ &B_0 \rightarrow B_t T_0 T_1 \\ &B_t \rightarrow b \end{aligned} \tag{6.6}$$

$$\begin{aligned} &B_1 \rightarrow B_t Z_1 \qquad B_1 \rightarrow B_t C T_0 T_1 \\ &B_1 \rightarrow B_t T_0 T_1 \end{aligned} \tag{6.7}$$

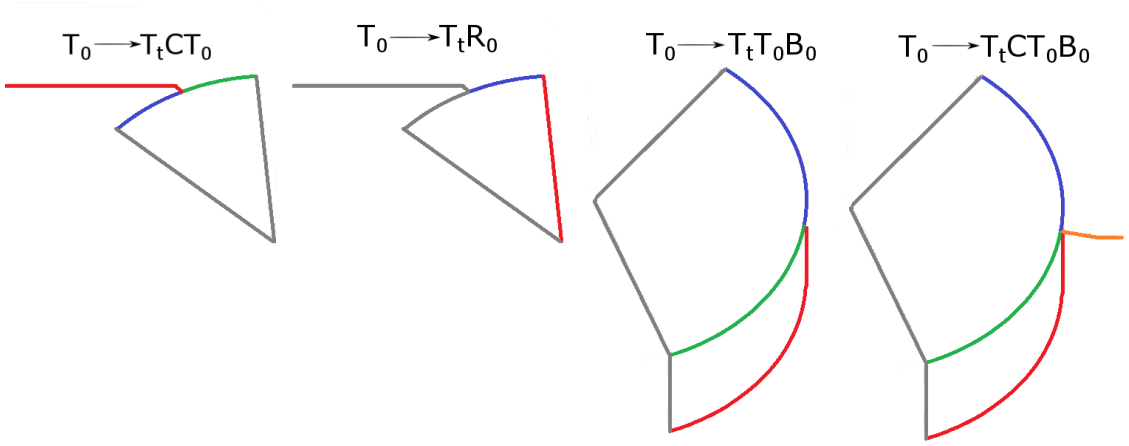


Figure 6.7: The location of the rules for T_0.

Equation 6.8 shows the substitution rules for the C, C_t, and L nonterminals. Each nonterminal has a single rule for substituting it. The C nonterminal represents the curve which connects the pie slice to the text label. There are no additional actions required to substitute the C nonterminal. Using the substitution rule $C_t \rightarrow c$ requires adding the label connection curve to the slice data structure. The substitution rule for the L nonterminal indicates adding the slice label to the slice data structure. This

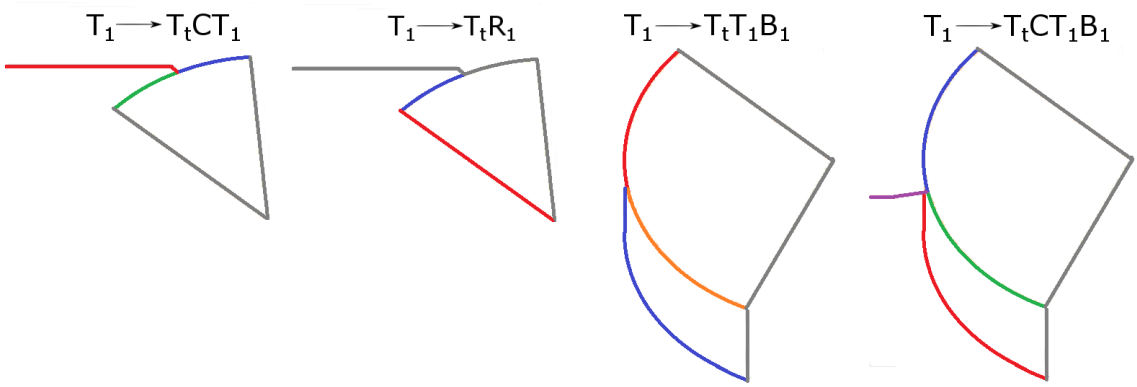


Figure 6.8: The location of the rules for T_1.

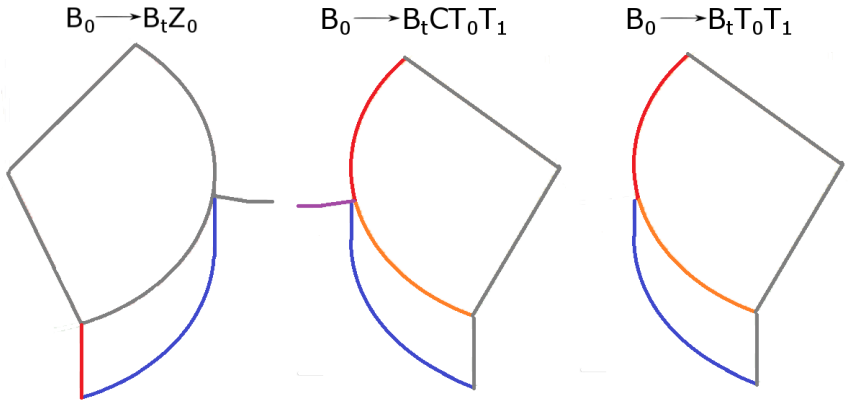


Figure 6.9: The location of the rules for B_0.

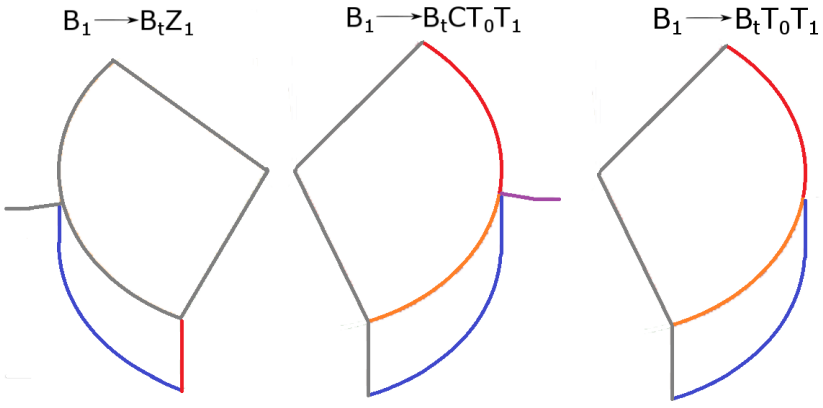


Figure 6.10: The location of the rules for B_1.

is described in more detail in section 6.1.5.

$$C \rightarrow C_t L \qquad C_t \rightarrow c \qquad L \rightarrow l \tag{6.8}$$

Equation 6.9 shows the rules for the R_0 and R_1 nonterminals. The r_0 and r_1 nonterminals indicate the final classification and linking of the two different radii curves.

$$R_0 \rightarrow r_0 \qquad R_1 \rightarrow r_1 \tag{6.9}$$

The rules for the Z_0 and Z_1 nonterminals are given in equation 6.10. The only possible connecting curves are the top edge curves and the radius curves. Only one direction is considered for the top edge curves since going in the other direction does not belong to the slice. The z_0 and z_1 terminals indicate the final classification and linking of the two different side curves. An example of the intersections represented by the Z_0 and Z_1 nonterminal can be seen in figure 6.11.

$$Z_0 \rightarrow Z_{t0} T_1 R_0 \qquad Z_1 \rightarrow Z_{t1} T_0 R_1 \qquad Z_{t0} \rightarrow z_0 \qquad Z_{t1} \rightarrow z_1 \tag{6.10}$$

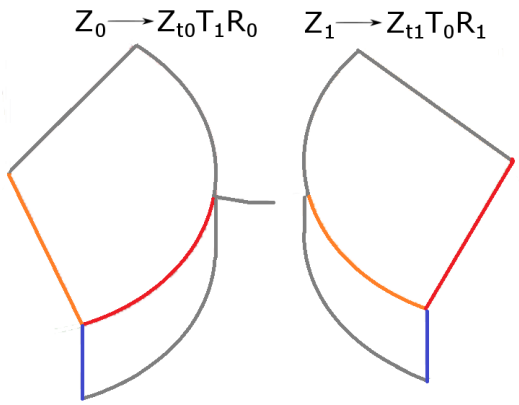


Figure 6.11: The location of the rules for Z_0 and Z_1.

The last set of rules, which are seen in equation 6.11, are different than the other rules. During the preprocessing step it is possible that a single logical curve is split into two different curves. These rules are designed to overcome this by connecting the curves which were split. These rules are used when the only curve detected is a curve at 180 degrees from the initial curve.

$$\begin{array}{llll} T_0 \rightarrow T_t T_0 & T_1 \rightarrow T_t T_1 & B_0 \rightarrow B_t B_0 & B_1 \rightarrow B_t B_1 \\ C \rightarrow C_t C & Z_0 \rightarrow Z_{t0} Z_0 & Z_1 \rightarrow Z_{t1} Z_1 & \end{array} \tag{6.11}$$

6.1.4 Pie Chart Example Derivation

This subsection provides a simple derivation to help the reader understand how the grammar operates. This example will classify and link all the curves in a single pie slice. At each step, which rule is used and why it is used is presented. The algorithm will get the same results by substituting the nonterminals in the string in any order. However, for consistency the first nonterminal in the string will always be substituted.

Figure 6.12 gives a pictorial example of the sequence in which the curves are classified and associated with the pie slice. Table 6.4 presents the derivation for the example. Figure 6.13 displays the parse tree which represents the execution order. In this figure, the leaf nodes represent terminals and the interior nodes represent the nonterminals. The colour of the leaf nodes is the same colour as the curve it corresponds to in figure 6.12. For example, the label connection is green in figure 6.12 and the terminal which represents the label connection is green in figure 6.13. The leaf nodes which do not have a colour do not correspond to a curve. The order of the terminals at the bottom of the parse tree is the same order that the components are classified. The remainder of the section explains the content of figures 6.13 and 6.12 and table 6.4.

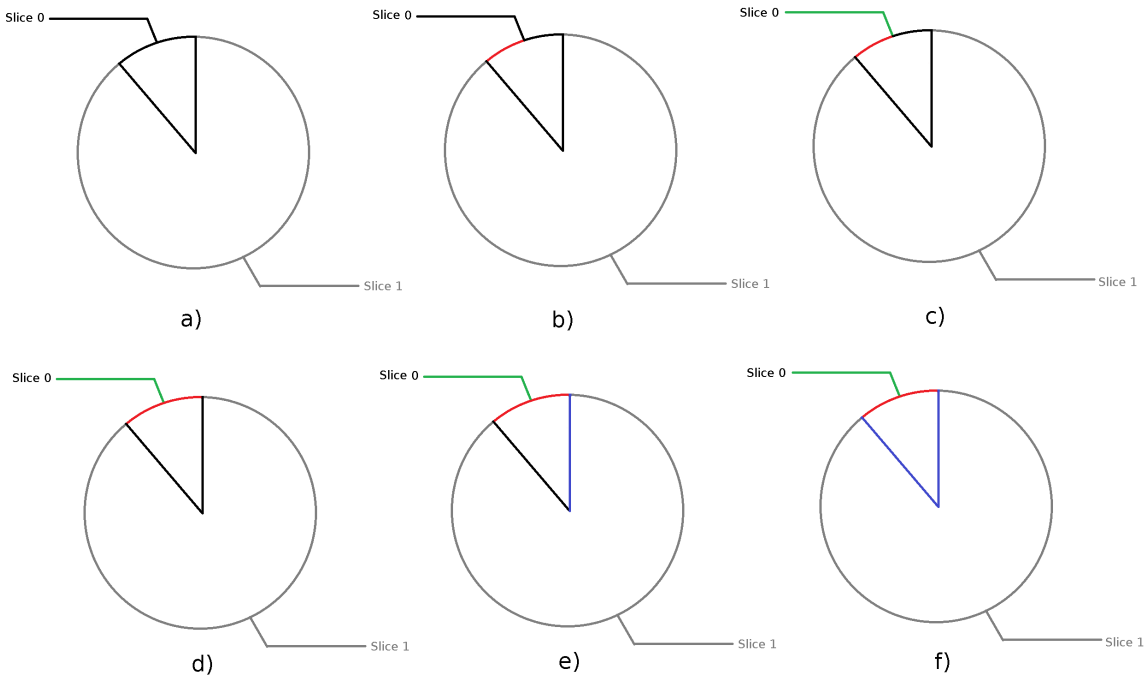


Figure 6.12: The order in which the curves are classified and linked to the pie slice. It is best if this figure is viewed in colour.

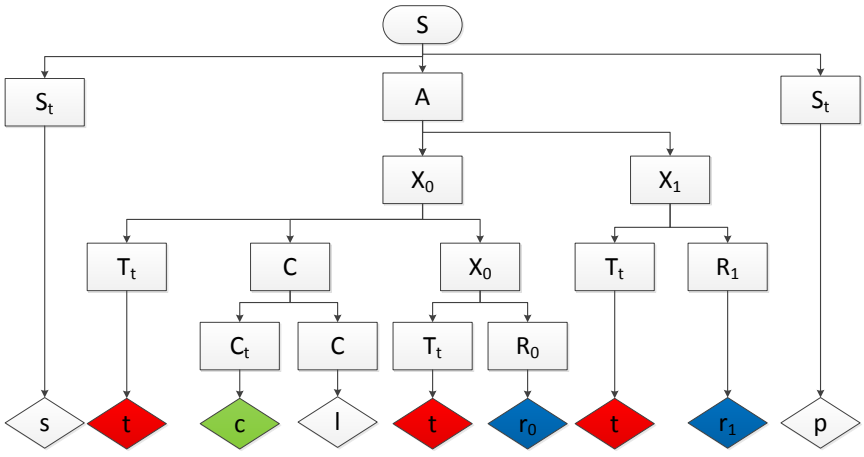


Figure 6.13: This tree presents the order in which the substitution rules are applied. The colour's of the nonterminal nodes are the same colour as the curves they classify in figure 6.12. It is best if this figure is viewed in colour.

Table 6.4: Process flow example.

	String	Substitution Rule	Fig. 6.12 sub-figure	Comment
0	S		b	Start nonterminal, curve on ellipse detected.
1	S_tAP_t	$S \rightarrow S_tAP_t$	b	
2	sAP_t	$S_t \rightarrow s$	b	Pie slice data structure created.
3	$sT_0T_1P_t$	$A \rightarrow T_0T_1$	b	
4	$sT_tCT_0T_1P_t$	$T_0 \rightarrow T_tCT_0$	b	There is a curve at 270 degrees and a curve at 180 degrees.
5	$stCT_0T_1P_t$	$T_t \rightarrow t$	b	Left half of top curve is classified.
6	$stC_tLT_0T_1P_t$	$C \rightarrow C_tL$	c	The label connection is not split into two, no additional curve detected at the end of label connection.
7	$stcLT_0T_1P_t$	$C_t \rightarrow c$	c	Label connection classified.
8	$stclT_0T_1P_t$	$L \rightarrow l$		Text label classified. Not shown in figure 6.12.
9	$stclT_tR_0T_1P_t$	$T_0 \rightarrow T_tR_0$	d	One curve at 90 degrees and another curve at 180 degrees.
10	$stcltR_0T_1P_t$	$T_t \rightarrow t$	d	Right half of top curve is classified.
11	$stcltr_0T_1P_t$	$R_0 \rightarrow r_0$	e	Radius curve is classified.
12	$stcltr_0T_tR_1P_t$	$T_1 \rightarrow T_tR_1$	e	One curve at 270 degrees and one curve at 180 degrees.
13	$stcltr_0tR_1P_t$	$T_t \rightarrow t$	e	Right half of top curve is classified. This step is redundant since this curve has already been classified.
14	$stcltr_0tr_1P_t$	$R_1 \rightarrow r_1$	f	Radius curve classified.
15	$stcltr_0tr_1p$	$P_t \rightarrow p$		Pattern or colour detected. Not shown in figure 6.12.

The derivation starts by finding a single unclassified curve on the outside of the pie chart. One can search the curves in any order to find an initial curve. In this example it is assumed that the red curve in figure 6.12b is the initial curve found on the circle.

Equation 6.2 presents the rules for the start nonterminal. There is only one possible rule for substituing the S nonterminal, the equation $S \rightarrow S_t A P_t$ (see line 1 in table 6.4). The character string is now $S_t A P_t$. The next nonterminal to substitute is the S_t nonterminal. There is only one substitution rule for the S_t nonterminal the rule $S_t \rightarrow s$. The s terminal indicates a new slice data structure needs to be created (see line 2 of the derivation 6.4). The character string is now sAP_t (see line 2 in table 6.4).

In order to substitute the A nonterminal, the $A \rightarrow T_0 T_1$ rule is used because the chart is a 2D pie chart. It is known that the chart is 2D since the bounding ellipse is circular. The string is now $sT_0T_1P_t$ (see line 3 in table 6.4).

The next step is to substitute the T_0 nonterminal. Recall that the T_0 nonterminal represents the top curve seen in figure 6.12. To substitute the T_0 nonterminal which curves are connected to the top curve are detected. The end point which is closest to the positive x-axis in the clockwise direction is used. The curve has connections at 270 degrees and 180 degrees. Table 6.3 shows that the label connection will always be at 270 degrees and the continuation of the top curve (T_0) will be at 180 degrees. Two nonterminals, C and T_0, should be considered for further linking. Thus the $T_0 \rightarrow T_t C T_0$ substitution rule is used (line 4 table 6.4). The string is now $sT_tCT_0T_1P_t$.

The next step in the derivation is to substitute the T_t nonterminal. The only rule for substituting the T_t nonterminal is the $T_t \rightarrow t$ rule (equation 6.4) and yields the t terminal (line 5 in table 6.4). The t terminal indicates adding the curve as a top curve to the slice data structure. The string is now $stCT_0T_1P_t$.

The C nonterminal represents the curve which connects the pie slice to the text label (see the green curve in figure 6.12c). The far end of this curve is not connected to any other curve and so the $C \rightarrow C_t L$ substitution rule is used (line 6 in table 6.4). Using this rule yields the string $stC_tLT_0T_1P_t$. There is only one rule for substituting the C_t nonterminal, the $C_t \rightarrow c$ rule (line 8 in table 6.4). The c terminal indicates adding the label connection curve to the slice object. There is also only one substitution rule for the L nonterminal. This is the $L \rightarrow l$ substitution rule and it indicates adding the pie slice label to the chart data structure(line 8 in table 6.4). The slice label is found by searching for the text block at the end of the label connection

curve. This is described in greater detail in section 6.1.5. The character string is now $stclT_0T_1P_t$.

The T_0 nonterminal corresponds to the second top curve connected to the original top curve. This is seen in figure 6.12d. To substitute the T_0 nonterminal the algorithm needs to detect the angles between the top curve and the curves connected to it. Recall that the T_0 nonterminal corresponds to a clockwise traversal. The curve has a connection at 90 degrees and so the substitution rule $T_0 \rightarrow T_tR_0$ is used (see line 9 in table 6.4). This results in the $stclT_tR_0T_1P_t$ character string.

The next step is to substitute the T_t nonterminal. This nonterminal also appeared earlier in this example. The only rule for substituting it is the $T_t \rightarrow t$ rule (see line 10 in table 6.4). Substituting the R_0 nonterminal uses the substitution rule $R_0 \rightarrow r$. This is the only rule for substituting the R_0 nonterminal. The string is now $stcltr_0T_1P_t$ (see line 11 in table 6.4).

The T_1 nonterminal represents the left side of the top curve. This curve can be seen in figure 6.12b. The reason it represents this curve and not the right side of the top curve is because the T_1 was added to the character string when the left side of the top curve was evaluated. In the parse tree, seen in figure 6.13 one will notice that the T_1 nonterminal shares a parent with the first T_0 nonterminal. Since this is the T_1 nonterminal it corresponds to a traversal in the counterclockwise direction (see figure 6.6). Thus which curves are connected to this curve on the left side need to be determined. Since there is a curve at 270 degrees, the $T_1 \rightarrow T_tR_1$ substitution rule is chosen (see line 12 in table 6.4). The resulting character string is $stcltr_0T_tR_1P_t$.

The T_t nonterminal has been encountered twice already in this example and there is only one rule for substituting it, the $T_t \rightarrow t$ substitution rule (see line 13 in table 6.4). The t terminal corresponds to adding the initial curve to the chart object. This curve has already been added to the chart structure with the same classification and so it is ignored. The character string is now $stcltr_0tR_1P_t$. In order to substitute the R_1 nonterminal the $R_1 \rightarrow r_1$ substitution rule is used (see line 14 in table 6.4). This is the only substitution rule for this nonterminal and results in adding the r_1 curve to the chart object. The location of this curve can be seen in figure 6.12f. The final nonterminal which needs to be substituted is the P_t nonterminal. The only substitution rule for this nonterminal is the $P_t \rightarrow p$ rule (see line 15 in table 6.4). This rule implies adding the chart pattern or colour to the chart object. The pattern or colour is not a curve and so it is not presented in figure 6.12. This process is described in more detail in section 6.1.5. The final string is $stcltr_0tr_1p$. The process

is finished since there are no nonterminals remaining in the string.

6.1.5 Arc Labels and Pie Slice Colour

The substitution rule associated with the L nonterminal is unique in that it is used to classify the text label associated with the pie slice. It is known that each pie slice has either one label or no label. It is also known that if a label connection curve is present, there must also be a label. At this stage it is assumed that the text block has already been segmented from the document. The algorithm detects the correct label by finding the first text block intersected by an imaginary line which continues from the end of the label connection. An example of this process can be seen in figure 6.14.

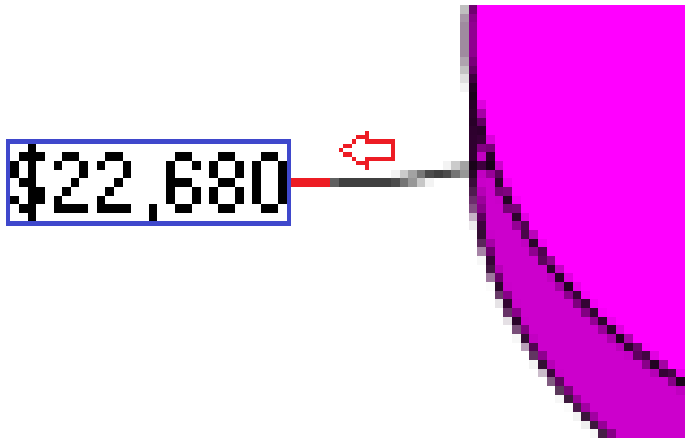


Figure 6.14: A zoomed in section of a pie chart. The red arrow above the label connection curve indicates the direction the algorithm searches for the pie slice label. The pie slice label is within the blue bounding box. The short red line indicates the imaginary line which is followed to find the text block. It is best if this figure is viewed in colour.

The terminal p recognizes which colour belongs with this pie slice. This is done by finding the colour of the region adjacent to the initial detected curve.

6.2 Axis Charts Grammar

The axis grammar is used to classify the components of 2D and 3D bar charts as well as line charts. Each derivation will classify all the components of a single chart image.

Similar to the pie chart grammar, the grammar for the axis charts are context-free and in Chomsky normal form.

In an axis chart there are three different primitive types: the bars are solid regions, the gridlines, axes, and polylines are curves, and the indices and numerical data are text blocks.

Figure 6.15 displays the components of a bar chart. This figure is the same as figure 5.4 and is repeated for ease of viewing.

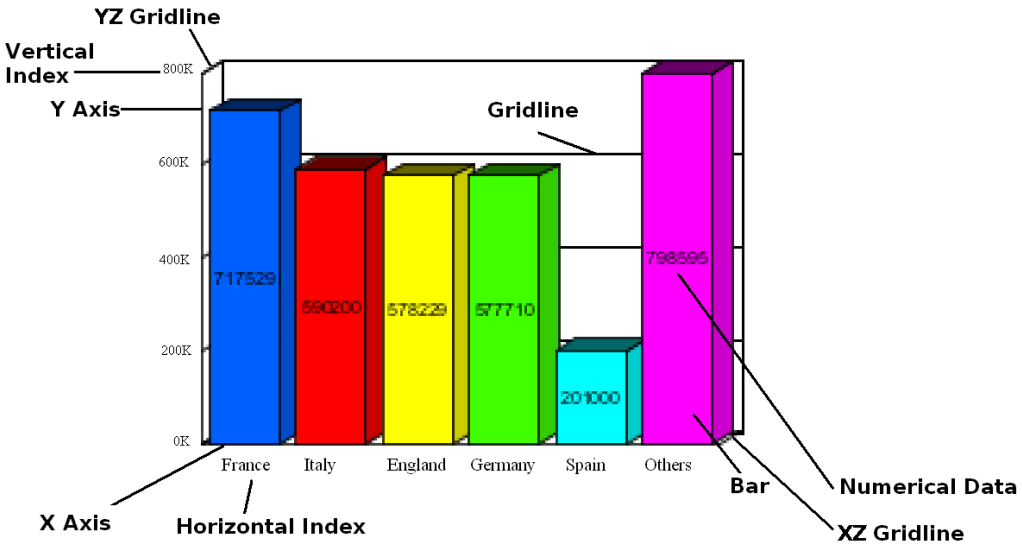


Figure 6.15: The primitives of a 3D bar chart. It is best if this figure is viewed in colour.

The primitives are classified by their spatial location and orientation relative to a previously classified primitive. At each step, a previously classified primitive is used to classify another primitive. The newly classified primitive will be used to classify another primitive. This process continues until all the primitives of the same class are classified.

Since there is no limit on the total number of some chart components, there is also no limit on the length of an output string. For example, some charts have 1 bar and some charts have 100 bars. Recursive substitution rules are used when there is an arbitrary number of components.

In a vertical bar chart the text directly below the bar is associated with the bar. That is, each bar has a single text block which describes the bar. Similarly, in a horizontal bar chart the text to the left of the bar is associated with the bar.

The terms *vertical bar text* and *horizontal bar text* are used to reference these terms respectively.

Figure 6.16 explains the dual role that a given text primitive can play, namely horizontal index and vertical bar text. For instance, the word "Austria" is both a horizontal index and a vertical bar text whereas the word "Finland" is only a horizontal index. "Finland" is not a horizontal bar text because there is no bar above it.

The text primitives associated with a bar are classified twice, as bar text (see section 6.2.7), and indices (see sections 6.2.4 and 6.2.5).

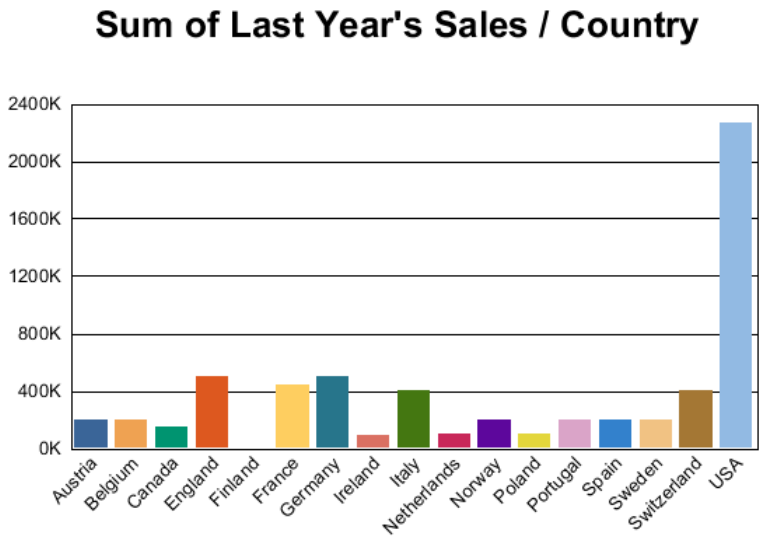


Figure 6.16: A bar chart with textual primitives that fulfill a dual role (horizontal index and vertical bar text). The word "Finland" is only a horizontal index because there is no bar above it.

This section is structured as follows. Subsection 6.2.1 outlines the preprocessing steps performed on the chart image. The axis chart terminals are presented in 6.2.2. The remaining sections present the substitution rules which are used to classify the different chart components. The substitution rules which initiate the detection and classification process are presented in 6.2.3. Subsections 6.2.4 and 6.2.5 outline the process for detecting the vertical and horizontal bars as well as the horizontal and vertical indices. The axis curves as well as the gridlines detection rules are presented in subsection 6.2.6. The rules in 6.2.7 detect the 3D bar structure, the vertical and horizontal bar text, and the numerical data. The chapter concludes with the

substitution rule which detects the polyline in subsection 6.2.8.

6.2.1 Preprocessing

This section outlines the preprocessing steps required for the axis charts. There are three different preprocessing procedures which are independent of each other. The gridline preprocessing (section 6.2.1) is only used by the substitution rules in subsection 6.2.6 which detect the gridlines and the axis. The polyline preprocessing is used in subsection 6.2.8 which detects the polyline. The background subtraction section is used by all the sections.

In addition to the preprocessing techniques outlined in the following subsections, the area Voronoi segmentation algorithm [29] is applied to the detected charts. This is done so that all the text indices are detected.

Gridline Preprocessing

A common problem with the bar charts in the database is that the bars often occlude the axis. Figure 6.17 shows an example of the occlusion by zooming in on the region.

Figure 6.17: A zoomed-in view of two bars occluding and the horizontal axis. This figure is from the same chart as figure 6.19. This figure should be viewed in colour.

This is solved by creating an image which contains the contours of the bars. A flow chart outlining this process is seen in figure 6.18. The process starts by performing a morphological erode operation on the chart. Since the gridlines, axis, and text are thin lines, this removes everything except the bars. The image is subtracted from the original image. An example of the resulting image can be seen in figure 6.19. The subtracted image is then thinned and vectorized using the method presented in section 5.2.3. The resulting image looks very similar to the image in figure 6.19 and so it is not presented here.

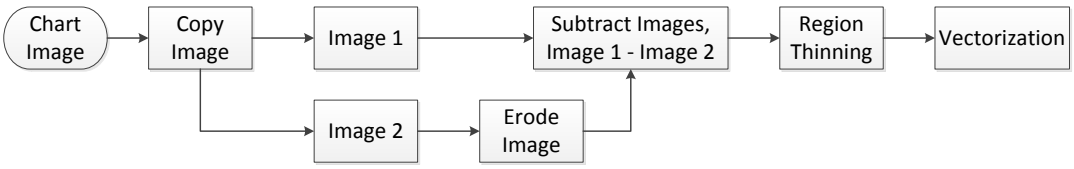


Figure 6.18: The process for detecting the curves and lines.

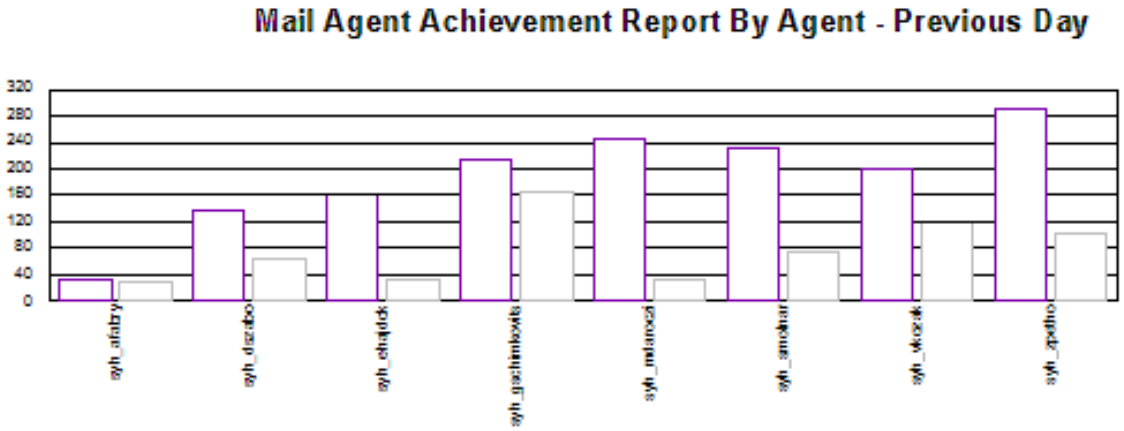


Figure 6.19: The chart without the bars but with the axis intact.

Polyline Extraction

The approach presented in this thesis separates polylines from each other on charts which contain more than one polyline. Many line charts use colour so that a human reader can distinguish between the different polylines. This approach uses also uses colour to distinguish the polylines. A limitation of this approach is that it does not work if the image is in grey-scale or if all the polylines are the same colour.

Figure 6.20 shows a flow chart with the preprocessing steps. The first step is to convert the image from the RGB colour spectrum to the HSV colour spectrum. This colour spectrum is used because only the hue is of interest. The intensity values found along the polyline are not consistent. To prevent the chart bars from being detected, the solid regions are removed from the image. This is done by setting all the pixels in the solid region to white.

The next step is to determine which hues are present within the chart image. Since there can be minor fluctuations on the same curve, a threshold is needed to determine if two hues are similar enough to be considered the same. A threshold of 3 was chosen as it was found to work well on the entire experimental database.

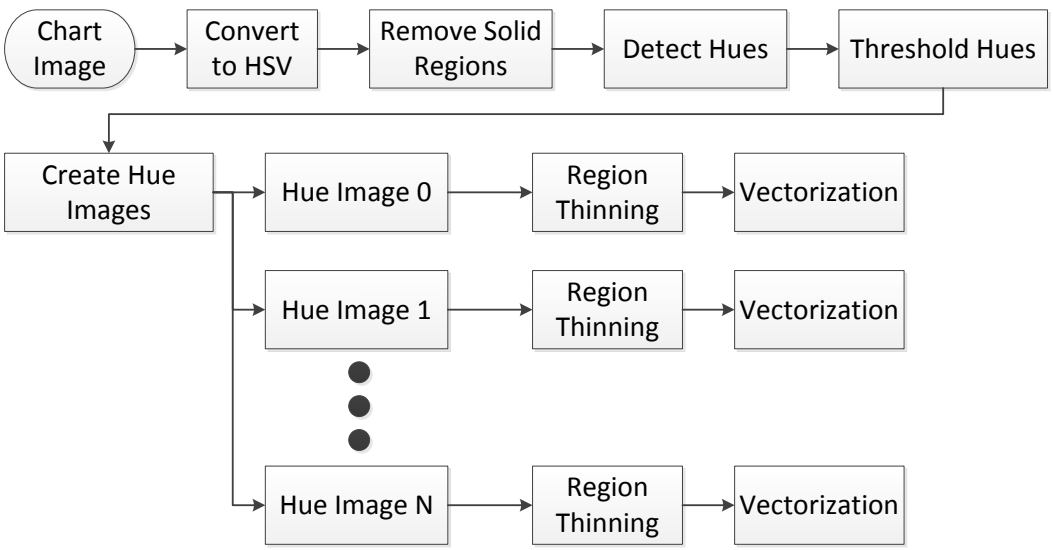


Figure 6.20: The preprocessing steps for finding the polyline curves. It is assumed that there are N distinct hues in the chart image.

We need to determine the number of distinct hues because this corresponds to the number of distinct polylines. Intersections between a polyline and other graphics contain pixels of variable hues, which are not representative for that polyline. These variations are ignored by counting and comparing the number of pixels represented by each hue. Hues represented by a number of pixels which is less than half of the number of pixels associated with the most frequent hue are ignored.

A separate image is created for each of the N remaining distinct hues. Region thinning and vectorization are performed on the images (see section 5.2.3). This results in N arrays of curves, where the curves in each array have the same hue.

Background Detection

Although most charts in the database have a white background, some charts are different. An example of a chart with a non-white background can be seen in figure 6.21. This is problematic as the background is detected as a solid region. Since the background is behind the gridlines, the background is split into many small regions. Without removing the background first, it can be difficult to distinguish between the background and the chart bars.

The background is removed by looking for any colour that covers more than 20%

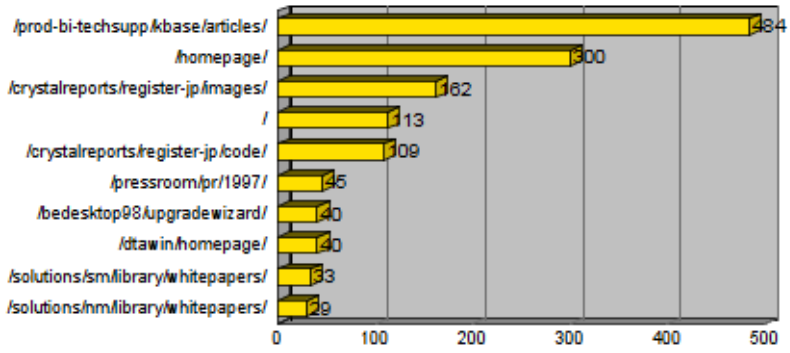


Figure 6.21: An example of a chart with a non-white background.

of the chart region.

6.2.2 Axis Chart Terminals

The presence of a terminal during a derivation signifies that a chart primitive is classified. Each terminal corresponds to a specific classification. For example, the terminal x corresponds to adding a section of the x-axis to the chart object.

Every rule which includes a terminal is of the form $A \rightarrow b$. That is, a single nonterminal is substituted by a terminal. This format is used so that the rules are in Chomsky normal form.

The terminals are presented in three tables, namely tables 6.5, 6.6, and 6.7. Table 6.5 is presented first and lists the terminals for the horizontal and vertical indices.

Table 6.5: The terminals for the vertical and horizontal indices.

Terminal Symbol	Nonterminal Symbol	Common Name	Primitive Classified	Description
t_{VI}	T_{VIt}	Text, Vertical Index	Text	Classifies the vertical indices.
t_{HI}	T_{HIt}	Text, Horizontal Index	Text	Classifies the horizontal indices.

The terminals for the axis and the gridlines are in table 6.6. There are six different classes of curves found within the grid structure of a chart. These are the X and Y axis, the vertical and horizontal gridlines, and the XZ and YZ gridlines (3D chart). A visual description of these components can be seen in figures 3.5 and 3.6 in chapter 3.

Table 6.6: The terminals for the axis and the gridlines.

Terminal Symbol	Nonterminal Symbol	Common Name	Primitive Classified	Description
x	X_t	X-Axis	Line	Classifies the x-axis
y	Y_t	Y-Axis	Line	Classifies the y-axis
g_V	G_{Vt}	Vertical Gridline	Line	Classifies the vertical gridline.
g_H	G_{Ht}	Horizontal Gridline	Line	Classifies the horizontal gridline
g_{XZ}	G_{XZt}	XZ Gridline	Line	Classifies the XZ gridline.
g_{YZ}	G_{YZt}	YZ Gridline	Line	Classifies YZ gridline

The vertical and horizontal bars have 12 terminals associated with them. These are presented in table 6.7. The first six terminals are associated with vertical bar charts and the last six are associated with horizontal bar charts. Within each of these two groups of six terminals, the first four terminals correspond to the bars and their 3D structure and the last two correspond to the text elements associated with each bar. See figure 3.5 in chapter 3 for a visual example of these chart components.

The terminal for the polyline is not presented in a table because there is only one terminal. The nonterminal and terminal for the polyline are P_t and p respectively.

Table 6.7: A list of the axis chart terminals for the bar structures and the nonterminals which are associated with them.

Terminal Symbol	Nonterminal Symbol	Common Name	Primitive Classified	Description
b_x	B_{Xt}	Vertical bar	Solid region	Bar classification, vertical bar chart
b_{xt}	B_{XTt}	Bar - X, Top	Solid region	Top of vertical bar.
b_{xr}	B_{XRt}	Bar - X, Right	Solid region	Right side of vertical bar.
b_{xl}	B_{XLt}	Bar - X, Left	Solid region	Left side of vertical bar.
t_{xhi}	T_{XHIt}	Text, Horizontal index	Text	Classifies the horizontal index directly below the vertical bar.
t_{xnd}	T_{XNDt}	Text, numerical data.	Text	Detects the numerical data.
b_y	B_{Yt}	Horizontal bar	Solid region	Bar classification, horizontal bar chart
b_{yt}	B_{YTt}	Bar - Y, Top	Solid region	Top of horizontal bar.
b_{yr}	B_{YRt}	Bar - Y, Right side	Solid region	Right side of horizontal bar.
b_{yb}	B_{YBt}	Bar - Y, Bottom	Solid region	Bottom of bar.
t_{yvi}	T_{YVIt}	Text, Vertical index	Text	Detects the text directly to the left the bar.
t_{ynd}	T_{YNDt}	Text, numerical data	Text	Detects the numeral data

6.2.3 The Start Nonterminal and the Initial Substitution Rule

The start nonterminal and the initial substitution rule initiate the derivation and the classification of the primitives. Equation 6.12 displays the start rule and table 6.8 presents a description of what each nonterminal represents. The substitution rule does not classify any primitives, instead it initiates four disjoint subsets of substitution rules, which are rooted by the V, H, G, and P nonterminals. Each of these four subsets are used to classify different chart components and are discussed in separate subsections. Figure 6.22 displays the parse tree for this rule.

$$S \rightarrow VHGP \tag{6.12}$$

Table 6.8: The nonterminals which appear in the initial substitution rule.

Symbol	Common Name	Description
S	Start	The start nonterminal.
V	Vertical bars	This subset will classify vertical bars and horizontal indices.
H	Horizontal bars	This subset will classify horizontal bars and vertical indices.
G	Grid	This subset will classify the axis and all the gridlines.
P	Polyline	This subset will classify the polyline.

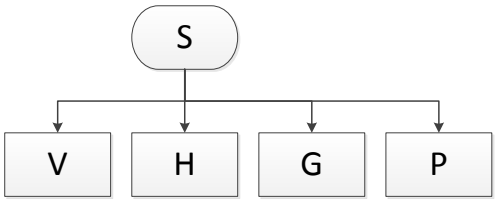


Figure 6.22: This parse tree represents the start of the axis chart grammar. The V, H, G, and P initiate four disjoint subsets which classify different components.

6.2.4 Substitution Rules for the Vertical Bars and Horizontal Indices

This subsection describes the classification of components associated with the x-axis, but not the axis itself. These components are the vertical bars and the horizontal indices. These rules are initiated with V nonterminal.

This set of rules are split into two subsets, one for the vertical bars and one for the horizontal indices. These are initiated with the B_V and T_{HI} nonterminals respectively.

The nonterminals for both the vertical bars and horizontal indices are presented in table 6.9. Every nonterminal except the B_X nonterminal results in the detection of an additional component of the chart. For example, the B_V nonterminal detects a single bar of the bar chart. The B_X nonterminal is discussed later in section 6.2.7.

Table 6.9: A list of the axis chart nonterminals on the vertical subset.

Symbol	Common Name	Primitive Detected	Primitive Required	Description
V	Vertical subset	Solid region	Line (x-axis)	Vertical bars and horizontal indices.
B_V	Vertical bar	Solid region		Detects a single vertical bar.
B_X	Vertical Bar		Solid region	Detects components of a single bar.
B_{VL}	Bar, vertical, Left	Solid region	Solid region	Detects bar to the left of another bar.
B_{VR}	Bar, vertical, Right	Solid region	Solid region	Detects bar to the right of another bar.
T_{HI}	Text, horizontal, index.	Text	Line (x-axis)	Detects a horizontal index.
T_{HIL}	Text, horizontal index, left	Text	Text	Detects a horizontal index left of a given horizontal index.
T_{HIR}	Text, horizontal, index right	Text	Text	Detects a horizontal index right of a given horizontal index.
T_{HIt}	Text, horizontal index.		Text	Replaced by the t_{HI} terminal.

The V nonterminal is substituted by looking for a single vertical bar (B_V) adjacent to the x-axis and a single horizontal index (T_{HI}). If a vertical bar is detected, the

$V \rightarrow B_V T_{HI}$ substitution rule is used, otherwise the $V \rightarrow T_{HI}$ rule is used. These substitution rules are found in equation 6.13. These rules assume all charts have horizontal indices but not all charts have vertical bars. The parse trees for these rules can be seen in figure 6.23.

$$V \rightarrow B_V T_{HI} \qquad V \rightarrow T_{HI} \tag{6.13}$$

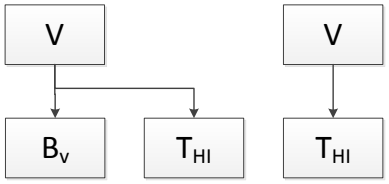


Figure 6.23: A visual representation of the substitution rules for the V nonterminal.

Vertical Bars: B_V

This section describes the subset of rules initiated by the B_V nonterminal. The rules which substitute the B_V nonterminal start the process of detecting the remaining bars. The rules start from the first bar that is identified (when substituting for the V nonterminal) and then go to the left and the right identifying the other bars.

There are four substitution rules for the B_V nonterminal, as seen in equation 6.14. The parse trees for these rules is seen in figure 6.24.

$$B_V \rightarrow B_x B_{VL} B_{VR} \qquad B_V \rightarrow B_x B_{VL} \qquad B_V \rightarrow B_x B_{VR} \qquad B_V \rightarrow B_x \tag{6.14}$$

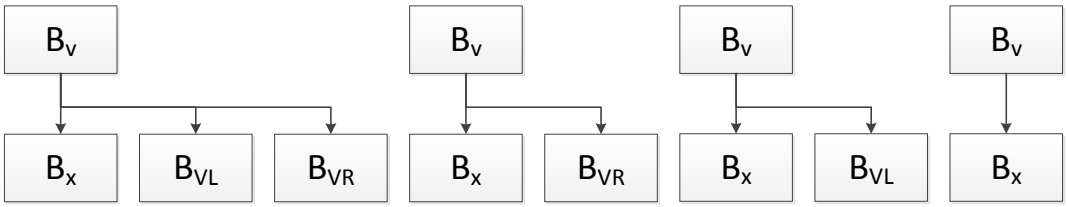


Figure 6.24: A visual representation of the substitution rules for the B_V nonterminal.

The substitution rule which is chosen depends on whether a bar to the left or right is detected. If a bar is detected to the left of the original bar, a rule with the B_{VL} nonterminal is chosen. If a bar is detected to the right of the original bar, a rule with the B_{VR} is chosen. The B_X, which appears in each rule, is discussed in section 6.2.7.

An example is presented to demonstrate the classification order. The green bar in figure 6.25 is assumed to be the first bar detected. The detection of this bar corresponds to the substitution of V by B_V (ie. $V \rightarrow B_V T_{HI}$). Substituting the B_V nonterminal is dependent on the presence of other bars to the left and right of the first bar. Since there is a bar to the left of the green bar and a bar to the right of the green bar (the yellow bar) the rule with both B_{VL} and B_{RL} is used. This is the rule $B_V \rightarrow B_X B_{VL} B_{VR}$.

The B_{VL} and B_{VR} are used to detect bars to the left and right respectively. To substitute these nonterminals, the substitution rules in equation 6.15 are used. These rules are recursive such that each recursive substitution detects an additional bar. When there are no additional bars to the left or right, the $B_{VL} \rightarrow B_x$ or $B_{VR} \rightarrow B_x$

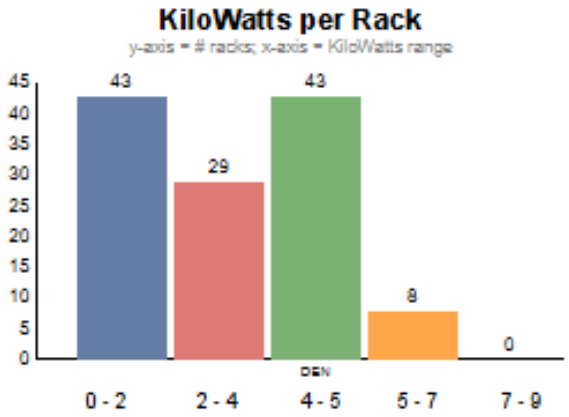


Figure 6.25: This chart is used to show the order in which the bars are classified.

is used. The parse trees for these rules is seen in figure 6.26.

$$B_{VL} \to B_x B_{VL} \qquad B_{VL} \to B_x \qquad B_{VR} \to B_x B_{VR} \qquad B_{VR} \to B_x \tag{6.15}$$

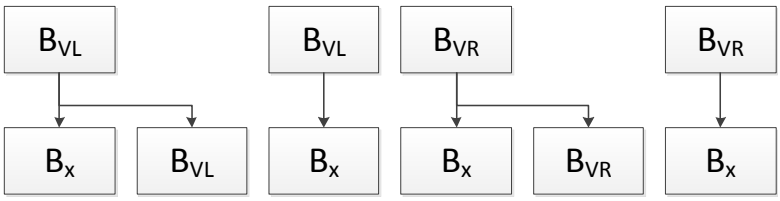


Figure 6.26: A visual representation of the substitution rules for the B_{VL} and B_{VR} nonterminals. The B_{VL} nonterminal indicates search to the left and the B_{VR} nonterminal indicates a search to the right.

Horizontal Indices: T_{HI}

These rules classify the horizontal indices and are initiated by the T_{HI} nonterminal. The rules which substitute the T_{HI} nonterminal start the process of detecting the remaining horizontal indices. These rules operate by detecting a single index and then searching left and right to detect the remaining indices.

There are four different rules which can be used to substitute the T_{HI} nonterminal as seen in equation 6.16. If a horizontal index is detected to the left of the original index, a rule with the T_{HIL} is chosen. If a horizontal index is detected to the right of the original index, a rule with the T_{HIR} is chosen. The parse trees for these rules can be seen in figure 6.28.

$$T_{HI} \rightarrow T_{HIt}T_{HIL}T_{HIR} \qquad T_{HI} \rightarrow T_{HIt}T_{HIL} \qquad T_{HI} \rightarrow T_{HIt}T_{HIR} \qquad T_{HI} \rightarrow T_{HIt} \tag{6.16}$$

An example is presented to demonstrate the classification order. The index with the red bounding box in figure 6.27 is assumed to be the first index detected. This index is detected during the substitution of V. Substituting the T_{HI} nonterminal requires knowing if there are indices beside the initial index. Since there is a index to the left of the initial index (the index with blue bounding box) and an index to the right of the initial index (the index with the green bounding box) the rule with both T_{HIL} and T_{HIL} is used. This is the rule $T_{HI} \rightarrow T_{HIt}T_{HIL}T_{HIR}$.

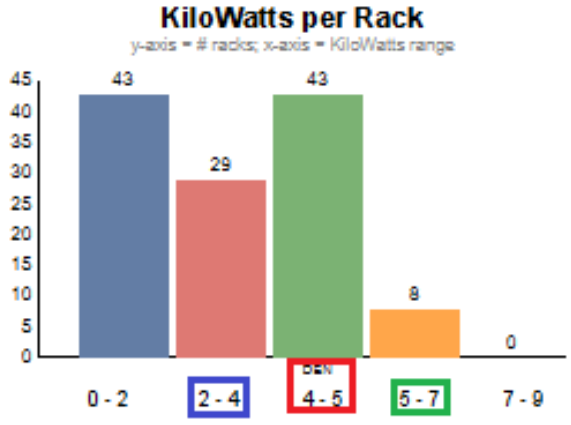


Figure 6.27: The chart analyzed in the example derivation.

The T_{HIR} and T_{HIL} nonterminals are used to detect indices to the left and the

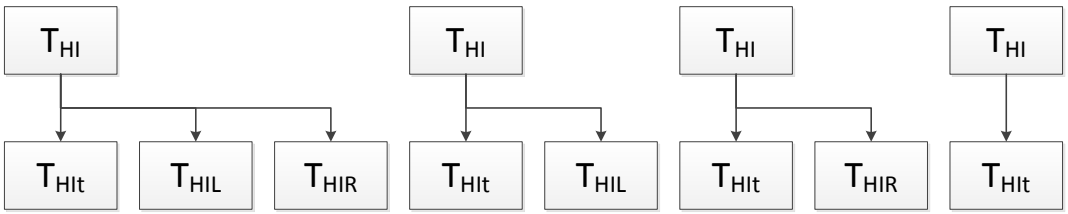


Figure 6.28: This is the subset for the T_{HI} nonterminal. This tree detects the horizontal indices. These rules are continued in figure 6.29.

right respectively. These nonterminals are substituted using the rules in equation 6.17. Similar to the rules for the bars, these rules are recursive such that each additional substitution detects an additional index. When there are no additional indices to the left or the right, the rules $T_{HIL} \rightarrow T_{HIt}$ or $T_{HIR} \rightarrow T_{HIt}$ are used.

$$T_{HIL} \rightarrow T_{HIt}T_{HIL} \qquad T_{HIL} \rightarrow T_{HIt} \qquad T_{HIR} \rightarrow T_{HIt}T_{HIR} \qquad T_{HIR} \rightarrow T_{HIt} \quad (6.17)$$

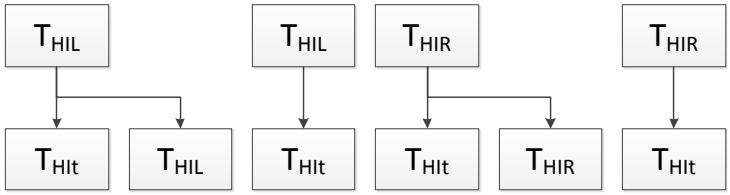


Figure 6.29: This is the subset for the T_{HIR} and T_{HIL} nonterminals. These trees are associated with the trees in figure 6.28.

6.2.5 Substitution Rules for the Horizontal Bars and Vertical Indices

This subsection describes the classification of the components associated with the y-axis, but not the axis itself. These components are the horizontal bars and vertical indices. These rules are initiated with the H nonterminal.

These rules are split into two subsets, one for the horizontal bars and one for the vertical indices. These are initiated with the B_H and T_{VI} nonterminals respectively.

The nonterminals for both the horizontal bars and the vertical indices are presented in table 6.10. Every nonterminal, except the B_Y nonterminal, results in the detection of an additional component of the chart. For example, the T_{VI} nonterminal detects a single vertical index. The B_Y nonterminal is discussed in section 6.2.7.

Table 6.10: A list of the axis chart nonterminals on the horizontal subset.

Symbol	Common Name	Primitive Detected	Primitive Required	Description
H	Horizontal subset	Solid region	Line (y-axis)	Horizontal subset
B_H	Horizontal bar	Solid region	Line (y-axis)	Detects a single horizontal bar.
B_Y	Horizontal Bar		Solid region	Detects sub-components of a single bar.
B_{HU}	Bar, horizontal, Up	Solid region	Solid region	Detects bar above another bar.
B_{HD}	Bar, horizontal, Down	Solid region	Solid region	Detects bar below another bar.
T_{VI}	Text, vertical index.	Text	Line (y-axis)	Detects a vertical index.
T_{VIU}	Text, vertical, index up	Text	Text	Detects a vertical index above a given vertical index.
T_{VID}	Text, vertical, index down	Text	Text	Detects a vertical index below a given vertical index.
T_{VIt}	Text, vertical index		Text	Replaced by the t_{VI} terminal

The H nonterminal is substituted by looking for a single horizontal bar (B_H) adjacent to the y-axis and a single vertical index (T_{HI}). If a horizontal bar is detected the $H \rightarrow B_H T_{VI}$ substitution rule is used, and if a horizontal bar is not detected the $H \rightarrow T_{HI}$ substitution rule is used. These substitution rules are found in equation

6.18. These rules assume all charts have vertical indices but not all charts have horizontal bars. A visual representation of these rules can be seen in figure 6.30.

$$H \rightarrow B_H T_{VI} \qquad H \rightarrow T_{VI} \tag{6.18}$$

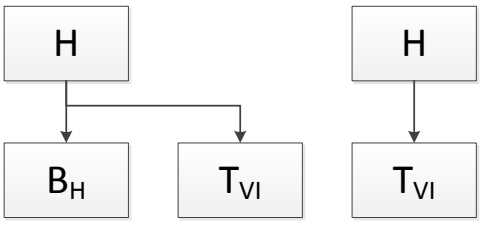


Figure 6.30: A visual representation of the substitution rules for the H nonterminal.

Horizontal Bars: B_H

This section describes the subset which is initiated by the B_H nonterminal. The process is similar to that of the vertical bars (section 6.2.4), except everything is rotated by 90 degrees. The rules which substitute the B_H nonterminal start the process of detecting the remaining bars. These rules start from the first bar that is identified (when substituting the H nonterminal) and then go up and down to identify the other bars.

To substitute the B_H nonterminal there are four possible rules as seen in equation 6.19. The parse tree for these rules is seen in figure 6.31.

$$B_H \rightarrow B_Y U D \qquad B_H \rightarrow B_Y U \qquad B_H \rightarrow B_Y D \qquad B_H \rightarrow B_Y \tag{6.19}$$

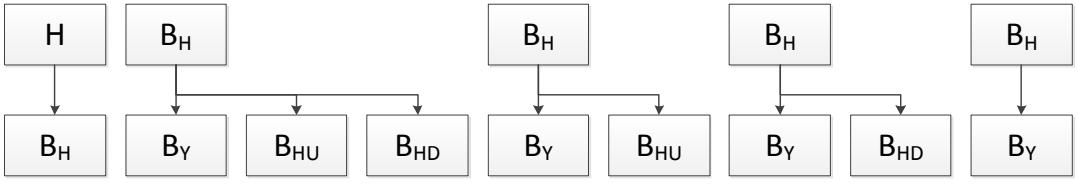


Figure 6.31: These trees show the possible subsets for the horizontal bar charts.

The substitution rule which is chosen depends on whether a bar is identified above the initial bar, below the initial bar, or both. If a bar is detected above the original bar, a rule with the B_{HU} nonterminal is chosen. If a bar is found below the original bar, a rule with the B_{HD} nonterminal is used. The B_Y nonterminal which appears in every rule, is discussed in section 6.2.7.

An example is presented to demonstrate the classification order. The yellow bar in figure 6.32 is assumed to be the first bar detected. The detection of this bar corresponds to the substitution of H by B_H (ie. $H \rightarrow B_H T_{VI}$). Substituting the B_H nonterminal is dependent on the presence of other bars above and below the first bar. Since the red bar is above the initial yellow bar and the green bar is below the initial yellow bar we need to choose a rule with both the B_{HU} and B_{HD} nonterminals. The only rule which fits this criteria is the $B_{HI} \rightarrow B_Y B_{HU} B_{HD}$ rule.

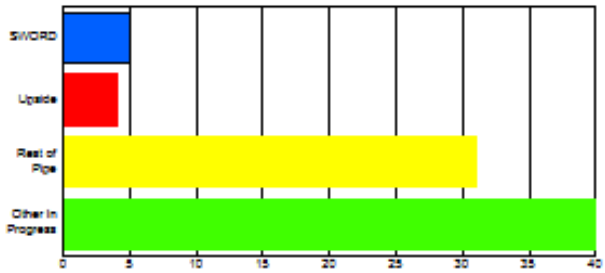


Figure 6.32: This chart is used in the example derivation.

The B_{HU} and B_{HD} are used to detect the bars above and below the first bar respectively. In order to substitute the B_{HU} and the B_{HD} nonterminals the substitution rules in equation 6.20 are used. These rules are recursive such that the same nonterminal can exist on both sides of the rule. When there are no additional bars above or below, the $B_{HU} \rightarrow B_Y$ rule or $B_{HD} \rightarrow B_Y$ rule is used. The parse tree for these rules can be seen in figure 6.33.

$$B_{HU} \rightarrow B_Y B_{HU} \qquad B_{HU} \rightarrow B_Y \qquad B_{HD} \rightarrow B_Y B_{HD} \qquad B_{HD} \rightarrow B_Y \tag{6.20}$$

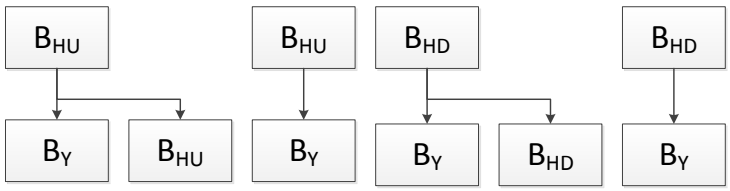


Figure 6.33: This tree detects the remaining bars of a horizontal bar chart.

Vertical Indices: T_{VI}

These rules classify the horizontal indices and are initiated by the T_{VI} nonterminal. The process works by detecting a single index and then searching up and down for the remaining indices.

There are four different rules which can be used to substitute the T_{VI} nonterminal as seen in equation 6.21 and figure 6.35. The rule which is chosen depends on whether a horizontal index is detected above (T_{VIU}), below (T_{VID}), or not at all. If a vertical index is found above the original index, a rule with the T_{VIU} nonterminal is chosen. If a vertical index is found below the original index, a rule with the T_{VID} nonterminal is chosen.

An example to help the reader understand the classification order. The index with the red bounding box in figure 6.34 is the first index detected. Since there is an index below the original index (green bounding box) and an index above the original index (blue bounding box) we need a rule with both the T_{VIU} and the T_{VID} nonterminals. This is the rule $T_{VI} \rightarrow T_{VIt}T_{VIU}T_{VID}$.

$$T_{VI} \rightarrow T_{VIt}T_{VIU}T_{VID} \qquad T_{VI} \rightarrow T_{VIt}T_{VIU} \qquad T_{VI} \rightarrow T_{VIt}T_{VID} \qquad T_{VI} \rightarrow T_{VIt} \tag{6.21}$$

The next step is to search for additional bars and is done using the T_{VIU} nonterminal and the T_{VID} nonterminal. The T_{VIU} rules are used to detect the vertical indices above the original index and the T_{VID} rules are used to detect the vertical indices below the original index. The rules for substituting the T_{VIU} and T_{VID} nonterminals are seen in equation 6.22.

$$T_{VID} \rightarrow T_{VIt}T_{VID} \qquad T_{VID} \rightarrow T_{VIt} \qquad T_{VIU} \rightarrow T_{VIt}T_{VIU} \qquad T_{VIU} \rightarrow T_{VIt} \tag{6.22}$$

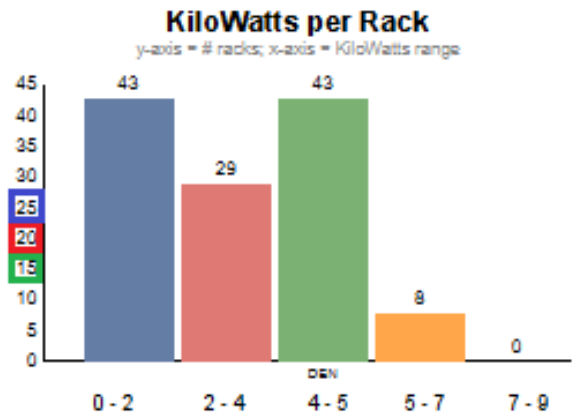


Figure 6.34: This chart is used in the derivation example.

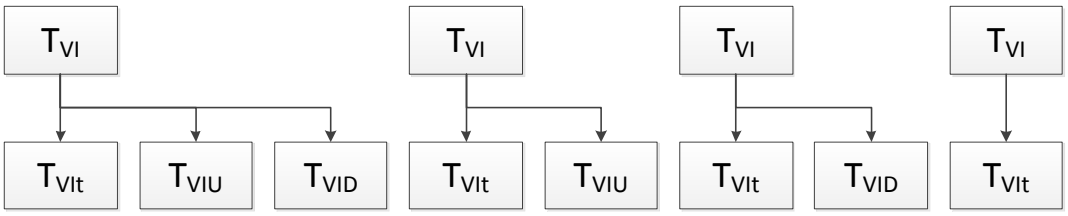


Figure 6.35: This is the subset for the T_{VI} rule. This tree detects the vertical indices. These rules are continued in figure 6.36.

6.2.6 Grid Substitution Rules

The grid substitution rules detect the curves and lines associated with the x and y axes and the gridlines. This includes the horizontal, vertical, the XZ and YZ gridlines. Examples of the XZ and YZ gridlines can be seen in figure 6.37.

The process starts with a single curve on the x-axis and a single curve on the y-axis, and follows the curves connected to them in order to classify the remaining curves. The final result of the derivation using this subset will be a collection of classified curves for the entire grid structure of the chart. It is important to recall from the gridline preprocessing section 6.2.1 that curves end at intersection points. A list of all nonterminals used in this subsection are presented in table 6.11.

The relative angle between the curves is used to determine the classification of the connecting curves. For example, in a 2D chart, the horizontal gridlines will have

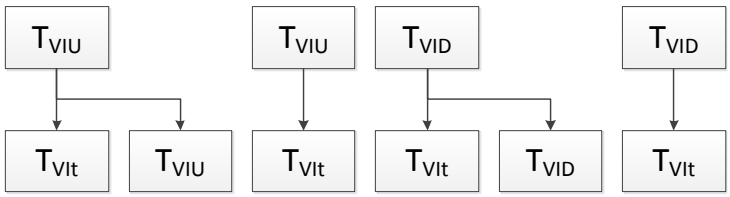


Figure 6.36: This is the subset for the T_{VIU} and T_{VI} nonterminals. These trees are associated with the trees in figure 6.35.

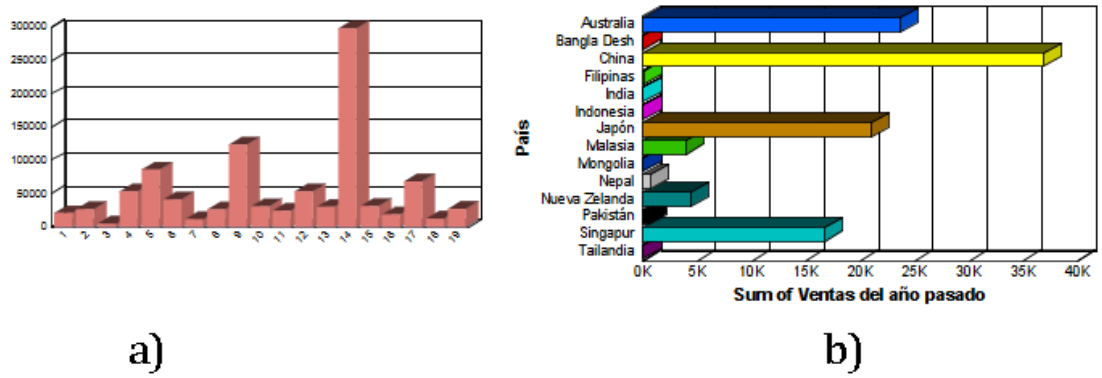


Figure 6.37: The two different Z-Axis gridlines can be seen here. Chart a) contains several gridlines in the yz plane. Chart b) contains gridlines in the xz plane.

an angle of 90 degrees from the y axis. Thus, if a curve is known to be part of the y axis and another curve is at 90 degrees from it, the other curve must be a horizontal gridline. An example of a curve intersection can be seen in figure 6.38.

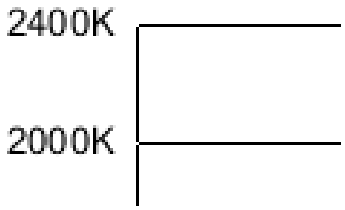


Figure 6.38: An example of a curve intersection.

All parse trees created using these rules will have G as the root nonterminal. The subset of rules described in this section are split into two smaller subsets, one for the x-axis, vertical gridlines and XZ gridlines, and the other for the y-axis, horizontal gridlines and YZ gridlines. These rules are initiated with the A_X (x-axis) and A_Y

Table 6.11: A list of the axis chart nonterminals for the grid structure.

Symbol	Common Name	Primitive Detected	Primitive Required	Description
G	Grid			Grid subset
A_x	X-Axis Start	Line	Line (x-axis)	Detects part of the x-axis, subset for x-axis.
A_y	Y-Axis Start	Line	Line (y-axis)	Detects part of the y-axis, subset for y-axis.
X_0	X-Axis, left	Arc/Line	Arc/Line	Detects a section of the x-axis, going left.
X_1	X-Axis, right	Arc/Line	Arc/Line	Detects a section of the x-axis, going right.
Y_0	Y-Axis, down	Arc/Line	Arc/Line	Detects a section of the y-axis, going down.
Y_0	Y-Axis, up	Arc/Line	Arc/Line	Detects a section of the y-axis, going up.
G_{H0}	Horizontal Gridline, left	Arc/Line	Arc/Line	Detects a horizontal gridline section, going left.
G_{H1}	Horizontal Gridline, right	Arc/Line	Arc/Line	Detects a horizontal gridline section, going right.
G_{V0}	Vertical Gridline, down	Arc/Line	Arc/Line	Detects a vertical gridline section, going down.
G_{V1}	Vertical Gridline, up	Arc/Line	Arc/Line	Detects a vertical gridline section, going up.
G_{XZ}	XZ Gridline	Arc/Line	Arc/Line	Detects a XZ gridline
G_{YZ}	YZ Gridline	Arc/Line	Arc/Line	Detects a YZ gridline

(y-axis) nonterminals.

The rules for substituting the G nonterminal can be seen in equation 6.23. Performing this substitution requires searching for an initial curve along the x-axis and an initial curve along the y-axis. It is important to recall from section 5.3.2 that the location of the axis was found using the vertical and horizontal projection profiles. The parse tree for these rules can be seen in figure 6.39.

There are rules in this section with nonterminals which have an i subscript. These rules are represented by a single equation such that i can be either 0 or 1. Hypothetically, if we had the equation $A_i \rightarrow A_t A_i$ it would represent both $A_0 \rightarrow A_t A_0$ and $A_1 \rightarrow A_t A_1$. This compact form is used to simplify the explanation in this thesis. A rule used in a derivation will always have a subscript of either 0 or 1.

$$G \rightarrow A_x A_y \tag{6.23}$$

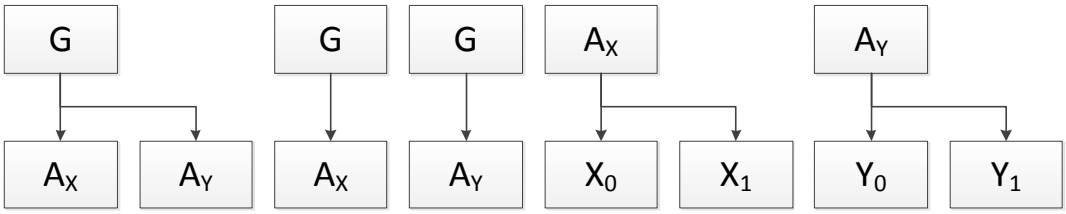


Figure 6.39: This is the top level of the subset for the axis and the gridlines.

The x-axis

This subsection describes the A_x nonterminal and the subset it is associated with. It starts with a single curve along the x-axis and uses this curve to detect the remaining curves along the x-axis as well as the curves connected to the x-axis. For a 2D chart these curves will be part of the vertical gridlines and for a 3D chart these will be the XZ gridlines. It is important to recall the position of the XZ gridline from figure 6.15.

The only rule for substituting the A_x nonterminal is the $A_x \rightarrow X_0 X_1$ rule seen in equation 6.24 and figure 6.39. This initiates the detection of the curves along the x-axis. This rule does not do any classification, however it initiates the search for the remaining part of the axis in both directions.

$$A_x \rightarrow X_0 X_1 \tag{6.24}$$

The next step involves finding the remaining parts of the x-axis and initiating the detection of the curves connected to the x-axis. The X_0 nonterminal starts with the initial curve and then detects curves in the direction of the chart origin (assumed to be the bottom left point) and the X_1 nonterminal will detect curves in the direction away from the chart origin. The direction of the nonterminals is seen visually in figure 6.40.

The substitution rules for the X_i nonterminals are found in equations 6.25 and 6.26. The rule which is chosen depends on the curves which are connected to the initial curve. The rule chosen is determined by observing the angle between the input curve and the curves which are connected to the end of the curve. For the X_0

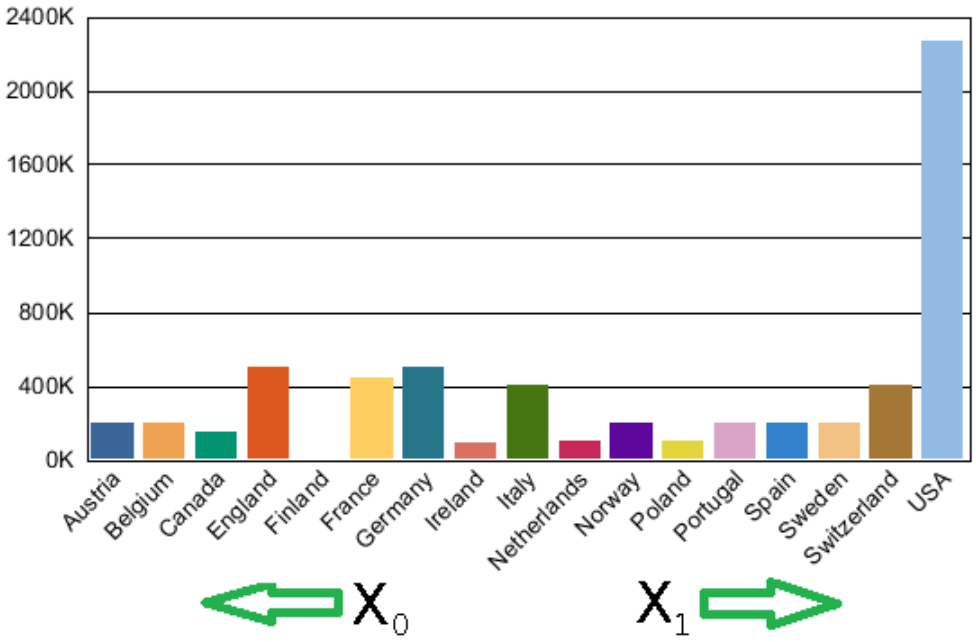


Figure 6.40: The directions for the X_0 and X_1 nonterminals. This is relative to the initial curve.

nonterminal, a vertical gridline is at 90 degrees and for the X_1 nonterminal, a vertical gridline is at 270 degrees. The XZ gridline is found at an angle of 25 degrees for the X_0 nonterminal and 205 degrees for the X_1 nonterminal. These angles were manually found via observation of many 3D bar charts. For both the X_0 and X_1 nonterminals, an additional axis curve is at 180 degrees. If a curve along the same line is detected, a rule with the X_i nonterminal on the right side is chosen. Likewise, if a vertical gridline or a XZ gridline is detected, a rule with the G_{V1} or G_{XZ} nonterminals is chosen. A visual representation of how the angles are measured can be seen in figure 6.43. The parse trees for these rules can be seen in figures 6.41 and 6.42.

$$X_i \rightarrow X_t G_{V1} X_i \qquad X_i \rightarrow X_t G_{V1} \qquad X_i \rightarrow X_t X_i \qquad X_i \rightarrow X_t \tag{6.25}$$

$$X_i \rightarrow X_t G_{XZ} X_i \qquad X_i \rightarrow X_t G_{XZ} \tag{6.26}$$

Each rule corresponds to a different intersection or curve end point. A graphical representation of each situation can be seen in figure 6.44. It is important to recall from figure 6.40 that X_0 is always going towards the left and the X_1 is always going towards the right.

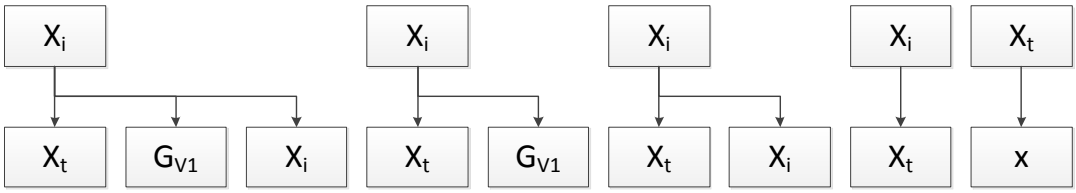


Figure 6.41: These parse trees show the rules for detecting the x-axis curves.

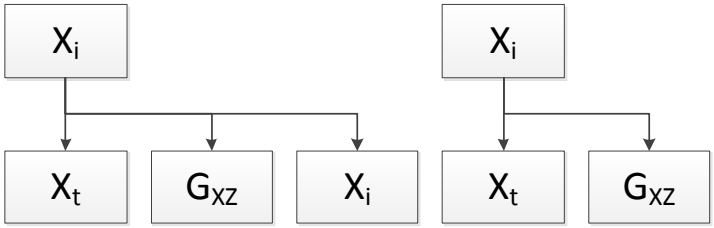


Figure 6.42: These parse trees show the rules for detecting the XZ gridlines.

Configurations a) through c) in figure 6.44 correspond to intersections between a vertical gridline and the x-axis. The rules which represent these intersections are the $X_i \rightarrow X_t G_{V1} X_i$ rule and the $X_i \rightarrow X_t G_{V1}$ rule.

Configuration d) in figure 6.44 corresponds to a small gap in the x-axis. The rule for this situation is the $X_i \rightarrow X_t X_i$ since there is only a single connecting x-axis curve (X_i) and no vertical gridline curve.

Configurations h) and i) in figure 6.44 correspond to unconnected ends coming from two different directions. In this situation, there is no connecting curve and the $X_i \rightarrow X_t$ rule is used.

Configurations e), f), and g) in figure 6.44 only exist on 3D charts. These are intersections between the x-axis and the XZ gridlines. The rules for these intersections correspond to the $X_i \rightarrow X_t G_{XZ} X_i$ rule and the $X_i \rightarrow X_t G_{XZ}$ rule. The two configurations in the bottom middle correspond to $i = 0$ and $i = 1$. That is, they correspond to going left and going right.

Vertical Gridlines

The G_{V0} and G_{V1} nonterminals represent the vertical gridlines. The G_{V0} nonterminal corresponds to searching in the downwards direction and the G_{V1} nonterminal

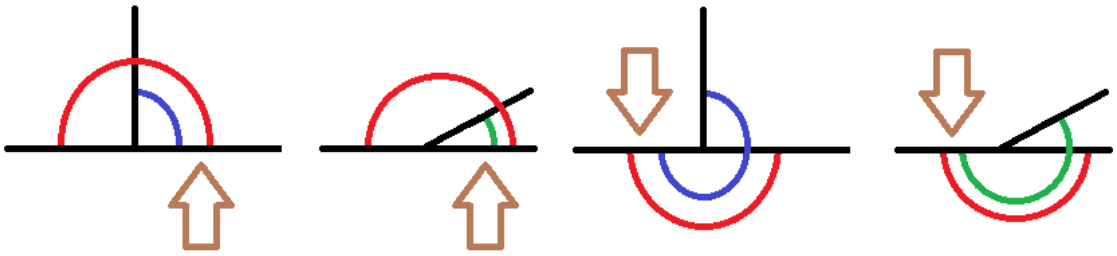


Figure 6.43: The angles for the vertical gridline substitution rules. The brown arrow indicates the initial curve. In the left two images, the initial curve is represented by the X_0 nonterminal, and in the right two images, the initial curve is represented by the X_1 nonterminal. The red, blue and green arcs represent the angle for the same nonterminal as the initial curve segment, the vertical gridlines, and the XZ gridline respectively.

corresponds to searching in the upwards direction, as shown in figure 6.45.

In addition to classifying the vertical gridlines, the curves connected to the vertical gridlines will be classified. These curves will be part of the horizontal gridlines.

The substitution rules for the G_{V0} and G_{V1} nonterminals are seen in equations 6.27, 6.28, and 6.29. Similar to other rules in this section on the grid structure, the choice of which rule is used depends on the curves which are connected to the input curve. The parse trees for these curves is seen in figure 6.46.

$$G_{Vi} \to G_{Vt}G_{H0}G_{H1}G_{Vi} \qquad G_{Vi} \to G_{Vt} \tag{6.27}$$

$$G_{Vi} \to G_{Vt}G_{H0}G_{Vi} \qquad G_{Vi} \to G_{Vt}G_{H1}G_{Vi} \qquad G_{Vi} \to G_{Vt}G_{H0}G_{H1} \tag{6.28}$$

$$G_{Vi} \to G_{Vt}G_{H0} \qquad G_{Vi} \to G_{Vt}G_{H1} \qquad G_{Vi} \to G_{Vt}G_{Vi} \tag{6.29}$$

The three possible connection types for the vertical gridline (G_{Vi}) are the horizontal gridline going left (G_{H0}), the horizontal gridline going right (G_{H1}), and the continuation of the vertical gridline (G_{Vi}). The horizontal gridlines are at 90 degrees and 270 degrees and the continuation of the vertical gridline is 180 degrees. A visual representation of the angles can be seen in figure 6.47.

There is a special case for the gridlines which is not present in the other curves. In figure 6.45 some of the vertical gridlines are intersected by the horizontal bars. To

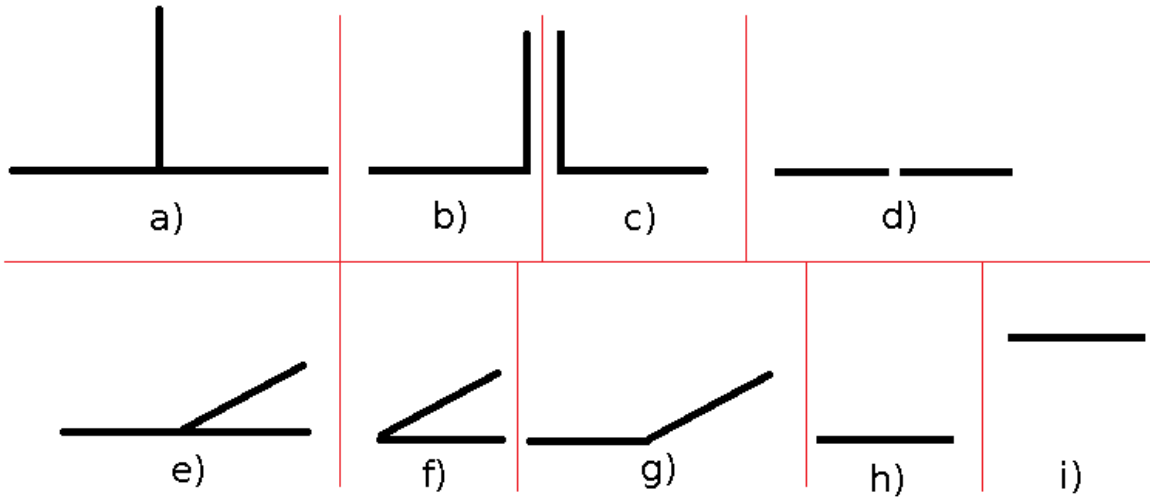


Figure 6.44: The six different configurations for the x-axis substitution rules. Each configuration corresponds to a different substitution rule.

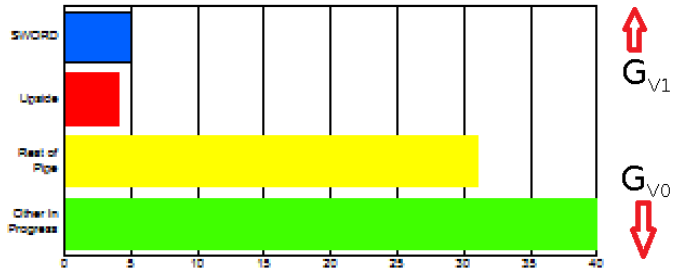


Figure 6.45: The directions for the G_{V0} and G_{V1} nonterminals. This is relative to the initial curve.

connect the gridlines which are divided by the bars, the search range for the connecting curves is much larger. A search range of 100 pixels was used for these connections compared with 5 pixels for other connections. If no additional unclassified curves are connected to the vertical gridline, the computation stops.

Each rule corresponds to a different intersection configuration. Figure 6.48 shows a graphical representation of each configuration. It is important to recall from figure 6.45 that the G_{V0} nonterminal is always going in the downwards direction and the G_{V1} nonterminal is always going in the upwards direction. All configurations except the two configurations at the bottom right correspond to a intersection with a horizontal gridline.

Configuration a) in figure 6.48 is an ordinary intersection in the middle of the

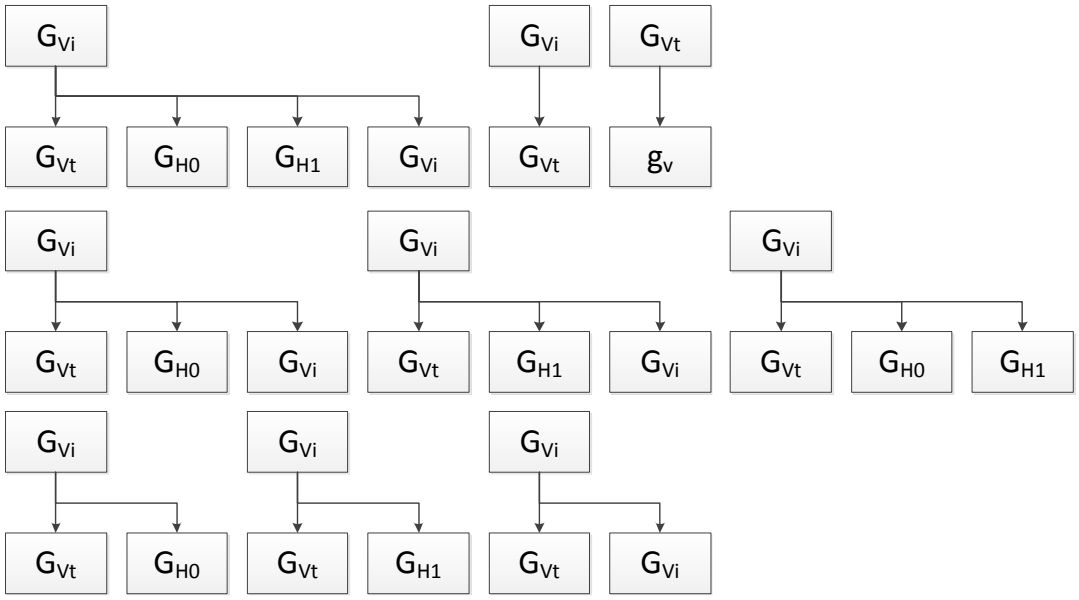


Figure 6.46: These parse trees show the rules for detecting the vertical gridlines.

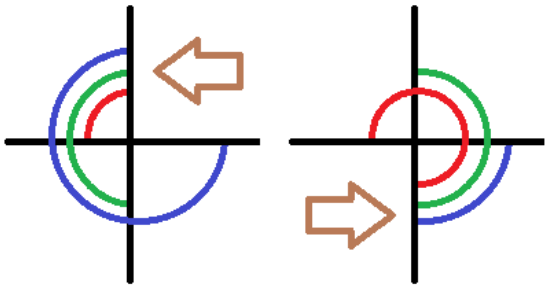


Figure 6.47: The angles for the substitution rules. The brown arrow indicates the initial curve. For the left figure, the initial curve is represented by the G_{V0} nonterminal and in the right figure, the initial curve is represented by the G_{V1} nonterminal. The blue, green, and red arcs are for the horizontal gridline going right (G_{H1}), the vertical gridline (G_{Vi}), and the horizontal gridline going left (G_{H0}) respectively.

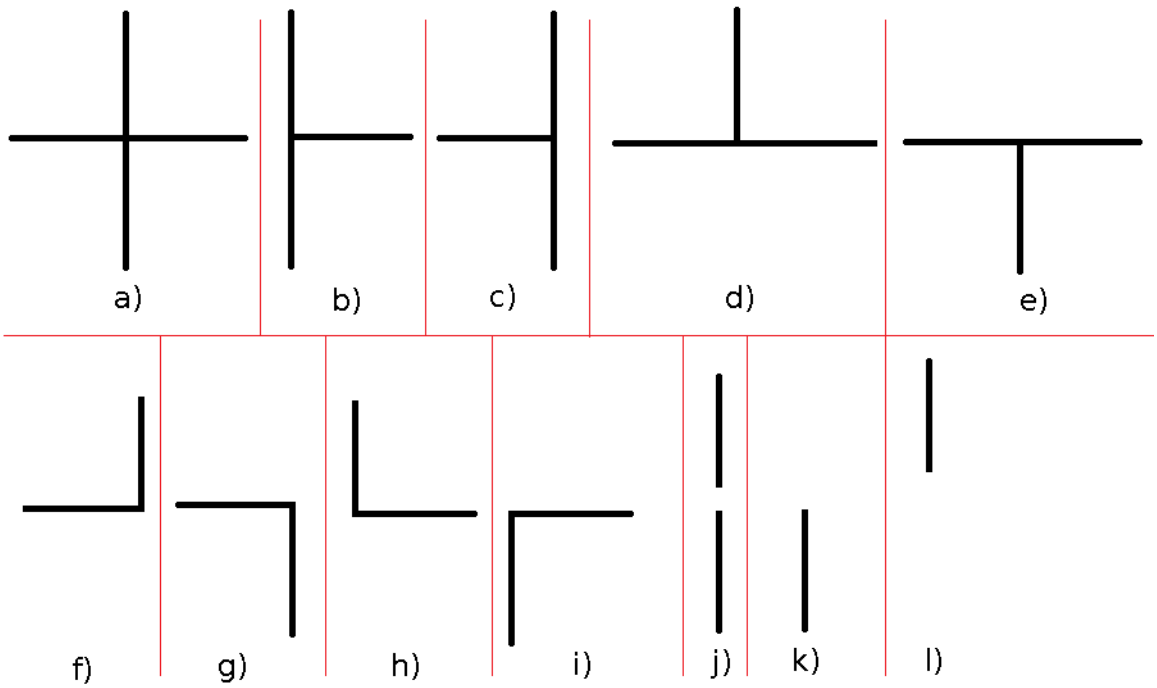


Figure 6.48: The directions for the G_{V0} and G_{V1} nonterminals.

chart where the horizontal gridline crosses the vertical gridline. This configuration corresponds to the $G_{Vi} \rightarrow G_{Vt}G_{H0}G_{H1}G_{Vi}$ substitution rule because it contains a horizontal gridline going right (G_{H1}), a horizontal gridline going left (G_{V0}) and a continuation of the vertical gridline (G_{Vi}).

Configurations *b*) and *c*) correspond to intersections where part of the horizontal gridline is missing. Examples of when this can occur is when the vertical gridline is at the edge of the chart or close to a bar. These configurations use the $G_{Vi} \rightarrow G_{Vt}G_{H1}G_{Vi}$ substitution rule and the $G_{Vi} \rightarrow G_{Vt}G_{H0}G_{Vi}$ substitution rule because they have either a horizontal gridline going left (G_{H0}) or horizontal gridline going right (G_{H1}) and the continuation of the vertical gridline (G_{Vi}).

Configuration *d*) and *e*) correspond to configurations where there are no continuation of the gridline. This can occur at the top of the chart or it can be caused by bars occluding the gridlines. These configurations use the $G_{Vi} \rightarrow G_{Vt}G_{H0}G_{H1}$ since there is a horizontal gridline going left (G_{V0}) and going right (G_{V1}) but no continuing vertical gridline (G_{Vi}). The left configuration corresponds to the G_{V0} nonterminal and the right configuration corresponds to the G_{V1} nonterminal. To connect the gridlines which are split by the bars, the connecting curve is allowed to be a much greater distance apart. In this implementation a distance of 100 pixels is used for these connections. The other connections have a maximum connection distance of 5 pixels. These values were determined empirically from the database used for testing, which has a constant pixels per inch.

Configurations *f*) through *i*) correspond to when there is only a horizontal gridline going left (G_{H1}) or a horizontal gridline going right (G_{H1}). This can occur at the corners of the chart or when a bar occludes the gridlines. These configurations correspond to the $G_{Vi} \rightarrow G_{Vt}G_{H0}$ and $G_{Vi} \rightarrow G_{Vt}G_{H1}$ substitution rules. It is important to recall that the G_{V0} nonterminal corresponds to approaching the intersection from the top direction and the G_{V1} corresponds to approaching the intersection from the bottom direction.

Configuration *j*) corresponds to a vertical gridline that has been split into two. This is common with horizontal bars but also can occur with polylines. This configuration uses the $G_{Vi} \rightarrow G_{Vt}G_{Vi}$ substitution rule. An example of a gridline which has been split into two can be seen in figure 6.45.

Configurations *k*) and *l*) correspond to unconnected end points. The left corresponds to the G_{V1} nonterminal which is approaching from the bottom and the right corresponds to the G_{V1} nonterminal which is approaching from the top. This

can occur when a polyline intersects a bar. This configuration corresponds to the $G_{Vi} \rightarrow G_{Vt}$ substitution rule.

XZ Gridlines

This subsection describes the rules associated with the XZ gridlines. The process involves searching for gridlines which connect to the XZ gridlines in the upwards direction. The XZ gridlines connect to the bottom horizontal gridline and vertical gridlines. For charts within this database, the XZ gridlines always appear at 25 degrees from the x-axis. Therefore, if a curve is at 25 degrees from X_0 nonterminal it is known that it is an XZ gridline (G_{XZ}). Since the X_1 nonterminal is going in the opposite direction, the angle required is 205 degrees. These angles were determined by manually measuring several charts.

$$G_{XZ} \rightarrow G_{XZt}G_{H0}G_{H1}G_{V1} \qquad G_{XZ} \rightarrow G_{XZt} \qquad G_{XZt} \rightarrow g_{xz} \tag{6.30}$$

$$G_{XZ} \rightarrow G_{XZt}G_{H0}G_{V1} \qquad G_{XZ} \rightarrow G_{XZt}G_{H1}G_{V1} \qquad G_{XZ} \rightarrow G_{XZt}G_{H0}G_{H1} \tag{6.31}$$

$$G_{XZ} \rightarrow G_{XZt}G_{H0} \qquad G_{XZ} \rightarrow G_{XZt}G_{H1} \qquad G_{XZ} \rightarrow G_{XZt}G_{V1} \tag{6.32}$$

There are eight rules for substituting the G_{XZ} nonterminal as seen in equations 6.30, 6.31, and 6.32. The rule which is chosen depends on the angles of the curves which are connected to the initial curve. The XZ gridlines connect to the same gridlines as the vertical gridline going in the upwards direction. The choice of which rule to use is based on the same principles as the vertical gridlines. The parse trees for these rules can be seen in figure 6.49 and a visual representation of the angles being observed can be seen in figure 6.50.

Figure 6.51 displays the possible configurations for the XZ gridlines. Each configuration corresponds to a different substitution rule.

Configuration *a*) corresponds to an intersection which contains all possible connections. That is, it connects to the vertical gridline going up (G_{V1}), the horizontal gridline going left (G_{H0}) and the horizontal gridline going right (G_{H1}). Thus it corresponds to the $G_{XZ} \rightarrow G_{XZt}G_{H0}G_{H1}G_{V1}$ substitution rule.

Configuration *b*) corresponds to the situation where no gridlines are detected. It

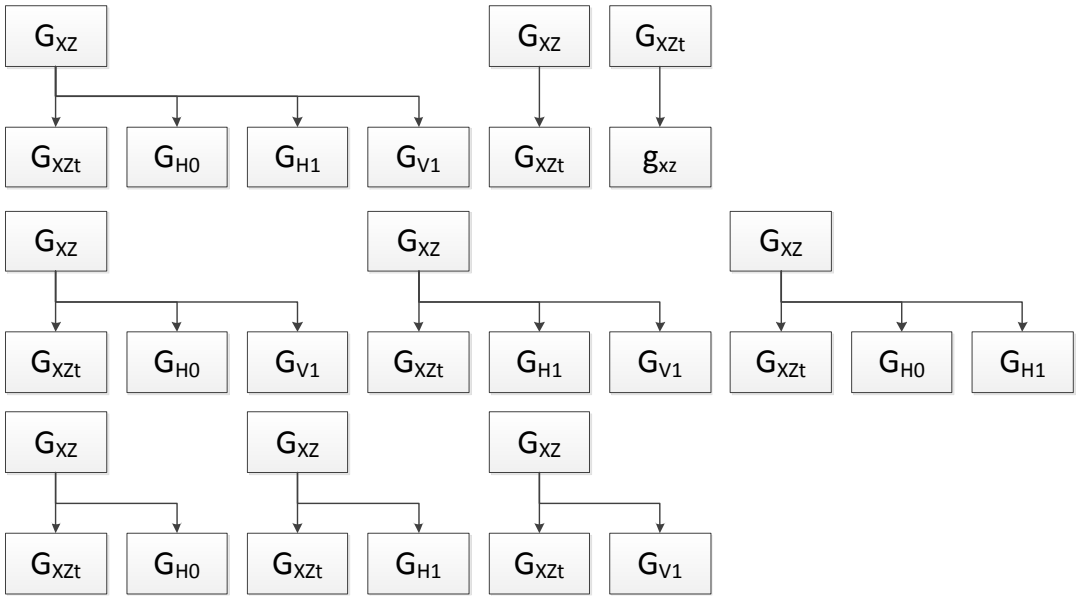


Figure 6.49: These parse trees show the rules for detecting the z gridlines along the xz plane.

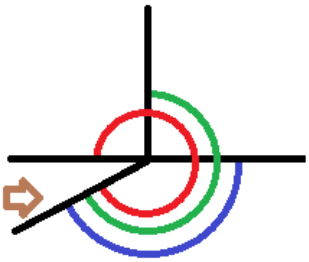


Figure 6.50: The angles for the substitution rules. The brown arrow indicates the initial curve. The initial curve is also represented by the G_{XZ} nonterminal. The blue, green, and red arcs are for the horizontal gridline going right (G_{H1}), the vertical gridline (G_{V1}), and the horizontal gridline going left (G_{H0}) respectively.

does not connect to any additional curves and so the $G_{XZ} \rightarrow G_{XZt}$ substitution rule must be used.

Configurations *c*) and *d*) correspond to configurations where the XZ gridline connects to the horizontal gridline going in only one direction and the vertical gridline (G_{V1}). The left configuration corresponds to the horizontal gridline going left (G_{H0}) and the right configuration corresponds to the horizontal gridline going right (G_{H1}).

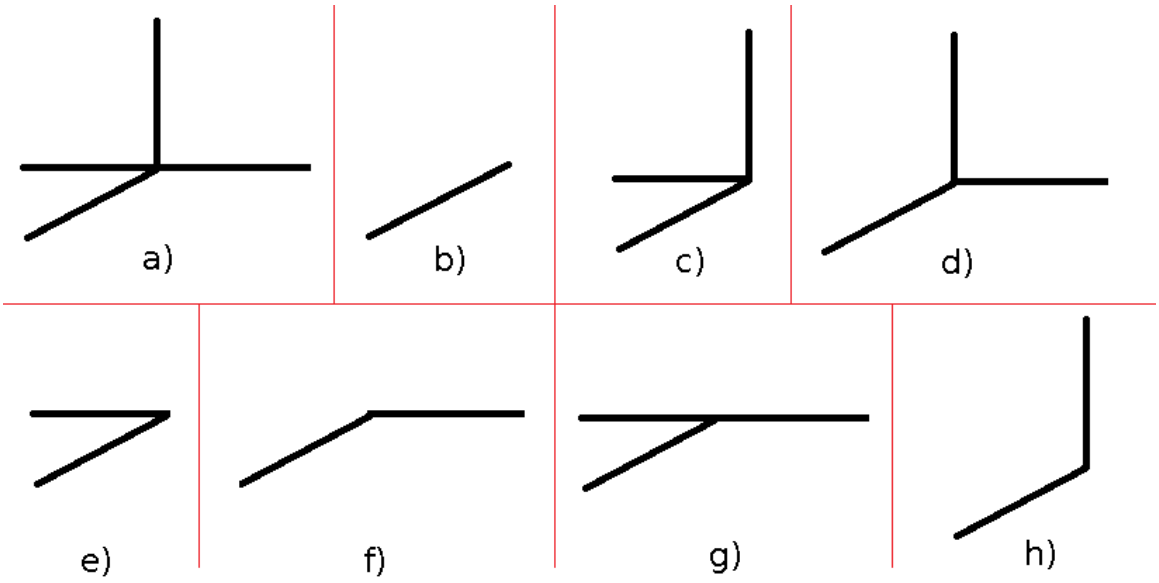


Figure 6.51: The directions for the G_{XZ} nonterminal.

The two substitution rules for these configurations are $G_{XZ} \rightarrow G_{XZt}G_{H0}G_{V1}$ and $G_{XZ} \rightarrow G_{XZt}G_{H1}G_{V1}$.

Configurations e) and f) are similar to the last two configurations, but without the vertical gridline connection. These configurations correspond to the $G_{XZ} \rightarrow G_{XZt}G_{H0}$ and $G_{XZ} \rightarrow G_{XZt}G_{H1}$ substitution rules.

Configuration g) corresponds to where the situation where the XZ gridline connects to a horizontal gridline in both directions, but there is no vertical gridline. This is the $G_{XZ} \rightarrow G_{XZt}G_{H0}G_{H1}$ substitution rule.

The last configuration, configuration h) corresponds to when only the vertical gridline is detected. This is the $G_{XZ} \rightarrow G_{XZt}G_{V1}$ substitution rule.

The y-axis

This subsection describes the A_y nonterminal and the subset it is associated with. It starts with a curve along the y-axis and uses this curve to detect the remaining curves along the y-axis. In addition to detecting the the y axis, the curves connected to the y axis will be detected. These curves will be part of the horizontal gridlines for 2D charts and the YZ gridlines for the 3D charts.

The only rule which substitutes the A_y nonterminal is $A_y \rightarrow Y_0Y_1$ which is seen in equation 6.33. This rule does not perform any calculations itself, instead it is substituted by the Y_0 and Y_1 nonterminals. The parse tree for this rule is seen in

figure 6.39.

$$A_y \rightarrow Y_0 Y_1 \tag{6.33}$$

The substitution rules for the Y_i nonterminals require searching for curves along the y axis. The Y_0 nonterminal searches for curves towards the origin and the Y_1 nonterminal searches away from the origin. These directions can be seen in figure 6.52.

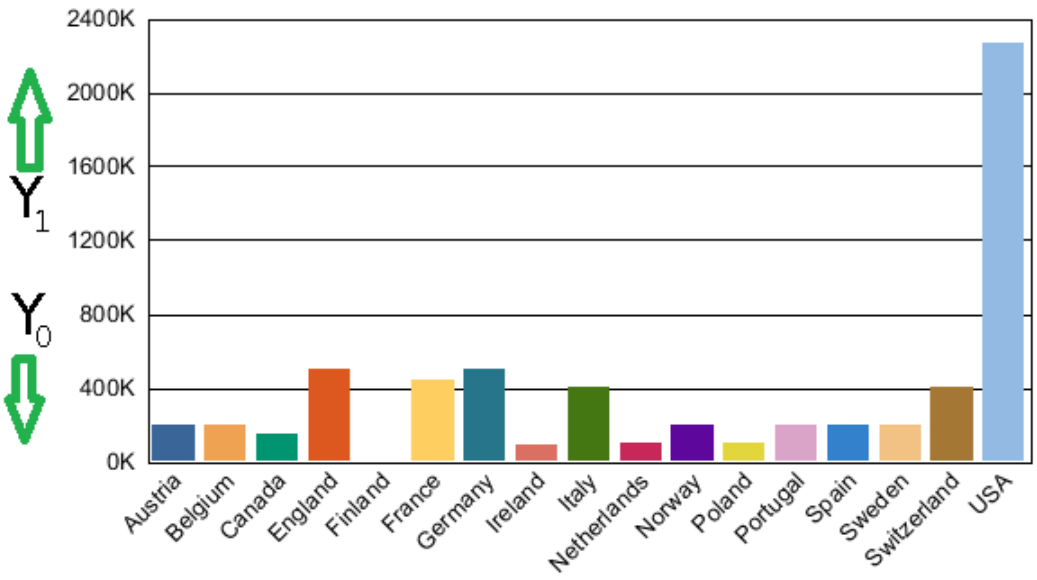


Figure 6.52: The directions for the Y_0 and Y_1 nonterminals. This is relative to the initial curve.

The rules are in equations 6.34 and 6.35. The rules associated with the Y_0 nonterminal start with the initial curve and then searches for additional curves in the direction of the origin. The rules associated with the Y_1 nonterminal search for additional curves in the direction away from the origin. The parse trees for these rules can be seen in figures 6.53 and 6.54.

$$Y_i \rightarrow Y_t G_{H1} Y_i \qquad Y_i \rightarrow Y_t G_{H1} \qquad Y_i \rightarrow Y_t Y_i \qquad Y_i \rightarrow Y_t \tag{6.34}$$

$$Y_i \rightarrow Y_t G_{YZ} Y_i \qquad Y_i \rightarrow Y_t G_{YZ} \tag{6.35}$$

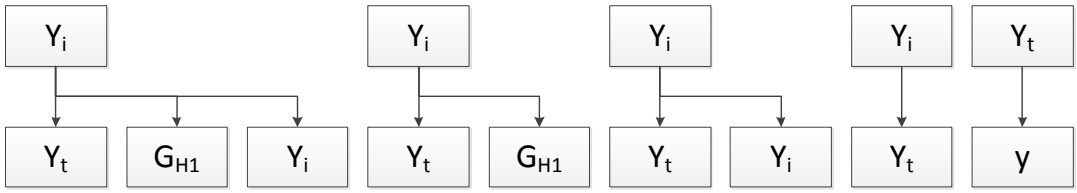


Figure 6.53: These parse trees show the rules for detecting the y-axis curves.

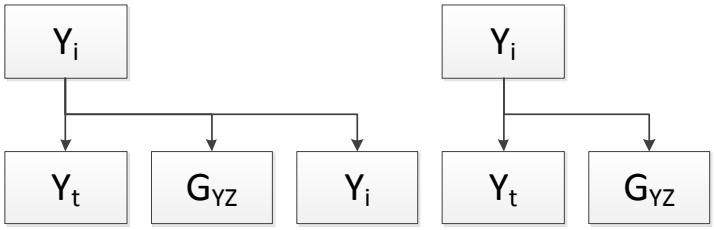


Figure 6.54: These parse trees show the rules for detecting the YZ gridlines from the y-axis.

There are five rules for substituting the Y_0 and Y_1 nonterminals. Similar to the X_i nonterminals, the choice of rule depends on which curves are connected to the end of the input curve. If a curve is detected along the same line as the input curve, a rule with the Y_i nonterminal on the right side is used. If a horizontal gridline is detected, a rule with the G_{H1} nonterminal on the right side is used and if a YZ gridline is detected, a rule with the G_{YZ} nonterminal on the right side is used. Similar to the previous grid rules, the curves are detected and classified by observing the angle between the input curve and the curves it is connected to. A horizontal gridline is at 270 degrees for the Y_0 nonterminal and 90 degrees for the Y_1 nonterminal. The YZ gridline (G_{YZ}) is at 295 degrees for the Y_0 nonterminal and at 115 degrees for the Y_1 nonterminal. The continuation of the y-axis is at 180 degrees for both Y_0 and Y_1 nonterminals. The y terminal adds a curve to the y-axis of the chart structure. These angles can be seen in figure 6.55.

Each rule corresponds to a different curve intersection type. These intersections can be seen in figure 6.56.

Configurations a), b), and c in figure 6.56 correspond to intersections between a

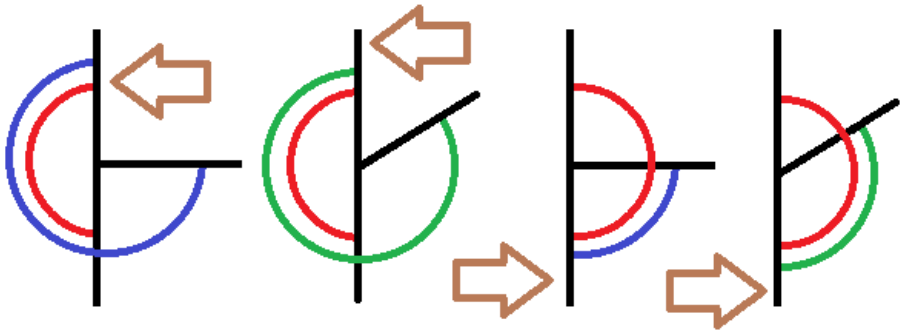


Figure 6.55: The angles for the substitution rules. The brown arrow indicates the initial curve. The left two images the initial curve is the Y_0 nonterminal curve and the right two images the initial curve is the Y_1 nonterminal curve. The red, blue, and green arcs represent the angle for same nonterminal as the initial curve segment, the horizontal gridlines, and the YZ gridline.

horizontal gridline (G_{H1}) and y-axis. The rules which represent these configurations are the $Y_i \rightarrow Y_t G_{H1} Y_i$ substitution rule and the $Y_i \rightarrow Y_t G_{H1}$ substitution rule. This rule is used because it contains the G_{H1} nonterminal. The leftmost configuration also contains a connecting y-axis curve (Y_i).

Configuration d) of figure 6.56 corresponds to a small gap in the y axis. The substitution rule for this configuration is $Y_i \rightarrow Y_t Y_i$. This rule is chosen because there is a single y-axis curve and no horizontal gridline.

Configurations e) and f) correspond to ends which are not connected to another curve. In this case there are no curves connected to the initial curve. The $Y_i \rightarrow Y_t$ substitution rule must be used here.

Configurations g) through i) only exist on 3D charts. These are intersections between the y-axis and the YZ gridlines. The substitution rules for these configurations are $Y_i \rightarrow Y_t G_{YZ} Y_i$ and $Y_i \rightarrow Y_t G_{YZ}$. The configuration at the bottom right correspond to both $i = 0$ and $i = 1$ in the substitution rule. That is, they correspond to going up and down.

Horizontal Gridlines

The G_{H0} and G_{H1} nonterminals represent the horizontal gridlines. The G_{H0} nonterminal classifies the curves by going towards the left of the chart and the G_{H1} classifies the curves by going to the right of the chart. The direction of the nonterminals is seen visually in figure 6.57.

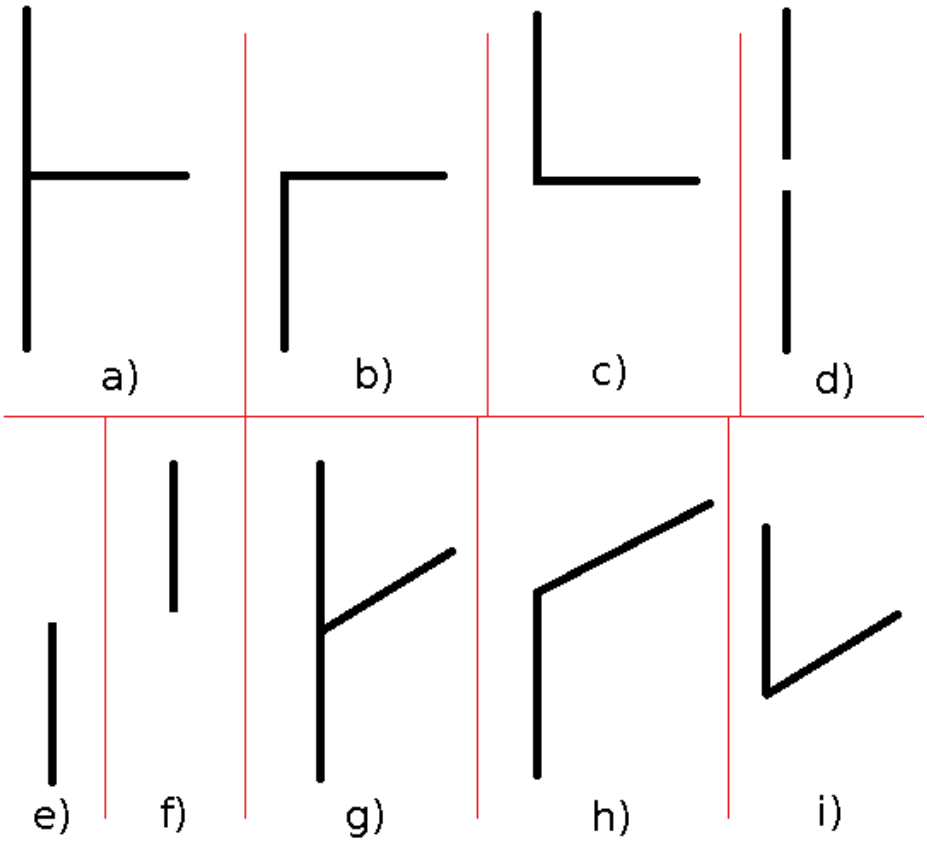


Figure 6.56: The six different configurations for the y-axis substitution rules. Each configuration corresponds to a different substitution rule.

In addition to classifying the horizontal gridlines, the curves connected to the horizontal gridlines will be classified. These curves are parts of the vertical gridlines.

There are eight rules for substituting the G_{Hi} nonterminals. These rules are seen in equations 6.36, 6.37, 6.38. Similar to other rules in the grid section, the choice of rule is dependent on the curves connected to the input curve. The parse trees for these rules can be seen in figure 6.58.

$$G_{Hi} \rightarrow G_{Ht}G_{V0}G_{V1}G_{Hi} \qquad G_{Hi} \rightarrow G_{Ht} \tag{6.36}$$

$$G_{Hi} \rightarrow G_{Ht}G_{V0}G_{Hi} \qquad G_{Hi} \rightarrow G_{Ht}G_{V1}G_{Hi} \qquad G_{Hi} \rightarrow G_{Ht}G_{V0}G_{V1} \tag{6.37}$$

$$G_{Hi} \rightarrow G_{Ht}G_{V0} \qquad G_{Hi} \rightarrow G_{Ht}G_{V1} \qquad G_{Hi} \rightarrow G_{Ht}G_{Hi} \tag{6.38}$$

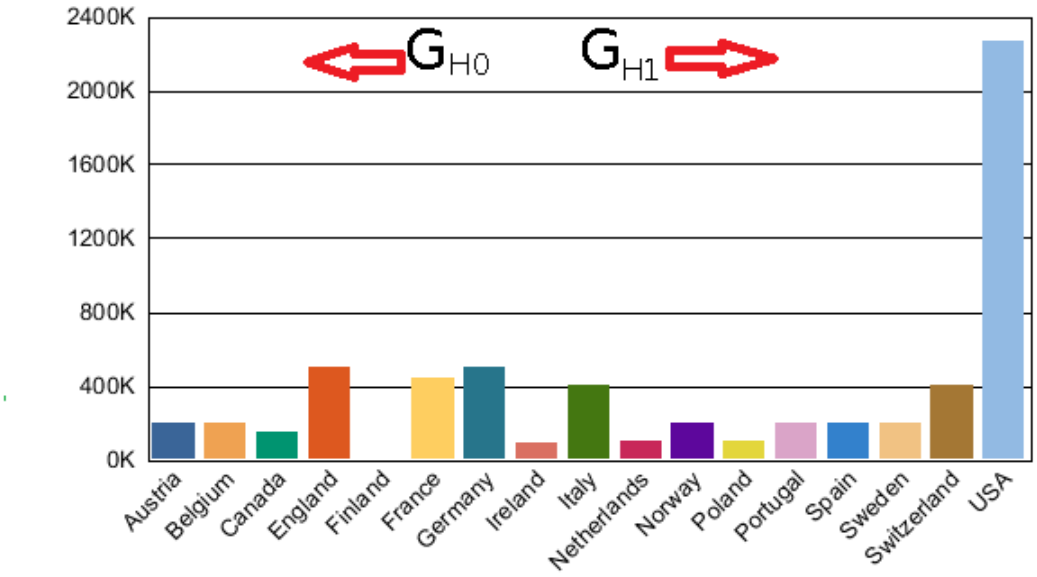


Figure 6.57: The directions for the G_{V0} and G_{V1} nonterminals. This is relative to the initial curve.

The possible connections for the horizontal gridlines G_{Hi} are the vertical gridline going up (G_{V1}), the vertical gridline going down (G_{V0}), and the horizontal gridline going in the same direction (G_{Hi}). The vertical gridlines are at 90 degrees and 270 degrees and the continuation of the horizontal gridline is at 180 degrees. Figure 6.59 shows these angles graphically.

Similar to the vertical gridlines, these curves intersect chart bars. An example of this can be seen in figure 6.57. To connect the gridlines which are split by the bars, the search range for the connecting curves is much larger. A search range of 100 pixels is used for these connections compared with 5 pixels for other connections.

Each rule corresponds to a different curve intersection configuration. These configurations can be seen in figure 6.60.

Configuration a of figure 6.60 contains all three possible connections. The substitution rule for this configuration is $G_{Hi} \rightarrow G_{Ht}G_{V0}G_{V1}G_{Hi}$.

Configurations b through e correspond to situations where one of the three possible connections are missing. From left to right, the missing connections are the vertical gridline going up (G_{V1}), the vertical gridline going down (G_{V0}), and the continuation of the horizontal gridline (G_{Hi}, last two configurations). For the two rightmost configurations, the left is for the horizontal gridline going right (G_{H1}) and the right is for the

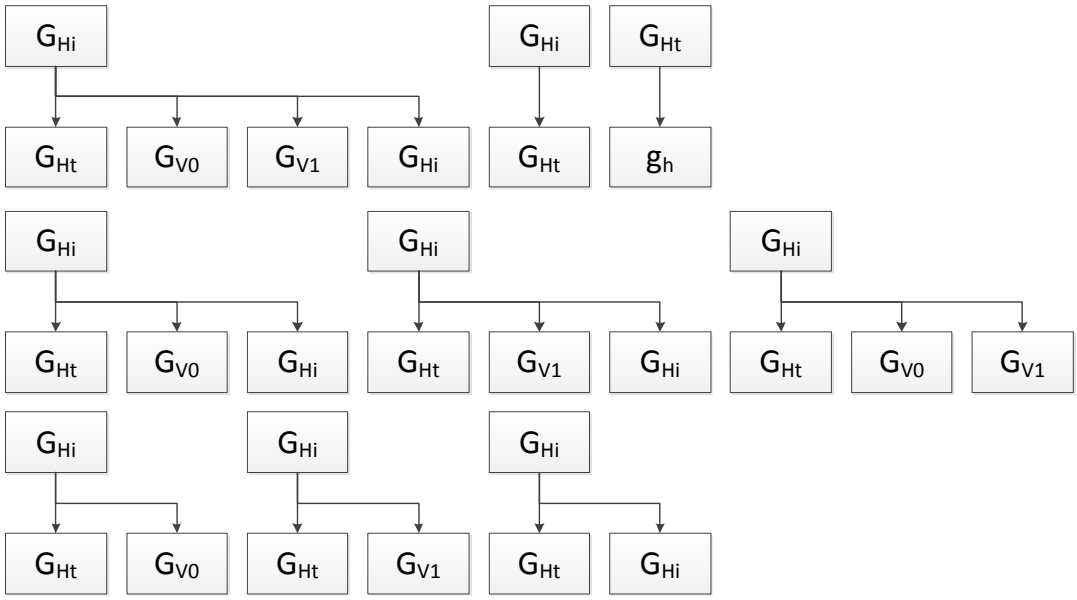


Figure 6.58: These parse trees show the rules for detecting the horizontal gridlines.

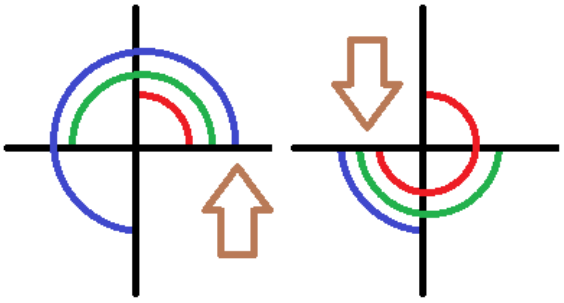


Figure 6.59: The angles for the substitution rules. The brown arrow indicates the initial curve. In the left figure, the initial curve is the G_{H0} nonterminal curve and in the right figure, the initial curve is the G_{H1} nonterminal curve. The blue, green, and red arcs are for the vertical gridline going down (G_{V0}), the continuation of the horizontal gridline (G_{Hi}), and the vertical gridline going up (G_{V1}).

horizontal gridline going left (G_{H0}). The substitution rules for these configurations are $G_{Hi} \rightarrow G_{Ht}G_{V0}G_{Hi}$, $G_{Hi} \rightarrow G_{Ht}G_{V1}G_{Hi}$, and $G_{Hi} \rightarrow G_{Ht}G_{V0}G_{V1}$.

Configurations f through j correspond to situations where only one connection is present. Starting from the left, the first connection is the continuation of the horizontal gridline (G_{Hi}), the next two correspond to the vertical gridline going up (G_{V1}), and the last two correspond to the vertical gridline going down (G_{V0}). The

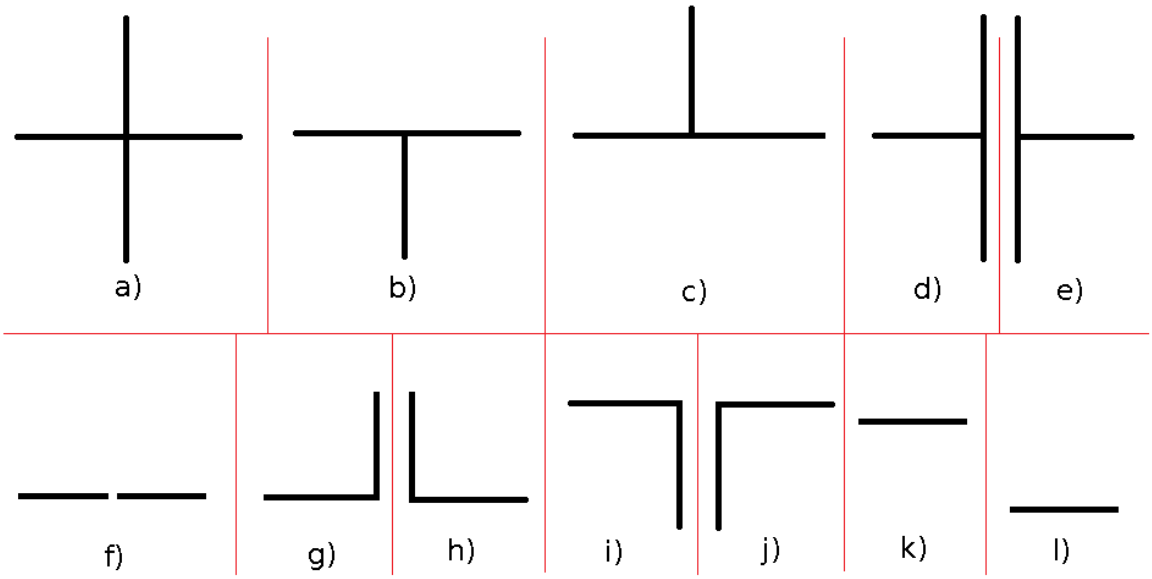


Figure 6.60: The directions for the G_{H0} and G_{H1} nonterminals.

substitution rules for these are $G_{Hi} \rightarrow G_{Ht}G_{Hi}$, $G_{Hi} \rightarrow G_{Ht}G_{V1}$, and $G_{Hi} \rightarrow G_{Ht}G_{V0}$.

Configurations k and l of figure 6.60 corresponds to curves which are not connected to any other curves. There are two configurations here because there are two different directions. The left configuration is for the G_{H1} nonterminal and the right configuration is for the G_{H0} nonterminal. The $G_{Hi} \rightarrow G_{Ht}$ substitution rule is used here.

YZ Gridlines

This section describes the rules associated with the YZ gridlines. The YZ gridlines always appear at 65 degrees from the y-axis (25 degrees from the x-axis). Since the angles are always taken in a counter-clockwise direction from the input curve, the angle searched for is actually 295 degrees for Y_0 and 115 degrees for Y_1.

The gridlines along the YZ plane are represented by the G_{YZ} nonterminal. This nonterminal corresponds to searching for additional curves to the right, away from the y-axis. There are eight rules which can be used to substitute the G_{YZ} nonterminal as seen in equations 6.39, 6.40, 6.41. The rule which is chosen depends on the angles of the curves which are connected to the initial curve. These rules are similar to the horizontal gridline rules (G_{Hi} nonterminal) such that many of the same nonterminals are present. The two key differences are G_{YZt} nonterminal which classifies the YZ gridline and the angles of the connecting gridlines. The vertical gridlines are at 65 degrees and 245 degrees and the horizontal gridline is at 155 degrees. These angles can be seen visually in figure 6.62. The parse trees for these rules can be seen in figure 6.61.

$$G_{YZ} \rightarrow G_{YZt}G_{V0}G_{V1}G_{H1} \qquad G_{YZ} \rightarrow G_{YZt} \tag{6.39}$$

$$G_{YZ} \rightarrow G_{YZt}G_{V0}G_{H1} \qquad G_{YZ} \rightarrow G_{YZt}G_{V1}G_{H1} \qquad G_{YZ} \rightarrow G_{YZt}G_{V0}G_{V1} \tag{6.40}$$

$$G_{YZ} \rightarrow G_{YZt}G_{V0} \qquad G_{YZi} \rightarrow G_{YZt}G_{V1} \qquad G_{YZ} \rightarrow G_{YZt}G_{H1} \tag{6.41}$$

Similar to the previous gridlines, each substitution rule corresponds to a different intersection configuration. These configurations can be seen in figure 6.63. There are three different curves which can connect to the YZ gridline. These are the vertical

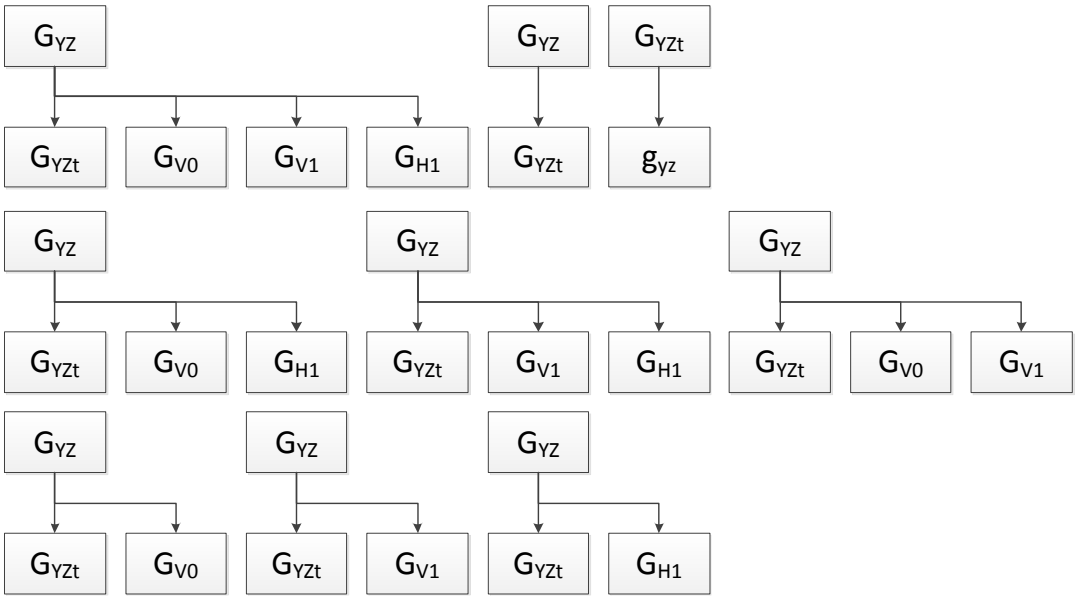


Figure 6.61: These parse trees show the rules for detecting the z gridlines along the zy-plane.

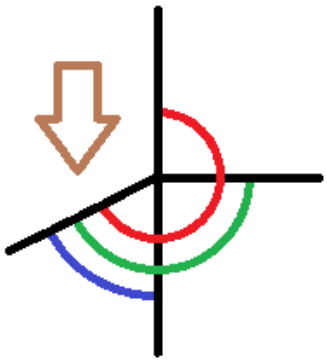


Figure 6.62: The angles for the substitution rules. The brown arrow indicates the initial curve. The blue, green, and red arc is for the vertical gridline going down (G_{V0}), the horizontal gridline (G_{H1}), and the vertical gridline going up (G_{V1}) respectively.

gridlines going up (G_{V1}) and down (G_{V0}) and the horizontal gridline going right (G_{H1}).

Configuration a) of figure 6.63 contains all three connections. The three other configurations on the top row, configurations b) through d) correspond to situations where only a single curve is connected to the YZ gridline. From b) to d), the connecting

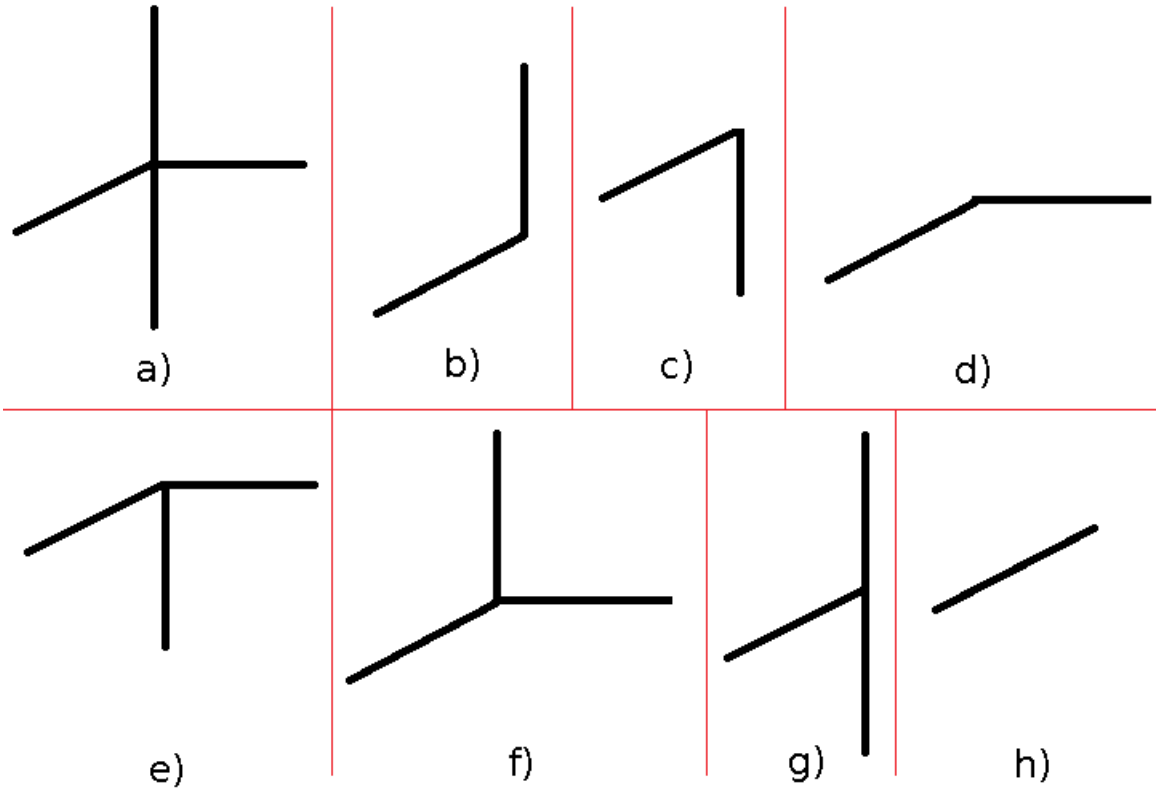


Figure 6.63: The directions for the G_{YZ} nonterminal.

curves are the vertical gridline going up (G_{V1}), the vertical gridline going down (G_{V0}), and the horizontal gridline (G_{H1}). The substitution rules for these configurations are $G_{YZi} \rightarrow G_{YZt}G_{V1}$, $G_{YZ} \rightarrow G_{YZt}G_{V0}$, and $G_{YZ} \rightarrow G_{YZt}G_{H1}$.

Configurations e) through g) of figure 6.63 corresponds to configurations where two curves are connected to the YZ gridline. Each configuration is missing a different curve. From right to left, the missing curves are the vertical gridline going up, the vertical gridline going down, and the horizontal gridline. The rules for these configurations are $G_{YZ} \rightarrow G_{YZt}G_{V0}G_{H1}$, $G_{YZ} \rightarrow G_{YZt}G_{V1}G_{H1}$, and $G_{YZ} \rightarrow G_{YZt}G_{V0}G_{V1}$.

Configuration h) in figure 6.63 corresponds to an YZ gridline which is not connected to any other gridline. In this case, the $G_{YZ} \rightarrow G_{YZt}$ substitution rule is used.

6.2.7 The Bar Structure: B_x and B_y

This subsection discusses the rules which classify the primitives associated with a single bar. These primitives correspond to the 3D structure, the bar text and the numerical data. Table 6.12 presents the nonterminals. These rules are used when a bar is detected and thus a B_X or B_Y nonterminal is encountered in the derivation. The B_X nonterminal represents a vertical bar and the B_Y nonterminal represents a horizontal bar.

Table 6.12: A list of the axis chart nonterminals for the 3D bar structures.

Symbol	Common Name	Primitive Detected	Primitive Required	Description
B_X	Bar - X		Solid region	Starts analysis of the vertical bar.
B_{XT}	Bar - X, Top	Solid region	Solid region	Top of vertical bar.
B_{XR}	Bar - X, Right	Solid region	Solid region	Right side of vertical bar.
B_{XL}	Bar - X, Left	Solid region	Solid region	Left side of vertical bar.
T_X	Text		Solid region	Starts the detection process for the text.
T_{XHI}	Text, Horizontal Index	Text	Solid region	Detects the text directly below the vertical bar.
T_{XND}	Text, numerical data.	Text	Solid region	Detects the numerical data.
B_Y	Bar - Y		Solid region	Starts analysis of the horizontal bar.
B_{YT}	Bar - Y, Top	Solid region	Solid region	Top of horizontal bar.
B_{YR}	Bar - Y, Right	Solid region	Solid region	Right side of horizontal bar.
B_{YB}	Bar - Y, Bottom	Solid region	Solid region	Bottom of bar.
T_{YVI}	Text, Vertical index	Text	Solid region	Detects the text directly to the left the bar.
T_{YND}	Text, numerical data	Text	Solid region	Detects the numeral data

Vertical Bars

When a vertical bar chart bar is classified the B_X nonterminal is encountered in the derivation. Substituting the B_X nonterminal requires searching for the 3D structure of the bar. This is done by looking for solid regions adjacent to the bar with a reduced luminance. From experimental observation it was discovered that the side has a luminance of about 80% of the front and the top has a luminance of about 40% of the front.

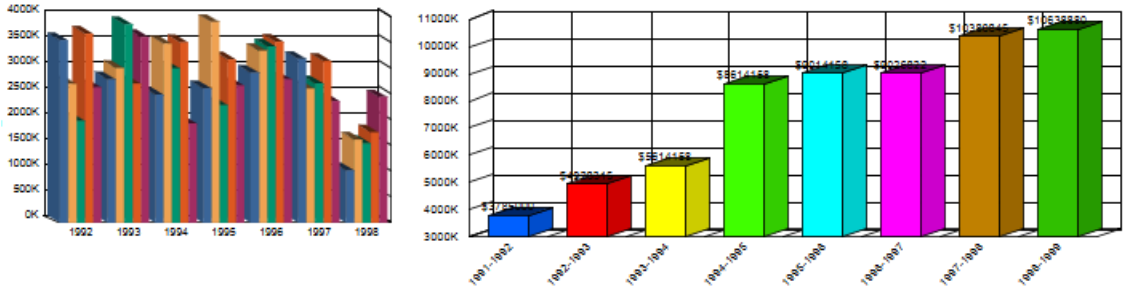


Figure 6.64: The two possible orientations for the bars. The left chart has the side of the bars to the left and the right chart has the side of the bars to the right.

There are three possible components in a 3D vertical bar chart. The components are the top (B_{XT}), the left (B_{XL}) and right sides (B_{XR}), and the text (T_X). A bar will have either a right side or a left side, but never both. A 2D chart will not have any sides but will still contain text elements.

Each rule in equation 6.42 corresponds to a different side configuration. On a single chart all bars should use the same rule because each bar should have the same side configuration. The leftmost rule corresponds to bars where the side is visible to the right of the bar. The middle rule corresponds to bars where the side is visible to the left of the bar. The rightmost rule corresponds to 2D charts.

$$B_X \rightarrow B_{Xt}B_{XR}B_{XT}T_X \qquad B_X \rightarrow B_{Xt}B_{XL}B_{XT}T_X \qquad B_X \rightarrow B_{Xt}T_X \tag{6.42}$$

Substituting the T_X nonterminal starts the classification of the text elements. The horizontal bar text (T_{XHI}) is classified by searching for text blocks below the chart bar. The numerical data (T_{XND}) is isolated by searching for text blocks superimposed on the bar or directly above it. It should be noted that the numerical data text blocks

are detected using the Gabor filter (See section 5.2.2 and appendix A.2). Equation 6.43 presents the rules used to substitute the T_X nonterminal.

$$T_X \rightarrow T_{XHI}T_{XND} \qquad T_X \rightarrow T_{XHI} \qquad T_{XHI} \rightarrow T_{XHIt} \qquad T_{XND} \rightarrow T_{XNDt} \quad (6.43)$$

Horizontal Bars

When a horizontal bar chart bar is classified, the B_Y nonterminal is encountered in the derivation. Substituting the B_Y nonterminal requires searching for the 3D structure of the horizontal bar. This is done by searching for solid regions adjacent to the bar with a reduced luminance. From experimental observation it was discovered that the side has a luminance of about 80% of the front and the top has a luminance of about 40% of the front.

Similar to the vertical bars, there are three possible components to a horizontal bar. The components are right side (B_{YR}), the top (B_{YU}) and bottom (B_{YD}), and text (T_Y). A bar will have either a top or a bottom, but never both. A 2D chart will not have a bottom or a top.

Each rule in equation 6.44 corresponds to a different side configuration. The same rule should be used for all bars of a chart. This is because all the bars have should have the same side configuration. The first rule corresponds to charts where the side of the bar is visible underneath. The second rule corresponds to charts where the side of the bar is visible above. The last rule corresponds to 2D charts.

$$B_Y \rightarrow B_{Yt}B_{YU}B_{YR}T_Y \qquad B_Y \rightarrow B_{Yt}B_{YD}B_{YR}T_Y \qquad B_Y \rightarrow B_{Yt}T_Y \qquad (6.44)$$

Substituting the T_Y nonterminal results in the isolation and classification of the text elements associated with a bar. The vertical index (T_{YVI}) is isolated and classified by searching for text blocks to the left of the chart bar. The numerical data (T_{YND}) is isolated and classified by searching for text blocks superimposed on the bar or directly to the right of it. It should be noted that the numerical data text blocks are often only segmented using the Gabor filter (See section 5.2.2 and appendix A.2). Equation 6.45 presents the rules used to substitute the T_Y nonterminal.

$$T_Y \rightarrow T_{YVI}T_{YND} \qquad T_Y \rightarrow T_{YVI} \qquad T_{YVI} \rightarrow T_{YVIt} \qquad T_{YND} \rightarrow T_{YNDt} \tag{6.45}$$

$$\begin{array}{llll} B_{Xt} \rightarrow b_{xt} & B_{Yt} \rightarrow b_{yt} & T_{XHIt} \rightarrow t_{xhi} & T_{XNDt} \rightarrow t_{xnd} \\ T_{YVIt} \rightarrow t_{ti} & T_{YNDt} \rightarrow t_{ynd} & & \end{array} \tag{6.46}$$

6.2.8 Line Chart Polyline

This subsection describes the P nonterminal and the rules used to detect the polylines. Each polyline is a series of coloured curves. These curves are initially detected using the preprocessing steps outlined in section 6.2.1. These polylines can occlude other polylines, the gridlines, or other chart components. In this approach it is assumed that every polyline has a different colour and that the each polyline can be uniquely identified by its colour. Not all charts will contain a polyline.

Substituting the P nonterminal requires finding the first coloured curve. If multiple coloured curves are remaining, then the first rule in equation 6.47 is chosen. If only a single coloured curve remains, then the second rule in equation 6.47 is chosen.

$$P \rightarrow P_tP \qquad P \rightarrow P_t \qquad P_t \rightarrow p \tag{6.47}$$

Chapter 7

Experimental Results

This chapter describes the methodology for evaluating the proposed approach and discusses the results of the evaluation process. The evaluation is performed by manually comparing the output of the approach with the original image. For each component there are several different types of errors which can occur during the detection and classification steps. Examples of each error are presented and discussed. A quantitative analysis was performed by aggregating all the errors occurring across the experimental database. The running time to process a page is around 1 second or less on an Intel i7 970 processor. The time is approximately the same for pages with pie charts as it is with axis charts.

Section 7.1 starts by describing the chart database used for evaluation. In order to evaluate the approach, a graphical user interface was developed. This is discussed in section 7.2. The system used for evaluation is expanded upon in section 7.3. The results for the pie charts are then presented in section 7.5 and the results for the bar charts are presented in section 7.6.

7.1 Experimental Database

A collection of digitally rendered document images which contain chart, tables, images, and other graphics have been supplied by SAP Vancouver. These documents have been generated using Crystal Reports, a business intelligence application which can create reports. Crystal Reports allows users to graphically combine data from various sources including many common SQL databases. These reports do not contain text in paragraph form and and sometimes do not have a Manhattan layout.

The experimental database contains over 14000 raster images from SAP. All the documents which contain charts have been selected from these 14000 images. Table 7.1 shows the number of each type of chart that was analyzed and how many pages were analyzed.

Table 7.1: The number of charts evaluated.

Chart Style	Chart Count	Page Count
2D Pie	51	38
3D Pie	72	69
2D Vertical Bar	101	93
2D Horizontal Bar	19	18
3D Vertical Bar	101	96
3D Horizontal Bar	24	23
Line	100	97
Total	468	424

7.2 Graphical User Interface

In order to test and evaluate the code a graphical user interface was developed. This interface has two different display windows. The first window (ie. the report window) displays the report which is currently being analyzed. The second window (ie. the information window) is a list of the detected properties of the report in text form. Both windows are dynamic such that the information displayed is dependent on the keyboard keys pressed and the location of the mouse cursor. We use the drawing functions which are built into OpenCV[1] to present information on the windows.

7.2.1 Report Window

The report window displays the current document image which is being analyzed. This window is also used to display the results at different stages of the analysis. Different key strokes as well as the mouse are used to control what is being displayed. This is extremely valuable in evaluating the algorithm as it allows a human user to visualize the results.

[1] http://opencv.org/

One of the display modes allows the user to see the results of the segmentation and classification. This places a bounding box around every segmented block. An example of this can be seen in figure 7.1. In this figure, the bars, the text, as well as the entire chart have all been segmented and classified. The colour of the bounding boxes indicate the classification results. The blue, red, and green bounding boxes indicate whether the block was classified, was not classified, and was classified as a chart respectively. Blocks left unclassified are not common with chart images, therefore an example showing this result is not provided.

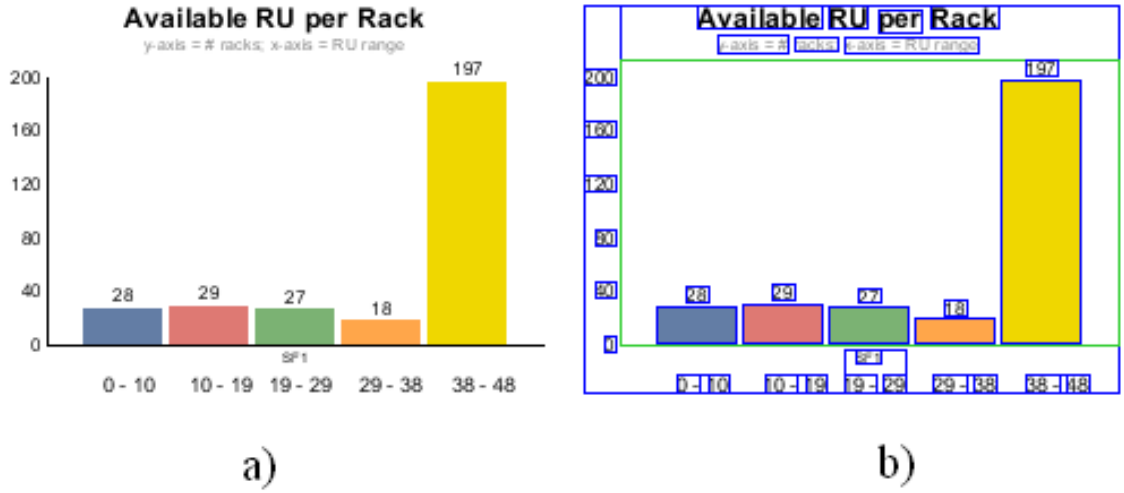


Figure 7.1: An example of what a segmented bar chart would look like if it was displayed by the GUI. a) The original chart image. b) Chart image showing the results of the segmentation. Blue rectangles indicate classified blocks, red rectangles indicate unclassified blocks, and green rectangles indicate charts. There are no red bounding boxes in this image.

The segmentation and classification mode is useful for quickly recognizing a segmentation or classification error. If a chart is not classified correctly it will have a red or blue bounding box instead of a green rectangle. If the rectangle is red, the chart is unclassified. If the bounding box is blue, it means the chart is classified as something other than a chart (for example, as a table). If the chart is not segmented correctly the bounding box which completely encompasses the chart will also contain graphics which are not part of the chart.

Another feature of the report window is the ability to draw the detected ellipse of a pie chart. An example of this is presented in figure 7.2. In addition to redrawing an ellipse, the GUI can also draw a box around the detected axis from the axis (see

figure 7.3). The width and height of the box indicate the length of the x and y axis. The bottom of the box is on the x-axis and the left side of the box is on the y-axis.

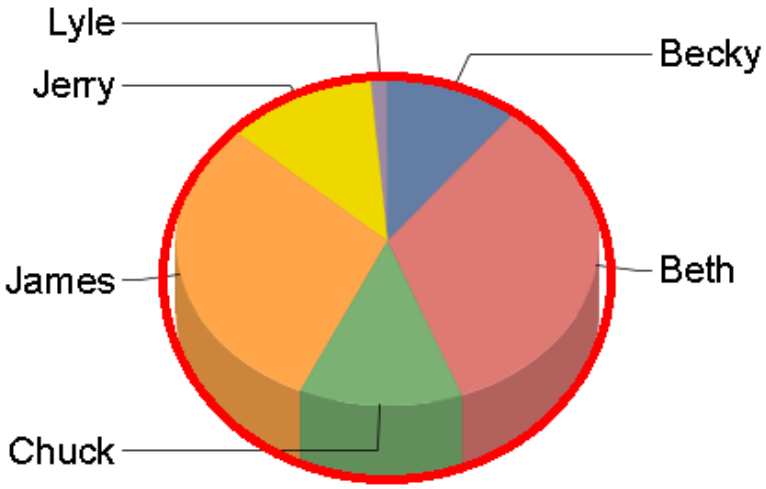


Figure 7.2: The ellipse which was detected in the pie chart classification stage.

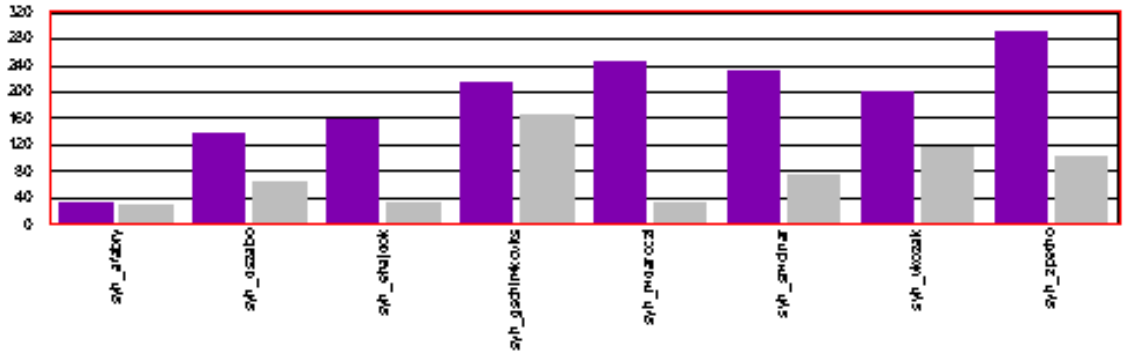


Figure 7.3: A bounding box showing the location and size of the detected axis.

7.2.2 Information Window

This window presents a dynamic summary of the report, as well as of the block that the mouse is hovering over. Table 7.2 lists some of the properties displayed. Several of these properties require the mouse hovering over the window. When the mouse hovers over a segmented block, the detected information about that block is displayed. In table 7.2, all properties which start with the word "mouse" convey properties at the pixel location specified by the mouse cursor position, while all properties which start

with the word "block" require the mouse be over a segmented block. The remaining properties belong to the entire page.

Table 7.2: Some of the information displayed in the information window.

Property	Example output	Description
Mouse position	(100,100)	The x and y coordinates of the point the mouse is over, in pixels.
Mouse colour	(255,255,255)	The RGB value of the pixel the mouse is over.
Block number	10	The block number of the block the mouse is over.
Block classification	Chart	The detected block classification.
Block sub-class	Pie chart	The sub-classification of the block.
Block bottom left	(100,100)	The bottom left coordinate of the block, in pixels.
Block size	(100,100)	The size of the block, in pixels.
Block text string	Country	Any text string, the output of the OCR performed on a text block.
Page Size	(612,792)	The size of the page, in pixels
Report number	766	Numeric count of the reports analyzed or detected.
Total blocks	155	Count of the segmented blocks.

7.2.3 Redrawing the Pie Charts

One of the more powerful abilities of the user interface is its ability to redraw the curves composing a chart that has been detected and recognized. Text blocks are not analyzed, they are just copied over because OCR is not applied in this evaluation. This is a qualitative method of chart evaluation which allows a user to easily spot errors. Figure 7.4 presents an example of a redrawn chart slice and a redrawn chart. The curves of the drawn chart have been colour coded to make it easy to see the classification of each curve. The redrawn curves and the original chart can be displayed in the same location so that the user can toggle between them in order to detect the differences.

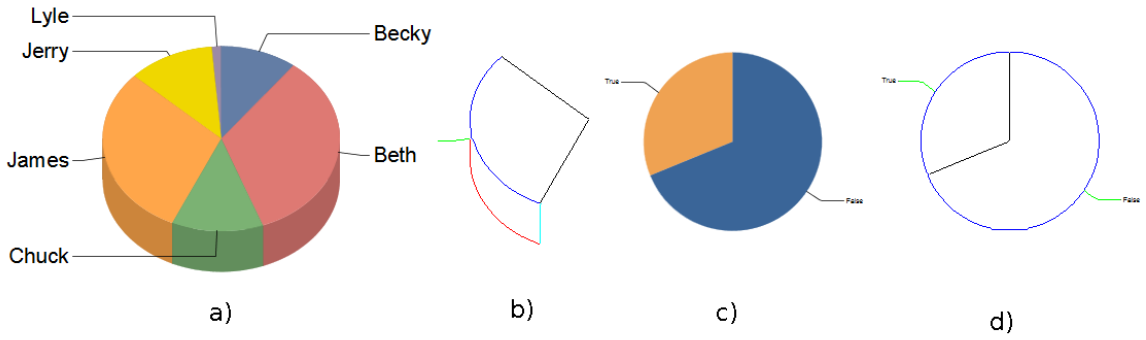


Figure 7.4: An example of a redrawn pie slice (a) and a redrawn chart (d). Figures a) and c) are the original images for figures b) and d) respectively. The colours indicate the classification. Black curves are radii curves, blue curves are top edge curves, red curves are bottom edge curves, green curves are label connection curves, and turquoise curves are side curves. It is best if this figure is viewed in colour.

7.2.4 Redrawing the Axis Charts

Similar to pie charts, the bar and line charts can also be drawn. Since bar and line charts have more components, there are more options for redrawing them. Individual components can be included or omitted. For example, we can only draw the detected bars. Figure 7.5a displays the detected curves within the chart image. Figure 7.5b displays a redrawn version of the chart. The curves of the gridlines and axis have been coloured to highlight their classification within the chart. Any components which are not detected or classified correctly are not drawn or are drawn incorrectly.

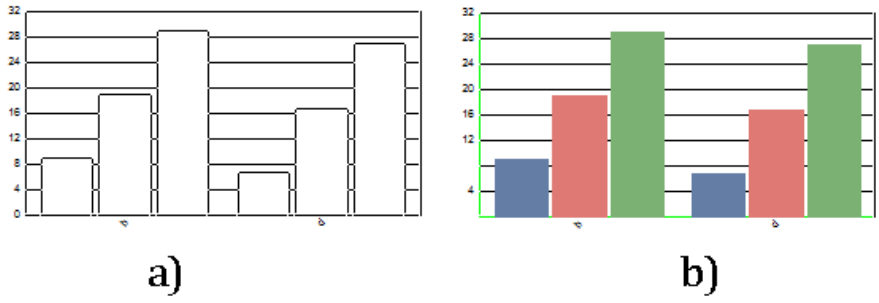


Figure 7.5: a) The original chart image. b) The detected and classified components drawn to screen. The curve colours indicate the classification. The green curves are the axis and the black curves are the gridlines.

7.3 The Semi-Automatic Evaluation System

In order to evaluate the algorithm we developed a semi-automated system. The system was designed to facilitate the recording of the errors and reduce the time required to perform the evaluation. Once each chart is analyzed, the results are printed to the terminal and to a text file.

The chart which is currently analyzed is displayed on the user interface. The chart redraw functionality is used to quickly detect and recognize errors. When an error is found the user places the mouse over the chart with the problem and presses key to indicate the nature of the error.

In order to evaluate the pie chart classification the user needs to specify how many pie slices the chart contains and if the chart contains slice labels. In addition, the user specifies the types of errors and how many errors are present.

Evaluating the classification of the bar and line charts is a bit more difficult because there are more components which need to be evaluated. The evaluation needs to include the axis, the grid lines, the text, the bars, and the polylines. In order to reduce the amount of information the user needs to enter into the system, the count of the detected components is displayed to screen.

7.4 Precision, Recall, and F-Score

In order to create a more meaningful interpretation of the results we have calculated the precision (P), recall (R) and F-Score for each component. The formulas are seen in equations 7.1, 7.2, and 7.3 where TP is true positive, FP is false positive, and FN is false negative.

$$\text{Precision} = P = \frac{TP}{TP + FP} \tag{7.1}$$

$$\text{Recall} = R = \frac{TP}{TP + FN} \tag{7.2}$$

$$\text{F-Score} = \frac{P \cdot R}{P + R} \tag{7.3}$$

7.5 Experimental Evaluation for Recognition of Pie Charts

We evaluate the approach on both 2D and 3D pie charts and present the results separately. The 2D pie chart database include 51 pie charts on 38 different reports. The 3D pie chart database include 72 pie charts on 69 different reports. Table 7.3 presents the number of reports analyzed in a more compact form.

Table 7.3: The number of 2D and 3D charts evaluated.

	Chart Count	Report Count
2D Pie Charts	51	38
3D Pie Charts	72	69

Experimental evaluation was performed using two different paradigms. The slice evaluation (described in section 7.5.1)investigates the ability of the proposed approach to correctly identify slices. The curve evaluation (described in section 7.5.2) explores the ability of the proposed approach to correctly classify each curve. In the following sections we will describe the possible errors and how our algorithm performed.

7.5.1 Slice Evaluation

This subsection analyzes the occurrence of errors in the slice identification process. The 5 different error types are splitting, merging, incomplete slice, missed slice, and false detection. We use visual examples to discuss of each error type.

The left and middle illustrations of 7.6 show the slice splitting error while the slice merging error is shown on the right. The most frequent cause for these errors are the presence of small gaps in the detected bottom or top curve. These gaps cause the curve to be split into several parts. The original chart image for figure 7.6 is shown in figure 7.7.

An incomplete slice detection occurs when not all curves belonging to a given slice have been detected or assigned to that slice. If the curves in the left figure of 7.6 were associated with a slice, but the curves in the middle figure of 7.6 were not associated with any slice, then the slice would be considered an incomplete slice.

A merged slice is when two different slices were detected as a single slice. The right side of figure 7.6 shows two slices merged into one.

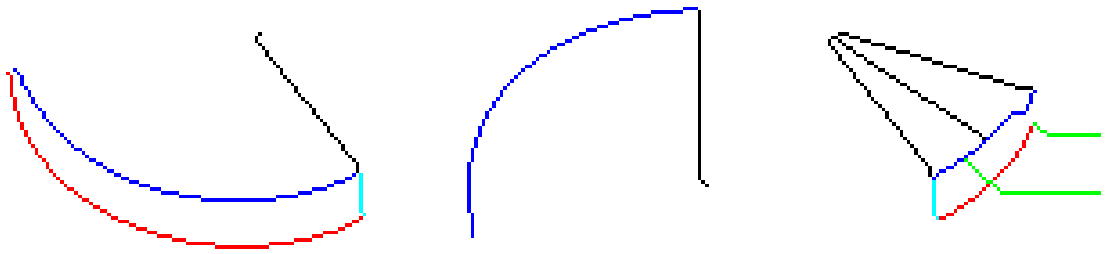


Figure 7.6: The first part of a split slice (left). The second part of a split slice (middle). Two slices merged into one (right). The original chart image is presented in figure 7.7. It is best if this figure is viewed in colour.

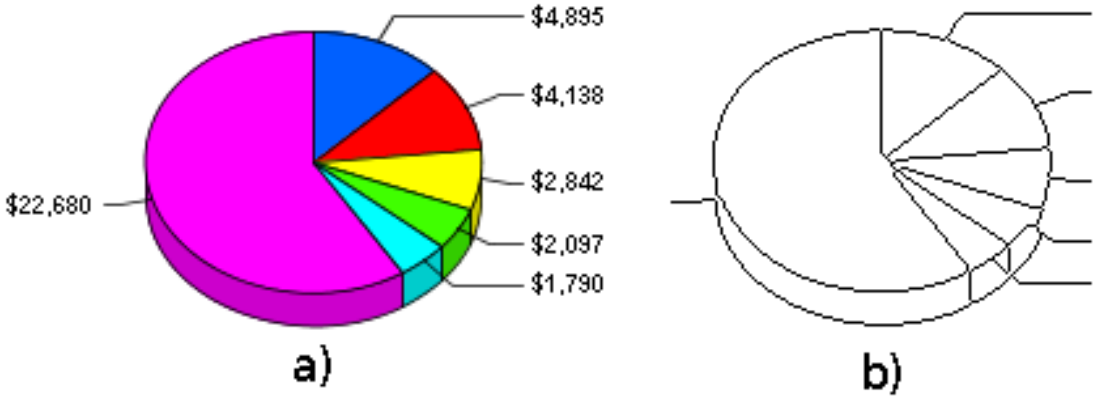


Figure 7.7: a) The original chart image for figure 7.6. b) The extracted curves. It is best if this figure is viewed in colour.

A missed slice occurs when an entire slice is not classified. This error occurs mostly for very thin slices. A false detection is when the algorithm detects a slice when there is no slice present.

Table 7.4 presents the results for both 2D and 3D pie slices. There are a few reasons for slices to not be detected properly. The first is imperfections in the curve detection process. If a person zooms in on the detected curves it is possible to see small gaps in some places. Another reason for the errors is when the chart contains very narrow slices. Often the details of these slices are too small to resolve and thus do not get properly detected. In this thesis, for the precision and recall calculations, the false detections are used for false positives, the missed slices are used for false negatives, and correct slices are used for true negatives.

Table 7.4: Results for the Pie Slices

	2D Charts	3D Charts
Split Slice	6	85
Merged Slice	14	23
Incomplete Slice	5	100
Missed Slice	4	66
False Detection	2	47
Correct slices	228	246
Precision	0.99	0.84
Recall	0.98	0.79
F-Score	0.99	0.81

7.5.2 Evaluation of the Curve Classification

This subsection analyzes the occurrence of errors in the curve classification process. It looks to see if the curves are classified correctly. The results for both the 2D and 3D pie charts are presented in table 7.5. We determined the number of false positives, false negatives and the true positives. This is used to calculate the precision and recall.

Sometimes a curve is classified incorrectly. An example of this is a curve along the top of the pie chart cylinder which is classified as a bottom curve. This is a false positive. Another error is when a curve is not classified. This is a false negative. This can occur when a section of the curve is missing near an expected intersection point. For example, in a 2D pie chart, the radius curves should connect to the top curve. If a section of the radius curve is missing next to this intersection, the radius curve will not be classified.

In addition to presenting the errors we also present the total number of correct detections. The precision, recall and F-Score are provided as a performance measure.

Table 7.5: Results for the curves of pie charts.

	2D Charts	3D Charts
Incorrect Classification (FP)	3	24
Missed Classification (FN)	37	176
Correct Classification (TP)	835	1838
Total Curves	872	2014
Precision	0.996	0.987
Recall	0.957	0.913
F-Score	0.977	0.948

7.6 Experimental Evaluation for Recognition of Axis Charts

To simplify the evaluation, the chart components have been abbreviated. These abbreviations are shown in table 7.6 and are used in the tables which present results in the later sections. The terminals for each component are also included for consistency with the grammar found in chapter 6.

Table 7.6: A list of the chart components evaluated.

Classification	Abbreviation	Terminal
Bar	Bar	b_x/b_y
Polyline	P	p
Horizontal Gridline	GX	g_H
Vertical Gridline	GY	g_V
XZ Gridline	GXZ	g_{XZ}
YZ Gridline	GYZ	g_{YZ}
X-Axis	X	x
Y-Axis	Y	y
Horizontal Index	HI	t_{HI}
Vertical Index	VI	t_{VI}
Bar Text	T	t_{xvi}/t_{yvi}
Numerical Data	ND	t_{xnd}/t_{ynd}

7.6.1 Descriptions of the Chart Errors

This subsection presents six different possible errors which can occur within the component classification. Theoretically all of these errors can exist for every chart component. In practice however, certain combinations were not seen. A list of these errors as well as their definitions are presented below.

- Split
- Merge
- Incomplete
- Added
- Missed

- Extra

The first error type occurs when a single component is *split* into multiple regions. For example, in a 3D bar chart, if the side of a bar is classified as another bar, it is considered to be split. An example of a bar which is split into two is seen in figure 7.8.

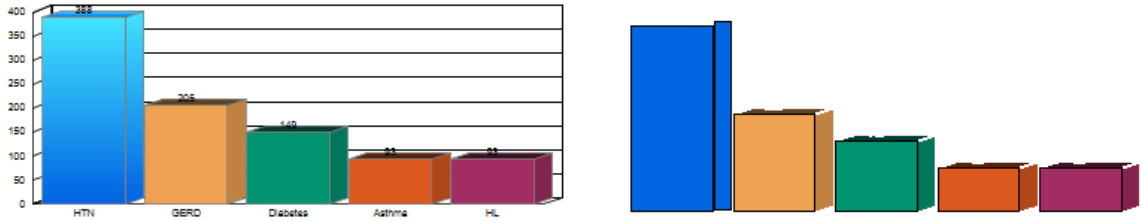


Figure 7.8: The left image is the original chart image and the right image is an example of a chart with a split bar. The blue bar on the left side was drawn as two distinct bars. Note that only the chart bars are shown in the right image. None of the other components were drawn to make it easier to see the split.

Sometimes a component can be split into more than two blocks. If a component is split into N blocks, it is recorded as $N-1$ splits. For example, if a single vertical index is split into three blocks, it has been split twice.

The second error type is when two components are *merged* together. This is the opposite of splitting. Figure 7.9 presents an example of vertical indices which were merged together. The bottom 4 indices were merged together and classified as a single index.

The third error type occurs when a chart component is correctly classified but not completely isolated. If part of a chart component is classified, but not the entire component, it is recorded as being incomplete. For example, in a 3D bar chart, if the front of the bar is classified but the side or top of the bar is not detected, it is labeled as incomplete. An example can be seen in figure 7.10 where two vertical indices were not completely detected.

Sometimes the detected chart component will contain graphics which are not part of the chart component. The term *added* is used to reference this fourth type of error because some graphics are added to the chart component. An example of this error can be seen in figure 7.11. An XZ gridlines is added to the detected x-axis. The red bounding box highlights the XZ gridline on the right image of figure 7.11. In most cases, when this error occurs the entire chart component is also detected.

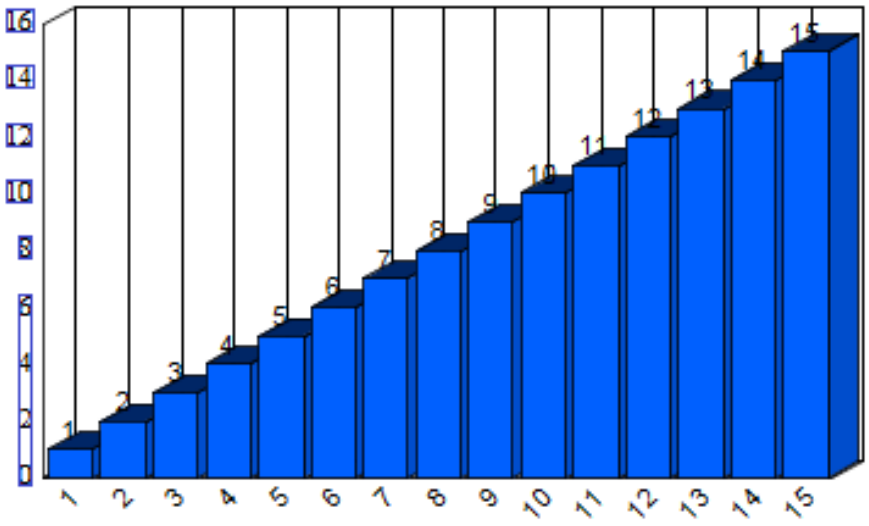


Figure 7.9: An example of a component that was merged. The bounding boxes around the vertical indices show how they were grouped together during classification. The top 5 vertical indices were detected properly. The bottom 4 vertical indices however were merged into a single block.

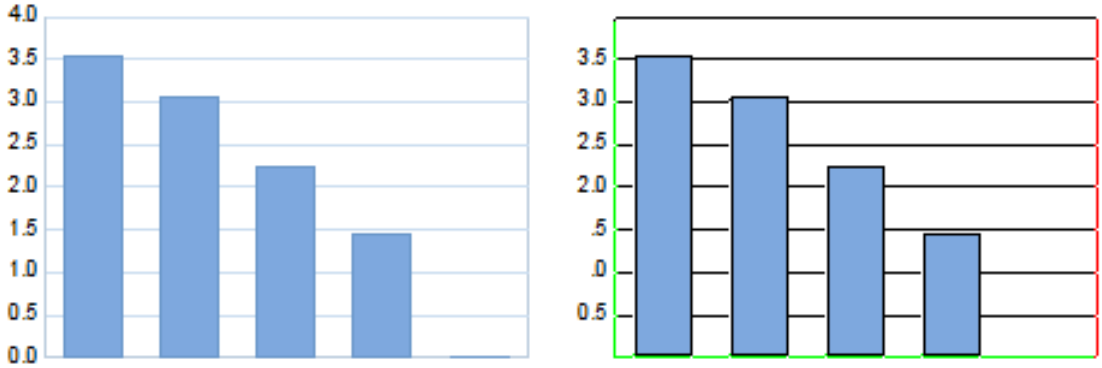


Figure 7.10: The original chart image (left). Two vertical indices were not completely detected. These indices have the values of 1.0 and 1.5. Two vertical indices are also missing. The 0.0 and 4.0 index values were not recognized. The colours of the axis and gridlines are changed intentionally.

The fifth type of error is a *missing* chart component and occurs when the entire chart component was not detected. Figure 7.10 shows an example of a missing vertical index. In this case the 0.0 and 4.0 index markers were not detected.

The sixth and last error type is when an *extra* component is detected. This is the opposite of a missing component. For example, if a bar chart has 10 bars but

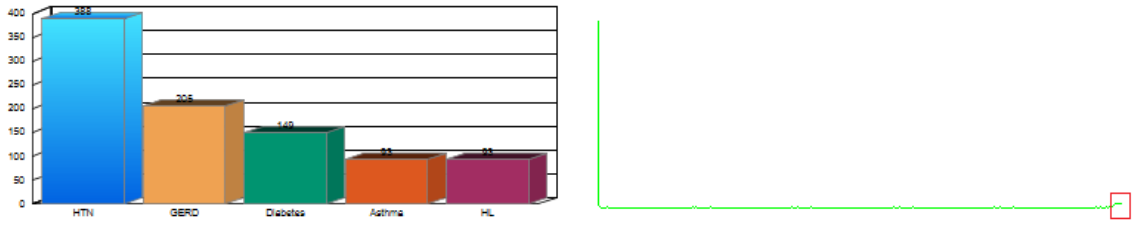


Figure 7.11: The XZ gridline of this chart was added to the x-axis. The original chart image is presented on the left. The right image shows the detected x and y axis. The red bounding box surrounds the XZ gridline which was added to the x-axis.

there were 11 solid regions classified as bars, then there is an extra bar detection. An example of an extra detection can be seen in figure 7.12.

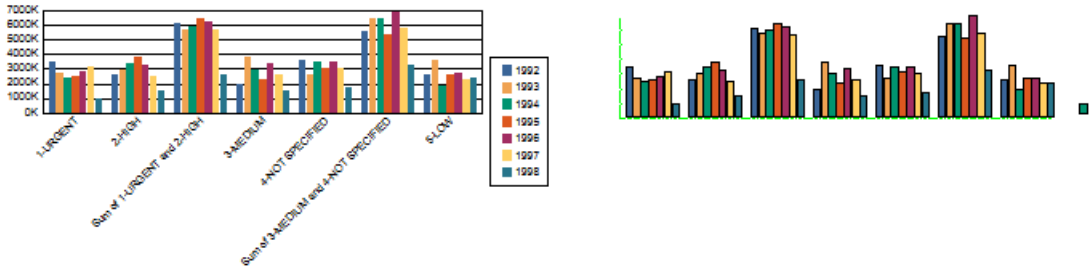


Figure 7.12: Left: The original chart image and the legend associated with it. Right: The redrawn version of the same chart image. Note that part of the legend was classified as a chart bar. This is an extra classification. Only the bars and the axis were redrawn.

One thing which is not obvious is which errors are false positives and which errors are false negatives. As mentioned in subsection 7.6.1, we have 6 types of errors in which the terms split, merge, incomplete, added, missed, extra are used to reference these errors.

In our calculations we used the missing components as the false negative and the extra components as a false positive. The reason the other errors were excluded is that the components were still detected, but were imperfect. For example, a common problem with 3D bar charts was that only part of the bar was detected. In almost all cases either the top of the bar (vertical bar) or the side of the bar was not detected. If the results were later transferred to a chart creation program, this error would not prevent the chart creation program from recreating the chart.

All of these errors are presented in the results. In addition to these errors, the ground truth count of each component is presented.

7.6.2 The Results

This subsection presents the results for the axis charts. The results for each chart type are presented individually. In addition there is a short discussion about the common problems specific to each chart type.

2D Vertical Bar Charts

The results for the 2D vertical bar charts are presented in table 7.7. A total of 101 charts were analyzed and the precision and recall were calculated for each component. The abbreviations for the columns can be seen in section 7.6.

Table 7.7: Results for 2D vertical bar charts.

Error Type	Bar	GX	GY	X	Y	HI	VI	T	ND
Split	0	0	0	0	0	14	0	1	0
Merge	15	3	0	0	0	126	99	127	0
Incomplete	18	33	20	9	4	36	2	228	9
Added	2	6	0	5	1	8	1	7	0
Missing	47	109	1	3	0	43	37	374	184
Extra	7	1	33	0	0	49	31	257	84
Count	2011	685	129	101	101	874	797	1866	252
Precision	1.00	1.00	0.80	1.00	1.00	0.94	0.96	0.85	0.45
Recall	0.98	0.84	0.99	0.97	1.00	0.95	0.95	0.80	0.27
F-Score	0.94	0.92	0.88	0.98	1.00	0.95	0.96	0.82	0.34

There were no problems which were specific to this chart type. There are a few problems which affect all axis charts. The gridlines between the bars were often missed. This is because they are often very short and have no connection with other gridlines. Another problem is text which touches the axis. This caused problems with the axis and gridline detection.

The most difficult component to detect is the numerical data (ND), mainly because this component often overlaps other components.

2D Horizontal Bar Charts

The results for the 2D horizontal charts are presented in table 7.8. A total of 19 charts were evaluated. Similar to the 2D vertical bar charts there are no problems

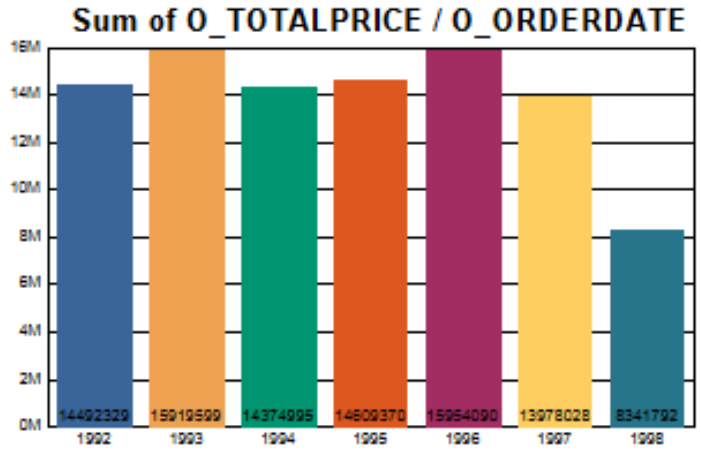


Figure 7.13: An example of the numerical data superimposed on top of the bar.

which are unique to this chart type.

Table 7.8: Results for 2D horizontal bar charts.

Error Type	Bar	GX	GY	X	Y	HI	VI	T	ND
Split	0	0	0	0	0	0	0	0	0
Merge	0	0	0	0	0	0	67	12	0
Incomplete	0	4	5	5	3	0	0	15	2
Added	3	3	2	0	0	0	0	0	0
Missing	18	2	28	0	2	21	14	29	13
Extra	0	23	5	0	1	12	5	1	2
Count	136	19	89	19	19	131	153	131	39
Precision	1	0.43	0.92	1.00	0.94	0.90	0.97	0.99	0.93
Recall	0.88	0.89	0.69	1.00	0.89	0.84	0.91	0.78	0.67
F-Score	0.94	0.58	0.79	1.00	0.91	0.87	0.94	0.87	0.78

3D Vertical Bar Charts

We analyzed a total of 101 3D charts. There are a few differences between 2D charts and 3D charts which should be highlighted. One difference is that the bars on a 3D chart have both a side and a top. These can sometimes be misclassified as another bar. When this occurs it is often a split error. When the front of the bar is detected but either the side or the top is missing, it is an incomplete error.

The 3D axis charts contain the XZ and YZ gridlines. These gridlines are typically very short and difficult to detect. For the vertical bar charts, many of the XZ gridlines are occluded by the bars. This reduces the number XZ gridlines on the chart and decreases the detection rate.

One observation is that the 3D vertical bar charts had significantly fewer bars than the 2D vertical bar charts. This is most likely because the 3D bars take up more space on the page than 2D bars.

Table 7.9: Results for 3D vertical bar charts.

Error Type	Bar	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	8	0	0	0	0	0	0	144	7	0	0
Merge	0	10	0	0	0	0	0	5	154	20	0
Incomplete	231	42	27	1	1	26	16	153	19	78	36
Added	8	116	21	5	2	8	0	0	0	0	0
Missing	79	126	47	52	203	8	3	13	52	76	183
Extra	89	35	64	158	7	7	0	33	31	44	152
Count	827	674	360	144	563	101	101	769	725	733	340
Precision	0.89	0.94	0.83	0.37	0.98	0.93	1	0.96	0.96	0.94	0.51
Recall	0.90	0.81	0.87	0.64	0.64	0.92	0.97	0.98	0.93	0.90	0.46
F-Score	0.89	0.87	0.85	0.47	0.77	0.92	0.98	0.97	0.94	0.92	0.48

3D Horizontal Bar Charts

There were 24 different charts analyzed in this dataset. Although the 3D horizontal bar charts had similar issues to the 3D vertical bar charts, there was one issue that stood out with the horizontal charts. There was significantly more charts which contained a background colour which was not white. This is the primary source of error for the bars and gridlines.

Table 7.10: Results for 3D horizontal bar charts

Error Type	Bar	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	0	0	0	0	0	0	0	4	2	0	0
Merge	0	0	0	0	0	0	0	2	60	0	0
Incomplete	124	3	6	3	0	0	15	0	0	32	0
Added	1	1	0	43	2	0	2	0	3	0	0
Missing	8	80	43	72	6	0	7	13	11	8	6
Extra	72	54	11	22	3	0	5	25	10	3	14
Count	226	114	169	162	13	24	24	189	221	219	9
Precision	0.75	0.38	0.92	0.80	0.70	1.00	0.77	0.88	0.96	0.99	0.18
Recall	0.96	0.30	0.75	0.56	0.54	1.00	0.71	0.93	0.95	0.96	0.33
F-Score	0.84	0.34	0.83	0.66	0.61	1.00	0.74	0.90	0.96	0.97	0.23

Line Charts

The results for the line charts are presented in table 7.11. A total of 100 charts were analyzed. Since these charts contained polylines they have their own set of challenges. Many of these charts have multiple intersecting polylines which increase the difficulty. An example of a chart with multiple intersecting polylines is presented in figure 7.14.

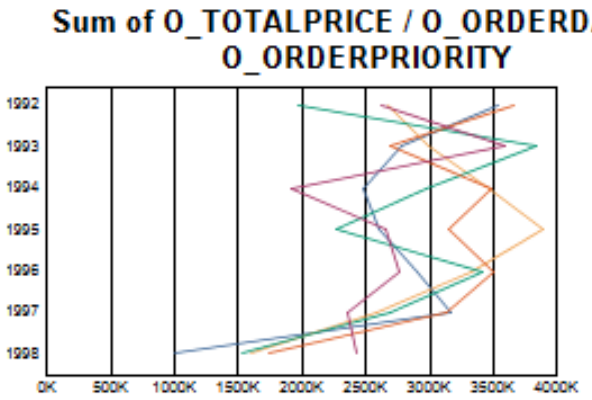


Figure 7.14: An example of a chart with multiple polylines.

Table 7.11: Results for line charts.

Error Type	P	GX	GY	X	Y	HI	VI
Split	0	3	0	0	0	66	0
Merge	15	10	0	0	0	65	54
Incomplete	99	97	3	9	17	270	8
Added	43	80	8	6	4	10	21
Missing	69	10	2	5	0	218	95
Extra	21	88	448	4	0	83	19
Count	316	685	175	100	100	1259	796
Precision	0.92	0.88	0.28	0.96	1.00	0.93	0.97
Recall	0.78	0.99	0.99	0.95	1.00	0.83	0.88
F-Score	0.84	0.93	0.44	0.95	1.00	0.88	0.92

A common problem with line charts is mistaking the polylines for gridlines. Some of the gridlines are not black and so colour cannot be used to distinguish between the gridlines and the polylines. In some situations, the polyline is superimposed on the gridline. When this occurs it is common to not detect the polyline.

In this dataset there were more horizontal gridlines than vertical gridlines. This resulted in more intersections between horizontal gridlines and the polylines and thus significantly increased the number of extra detections for vertical gridlines.

7.6.3 Aggregating the Results

To provide more detail into the performance of the algorithm the results are aggregated into groupings with a similar style or format. These formats are 2D bar charts 3D bar charts, vertical bar charts, horizontal bar charts and all charts. To calculate these tables we have combined the results from the tables in the previous subsection together. The precision and recall was recalculated for the new values.

The combined results for the both 2D vertical bar charts and 2D horizontal bar charts can be seen in table 7.12. A total of 120 charts from 111 pages are represented in this aggregation. Individually, the vertical bar chart has few vertical gridlines compared to the horizontal gridlines. A lot of vertical bar charts have only a single vertical gridline, as seen in figure 7.13. Similarly, the horizontal bar charts have few horizontal gridlines. Combining them creates a more even distribution of chart components.

Table 7.12: Results for 2D bar charts.

Error Type	Bar	GX	GY	X	Y	HI	VI	T	ND
Split	0	0	0	0	0	14	0	1	0
Merge	15	3	0	0	0	126	166	139	0
Incomplete	18	37	25	14	7	36	2	243	11
Added	5	9	2	5	1	8	1	7	0
Missing	65	111	29	3	2	64	51	403	197
Extra	7	24	38	0	1	61	36	258	86
Count	2147	704	218	120	120	1005	950	1997	291
Precision	1.00	0.96	0.83	1.00	0.99	0.94	0.96	0.86	0.52
Recall	0.97	0.84	0.87	0.98	0.98	0.94	0.95	0.80	0.32
F-Score	0.98	0.90	0.85	0.99	0.98	0.94	0.95	0.83	0.40

Similar to how the 2D charts are combined together, the 3D vertical bar charts and the 3D horizontal bar charts are combined together. Combining the 3D bar charts is useful for the same reason as the 2D charts, it has a more even distribution of chart components. For this combination, there was 125 charts from 119 pages. The results for these charts can be seen in table 7.13.

Another way to aggregate the results is to combine both 2D and 3D vertical bar charts together. This combination included 202 charts from 189 pages. This combination is presented in table 7.14. There are no problems for these charts which have not already been specified in the previous section.

Similar to the vertical bars, the horizontal bar charts were combined together and are present the results in table 7.15. For this aggregation 43 charts were analyzed from

Table 7.13: Results for 3D bar charts.

Error Type	Bar	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	8	0	0	0	0	0	0	148	9	0	0
Merge	0	10	0	0	0	0	0	7	214	20	0
Incomplete	355	45	33	4	1	26	31	153	19	110	36
Added	9	117	21	48	4	8	2	0	3	0	0
Missing	87	206	90	124	209	8	10	26	63	84	189
Extra	161	89	75	180	10	7	5	58	41	47	166
Count	1053	788	529	306	576	125	125	958	946	952	349
Precision	0.86	0.87	0.85	0.50	0.97	0.94	0.96	0.94	0.96	0.95	0.49
Recall	0.92	0.74	0.83	0.59	0.64	0.94	0.92	0.97	0.93	0.91	0.46
F-Score	0.84	0.80	0.84	0.54	0.77	0.94	0.94	0.95	0.94	0.93	0.47

Table 7.14: Results for vertical bar charts.

Error Type	Bar	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	8	0	0	0	0	0	0	158	7	1	0
Merge	15	13	0	0	0	0	0	131	253	147	0
Incomplete	249	75	47	1	1	35	20	189	21	306	45
Added	10	122	21	5	2	13	1	8	1	7	0
Missing	126	235	48	52	203	11	3	56	89	450	367
Extra	96	36	97	158	7	7	0	82	62	301	236
Count	2838	1359	489	144	563	202	202	1643	1522	2599	592
Precision	0.97	0.97	0.82	0.37	0.98	0.96	1.00	0.95	0.96	0.88	0.49
Recall	0.96	0.83	0.90	0.64	0.64	0.95	0.99	0.97	0.94	0.83	0.38
F-Score	0.96	0.89	0.86	0.47	0.77	0.95	0.99	0.96	0.95	0.85	0.43

41 different document images. As in the vertical bar charts, there are no problems for these charts which have not already been specified in the previous section. One common difference in this set, which was mentioned previously, is that some of the charts contained a non-white background. This was the primary source of error for the bars and gridlines.

Table 7.15: Results for horizontal bar charts

Error Type	Bar	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	0	0	0	0	0	0	0	4	2	0	0
Merge	0	0	0	0	0	0	0	2	127	12	0
Incomplete	124	7	11	3	0	5	18	0	0	47	2
Added	4	4	2	43	2	0	2	0	3	0	0
Missing	26	82	71	72	6	0	9	34	25	37	19
Extra	72	77	16	22	3	0	6	37	15	4	16
Count	362	133	258	162	13	43	43	320	374	350	48
Precision	0.82	0.40	0.92	0.80	0.70	1.00	0.85	0.89	0.96	0.99	0.64
Recall	0.92	0.38	0.72	0.56	0.54	1.00	0.79	0.89	0.93	0.89	0.60
F-Score	0.87	0.39	0.81	0.66	0.61	1.00	0.82	0.89	0.94	0.94	0.62

The results for all five axis chart types are combined together in table 7.16. During the evaluation, the charts were separated into distinct styles (ie. 2D bar, 3D bar etc.), however the algorithm itself did not know the chart type before analyzing it. If the algorithm was evaluated by mixing all the axis charts together, this is what the results would be. This represents 245 chart images from 230 pages.

Table 7.16: Results for all bar charts

Error Type	Bar	P	GX	GY	GXZ	GYZ	X	Y	HI	VI	T	ND
Split	8	0	3	0	0	0	0	0	228	9	1	0
Merge	15	15	23	0	0	0	0	0	198	434	159	0
Incomplete	373	99	179	61	4	1	49	55	459	29	353	47
Added	14	43	199	31	48	4	19	7	18	25	7	0
Missing	191	69	327	121	124	209	16	74	298	209	487	386
Extra	103	21	201	561	180	10	11	66	202	96	305	252
Count	3200	316	2177	922	306	576	345	345	3860	2692	2949	640
Precision	0.97	0.92	0.92	0.62	0.50	0.97	0.97	0.84	0.95	0.97	0.91	0.72
Recall	0.94	0.78	0.87	0.88	0.59	0.64	0.96	0.82	0.93	0.93	0.86	0.62
F-Score	0.96	0.84	0.89	0.73	0.54	0.77	0.96	0.83	0.94	0.95	0.88	0.67

Chapter 8

Conclusions

This thesis proposes a new system for classifying the components of chart images. The system consists of two distinct syntactic grammars, one for classifying the pie chart components and the other for bar and line charts. For each of these chart types, the entire process is outlined and presented. This process was outlined in chapter 1.

The processes required prior to the chart grammars are outlined in chapter 5. These include segmentation, primitive detection and chart detection. One of the contributions of this research is the segmentation at oblique cuts which was published in [52].

The primary contribution in this thesis is the chart grammars presented in chapter 6. The two grammars, one for pie charts (section 6.1) and one for bar and line charts (section 6.2), are new methods for chart information extraction. For pie charts this involves both recognizing the roles of each component, but also which slices they belong to. For bar and line charts, the grammar classifies the bars, gridlines, axes, horizontal and vertical indices, numerical data, and polylines. This a complete system for chart understanding.

In order to evaluate the approach a semi-automatic system was developed. This system draws each chart one component at a time, which allows the user to verify that each component is detected correctly. The differences can be easily seen because the semi-automatic system allows a human user to quickly compare the redrawn chart to the original. The user then enters detailed information about the error that occurred. This resulted in a detailed analysis of the limitations of the system.

There are several ways in which this research can be extended. For example, one could analyze different styles of charts. These can include hi-low charts, doughnut charts, pie charts where a slice is removed and area charts. All of these chart types

appeared in SAP's testing database, however not in sufficient quantities. However, these charts could come from a different dataset, for example the IAPR contests on graphics recognition [17]. The charts could also be custom generated using a chart reporting tool. This would allow a comparison between the extracted information and original source.

Another possible area of improvement would be to use different algorithms for detecting the primitives. This could increase the accuracy of the detection process as well as allow the system to detect more complex charts. For example, it may be possible to detect a chart similar to the one found in figure 8.1[1].

Figure 8.1: An example of a chart that would not work under the current system. Licensed under the creative commons (Zemanta).

Additionally, the missing components based on contextual information. For example, if the numbers 0, 1, 2, and 4 were detected as vertical indices, but not 3, the computer could predict that an index with the a value of 3 should be placed between the indices 2 and 4.

The evaluation of the approach may be improved by reporting the number of

[1] http://pietistschoolman.com/2011/07/23/that-was-the-week-that-was-5/

errors on a per-chart basis. This would allow show how many charts were detected perfectly as well as how many had major problems.

Another method for evaluating the results would be to create a simple baseline method and compare my results to this method. This could be done on a per-component basis as well as for chart detection.

In addition to extending the capabilities of the proposed system, the syntactical approach for chart recognition will be submitted for publication in a relevant journal, such as the Springer Journal on Document Analysis and Recognition (IJDAR).

Appendix A

Third Party Code

This is a list of third party code used within the project. Since I did not develop this code myself it is included as an appendix and not in the main body of the thesis.

A.1 Tesseract

Tesseract is an open source optical character recognition system (OCR). The OCR is applied on a block by block basis. When a block is classified as text it is handed to the Tesseract engine to read the text. The output is captured and stored as a string. It is freely available online on Google's code archives [1]. More information can be found at [51].

Shortly after the Tesseract engine was included in the project it was realized that most of the text has a resolution which is too small to read. The Tesseract documentation states that the OCR will have very poor results with text which has a vertical height of less than 10 pixels [2].

This feature was implemented as an option to the program. Since it makes the program take a lot longer to run and has low accuracy, it is turned off by default.

A.2 The Gabor Transform

The Gabor transform is a special case of the short-time Fourier transform. It's primary purpose is to measure sinusoidal frequency and phase changes of a local window

[1] http://code.google.com/p/tesseract-ocr/

[2] `http://code.google.com/p/tesseract-ocr/wiki/FAQ#Is_there_a_Minimum_Text_Size?_%28It_won%27t_read_screen_text!%29`

of a signal as it changes in time. The filter is a linear filter which can be used for edge detection. Since the filter is linear it requires an orientation to be provided by the program or user.

The source code for the Gabor function can be found on the github website [3]. The paper by Jain and Sushil [26] describes the Gabor filter as well as how it can be applied for text recognition.

A.3 OpenCV

The open source computer vision library, OpenCV is by far the most used 3rd party code in this project. The OpenCV library is a general purpose library for making computer vision related software. More information as well the source code can be found on the openCV website[4].

[3] `https://github.com/EOL/species-identification-by-uploading-a-photo/blob/master/src/cvgabor.cpp`

[4] http://opencv.org/

Appendix B

Input File

The input file is used to control the program. This file is loaded when the program runs and it contains information required to run the program. This appendix will discuss some of the options and how they are used by the program. One benefit of using this file to run the program is that you can run it several times with the same parameters without having to either repeatedly enter them into the program or hard code them into the program. We present a few of the options below.

The most important entry in the file is the path of the images. This is the only entry which is required to run the program. This points to a folder in which the algorithm will analyze all images in that folder as well as all images in the subfolders.

The next two entries are the whitelist and the blacklist. These entries are lists of charts for the computer to analyze. The program will analyze all reports which are in the whitelist, but none of the reports in the blacklist. This is how reports with a common property are analyzed. For example, all the 2D vertical bar charts are in a single list. This allows the algorithm to only run on the 2D vertical bar charts.

Another input is the ability to skip to a specific report. When analyzing the reports all of the reports are given a number based on the order in which they are analyzed. Using this number it is possible to skip to a specific report.

Appendix C

Published Works

Over the term of the PhD, several paper were published. This appendix lists these papers and gives a brief description of them.

The segmentation approach presented in this approach was published in [52]. More information about this algorithm can be seen in section 5.1.1.

At the beginning of the PhD, some research was performed in the area of comparing two different document images. The images being analyzed were supposed to be the same, however in reality they contained minor differences. A method for classifying the changes in these reports was published in [53].

During the later part of the PhD research two patent applications were started, one for the pie charts and one for the bar and line charts. This process was a combined effort between SAP and UVic. Unfortunately, a few months before the completion of this thesis a key employee at SAP left the company which has put this application on hold.

Glossary

Added (error type) When a detected chart component contains graphics which should not be part of the component. See section 7.6.1.

Axis Chart A chart which contains an axis. This is to distinguish these charts from charts which do not contain an axis (ex. pie charts). The axis charts include bar charts and line charts..

Chomsky Normal Form A grammar is in Chomsky Normal Form if all the substitution rules are of the form $A \to BC$, $A \to d$ or $A \to \varepsilon$. Here A,B,C are nonterminals, d is a terminal, and ε is the empty string. The start symbol cannot exist on the right hand side of any substitution rule. That means B and C cannot be the start symbol. See section 4.1.

Context sensitive grammar A grammar is considered context sensitive if it is not *context free*. See section 4.1.

Context free grammar A language is considered context free if all the rules are of the form $A \to b$ where A is a single nonterminal and b is a string of terminals and nonterminals (including the empty string). See section 4.1.

Derivation A sequence of substitutions using a mathematical grammar. See chapter 4.

Extra (error type) When a detected component is not the chart component. See section 7.6.1.

Formal language A set of strings of symbols constrained by the rules associated to it. See chapter 4.

Grammar A set of substitution rules for strings in a formal language. See chapter 4.

Horizontal Bar Text The vertical index to the left of a horizontal bar..

Horizontal Bar Chart A bar chart in which the bars extend in the horizontal direction and touches the vertical axis..

Incomplete (error type) When a detected chart component is missing part of the component. See section 7.6.1.

Manhatten Layout When the contents of a page are aligned to a grid..

Merge (error type) When two chart components are detected as a single component. See section 7.6.1.

Missing (error type) When a chart component is not detected at all. See section 7.6.1.

Nonterminal A symbol which appears as either an input to a substitution rule or an output from a substitution rule which must be replaced using a substitution rule. In this thesis all nonterminals are represented by uppercase letters. See chapter 4.

Parse tree The mathematical tree formed during a derivation of a string using a context-free grammar. See chapter 4.

Primitive The smallest components which can not be logically broken into smaller components. See section 3.2.

Solid Region A 8-connected region of a similar colour..

Split (error type) When a chart component is detected as two components instead of one. See section 7.6.1.

String (formal language) A series of terminal symbols and nonterminal symbols. See chapter 4.

Substitution rule A rule for substituting a nonterminal in a character string. See chapter 4.

Terminal A symbol which appears as an output from one or more substitution rules which cannot be changed and is not replaced using a substitution rule. In this thesis all terminals are represented by lowercase letters. See chapter 4.

Vertical Bar Text The horizontal index directly below a vertical bar..

Vertical Bar Chart A bar chart in which the bars extend in the vertical direction and touches the horizontal axis..

XZ gridline The gridlines in the XZ plane. See section 3.2.

YZ gridline The gridlines in the YZ plane. See section 3.2.

Bibliography

[1] H.K. Aghajan and T. Kailath. Slide: Subspace-based line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1057–1073, 1994.

[2] S. Ahmed, M. Liwicki, and A Dengel. Extraction of text touching graphics using SURF. In *10th IAPR International Workshop on Document Analysis Systems (DAS).*, pages 349–353, March 2012.

[3] S. Ahmed, M. Weber, M. Liwicki, and A Dengel. Text/graphics segmentation in architectural floor plans. In *International Conference on Document Analysis and Recognition (ICDAR).*, pages 734–738, Sept 2011.

[4] Henry S. Baird. Background structure in document images. In *In Advances in Structural and Syntactic Pattern Recognition*, pages 17–34. World Scientific, 1992.

[5] HenryS. Baird. Document image defect models. In HenryS. Baird, Horst Bunke, and Kazuhiko Yamamoto, editors, *Structured Document Image Analysis*, pages 546–556. Springer Berlin Heidelberg, 1992.

[6] William Brouwer, Saurabh Kataria, Sujatha Das, Prasenjit Mitra, and C Lee Giles. Automatic identification and data extraction from 2-dimensional plots in digital documents. *arXiv preprint arXiv:0809.1802*, 2008.

[7] Nawei Chen and Dorothea Blostein. A survey of document image classification: problem statement, classifier architecture and performance evaluation. *International Journal on Document Analysis and Recognition*, 10:1–16, 2007.

[8] Beibei Cheng, Sameer Antani, R. Joe Stanley, and George R. Thoma. Automatic segmentation of subfigure image panels for multimodal biomedical document retrieval. *Proc. SPIE*, 7874, 2011.

[9] Beibei Cheng, Sameer Antani, R Joe Stanley, and George R Thoma. Graphical image classification combining an evolutionary algorithm and binary particle swarm optimization. *Proceedings of SPIE Electronic Imaging, San Francisco, California*, 2012.

[10] Beibei Cheng, R.J. Stanley, S. Antani, and G.R. Thoma. Graphical figure classification using data fusion for integrating text and image features. In *12th International Conference on Document Analysis and Recognition (ICDAR).*, pages 693–697, Aug 2013.

[11] D. Demner-Fushman, S. Antani, and G.R. Thoma. Automatically finding images for clinical decision support. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 139–144, Oct 2007.

[12] Dina Demner-Fushman, Sameer Antani, Matthew Simpson, and George R. Thoma. Annotation and retrieval of clinically relevant images. *International Journal of Medical Informatics*, 78(12), 2009. Mining of Clinical and Biomedical Text and Data Special Issue.

[13] L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 10(6):910 –918, nov 1988.

[14] Herbert Freeman. Computer processing of line-drawing images. *ACM Comput. Surv.*, 6(1):57–97, March 1974.

[15] B. Gatos, I. Pratikakis, and S.J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39(3):317 – 327, 2006.

[16] R.C Gonzales and Woods RE. *Digital Image Processing.* Prentice Hall, 2nd edition, 2001.

[17] Xavier Hilaire and Karl Tombre. Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:890–904, 2006.

[18] C. J. Hilitch. Linear skeletons from square cupboards. In B. Meltzer and Donald Michie, editors, *Machine Intelligence 4*, page 403. Edinburgh University Press, 1969.

[19] Thai V. Hoang and Salvatore Tabbone. Text extraction from graphical document images using sparse representation. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, pages 143–150, New York, NY, USA, 2010. ACM.

[20] Weihua Huang and Chew Lim Tan. Locating charts from scanned document pages. In *Ninth International Conference on Document Analysis and Recognition.*, volume 1, pages 307 –311, September 2007.

[21] Weihua Huang, Chew-Lim Tan, and Wee Kheng Leow. Elliptic arc vectorization for 3d pie chart recognition. In *International Conference on Image Processing, ICIP.*, volume 5, pages 2889–2892 Vol. 5, 2004.

[22] Weihua Huang, Chew Lim Tan, and Wee Kheng Leow. Model-based chart image recognition. In Josep Llads and Young-Bin Kwon, editors, *Graphics Recognition*, volume 3088 of *Lecture Notes in Computer Science*, pages 87–99. Springer Berlin / Heidelberg, 2004.

[23] Weihua Huang, Chew Lim Tan, and Wee Kheng Leow. Associating text and graphics for scientific chart understanding. *International Conference on Document Analysis and Recognition.*, 0:580–584, 2005.

[24] Weihua Huang, ChewLim Tan, and Jiuzhou Zhao. Generating ground truthed dataset of chart images: Automatic or semi-automatic? In Wenyin Liu, Josep Llads, and Jean-Marc Ogier, editors, *Graphics Recognition. Recent Advances and New Opportunities*, volume 5046 of *Lecture Notes in Computer Science*, pages 266–277. Springer Berlin Heidelberg, 2008.

[25] Weihua Huang, Siqi Zong, and Chew Lim Tan. Chart image classification using multiple-instance learning. In *IEEE Workshop on Applications of Computer Vision.*, page 27, February 2007.

[26] Anil K. Jain and Sushil Bhattacharjee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5(3):169–184, 1992.

[27] Rangachar Kasturi, Lawrence O'Gorman, and Venu Govindaraju. Document image analysis: A primer. *Sadhana*, 27:3–22, 2002.

[28] F. Kboubi, A.H. Chabi, and M.B. Ahmed. Table recognition evaluation and combination methods. In *Eighth International Conference on Document Analysis and Recognition.*, pages 1237 – 1241 Vol. 2, 2005.

[29] Koichi Kise, Akinori Sato, and Motoi Iwata"¨Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370 – 382, 1998.

[30] Ruizhe Liu, Weihua Huang, and Chew Lim Tan. Extraction of vectorized graphical information from scientific chart images. In *Ninth International Conference on Document Analysis and Recognition.*, volume 1, pages 521 –525, September 2007.

[31] Yan Liu, Xiaoqing Lu, Yeyang Qin, Zhi Tang, and Jianbo Xu. Review of chart recognition in document images. *Proc. SPIE*, 8654, 2013.

[32] Xiaonan Lu, J.Z. Wang, P. Mitra, and C.L. Giles. Automatic extraction of data from 2-d plots in documents. In *Ninth International Conference on Document Analysis and Recognition.*, volume 1, pages 188–192, 2007.

[33] Akira Maruoka. Context-free languages. In *Concise Guide to Computation Theory*, pages 81–106. Springer London, 2011.

[34] A Mishchenko and N. Vassilieva. Chart image understanding and numerical data extraction. In *Sixth International Conference on Digital Information Management (ICDIM).*, pages 115–120, Sept 2011.

[35] Ales Mishchenko and Natalia Vassilieva. Model-based chart image classification. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Song Wang, Kim Kyungnam, Bedrich Benes, Kenneth Moreland, Christoph Borst, Stephen DiVerdi, Chiang Yi-Jen, and Jiang Ming, editors, *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, pages 476–485. Springer Berlin Heidelberg, 2011.

[36] Henning Muller, Nicolas Michoux, David Bandon, and Anoine Geissbuhler. A review of content-based image retrieval systems in medical applications-clinical benefits and future directions. *International Journal of Medical Informatics*, 73, 2003.

[37] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10 –22, July 1992.

[38] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1162–1173, 1993.

[39] L. O'Gorman. Primitives chain code. In *International Conference on Acoustics, Speech, and Signal Processing. ICASSP-88.*, pages 792–795 vol.2, April.

[40] Lawrence O'Gorman. kxk thinning. *Computer Vision, Graphics, and Image Processing*, 51(2):195 – 215, 1990.

[41] William Playfair. Commercial and political atlas: Representing, by copper-plate charts, the progress of the commerce, revenues, expenditure, and debts of england, during the whole of the eighteenth century. *London: Corry*, 1786.

[42] William Playfair. *The statistical breviary.* Wallis, 1801.

[43] V.S.N. Prasad, B. Siddiquie, J. Golbeck, and L.S. Davis. Classifying computer generated charts. In *Content-Based Multimedia Indexing, 2007. CBMI '07. International Workshop on*, pages 85 –92, June 2007.

[44] Joseph Priestley. *A Chart of Biography.* Warrington, 1765.

[45] Barry Rafkind, Minsuk Lee, Shih-Fu Chang, and Hong Yu. Exploring text and image features to classify images in bioscience literature. In *Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis*, BioNLP '06, pages 73–80, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[46] Prateek Sarkar. Document image analysis for digital libraries. In *Proceedings of the 2006 international workshop on Research issues in digital libraries*, pages 12:1–12:9, New York, NY, USA, 2007. ACM.

[47] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[48] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. Revision: automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 393–402. ACM, 2011.

[49] Mingyan Shao and RobertP. Futrelle. Recognition and classification of figures in pdf documents. In Wenyin Liu and Josep Llads, editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 231–242. Springer Berlin Heidelberg, 2006.

[50] M. Shilman, P. Liang, and P. Viola. Learning nongenerative grammatical models for document analysis. In *Tenth IEEE International Conference on Computer Vision.*, volume 2, pages 962–969 Vol. 2, 2005.

[51] R. Smith. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition.*, volume 2, pages 629–633, 2007.

[52] Jeremy Svendsen and Alexandra Branzan-Albu. Document segmentation via oblique cuts. *SPIE Proceedings, Document Recognition and Retrieval XX*, 8658, 2013.

[53] Jeremy Svendsen and Alexandra Branzan-Albu. Change clasification in graphics-intensive digital documents. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, September 2015.

[54] Y.Y. Tang, Hong Ma, Jiming Liu, Bing Fa Li, and Dihua Xi. Multiresolution analysis in extraction of reference lines from documents with gray level background. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(8):921 –926, August 1997.

[55] Karl Tombre, Salvatore Tabbone, Loic Pelissier, Bart Lamiroy, and Philippe Dosch. Text/graphics separation revisited. In Daniel Lopresti, Jianying Hu, and Ramanujan Kashi, editors, *Document Analysis Systems V*, volume 2423 of *Lecture Notes in Computer Science*, pages 615–620. Springer Berlin / Heidelberg, 2002.

[56] N. Vassilieva and Y. Fomina. Text detection in chart images. *Pattern Recognition and Image Analysis*, 23(1):139–144, 2013.

[57] T. Watanabe, Qin Luo, and N. Sugie. Layout recognition of multi-kinds of table-form documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 17(4):432 –445, April 1995.

[58] Liu Wenyin and Dov Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications*, 9(5-6):240–250, 1997.

[59] Li Yang, Weihua Huang, and ChewLim Tan. Semi-automatic ground truth generation for chart image recognition. In Horst Bunke and A.Lawrence Spitz, editors, *Document Analysis Systems VII*, volume 3872 of *Lecture Notes in Computer Science*, pages 324–335. Springer Berlin Heidelberg, 2006.

[60] Naoko Yokokura and Toyohide Watanabe. Layout-based approach for extracting constructive elements of bar-charts. In Karl Tombre and Atul Chhabra, editors, *Graphics Recognition Algorithms and Systems*, volume 1389 of *Lecture Notes in Computer Science*, pages 163–174. Springer Berlin / Heidelberg, 1998.

[61] Bo Yuan and Chew Lim Tan. A multi-level component grouping algorithm and its applications. In *Eighth International Conference on Document Analysis and Recognition.*, pages 1178 – 1181 Vol. 2, August-September 2005.

[62] Richard Zanibbi, Dorothea Blostein, and James R. Cordy. A survey of table recognition. *International Journal on Document Analysis and Recognition*, 7:1–16, 2004.

[63] Yefeng Zheng, Changsong Liu, Xiaoqing Ding, and Shiyan Pan. Form frame line detection with directional single-connected chain. In *Sixth International Conference on Document Analysis and Recognition.*, pages 699–703, 2001.

[64] Yan Ping Zhou and Chew Lim Tan. Hough technique for bar charts detection and recognition in document images. In *International Conference on Image Processing.*, volume 2, pages 605 –608 vol.2, September 2000.

[65] Yanping Zhou and Chew Lim Tan. Chart analysis and recognition in document images. In *Sixth International Conference on Document Analysis and Recognition.*, pages 1055 –1058, 2001.

[66] Yanping Zhou and Chew Lim Tan. Learning-based scientific chart recognition. In *4th IAPR International Workshop on Graphics Recognition, GREC2001*, pages 482–492, 2001.

[67] YanPing Zhou and ChewLim Tan. Bar charts recognition using hough based syntactic segmentation. In Michael Anderson, Peter Cheng, and Volker Haarslev,

editors, *Theory and Application of Diagrams*, volume 1889 of *Lecture Notes in Computer Science*, pages 494–497. Springer Berlin Heidelberg, 2000.