SCIENTIFIC CHART IMAGE RECOGNITION AND INTERPRETATION

WEIHUA HUANG

NATIONAL UNIVERSITY OF SINGAPORE

2008

SCIENTIFIC CHART IMAGE RECOGNITION AND INTERPRETATION

BY

WEIHUA HUANG

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

AT

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

NOVEMBER 2008

Name: Weihua Huang

Degree: Doctor of Philosophy

Department: Department of Computer Science

Thesis Title: **Scientific Chart Image Recognition and Interpretation**

Abstract: This dissertation presents the research work on scientific chart image recognition and interpretation, a relatively new area of document image analysis. First of all, we introduce the background and objective of the project. Next we conduct a literature review to summarize previous research activities that are relevant to ours and find out their limitations that are to be overcome. This dissertation then provides a general chart recognition and interpretation paradigm, and investigates all the major aspects of the research problem, including chart image recognition, chart interpretation and its applications, and ground truth dataset generation. Chart image recognition focuses on extracting low-level graphical symbols and text symbols and using model based method or learning based method to achieve classification and construction of chart components. Chart interpretation performs high-level association of textual and graphical information to capture the semantics of chart images and generate descriptions. The result of interpretation can be used by other applications to enhance their performance. This dissertation also investigates two good examples of such application: optical character recognition (OCR) and question answering (QA). The generation of public dataset and ground truth is also an important issue. In this dissertation, we apply both automatic and semi-automatic approaches for generating public dataset with ground truth.

Keywords: Chart Recognition, Chart Interpretation, Model Based Method, Machine Learning, Information Extraction, Ground Truth Generation.

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor, Professor Tan Chew Lim, for his deep insights and dedication to provide continuous guidance and help to me since I was an undergraduate student. With his valuable advice and encouragement, I keep the passion to explore into the document analysis field throughout the past ten years.

I also sincerely appreciated the suggestions and insights received from the following people that help me to polish up my works and complete my thesis: Dr. Huang Zhiyong, currently with the Institute for Infocomm Research (I2R); Dr. Terence Sim, Dr. Low Kok Lim and Dr. Kan Min-Yen, from School of Computing, National University of Singapore.

I also want to express my thanks to the following people for their contributions during their work on final year projects in the Center for Information Mining and Extraction (CHIME): Mr. Liu Ruizhe worked with me on the vectorization techniques; Ms. Zong Siqi helped me to conduct experiments on the learning based chart image classification; Ms. Yang Li helped me to implement the semi-automatic system for extracting ground truth from chart images; Mr. Zhao Jiuzhou also helped me to implement the graphical user interface for the automatic generation of ground truthed chart images.

Last but not least, I must thank my parents and my wife Coco, for their selfish-less support and encouragement through all the years of my study.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Soon after the personal computer became popular in the early 1980's, it was believed that the world is moving towards paperless work environment. However in our society today, paper is still an important medium for exchanging information in literary, scientific and commercial fields. Converting paper-based documents into a computer readable electronic format is a crucial step for broadening the scope of the information source. As far back as 1985, it was stated that about one trillion statistical graphs were printed each year [1]. Many more of such graphs are expected with the proliferation of printed paper documents today. Most statistical graphs appearing in scientific papers are scientific charts or diagrams. While forms or tables are good tools to convey information from structurally arranged data, scientific charts are a very powerful visual tool for representing data, because people understand symbolic graphs better and faster than the corresponding text [2].

In this dissertation, the most accurate definition of a chart we are dealing with states that "a chart is a type of information graphic or graphic organizer that represents tabular numeric data and/or functions" [90]. As the term "chart" can also refer to other meanings, such as music popularity rankings, we use a more specific phrase “scientific chart” here to avoid confusion. “Scientific chart” does not mean a chart is only used for scientific purposes. Rather it is mainly because plotting data or functions is a common practice in most scientific fields. There are also other types of information graphic that use the term “chart”, such as flow chart, organization chart etc. These types of information graphics are not in the scope of this dissertation.

According to Zhou [3], the process of converting a scientific chart image into computer readable form is called *scientific chart recognition*, while *scientific chart interpretation* refers to the process of understanding the semantic meaning of a scientific chart and obtaining the tabular data from it. In the literature, there is little research work and practical results reported on recognizing and interpreting scientific chart images, comparing to those on other types of document images such as forms or tables. Thus it is a relatively young research field to explore into. While early attempts mainly focused on scientific chart recognition and only touched a little on the interpretation part, this research project aims at providing more details and proposing new methods for both recognition and interpretation of scientific chart images. The work reported here has the following significance:

- In their book “Document Image Analysis” [4], O’Gorman and Kasturi stated the ultimate goal of document image analysis: “to recognize the text and graphics components in images and extract the intended information as a human would”. To meet this goal, chart images, as one of the various types of document images that are frequently used, should be made machine readable.

- Recognition and interpretation of chart images fill in the blanks of existing information retrieval systems and document recognition systems. For example, more powerful content-based retrieval of graphics images can be achieved for image-based search engines. More complete content of a scanned document page can also be captured by an OCR system if scientific charts in the document page are converted into a machine readable form.
- Researching in the field of scientific chart image interpretation reveals new problems that were not studied in depth before. For example, traditional document image recognition handles textual and graphical information separately. In fact, most researchers treat text recognition and graphics recognition as two separate problems, as can be seen from the survey paper by George Nagy [5]. However, to achieve chart image interpretation, we are facing the newly discovered problem of associating these two kinds of information at both structural and semantic level, to which no satisfying solution exists yet.

1.2 Challenges

Zhou listed four major challenges in scientific chart recognition [3], including: (i) the great diversity of chart types and styles, (ii) the flexibilities in the structural arrangement, (iii) the difficulties in describing the syntax and semantics of the complex charts and (iv) the difficulty in dealing with degraded, distorted or noisy input. Although Zhou has done substantial work to deal with these challenges, she also pointed out several aspects that need to be further explored, including:

- **Broadening chart types to be recognized and interpreted**. In Zhou's work, most methods were developed specifically for bar charts only, except for the coordinate line detection that works for all 2D and 3D chart types with coordinate lines. Thus it still remains open how a more general chart classification can be constructed to discriminate a wide range of different chart types.
- **Associating information from multiple modalities in the interpretation process.** It is impossible to achieve full interpretation by examining just the information from a single modality, say graphical information alone or textual information alone. Text and graphics must be associated to provide both structural and semantic information. However, as we have already mentioned, there is no existing solution to this problem yet.
- **Enriching the types of text labels in the charts to be handled.** In Zhou's work, only axis labels, axis titles and figure titles are defined for the text blocks in the charts. When more types of labels are included, such as legends, data labels and data values, especially when some chart types do not even have axis lines, the original method for assigning text labels no longer works.
- **Building a publicly available dataset with ground truth.** The existence of a ground truth dataset allows the evaluation and comparison of performance of different

scientific chart recognition and interpretation systems from various ways. However, so far such a dataset does not exist in this relatively new research field, making the evaluation and comparison between different systems difficult.

1.3 Objectives of the Research

This dissertation aims to investigate into the problems in both chart image recognition and chart image interpretation. The investigation and solution proposed leads to a working system. Within the time frame of the present dissertation, the chart images handled by the system are of three most commonly used types only: bar charts, pie charts and line charts. The main objective is to provide a general chart recognition paradigm and to find a solution for each of the major modules in the paradigm, which can be further broken down into the following objectives:

1. Chart recognition and interpretation: To propose a general framework for the recognition and interpretation of scientific chart images. To propose new techniques and evaluate existing techniques to be applied in each module in the framework.
2. Chart segmentation: Investigate the top-down process of segmenting a chart into various components. Identify the key components in scientific charts. Extraction of such components involves bottom-up symbol construction using graphical primitives. Two of the most important issues are investigated: the extraction of a set of coordinate lines and the extraction of data components.

3. Chart classification: Investigate both model-based and learning-based chart classification. Features used for classification include image features, graphical primitives and chart components extracted from the chart images.
4. Text/graphics association and chart interpretation: Investigate the problem of associating text with corresponding graphical symbols to identify the role of text blocks in a chart and to obtain semantic information of a chart image. Chart interpretation is achievable by applying a domain-specific interpretation method on the associated textual and graphical information.
5. Generation of ground truth dataset for public use: Build up a collection of chart images and create its corresponding ground truth data. Both synthetic chart images and real-life chart images are to be included. Formulate the ground truth data so that it can be used for evaluating the performance of chart recognition systems, chart interpretation systems, and other systems such as graphical symbol construction system or text recognition system etc.

1.4 Contributions

We aim to make contributions from three problem domains that we will investigate in this dissertation: chart classification and recognition, chart interpretation and applications, and the generation of ground truth dataset.

In the problem domain of chart classification and recognition, the contributions will be as follows:

1. We will propose a method for extracting line information based on vectorization. The vector information of straight lines, circular arcs and elliptic arcs are extracted and used in following steps. The vectors are used to construct higher level graphical symbols for chart recognition.
2. We will propose a method for identifying the coordinate lines in a chart image. Unlike the traditional approach, the proposed method makes use of both textual and graphical information.
3. We will apply domain knowledge to build chart models for classification of chart images and the recognition of chart components. Comparing to Zhou's work, we enrich the domain knowledge to handle multiple chart types.
4. We will also explore machine learning based method for training the system to automatically classify input chart images, hence this is a data based approach.

In the problem domain of chart interpretation and applications, the contributions are as follows:

1. A more general set of text classes is designated and their text labels are assigned accordingly, comparing to what was defined by Zhou. We will also propose a set of features for learning the relationship between text blocks and graphical symbols in a chart image. By associating the text with graphics, the structural information of the chart image can be re-constructed.
2. We will study the interpretation of different chart types and the extraction of

tabular data. The result of chart interpretation will be stored in both XML format and natural language format.

3. We will explore how chart interpretation is integrated with the existing techniques such as OCR systems and question answering (QA) systems.

In the problem domain of generation of a ground truth dataset, the contributions can be summarized as follows:

1. We will define ground truth data in the context of chart image recognition and interpretation. The defined ground truth has multiple levels of details.
2. We will propose a semi-automatic ground truthing system for extraction ground truth data from existing chart images.
3. We will also propose an automatic system for synthesizing chart images and generating ground truth data at a large scale.

1.5 Outline of the Dissertation

The remaining chapters in this dissertation are organized in the following way:

Chapter 2 surveys related works in scientific chart recognition and interpretation, including parsing and interpretation of electronic charts, and recognition and interpretation of chart images. The limitation of these works is identified as well.

Chapter 3 introduces the terminology used in chart generation, which we adopt in our work. We then revisit design principles and other key issues on chart generation that are useful for designing the general chart recognition and interpretation paradigm.

Chapter 4 focuses on chart image classification and recognition. Recognition of graphics and text are being presented separately. At the beginning of graphics recognition, a vectorization method is proposed to obtain graphical primitives such as straight lines and arcs. Two graphical symbol construction methods are explored: a parsing based method using available domain knowledge, and another graph based method without domain knowledge. The former is used for model based chart classification, while the latter is used for machine learning based classification. Methods for recognition of chart components, namely coordinate lines and data components, are also introduced. For text recognition, a text segmentation method is applied to form text blocks. Then optical character recognition is applied to obtain text information.

Chapter 5 discusses methods for text/graphics association and chart interpretation. Association of text and graphics is achieved through learning based classification of text blocks based on a set of features proposed by us. After these two types of information are combined, chart interpretation is then carried out to recover tabular data in the chart. The result of interpretation is stored in XML format or plain natural language text format.

Chapter 6 discusses practical applications of chart interpretation. Two sample applications are illustrated here. The first one is optical character recognition (OCR) system whose output becomes more complete with the chart information added. The second one is question answering (QA) systems that can handle more variety of questions using the supplement information provided by chart interpretation.

Chapter 7 addresses the issue of ground truth and public datasets in chart image recognition and interpretation. It then presents two systems for generating ground truth chart image dataset. First, a semi-automatic system has been developed to extract multi-level ground truth data from real-life chart images. Second, an automatic system is developed to synthesize large scale chart images with ground truth recorded at the same time.

Chapter 8 concludes the dissertation by summarizing the contributions of this dissertation. It also points out some future directions to be further explored.

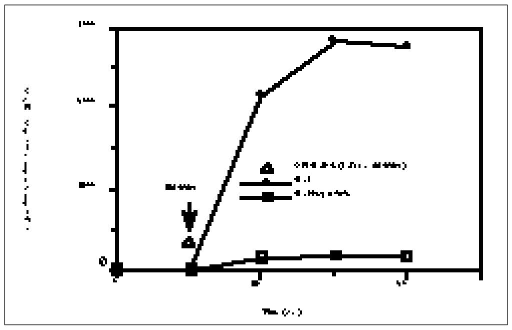
Chapter 2

Literature Review

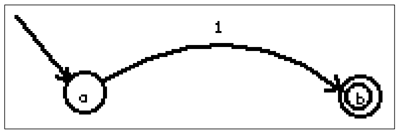
When a chart is generated using graphics software, it is in a structured graphical form, in which the individual chart elements can be accessed and some elements are still modifiable. On the other hand, when a chart is converted into a raster image format, such as BMP or JPG, all the structural information is lost and everything turns into pixels. To differential the two, the former is denoted as “graphic charts” and the latter is denoted as “chart images”. Depending on the nature of the input, previous works related to scientific chart recognition and interpretation can be further divided into graphic chart recognition and chart image recognition. For graphic charts, primitive information such as the text strings and the graphical elements can be extracted from the chart itself. Structural information such as the correspondence between text strings and graphical elements can also be captured through methods such as parsing. Thus the emphasis is on the interpretation of the chart based on these raw pieces of information. On the other hand, the problem of chart image recognition is obviously more challenging and involves more techniques. Image processing techniques are required to process the image and to separate text components from graphical elements. Raster-to-vector conversion is needed to change raster pixels into vector format graphical primitives so that the graphical symbols can be constructed. Text recognition methods are needed to convert text components into electronic form. If the input is a scanned document page, then layout analysis is also needed to locate chart areas in the page. In this chapter, we will review works in both graphic chart interpretation and chart image recognition.

2.1 Graphic Chart Recognition

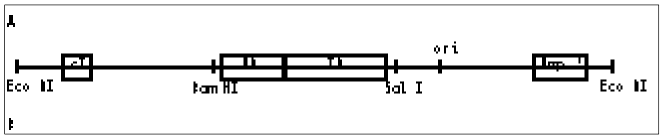
The earliest research work related to graphic chart recognition was done by Futrelle et al. The initial framework was proposed in 1985 [6], and a system named Diagram Understanding System (DUS) was reported subsequently [7-9]. The diagram understanding system developed by them became complete and operational in 1996, and was claimed to be the first working system to fully parse a variety of actual diagrams drawn from the research literature, including x-y plots, linear gene diagrams and finite state machines. Example of such diagrams is given in Figure 2.1.



(a) X-Y plots



(b) Finite state machine



(c) Linear gene diagram

Figure 2.1. Three types of diagrams handled by DUS

The core method used in the system is called context-based constraint grammars that build hierarchical parse tree for each diagram type for classification and description generation. Example of such grammar is shown in Figure 2.2. There are two strengths of the approach. Firstly, the parsing process captures the structural information of the diagram and geometric relationships among the graphical elements. The structured description is more useful than a collection of primitive objects. Secondly, the parse tree built by the system facilitates further automated reasoning about a diagram. The input diagrams are in the vector format rather than raster format. Both the grammars and Graphical User Interface of the system was written in LISP.

```
;;; ****************** < Y-Ticks > ***************
( Y-Ticks -> Ticks Y-Line
             (Y-Line)
(Ticks (touch Y-Line '?) :constraints (> (size Ticks) 2)))

;;; ****************** < X-Line > ***************
( Y-Line -> Line
(:constraints (vertp Line)
(long Line)))

;;; ****************** < Ticks > ***************
( Ticks -> Set(Line)
(:element-constraints (horizp Line)
(short Line)))
```

Figure 2.2. Sample grammar with three rules in the DUS

Futrelle et al. also proposed a scheme for recognizing and classifying vector format graphics in PDF documents [10-11]. The scheme includes three major stages, as shown in Figure 2.3: (1) extraction of the PDF vector entities and converting them to self-contained Java 2D objects; (2) grapheme recognition based on spatial analysis; and (3) figure classification and recognition. In this work, the concept “grapheme” is defined as a combination of graphical primitives based on constraints. The classifier classifies each detected figure in the PDF document into five figure types: data point figure, line figure, bar chart, curve figure and tree/hierarchy figure. The feature vector contains 16 numerical values corresponding to the count of 16 types of graphemes defined in a figure. Futrelle et al. admitted that majority of the figures in the PDF documents are in raster format rather than vector format, but they assumed that the raster format can easily be converted to vector format using vectorization. This may be true for the graphical part of the figure, but obtaining vector format textual information and the overall structural information from a raster format figure requires additional techniques other than vectorization.

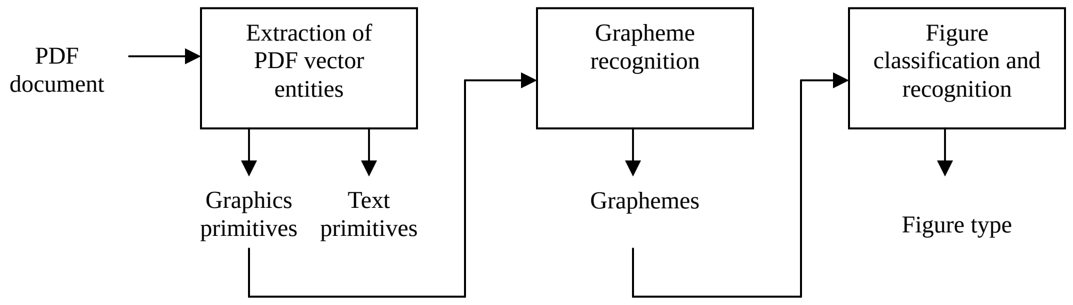


Figure 2.3. Three stages for recognition and classification of figures in PDF documents.

Carberry et al. also studied the problem of understanding information graphics for users who have serious sight impairments. They modeled the problem as a discourse level problem in which knowledge from multiple sources (text, graphics, etc.) are to be obtained and combined [12]. Notably, the term “information graphics” used by them has a broad scope that covers any graphical representation of information, including maps, diagrams, charts, etc. However, the major type of information graphics being studied by Carberry et al. is graphic scientific chart. The probabilistic framework of their system can be seen in Figure 2.4.

The visual extraction module (VEM) in Carberry et al.'s system extracts both textual and graphical primitives and then groups them into meaningful components at the level of abstraction [15]. For textual information, the text primitives are characters and the highest level of meaningful text is the phrase level. For graphical information, the graphical primitives are the line primitives and then they are grouped to form the x-y axis, axis tick marks, data components etc. based on the domain knowledge specified manually. These captured components are then stored in an XML format.

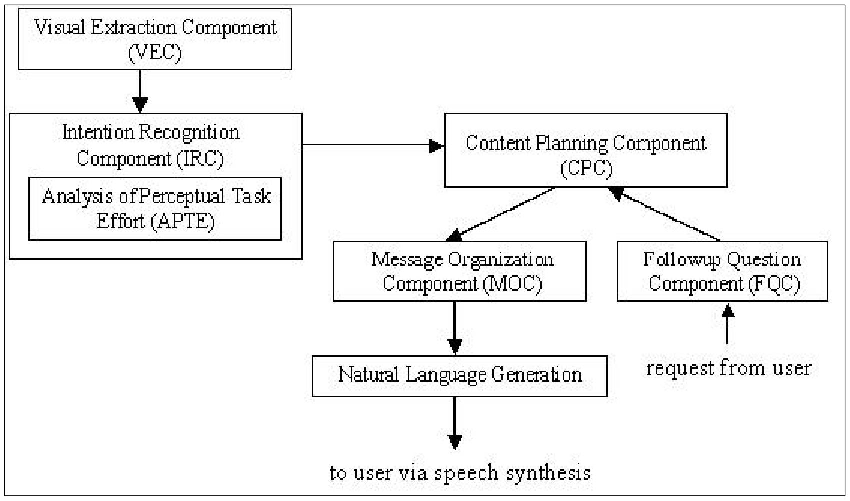

Visual Extraction Component (VEC)
Intention Recognition Component (IRC)
Analysis of Perceptual Task Effort (APTE)
Content Planning Component (CPC)
Message Organization Component (MOC)
Followup Question Component (FQC)
Natural Language Generation
request from user
to user via speech synthesis

Figure 2.4. Main architecture of Carberry et al.'s system

In [13, 14], Carberry et al. explained how the intention recognition component (IRC) in their proposed system recognizes the intended message embedded in an information graphic through a plan inference process which relies on a Bayesian network for hypotheses analysis. The hypothesized message is then summarized and returned to the user. Two sample operators in the system are shown in Figure 2.5. Each operator consists of the following fields:

- *Goal*: the goal that the operator achieves.

- *Data-requirements*: requirements that the data must satisfy in order for the operator to be applicable in a graphic planning paradigm.
- *Display-constraints*: features that constrain how the graphic is eventually constructed if this operator is part of the final plan.
- *Body:* lower-level sub-goals that must be accomplished in order to achieve the overall goal of the operator.

Although the types of information graphics are still limited to several commonly used scientific charts, their work showed great potential of extending traditional language understanding and document summarization to graphical documents.

```
Goal:  Find-value(<viewer>, <g>, <e>, <ds>, <att>, <v>)
Gloss:  Given graphical element <e> in graphic <g>, <viewer> can find the value <v>
in dataset <ds> of attribute <att> for <e>
Data-req:  Dependent-variable(<att>, <ds>)
Body:      1. Perceive-dependent-value(<viewer>, <g>, <att>, <e>, <v>)
```

(a) Operator for achieving a goal perceptually

```
Goal:     Find-value(<viewer>, <g>, <e>, <ds>, <att>, <v>)
Gloss:   Given graphical element <e> in graphic <g>, <viewer> can find the value <v>
in dataset <ds> of attribute <att> for <e>
Data-req:  Natural-quantitative-ordering(<att>)
Display-const:  Ordered-values-on-axis(<g>, <axis>, <att>)
Body:     1. Perceive-info-to-interpolate(<viewer>,<g>,<axis>,<e>,</1>,</2>,<f>)
          2. Interpolate(<viewer>, </1>, </2>, <f>, <v>)
```

(b) Operator that employs both perceptual and cognitive subgoals

Figure 2.5. Sample operators in Carberry's system

2.2 State of the Art in Chart Image Recognition

The most recent systematic work in chart image recognition was done by Zhou, who has made contributions in four aspects of scientific chart image recognition and interpretation.

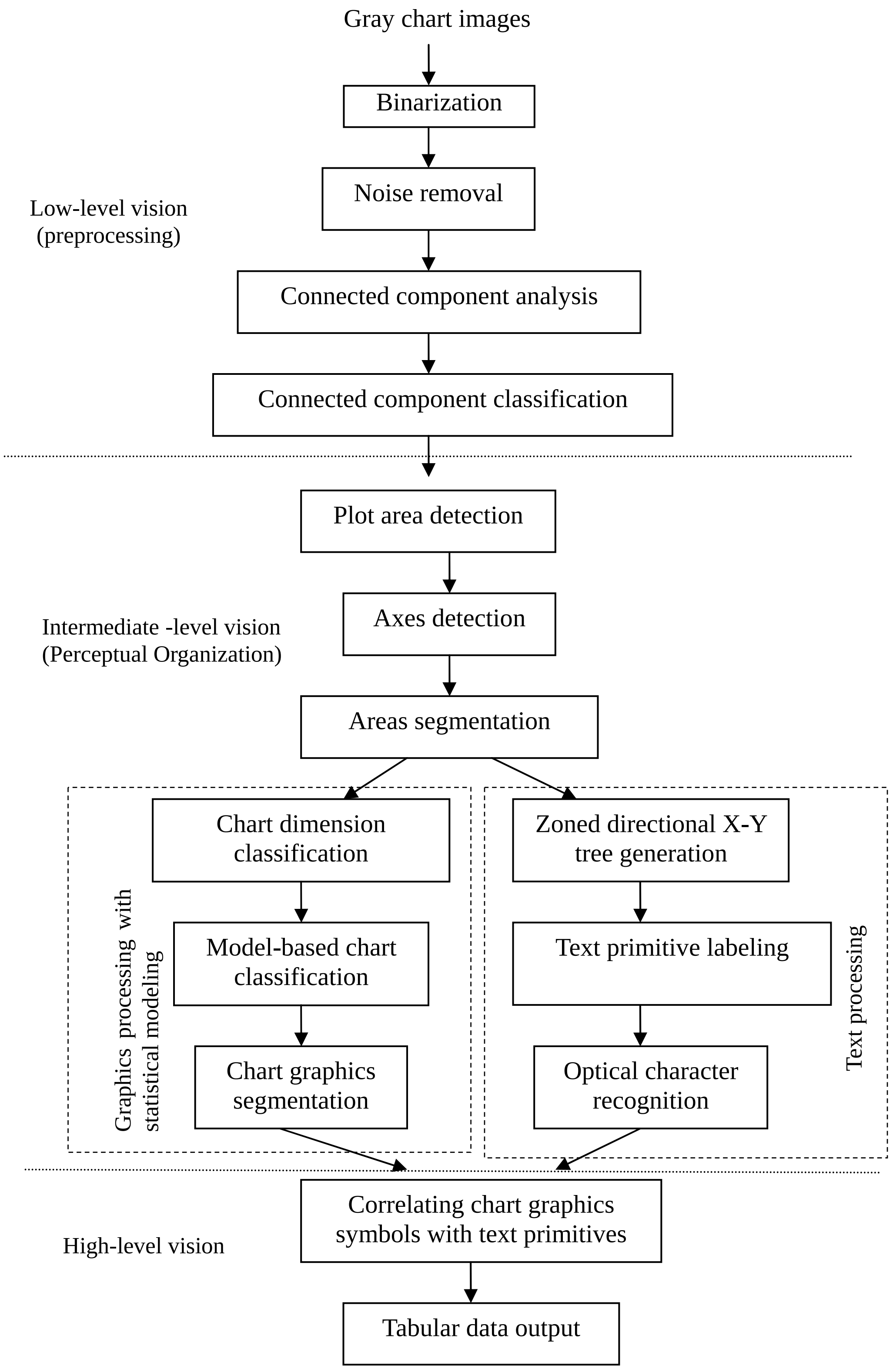

Gray chart images
Binarization
Noise removal
Low-level vision (preprocessing)
Connected component analysis
Connected component classification
Plot area detection
Axes detection
Intermediate -level vision (Perceptual Organization)
Areas segmentation
Chart dimension classification
Model-based chart classification
Chart graphics segmentation
Graphics processing with statistical modeling
Zoned directional X-Y tree generation
Text primitive labeling
Optical character recognition
Text processing
Correlating chart graphics symbols with text primitives
High-level vision
Tabular data output

Figure 2.6. Zhou's scientific chart image recognition framework

Firstly, she gave notation definitions of a scientific chart and investigated the mechanism of human visual perception on chart recognition. Based on these findings, she proposed a hierarchical statistical-model-based chart recognition framework which focuses on the intermediate level of vision [3], as shown in Figure 2.6. From the diagram, we can see that the tasks in the framework are divided into three levels: low-level vision that preprocesses an input image, intermediate-level vision that performs the recognition, and high-level vision that achieves the interpretation.

Secondly, she suggested an improved projection-based approach for plot area detection, comparing to earlier works. A method for coordinate system reconstruction based on Hough feature clustering and geometric analysis was proposed to detect 2-D and 3-D axes in chart images [17, 18].

Her third proposal for a framework of chart classification and segmentation was based on a statistical modeling [16]. The chart models are defined making the use of Hidden Markov Model (HMM). Selected feature points are located in the chart images. They are used for both training and matching the HMM models.

Last but not least, she proposed a zoned directional X-Y tree structure to hierarchically represent the text in graphical documents [3]. Structural analysis is based on the X-Y tree constructed and the text primitives are labeled. Procedures to extract axes tick labels and titles were illustrated.

There were not many other works, publications found in the open literature on scientific chart image recognition and interpretation. In 1997, Yokokura et al. proposed a layout-based network which is a schema-based framework to graphically describe the layout relationship information of bar charts [19]. Vertical and horizontal projection is used for the segmentation of graphical symbol objects and the extraction of the bar chart layout information while constructing graphical and text primitives. Scanned bar chart images were tested and the performance was reported. The performance of coordinate system detection was later compared with that of Zhou's approach [17]. Due to the simplicity of the segmentation method, the types of bar charts that can be recognized are constrained by many assumptions. Furthermore, their work did not really cover other chart types. The most recent work that is relevant to ours was reported by W. Browuer et al. on segregating and extracting information from 2D plots [88]. The proposed method attempts to detect axes features and text features for classification of figure images. Then both graphical symbols, such as the axis ticks and data points, and textual components, such as the axis labels and legends, are detected in the 2D plots identified. The extracted information has been combined for data set generation.

2.3 Limitations of Previous Works

While the related works provided valuable sketch and direction on the problems we intend to solve, however, they do have the following limitations and short comings:

- Some of the proposed methods are designed for a specific chart type only. For example, Zhou's methods work for charts with coordinate lines, Yokokura's work focuses on only the bar chart, and W. Browuer et al.'s method works only for 2D plots. Although works on graphic chart interpretation cover more variety of chart types, the nature of the problem is quite different. Recognition and interpretation of scientific chart images is far more challenging as it requires image processing and re-construction of both syntactic and semantic information. A more general framework for recognition and interpretation of various chart types will make the implemented system more easily expansible, by adding new domain knowledge, or by machine learning techniques. In short, it is our intention to make our system more robust.

- Most works on scientific chart image recognition concentrate only on low-level features to retrieve partial information from the chart images, without constructing high-level components for performing further interpretation of the chart content. In other words, only low-level and intermediate-level vision tasks (as shown in Figure 2.6) were studied intensively, while exploration into high-level vision is still quite limited. However, the methods proposed for graphic chart interpretation give good hints on how to achieve high level vision.
- Most works completely ignore the textual information in the chart images or only make use of the textual information to a very limited extent. Futrelle's work extracts only the x-y axes labels. In Carberry's proposed approach, there is an OCR module for text recognition, but how to obtain the role of the text strings is not even very clearly defined and formulated. Zhou's method of labeling text based on X-Y tree structure assumes Manhattan layout of the text strings in the chart image, which may not be true for chart types other than bar chart. Furthermore, only axis labels and titles were identified using her method.
- Although different researchers have reported their results to some extent, there is no public dataset with the ground truth and the performance evaluation tool available, thus it is difficult to compare the performance among different systems. Futrelle et al.

tested their system using figures extracted from the biological research papers. Carberry et al. tested their system using their own corpus of graphics. Zhou et al. collected a set of scientific chart images which were taken from scanned technical journal pages. The images were scanned using a standard resolution of 300 dpi, with the existence of noise, discontinuity and skew angle. The majority of the images are bar charts and line charts. However, the major problem is that there is no ground truth data available with this collection.

Some of these points were mentioned in Zhou’s work [3] as a suggestion for some future directions. In this dissertation, however, we do aim to overcome these limitations and come up with practical solutions to these yet unsolved problems as well.

Chapter 3

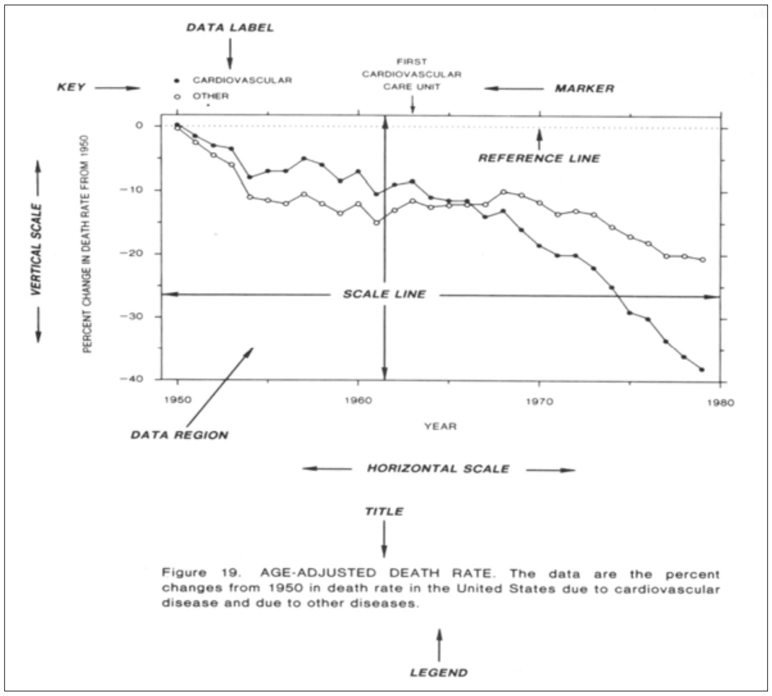
Chart Generation and Chart Recognition

Before we go into some detailed problem formulations and solutions for chart recognition and interpretation, it is important to define first the terminology for the various elements in a chart and to revisit the key issues in generation of charts from data. As chart recognition and interpretation can be viewed as the reverse process of chart generation, the design principles and other relevant studies on chart generation provide helpful guidelines for us to solve problems in chart recognition and interpretation.

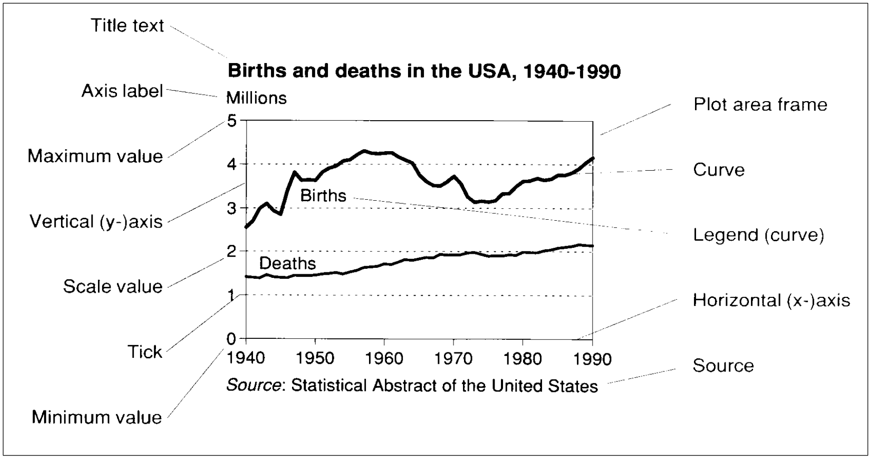
3.1 Terminology

The key elements in a chart have been defined by various researchers in the past. The terminology used for these elements can be different as appeared in different works, and its usage is not consistent even in recent literature. Some of the terms may have different names, although the meaning carried by each term is more or less the same. For example in Figure 3.1, we can see the obvious difference between the terminology used by William S. Cleveland [20] and the terminology used by Anders Wallgren et al. [22]. In our work, we adopt the terminology similar to the one proposed by Anders Wallgren et al. The set of terms are defined in the way that it is applicable to all types of charts and every key element is specifically distinguished from others. The terminology is summarized in

Table 3.1, and it will be used throughout the whole dissertation. Table 3.1 also specifies the type of information for each element, which is either text or graphics. Note that although many terms are related to coordinate lines or the plot area specified by them, the remaining terms are sufficient to be applied to chart types without coordinate lines.



(a) Terminology used by William Cleveland



(b) Terminology used by Anders Wallgren et al.

Figure 3.1 Different terminology used by researchers

Table 3.1 Terminology used in our work

Term	Type	Meaning
Chart title	Text	The title of the whole chart
Plot area	Graphics	The area in which data is plotted
Axis	Graphics	The coordinate lines that defines the plot area
Tick	Graphics	The small ticks along an axis line to specify units
Axis title	Text	The title of an axis line
Axis unit	Text	The unit of an axis line
Axis label	Text	Labels placed along an axis line
Grid line	Graphics	Horizontal or vertical lines placed in the plot area to assist with the measurement of data values
Data component	Graphics	A graphical symbol representing data plotted
Data label	Text	A label assigned to a data component
Data value	Text	The value corresponding to a data component
Legend	Text	The legend distinguishes different data series

3.2 Key Issues in Chart Generation

To make the graphical representation more effective in plotting data and delivering the intended message, there are several key issues to be considered. First of all, the general principles of graphing data need to be enforced in order to guarantee a clear and meaningful plotted chart. Secondly, although there is no conclusion about under which situation a chart type gives best representation, the choice of chart type follows certain convention. Thirdly, there are a number of attributes of the graphical symbols to represent data, that are useful for data recovery. Lastly, the amount of information represented by graphics or text need to be well determined to make the chart both information packing and easy to interpret.

3.2.1 The Principles of Graphing Data

The principles listed by Cleveland [20] were originally meant not only for chart construction but also for construction of general graphic information, such as figures and diagrams. By following these principles, the graphical symbols can be visualized without difficulties and ambiguities, making the graphical information easy to be viewed and understood. According to Cleveland, the important principles include:

- **Clear vision:** the data should stand out and must be represented by prominent graphical symbols. Symbols in the data region should not interfere with each other, and must be visually distinguishable.
- **Clear understanding:** major conclusions should be put into graphical form. Textual description should be comprehensive and informative. Label, scales and data symbols should be consistent with each other.
- **Scales:** the range of the tick marks should include the range of data. The data should fill up as much of the data region as possible. Choose appropriate scales when graphs are compared.
- **General strategy:** Pack large amount of quantitative information into a small region. The process of graphing data is an iterative and experimental process.

3.2.2 The Choice of Chart Types

There are almost an infinite number of different kinds of charts. However, most of them can be traced back to a limited number of basic types [22]. Bar charts are more suitable to show numbers, proportions, frequencies or other ratios. In a bar chart, the variable is qualitative or discrete. A variation of bar chart is the histogram, where the variable to be illustrated is continuous. Pie charts are most appropriate for giving a general picture in situations where we want to compare proportions. Scatter plots are used to show how two variables co-vary (or how they do not co-vary). Line charts are normally used for describing developments of a continuous variable. As the variable is continuous, the different values can be joined by lines. Area charts are also used to show developments over a continuous variable. Maps and flow charts are also categorized as charts, but unlike other charts that plot statistical data, they emphasize more on geographical information and procedural information.

3.2.3 The Data Representation versus Perceptual Judgments

A representation of data refers to the way we visualize data. Cleveland listed ten basic perceptual judgments that a human being performs to visually decode quantitative information encoded on graphs, including:

1. Angle
2. Area
3. Color hue
4. Color saturation
5. Density (Amount of black)
6. Length (Distance)
7. Position along a common scale
8. Position along identical, nonaligned scales
9. Slope

10. Volume

By encoding corresponding characteristics into data representations, in our case the graphical symbols, it is highly possible that the same quantitative information is recoverable from the graphical symbols. For example, data values are represented by the lengths of the bars in a bar chart. Thus by calculating the lengths of the bars, and by obtaining the scale information from the coordinate lines, the original data value can be calculated.

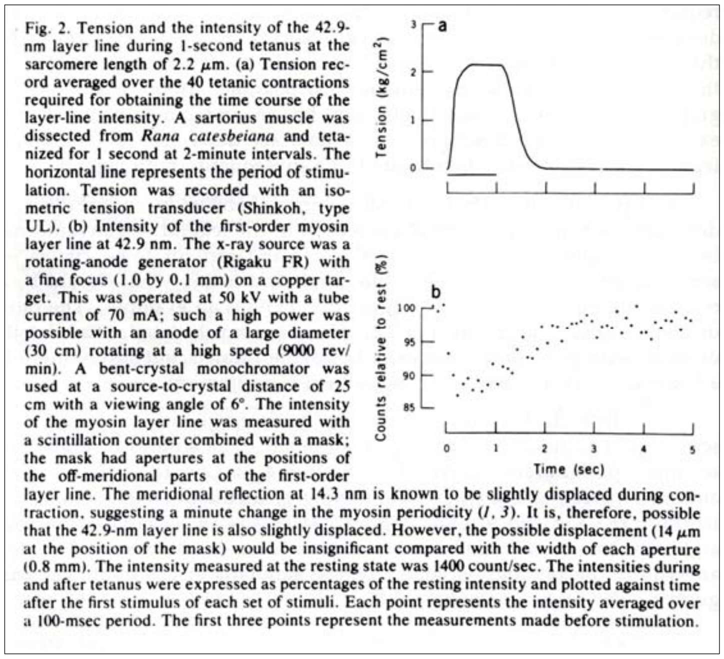


Fig. 2. Tension and the intensity of the 42.9-nm layer line during 1-second tetanus at the sarcomere length of 2.2 μm. (a) Tension record averaged over the 40 tetanic contractions required for obtaining the time course of the layer-line intensity. A sartorius muscle was dissected from *Rana catesbeiana* and tetanized for 1 second at 2-minute intervals. The horizontal line represents the period of stimulation. Tension was recorded with an isometric tension transducer (Shinkoh, type UL). (b) Intensity of the first-order myosin layer line at 42.9 nm. The x-ray source was a rotating-anode generator (Rigaku FR) with a fine focus (1.0 by 0.1 mm) on a copper target. This was operated at 50 kV with a tube current of 70 mA; such a high power was possible with an anode of a large diameter (30 cm) rotating at a high speed (9000 rev/min). A bent-crystal monochromator was used at a source-to-crystal distance of 25 cm with a viewing angle of 6°. The intensity of the myosin layer line was measured with a scintillation counter combined with a mask; the mask had apertures at the positions of the off-meridional parts of the first-order layer line. The meridional reflection at 14.3 nm is known to be slightly displaced during contraction, suggesting a minute change in the myosin periodicity (*1*, *3*). It is, therefore, possible that the 42.9-nm layer line is also slightly displaced. However, the possible displacement (14 μm at the position of the mask) would be insignificant compared with the width of each aperture (0.8 mm). The intensity measured at the resting state was 1400 count/sec. The intensities during and after tetanus were expressed as percentages of the resting intensity and plotted against time after the first stimulus of each set of stimuli. Each point represents the intensity averaged over a 100-msec period. The first three points represent the measurements made before stimulation.

Figure 3.2. Inappropriate ratio of text caption versus graphical content

3.2.4 Textual Content versus Graphical Content

As the chart is a powerful tool for plotting statistical data in a visual way, it is suggested that the major conclusions should be put into graphical form. The text accompanying a chart, typically a figure caption, should be both comprehensive and informative.

Cleveland suggested a framework for generating text that contributes to a clear explanation of a chart, which contains three aspects: describe everything that is graphed; draw attention to the important features of the data; describe the conclusions that are drawn from the data on the graph.

The figure caption, the explanation text and the chart itself combines to provide complete information to a reader. We further suggest that minimum redundancy should be enforced to the information carried by graph and by text, to make the chart more information packing and to avoid extra workload of the reader. This means whatever presented in the graph need not be repeated in the text, unless it is worth being pointed out. A short explanation text may be combined with the figure caption so that all relevant information is kept close to each other. However, if the explanation text is significantly long, then it should be placed in the paragraph before or after the chart, instead of in the figure caption, otherwise it reduces the comprehensiveness of the figure caption. Figure 3.2 shows a negative example where explanation is overdone in the caption.

3.3 The Task of Chart Recognition

By examining the basic principles of graphing data, the different usage of each chart type to plot data and the set of perceptual judgments commonly used for data representation, we can properly design methods to recover both textual and graphical information from a chart that is converted into imaged format and to perform further interpretation to get back the data plotted in the chart. The whole task can be divided into the following sub-goals:

3.3.1 Recognizing the Chart Type

To recognize the type of a given chart, the key point is to extract various graphical elements from the chart. As different charts use different set of graphical elements, most importantly the different data representations, the extraction of graphical elements is crucial to distinguish different chart types. Based on this fact, a model based chart recognition method is the appropriate approach. Each chart type is represented as a model in which common graphical elements appearing in the chart type are specified. The model can be obtained by either manual specification or machine learning. Recognition of chart type is done through model matching. On the other hand, the text is mainly used to annotate individual graphical symbols or provide explanations to the whole chart. Thus textual information does not play a crucial role in chart type recognition.

3.3.2 Recognizing the Chart Components

During the recognition of the type of a given chart, the major graphical symbols are extracted. Thus the chart components represented by these graphical symbols are also obtained at the same time. This is an advantage of the model based approach that the outcome of a step can be applied for multiple purposes, which guarantees the efficiency of the method. However the chart components obtained so far only include graphical information in a chart. Textual information in the same chart still needs to be recognized. To obtain textual information, text segmentation and recognition techniques should be applied to extract text blocks and the actual content of each block. Furthermore, as text plays multiple roles in a chart, the specific role of each text block needs to be identified. This task is done by associating text blocks with graphical symbols.

3.3.3 Recognizing the Data in a Chart

As section 3.2.3 mentioned, data values in a chart are perceptually judged in a number of ways. In other words, a data is represented by a number of attributes of the data components. For a specific chart type, the way to represent data is fixed. Thus one way to specify how data is represented is to directly associate the data representation with each chart type. Once the chart type is identified, corresponding interpretation of data components can be applied to calculate detailed data values. Another possibility is to let the system find it out automatically through machine learning, given the number of representations can be exhaustively enumerated. One key issue here is that to calculate absolute data values, textual information must be correctly associated with graphical information, to provide scale information and index for data etc., otherwise the data obtained can only be relative.

3.3.4 Recognizing the Intended Message Carried by a Chart

After data values are obtained and other textual information is available, a summary of the chart content can be generated. The most direct way of summarization is to generate a description listing the facts obtained such as the type of the chart, the data indices and values, and other textual information. From this, the intended message that the author of the chart wants to deliver to the readers can be estimated. Carberry et al. implemented this step using Bayesian network and hypothesis analysis [13, 14]. The output of their method is a summary of the hypothesized message in natural language form. Natural language form summary of a given chart is important for information extraction applications such as question answering (QA) or web based search etc.

3.4 General Chart Recognition and Interpretation Paradigm

Figure 3.3 summarizes an overall picture of chart recognition and interpretation. It illustrates detailed steps on how each sub-goal discussed in section 3.3 is to be achieved. Comparing with that of Zhou’s framework in Figure 2.6, the proposed paradigm here is much more general in the sense that it places no assumption on the existence of coordinate lines. The coordinate lines are treated as part of the chart components that may or may not exist. The paradigm also shows much more clearly how the textual and graphical information is being processed separately during recognition and associated during the interpretation.

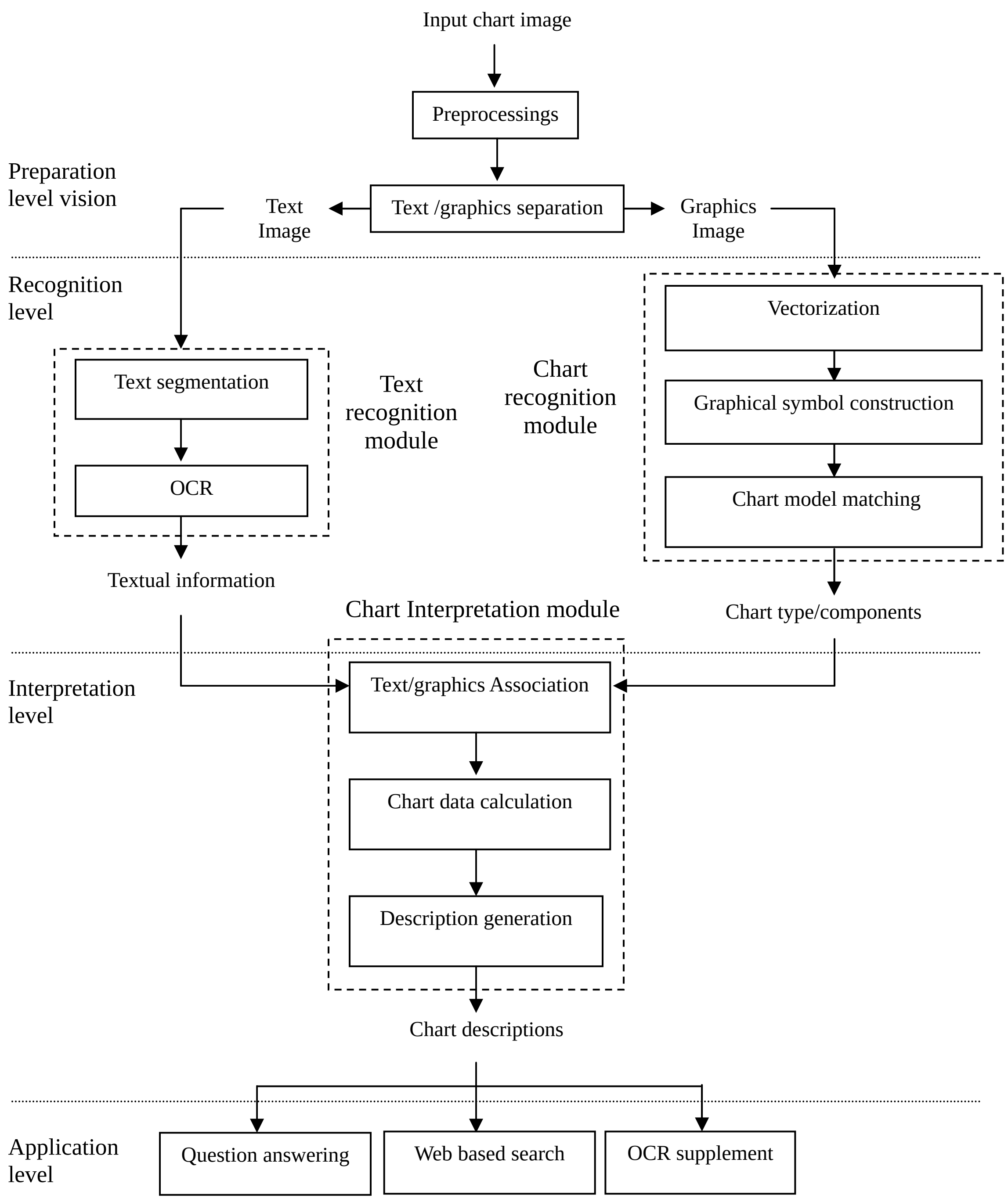


Figure 3.3. General chart recognition and interpretation

Chapter 4

Chart Image Recognition

The task of chart recognition is the extraction of textual and graphical information from a given chart image without further interpretation. The extraction of graphical information further includes the recognition of the chart type as well as the construction of major chart components, namely the coordinate lines and data components. For the graphical information extraction, a model based approach is adopted first, then an alternative approach based on machine learning is explored. Text information extraction is done by segmenting the text into appropriate blocks followed by optical character recognition (OCR). Experiments for the works reported in this chapter were done using both synthetic chart images and real-life chart images downloaded from the internet or scanned from books and papers. At the beginning of the chapter, low level vision tasks such as image preprocessing and separation of text and graphics are also introduced.

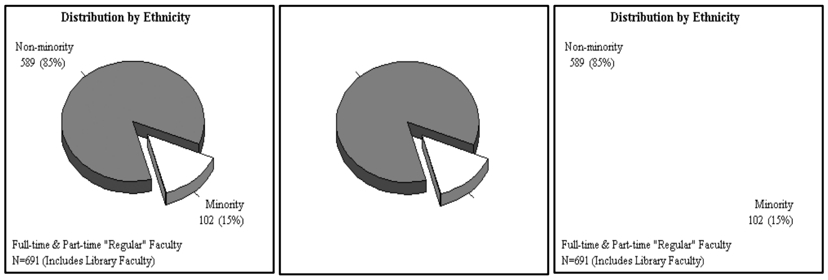
4.1 Low Level Vision Tasks

4.1.1 Image preprocessing

The main task of image preprocessing is to apply color conversion to the input image for future steps. If an image is in RGB color scheme, it is converted to a grayscale image [22] using the following color conversion. I(x, y) is the intensity of a pixel with coordinate of (x, y), R(x, y), G(x, y) and B(x, y) represent the values in RGB channels of the same pixel in the color image. The set of coefficients in formula 4.1 is commonly used due to the fact that human eyes are more sensitive to green channel. A simpler alternative is to assign equal weights to all three channels.

$$I(x, y) = 0.3 \times R(x, y) + 0.59 \times G(x, y) + 0.11 \times B(x, y) \qquad (4.1)$$

Changing RGB triples into grayscale intensities is required by certain steps such as edge detection etc. However, the original image is not overwritten by the new grayscale image. Thus both RGB colors and grayscale intensities are accessible for further processing.



(a) Original image (b) Graphics extracted (c) Text extracted

Figure 4.1. Example of text/graphics separation

4.1.2 Text/Graphics Separation

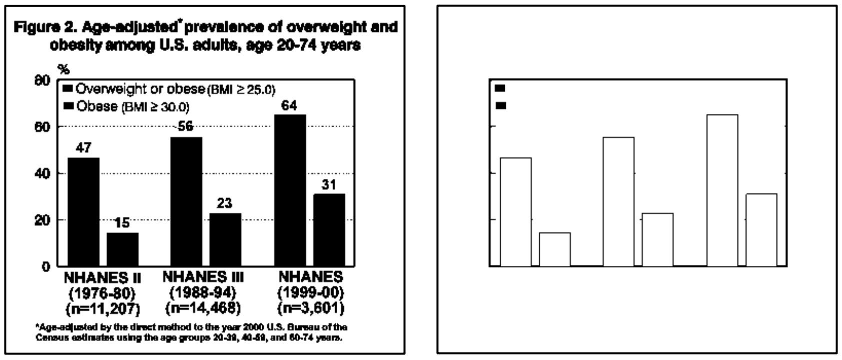
From the grayscale image, connected component analysis is performed within the image area to separate text from graphics [23]. Firstly, connected components are constructed by grouping pixels with similar intensities and are 8-neighbors of each other [22]. A series of filters are applied to classify the connected components into textual components and graphical components. These filters take into consideration the size (number of pixels), the height and width, the height/width ratio and the black pixel density of each connected component constructed. Thresholds that are obtained through training examples are used for classification purpose. In this way, the connected components are classified into three types: graphical component, textual component and noise. Textual components and graphical components are stored separately, as shown in Figure 4.1, to be processed by corresponding recognition units. Connected components that are treated as noise are discarded.

4.2 Graphics Recognition

In this work, graphics recognition is modeled as a bottom-up process that gradually constructs higher level graphical symbols using lower level entities. The most basic entities are the edge pixels. From the edge pixels, graphical primitives such as straight lines and curves are constructed through vectorization, which is a common approach for extracting graphical information from raster format inputs. The graphical primitives in vector form are useful for construction of higher level 2D and 3D shapes using geometric methods. The shapes are then used for more than one purpose: classification of the input image as well as the construction of chart components. Thus the classification of the input image is achieved right after all chart components are constructed, instead of being performed beforehand. In this work, we focus on the most important chart components: the coordinate lines and the data components. Each of the following sub-sections presents details in this bottom-up process.

4.2.1 Edge Detection

An edge map is obtained for the graphical components. This is done by applying edge detection on the image storing graphical components. The Canny edge detector which is a commonly used edge detector is used here. As an input image may contain line drawings that should be preserved, an additional heuristic rule is applied before edge detection to skip such line drawings. The rule specifies the maximum expected line thickness. For each pixel, the number of consecutive pixels with the same intensity in the horizontal or vertical direction is calculated. If this number is smaller than the maximum expected line thickness, the pixel is skipped during edge detection. The edge map is passed to the vectorization step to extract straight lines and curves. An example is shown in Figure 4.2.



(a) Original image (b) Edge map obtained

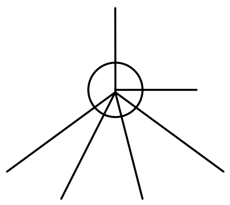
Figure 4.2. Example of edge detection

4.2.2 Vectorization

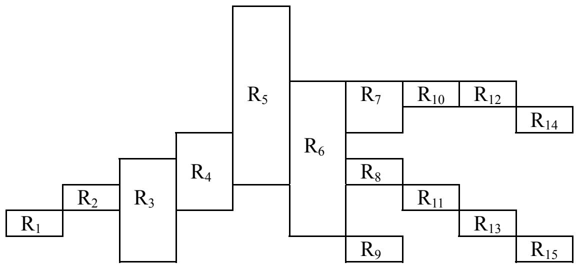
Before graphical symbols are constructed, the task of vectorization is performed to group the edge pixels. Vectorization performs the raster-to-vector conversion and extracts graphical primitives such as straight lines and arcs. Examples of straight line vectorization include [24, 25, 27] and examples of arc vectorization include [25-28]. Noise in the input image and line junctions are the common obstacles to achieving satisfactory vectorization performance, while another major concern is the computational complexity. Based on the nature of the approaches, methods in the literature can be divided into two categories: geometric based approaches and curve fitting. The former applies geometric constraints on pixel segments to construct vectors. The main drawback is that only straight lines or circular arcs are covered, as geometric constraints are hard to be specified for more complex graphical entities such as ellipse. Curve fitting is applicable to all kinds of curves that can be expressed in parametric form. However, the fitting requires a proper set of points and is fragile towards noise. Here we propose a novel vectorization method that combines the idea of the two approaches. The method is based on a data structure called Directional Single-Connected Chain (DSCC) and curve fitting. It constructs straight lines, circular arcs and elliptic arcs from the edge map.

4.2.2.1 The Directional Single-Connected Chain

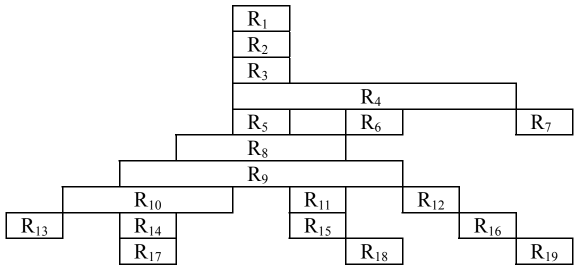
The Directional Single-Connected Chain (DSCC) was originally proposed to extract frame lines in tables [29]. It is easy to construct and computationally efficient. A DSCC is composed by a set of short segments called run-lengths, and it can be horizontal or vertical. A horizontal (vertical) chain is formed by doing a linear regression of the middle points of its run-lengths, and satisfying the constraint that the line has a degree $< (\geq) \pi/4$. A horizontal chain contains vertical run-lengths (Figure 4.3(b)) and a vertical chain contains horizontal run-lengths (Figure 4.3(c)).



(a)



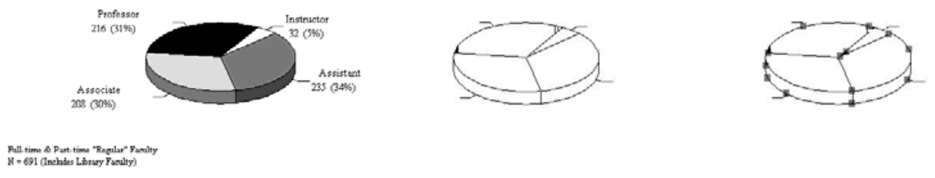
(b)



(c)

Figure 4.3. Example of DSCC. **(a): real lines; (b): vertical run-lengths in the circled area in (a); horizontal chains: C_1={R_1,R_2}; C_2={R_3,R_4}; C_3={R_5,R_6}; C_4= {R_7,R_{10},R_{12},R_{14}}; C_5={R_8,R_{11},R_{13},R_{15}}; C_6= {R_9} (c): horizontal run-lengths in the circled area in (a); vertical chains: C_1={R_1,R_2,R_3}; C_2={R_4}; C_3={R_5}; C_4={R_6}; C_5={R_7}; C_6= {R_8,R_9}; C_7={R_{10}}; C_8={R_{11},R_{15},R_{18}}; C_9= {R_{12},R_{16},R_{19}}; C_{10}={R_{13}}; C_{11}={R_{14},R_{17}}**

A vertical run-length is defined as $R_i\,(x_i, ys_i, ye_i) = \{(x,y) \mid \forall p(x,y) = 1,\, x = x_i,\, y \in [ys_i, ye_i]$ and $p(x_i, ys_i\text{-}1) = p(x_i, ye_i{+}1) = 0\}$, while $p(x,y)$ is the location of a pixel in the image, 1 is black pixel (foreground), 0 is white pixel (background). This run-length starts from (x_i, ys_i) and ends at (x_i, ye_i). In a horizontal chain C_h, (Figure 4.3(b)) every run-length R_i is arranged in a horizontal sequence, and any two run-lengths R_i and R_{i+1} are connected horizontally. Except for the run-lengths at both ends of the chain, R_l and R_r, any R_i has one and only one run-length R_j connected on each side. For the left side of R_l and the right side of R_r, either there is no run-length or there are more than one run-lengths connected. The connection here refers to 8-neighbors connection. Also, the run-lengths in the same chain should have similar length within some range, which is set between half of the average length and twice of the average length. Otherwise, the chain is considered to be broken. The horizontal run-length is defined similarly as the vertical run-length described above. For a horizontal run-length R_i, $R_i(y_i, xs_i, xe_i) = \{(x,y) \mid \forall p(x,y) = 1, y = y_i, x \in [xs_i, xe_i] \text{ and } p(xs_i-1, y_i) = p(xe_i+1, y_i) = 0\}$. This run-length starts from (xs_i, y_i) and ends at (xe_i, y_i). Similarly, a vertical chain C_v is formed by horizontal run lengths (Figure 4.3(c)).


Professor
216 (31%)
Instructor
32 (5%)
Assistant
235 (34%)
Associate
208 (30%)
Full-time & Part-time "Regular" Faculty
N = 691 (Includes Library Faculty)

(a) Original Chart (b) Edge map of the chart (c) Chains and joints detected

Figure 4.4. Example of processing a 3D pie chart

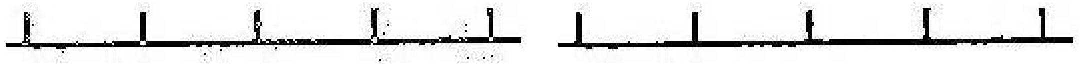
4.2.2.2 DSCC construction and post-processing

DSCCs are constructed based on the above definitions. For vertical chains, the pixels in the image are scanned from top to bottom, while for horizontal chains, the scanning process is from left to right. An example is shown in Figure 4.4. Since the input image may be noisy, there are several post-processing steps performed to refine the resulting chains:

- Filtering: the run-lengths with length 1 and no neighbors are treated as black dots and are removed. Run-lengths with length 1 and have only one neighbor are also removed.

Such run-lengths are treated as protrusions into actual line segments. Tiny chains with the number of run-lengths being one or two are also treated as noise and removed.

- Smoothing: for vertical run-lengths, if two of them, R_i and R_j, are in the same column, i.e., $x_i = x_j$, and the blank area between them are less than 3 pixels, then they are combined to form a new run-length, R_k, where $x_k = x_i$, $y_{sk} = \min(y_{si}, y_{sj})$, and $y_{ek} = \max(y_{ei}, y_{ej})$. Similar process is carried out for horizontal run-lengths. With this step, the holes in the lines are filled. It also prevents broken line segments from appearing. An example of filtering and smoothing is shown in Figure 4.5.
- Splitting: a DSCC may be a straight line, or a curve or even a polyline. To distinguish between a curve and a polyline, we need to record down the turning points along the DSCC. The idea is to use a divide-and-conquer strategy. The starting point and ending point of the DSCC is linked to form a virtual line *L*. A point *p* on the DSCC with the largest distance to *L* is treated as a turning point if the distance is greater than a predefined threshold. The turning point is stored and the same search process is then applied to the two sub-chains of the DSCC on the two sides of the point. In the end, a set of turning points $\{p_1, p_2, \ldots, p_n\}$ is obtained.



a. Chains before refinement b. Chains after refinement

Figure 4.5. Smoothing run-lengths

4.2.2.3 Ellipse-specific fitting theory using least square method

A. Fitzgibbon et al. propose an ellipse-specific fitting [30]. This fitting method always constructs an ellipse from a given point set. In our case, the mid-points of the set of run-lengths stored with a DSCC are used as the point set for the method. By fitting the points to a hypothetical ellipse, and computing the ratio of the maximum radius versus the minimum radius, we are able to tell if a DSCC is a straight line, a circular arc or an elliptic arc. The main idea of their method is to represent an arc as a second order polynomial:

$$F(A; X) = A \cdot X = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{4.2}$$

where $A = [\ a\ b\ c\ d\ e\ f\ ]^T$ and $X = [\ x^2\ xy\ y^2\ x\ y\ 1]^T$. $F(A; X_i)$ is called the "algebraic distance" of a point (x_i, y_i) to the conic $F(A; X) = 0$. Then the fitting of a general conic is done by minimizing the sum of squared algebraic distances:

$$D_A(A) = \sum_{i=1}^{N} F(A; X_i)^2 \tag{4.3}$$

of the curve to the N data points X_i. Enforcing quadratic constraint $4ac - b^2 = 1$ on the parameters helps to avoid the trivial solution to equation 4.3 and degenerate the problem to ellipse fitting. After a series of transformations, the minimization of equation 4.3 can be solved by solving a system of simultaneous equations:

$$\mathrm{SA} = \lambda \mathrm{CA} \tag{4.4}$$

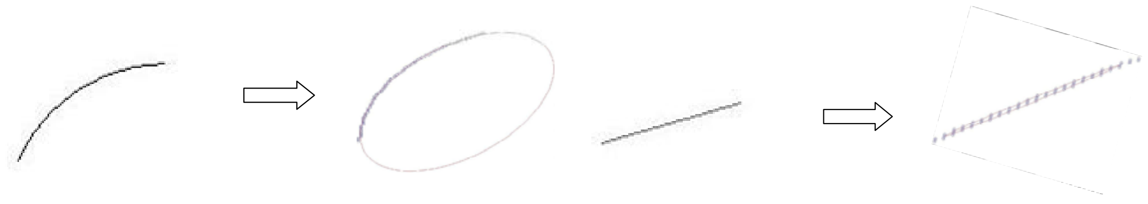
$$\mathrm{A}^T\mathrm{CA} = 1 \tag{4.5}$$

where S is the *scatter* matrix D^TD. $D = [\ x_1\ x_2\ \dots\ x_n\ ]^T$ is called the *design matrix* and C is the matrix that expresses the constraint. λ is the Lagrange multiplier. For the complete derivation of the equations, please refer to [30].

After the solution vector $A = [\ a\ b\ c\ d\ e\ f\ ]^T$ (i.e. all parameters) is obtained, an affine transformation is performed to transform the ellipse from general quadratic form to an standard form:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1 \tag{4.6}$$

where the center is (x_0, y_0) with major radius *max(a, b)* and minor radius *min(a, b)*. This is a basic operation in linear algebra. An example is shown in Figure 4.6(a).



(a) An arc fitted on an ellipse

(b) A straight line segment fitted on a very flat ellipse

Figure 4.6. Example of ellipse fitting

4.2.2.4 Extracting straight lines and circular arcs

Because the curve fitting method always returns an elliptic arc (or a complete ellipse) from the given set of points, we still need to come up with rules to extract straight lines and circular arcs. The basis of the rules is the ratio of the maximum radius versus the minimum radius of the extracted elliptic arc.

- Rule 1: if the ratio is greater than a predefined threshold value, then the original DSCC is treated as a straight line and the fitted elliptic arc is rejected. An example is shown in Figure 4.6(b).
- Rule 2: if the ratio is close to 1 (with a small allowed error range), then the fitted arc is treated as circular arc instead of elliptic arc.

As the DSCC may also be a polyline, one extra step is to calculate the least square error between the fitted arc and the original set of run-lengths stored with the DSCC. If the error is significant, then the DSCC is considered not an arc but a polyline. The turning points obtained in the splitting step illustrated in section 4.2.2.2 are used as the turning points along the polyline.

The last step is to combine straight lines and arcs that were originally broken. Firstly, we define a connected area. That is, if the starting or ending points of two chains have Euclidean distance less than a threshold value, they are considering in the same connected area. To combine two arcs, we define the following rules:

1. The two arcs must be within the connected area.
2. The two tangent lines of the staring or ending points of the two arcs should be angled less than 10 degrees.

The first rule is straightforward. The second rule is implemented by computing the five continuous run-lengths counting from the starting or ending run-lengths and the linear regression on the midpoints of the five continuous run-lengths. The angle of the two lines is then computed.

To combine two straight lines, we also define two similar rules as stated below. If a straight line is broken into several smaller segments with gaps in between, our method is able to re-join the segments to form the original line.

1. The two lines must be within the connected area.
2. The two lines should be angled less than 10 degrees.

4.2.3 Coordinate Line Detection

The first graphical symbol to be identified is the set of coordinate lines in a chart. Coordinate lines are very important because they provide both index information and scale information for plotting data. The existence of the coordinate lines also helps to differentiate chart types. Some chart types such as bar chart or line chart have coordinate lines while others such as pie chart do not have coordinate lines. Both Yokokura et al. and Zhou et al. proposed methods to detect the coordinate lines [17-19].

Yokokura's method is based on the vertical and horizontal projection profiles of the chart image [17]. The left-most peak and the lowest peak in the vertical and horizontal projections are extracted and labeled as Y-axis and X-axis respectively. On the other hand, Zhou et al. use Hough transform to detect long vertical and horizontal lines and performs geometric constraint checking to specify the Y-axis and X-axis [19]. Zhou et al.'s method is more robust towards skew angles and is able to detect coordinate lines in 3D charts. However, both previous methods cannot handle false positives like frames or border lines in input images, such as the one shown in Figure 4.7(b). This is because the methods only consider graphical information and completely ignore textual information, which is also a crucial part of the domain knowledge that we have to deal with.

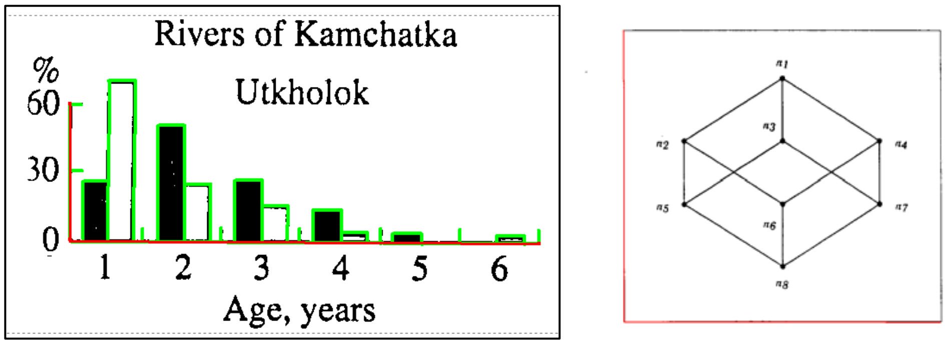
Our method is similar to Zhou et al.'s method reported in [19]. Straight lines are detected using the vectorization method introduced previously, instead of using Hough transform. The following geometric constraints are checked:

- As the Y-axis is most likely placed at the left of a chart image, the preference of vertical lines descends from left to right.
- As the X-axis is most likely placed at the bottom of a chart image, the preference of horizontal lines descends from bottom to top.
- The two candidate coordinate lines must be perpendicular to each other, with a small angle tolerance of T_θ.

Furthermore, extra heuristics are applied to refine the detection of coordinate lines, taking into consideration of the domain knowledge.

- For each pair of candidate lines, there should be small text blocks along the line. For the Y-axis candidate, the text blocks are checked on the left of the line. The text blocks found are sorted by vertical position of the bounding box of each text block. The distance between the center of the top-most text block and the center of the bottom-most text block is compared against the length of the Y-candidate and is stored as *Coverage_Y*. For the X-axis candidate, the text blocks are checked below the line. Similar checking and sorting is performed. The distance between the center of the left-most text block and the center of the right-most text block is compared against the length of the X-candidate and is stored as *Coverage_X*. Both *Coverage_Y* and *Coverage_X* must be higher than a threshold value, otherwise the candidate lines will not be selected as coordinate lines.

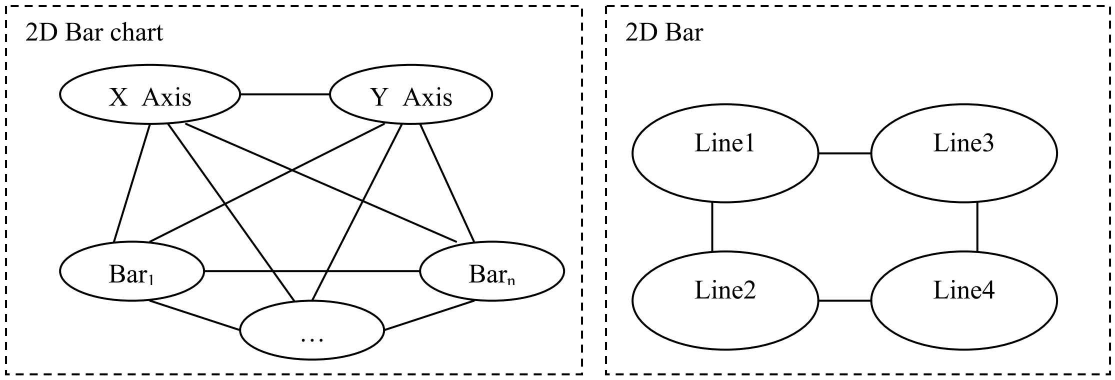
- The area bounded by the candidate coordinate lines must contain most of the graphical objects. A bounding box is formed by the endpoints of the candidate lines. Then the center of each of the remaining graphical components is checked to see whether it is within the bounding box. The number of graphical components within the bounding box divided by the total number of graphical components is denoted as *Total_coverage*. *Total_coverage* must be higher than a threshold value for the two candidate lines to become coordinate lines.



(a) Coordinate lines detected

(b) A false positive returned by existing methods

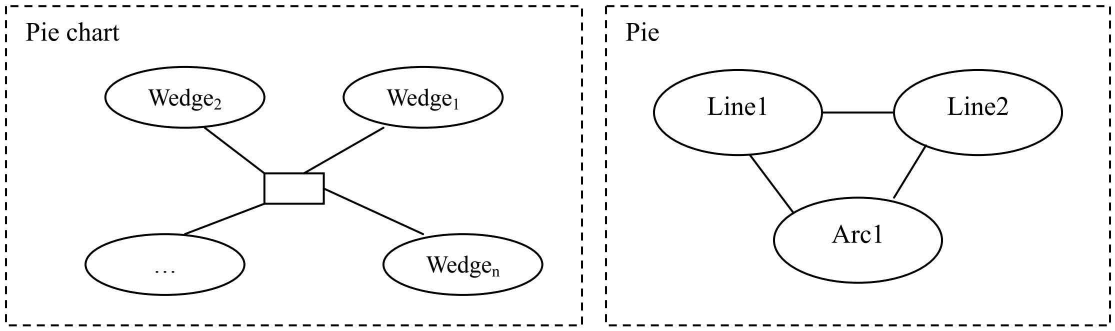
Figure 4.7. Coordinate line detection and false positives.



(a) Chart level specification

(b) Component level specification

Figure 4.8. Domain knowledge for 2D bar chart.



(a) Chart level specification

(b) Component level specification

Figure 4.9. Domain knowledge for pie chart

Figure 4.7(a) shows an example of the coordinate lines detected. Experiments on coordinate line detection will be discussed in section 4.4.

4.2.4 Data Component Recognition

To identify graphical symbols representing data for each chart type, domain knowledge is applied. There are two levels of domain knowledge. At the top level, the domain knowledge specifies the kind of graphical symbols that are expected to appear in each chart type. At the bottom level, the set of primitive graphical primitives and the constraints among them further determines how each graphical symbol is formed. Graphical form of the domain knowledge for 2D bar chart and pie chart are shown in Figure 4.8 and Figure 4.9 respectively.

The domain knowledge can also be specified by a set of parsing rules, as shown below.

- Bar chart:

  $BarChart = \{x\text{-}axis, y\text{-}axis, BarSet\}$, where

  $BarSet = \{Bar\}$, where number of elements ≥ 2 and

  $Bar = \{l_1, l_2, l_3 \mid l_1 \perp l_3, l_2 \perp l_3, l_3 \parallel x\text{-}axis, \mathrm{CE}(l_1, l_3), \mathrm{CE}(l_2, l_3), \mathrm{EL}(l_1, x\text{-}axis), \mathrm{EL}(l_2, x\text{-}axis)\}$

- Pie chart:

  $PieChart = \{Wedge\}$, where number of elements ≥ 2 and

  $Wedge = \{l_1, l_2, a_1 \mid \mathrm{CE}(l_1, l_2), \mathrm{EL}(l_1, a_1), \mathrm{EL}(l_2, a_1)\}$

- Line chart:

LineChart = {*x-axis*, *y-axis*, *Polyline*}, where

Polyline = {l_i | CE(l_i, l_{i+1}), for $i = 1, \ldots, n-1$} where $n \geq 2$

Here, l_i refers to a line and a_i refers to an arc, where $i = 1, 2, 3 \ldots n$. A set of binary constraints between two primitive graphical entities *a* and *b* are defined:

- $a \parallel b$: line a is parallel to line *b*.
- $a \perp b$: line a is perpendicular to *b*.
- CE(*a*, *b*): shape *a* and *b* share one common endpoint.
- EL(*a*, *b*): one end point of shape *a* lies on shape *b*.

There are also global constraints for some chart models. For example, in a bar chart model, all bars must have similar width. The bars in the same data category are filled with the same color or texture. And in a pie chart, the summation of the angles from all wedges must be 2π. All the wedges have similar radius. These global constraints are hard to express using simple symbols thus they are not shown above.

The actual graphical symbol construction process is done through extraction of all possible shapes or polylines specified in the domain knowledge of all chart models available. Based on the vectorized lines and arcs, a graph *G(V, E)* is formed where *V* is the set of intersection points among the lines and arcs, and *E* is the set of segments (straight line segment or arc segment) between intersection points. Considering all primitive graphical entities and their intersections, construction of graphic symbols is done using a constrained graph search on *G*. Both binary constraints and global constraints are checked during the search process.

All constructed target graphical symbols, together with the coordinate lines detected in the previous section, are checked against every chart model available in the domain knowledge to calculate the similarities between a given chart image and the known chart types. The similarity is calculated as below:

$$Similarity_{ij} = \max(0, \sum_{k=0}^{n-1} W_{ik} C_{ijk} - \sum_{l=0}^{m-1} W_{jl} O_{jl}) \tag{4.7}$$

where W_{ik} is the weight of k_{th} component in the chart model i, W_{jl} is the weight of l_{th} unexpected component in the image j, thus $\Sigma W_{ik} = 1$ and $\Sigma W_{jl} = 1$. C_{ijk} is 1 if the k_{th} chart component from model i exists in the image j, and it is 0 otherwise. O_{jl} indicates the existence of the l_{th} component that is unexpected to appear in model i. The weights can be predefined or calculated based on a collecting of training samples.

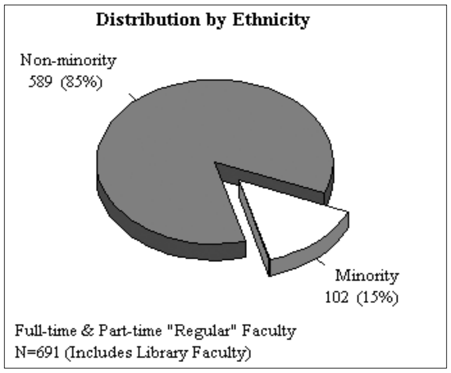
The chart type returning the highest similarity is deemed to be the type of the input infographic image. If all the similarity values are smaller than a cut-off value, then the given image is not treated as a chart. Thus formula 4.7 can also be used for differentiate charts from other types of figures. Once the chart type is determined, all relevant chart components are stored and passed to the interpretation process.

4.2.5 Data Component Recognition and Chart Classification through Machine Learning

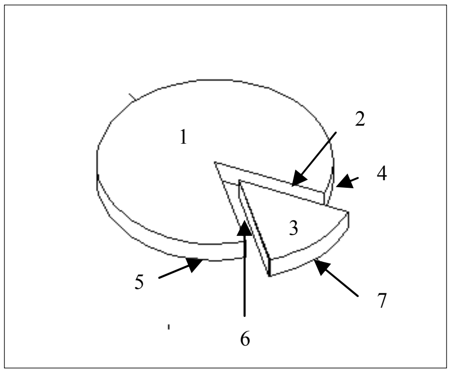
The main drawback of the model based approach is that the domain knowledge is predefined and the types of chart that the system handles are also predefined. If a new type of chart is encountered, both the graphical symbols and constraints in the domain knowledge need to be manually expanded to handle the new chart type. This reduces the integrity and expandability of the system. Here we explore the use of machine learning techniques to achieve both data component recognition and chart classification.

Traditional unsupervised or supervised learning methods require a class label to be assigned to each feature vector. That is to say, the training examples are represented by ($y_i, x_{i1}, x_{i2}, ..., x_{in}$) tuples, where i is the index of examples: $i = 1, 2, 3, ...$ and there are n features, each feature vector ($x_{i1}, x_{i2}, ..., x_{in}$) is tagged with the correct label yi. If we adopt this representation, then we have two choices. The first choice is to perform classification of the chart types based on features extracted from the whole image. And the second choice is to extract features from components in the chart for classification purpose. The first choice is the same as previous works, where the features used for classification are global features that cannot be used for future stages. For the second choice, a crucial requirement is that the image can be properly decomposed into components. Fortunately charts are line drawings that can be decomposed. Thus now the problem becomes how to classify chart images that contain a variable number of components which are represented by feature vectors.

The answer is to use multiple instance learning, or sometimes called semi-supervised learning. Please refer to Appendix A for detailed background information and formulations. The motivation is based on the observation that in a chart type, data always have a homogeneous representation and the representation is often shape-based graphical symbols. Furthermore, different chart types represent data using different shape (or combination of shapes). Based on the correlations among shapes, data representations and chart types, the chart classification problem can be modeled as a multiple-instance learning problem. The result of learning helps to answer two questions: which shape (or combination of shapes) represents data in a chart? How closely is a shape (or combination of shapes) correlated to the chart type? The answer to the first question can be used for further interpretation of the chart content and the answer to the second question is for chart image classification.



(a) The original image



(b) Basic shapes obtained.

Figure 4.10. Example of shape construction from an input image. There are seven shapes in (b) labeled with numbers

The preprocessing steps and vectorization step remain the same for the learning based approach. The major difference is that now we cannot assume the graphical symbols to be known beforehand. Thus all possible graphical symbols are enumerated in this case. Based on the vectorized lines and arcs, a graph G(V, E) is formed where V is the set of intersection points among the lines and arcs, and E is the set of segments (either straight line segment or arc segment) between intersection points. Shape construction is a process of finding the Minimum Cycle Basis (MCB) [86] on the graph G, and an efficient algorithm was proposed by Ferreira et al. [87]. However the original algorithm only finds polygons from a set of straight lines, thus some of the steps are modified to take care of arc segments. Figure 4.10 shows an example of graph constructed from a 3D pie chart, from which 7 shapes are constructed. Each number in the figure represents one shape.

Now a collection of shapes are obtained. The edges in a shape are classified into three types: (1) straight line, (2) circular arc or (3) elliptic arc. Although these three types of edges are not sufficient for general shapes, they can cover all the edges extracted from the chart images we examined since shapes in chart images are relatively more regular. Four shape descriptors are used to form the feature vector for each shape constructed: number of edges n_i for each edge type i; order o among the edges (represented as a sequence of edges); number of parallel edge pairs n_p; number of symmetric axes n_s. Thus a feature vector can be represented as $<n_1, n_2, n_3, o, n_p, n_s>$. We choose these four shape descriptors because they are all invariant to translation, rotation and scaling, and a combination of them can uniquely define a shape class.

To train a chart type A, a set of positive bags $B^+ = \{B_j^+, j = 1,2,...,n\}$ and a set of negative bags $B^- = \{B_k^-, k = 1,2,...,m\}$ are provided by the user. Each bag is an image containing shapes. The first step is to find out the universal set of components (shapes) $C = \{C_i, i = 1, 2, ..., l\}$ where $C = B^+ \cup B^-$. Then the correlation factor $CF(C_i, A)$ between each C_i and a chart type A is derived from the conditional probability $P(C_i \mid B^+, B^-)$. Assuming that the training examples are conditionally independent given the chart type A, and by applying Bayes' rule (assuming an uninformative prior over the shape C_i), we can get:

$$P(C_i \mid B^+, B^-) = \prod_{j=1}^{n} P(C_i \mid B_j^+) \prod_{k=1}^{m} P(C_i \mid B_k^-) / P(C_i)^{n+m-1} \tag{4.8}$$

where

$$P(C_i \mid B_j^+) = 1 - \exp(-Num(Ci, B_j^+)) \tag{4.9}$$

$$P(C_i \mid B_k^-) = \exp(-Num(Ci, B_k^-)) \tag{4.10}$$

As $P(C_i)$ is independent of the training examples, we take it out from the expression in (4.8) to get:

$$CF(C_i, A) = \prod_{j=1}^{n} P(C_i \mid B_j^+) \prod_{k=1}^{m} P(C_i \mid B_k^-) \tag{4.11}$$

The derivation of formula (4.8) can be found in [83]. Both $P(C_i \mid B_j^+)$ and $P(C_i \mid B_k^-)$ are exponential functions that depend on the number of shapes matching the component C_i. $Num(C_i, B_j^+)$ calculates the number of shapes that match component C_i in the positive example B^+_j. As $Num(C_i, B^+_j)$ increases, $P(C_i \mid B^+_j)$ increases and is approaching 1. $Num(C_i, B_k^-)$ calculates the number of shapes that match component C_i in the negative example B_k^-. As $Num(C_i, B_k^-)$ increases, $P(C_i \mid B_k^-)$ decreases and is approaching 0. Exact matching between feature vectors is required.

For a new image that is also treated as a bag containing a number of components C' = $\{C_g, g = 1, 2, ..., h\}$, we calculate the similarity between C' and type A as:

$$Sim(A, C') = \sum_{g=1}^{h} Num(C_g, C')CF(C_g, A) \tag{4.12}$$

where $Num(C_g, C')$ counts the number of occurrences of component C_g in the given image C'. If $C_g \in C$, then $CF(C_g, A)$ is pre-computed during the training process, otherwise $CF(C_g, A) = 0$. Similarity between the new image and every existing chart type is calculated, and the chart type that results in the highest similarity value is deemed to be the type of the new image. It is also possible that the new image belongs to a new chart type that was not presented during the training process. Thus a manually set cut-off value can be set to judge whether the new images belongs to any of the existing types.

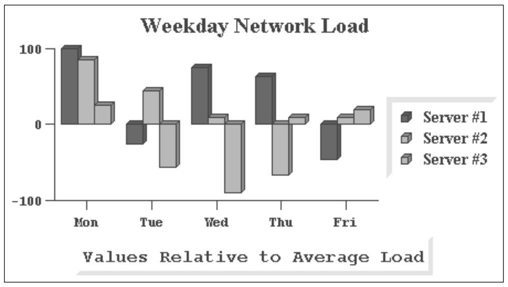


Figure 4.11. A 3D bar chart

In some cases, the data components in a chart are represented using more complex graphical symbols that are combinations of shapes. This happens very often for 3D chart types. Figure 4.11 shows an example of 3D bar chart in which the data components are represented as cuboids that consist of 1 rectangle and 2 parallelograms. Based on the observation of homogenous representation of data in charts, the shapes that form a data component should have a high degree of co-occurrence in the set of shapes recognized. Another heuristic is that the shapes forming a single symbol are not separated, which means they are neighbors of each other. Thus an extra step to detect a complex symbol (data component) is to identify those shapes that are neighbors of each other and have high degree of co-occurrence. Two shapes are neighbors if they share a common edge or part of an edge.

To find out the probability of a combination of shapes being a complex symbol, we can compute the degree of neighborhood *DoN* between two shape types T_1 and T_2. If the number of T_1 shapes in a given image is N_1 and the number of T_2 shapes in the same image is N_2, we can find out the number of T_1-T_2 neighboring pairs $N_{neighbor}$. Then *DoN* is calculated as:

$$DoN = \left(\frac{N_{neighbor}}{N_1} + \frac{N_{neighbor}}{N_2} \right) / 2 \tag{4.13}$$

Note that the value of *DoN* falls in interval [0, 1]. When none of the T_1 shapes is neighbor of T_2 shapes, $N_{neighbor}$ becomes 0 and thus *DoN* becomes 0. In this situation, the two types of shape never appear together as neighbors. When all T_1 shapes are neighbors of T_2 shapes, and $N_1 = N_2$, *DoN* reaches its maximum value of 1. In this situation, the two types of shape always appear together as neighbor of each other. Thus we can see that the higher the *DoN* is, the more possible that the two types of shapes form a complex symbol.

4.3 Text Recognition

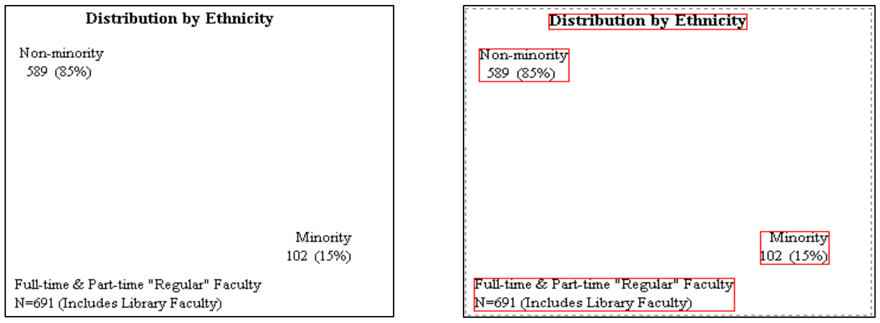
To obtain textual information in a chart image, two steps are carried out. First of all, text components are grouped to form text blocks. Secondly, the electronic content of each text block is recognized using optical character recognition. Thus the output of text recognition consists of two types of information: zoning information for each text block and electronic content of each text block.

4.3.1 Text Grouping

Text components need to be grouped properly to form logical blocks. Each text block contains a sentence, an alphabetic phrase or a numerical string. The method proposed by Yuan et al. [31] is used to perform this task. The grouping function is defined as:

$$f(s_1, s_2) = \sqrt{\frac{ks_1 s_2}{s_1 + s_2}} \tag{4.14}$$

where s_1 and s_2 are the sizes of the two components and *k* is an adjustable parameter that is used to determine the grouping level. The size of a component is defined as the number of black pixels belonging to the component. If the calculated *f* is smaller than the distance between the two components, the components are considered belonging to the same text block. *k* is the only parameter to be manually specified here. A large k results in large text blocks while a small k results in more isolated blocks. In our case, the value of *k* is set to 10. An example is given in Figure 4.12, where the text blocks formed are indicated by bounding boxes.


Distribution by Ethnicity
Non-minority
589 (85%)
Minority
102 (15%)
Full-time & Part-time "Regular" Faculty
N=691 (Includes Library Faculty)

(a) Text components (b) Text blocks

Figure 4.12. Example of text grouping

An advantage of this method is that the calculation is based on characteristics of connected components, which are already obtained during the text/graphics separation step. This guarantees the efficiency of the method. Furthermore, the method is rotation invariant, as the size of the connected components and Euclidean distance between them are both rotational invariant. Even if the input image is skewed, or even rotated by a large degree, the method still returns the same result. On the other hand, the sparsely distributed text in charts and the possible existence of skew angle make it unsuitable to use the traditional methods mentioned in [89], including projection profile based methods, texture based methods, or other methods that assume existence of text lines.

4.3.2 Optical Character Recognition

Optical character recognition (OCR) is applied to each text block to recognize the electronic content. To achieve this, the OCR modules provided in the *Scansoft Omnipage Capture SDK* package are used. The OCR modules generally work well. However, the error rate of the character recognition process heavily depends on the quality of the input image. Since the OCR accuracy is not the main focus of our work here, such errors are manually corrected.

4.4 Experiments and Discussions

4.4.1 Test Data Set

200 chart images were collected to form a dataset for training and testing the graphics recognition module in the system. Most of the images are scanned black-and-white images, while the rest are color images downloaded from the web. Out of these 200 imaged infographics, there are 80 2D bar charts, 60 2D line charts, and 60 pie charts that are 2D or 3D. Beside the images, multi-leveled ground truth information is also available for performance evaluation, including vector level information of straight lines, circular and elliptic arcs as well as semantic level information such as graphical symbols and text blocks etc. The extraction of such ground truth information will be presented in Chapter 7.

4.4.2 Experiment for Vectorization

The DSCC-based vectorization function was applied to the images in the dataset to extract straight line segments, circular arcs and elliptic arcs from them. For straight line segments, the attributes stored are the starting point, the ending point and the thickness of the line. For circular arcs, the attributes stored are the starting point, the ending point, the center point, the radius of the circle and the thickness of the arc. For elliptic arcs, the attributes stored are the starting point, the ending point, the center of the ellipse, the maximum and minimum radii of the ellipse and the thickness of the arc.

Table 4.1. Performance of vectorization

Category		Correct %	Broken %	Incorrect %
Bar chart	Straight line	84.72	6.94	8.34
	Arc	-	-	-
Pie chart	Straight line	83.45	15.11	1.44
	Arc	82	13.72	4.28
Line chart	Straight line	93.06	3.57	3.37
	Arc	-	-	-

To evaluate the vector information obtained, we compare the extracted vectors with the vectors provided in the ground truth data. For comparison purpose, the overlapping segment s is calculated between an extracted vector v_d and the corresponding vector in the ground truth v_g [16]. *Coverage(s, v_i)* is calculated as the length of s divided by the length of v_i, where v_i is either v_d or v_g. If both *Coverage(s, v_d)* and *Coverage(s, v_g)* are ≥ 90%, then the extracted vector is deemed correct. If *Coverage(s, v_d)* is greater than 90% but *Coverage(s, v_g)* is not, then v_d is treated as a broken subpart of v_g. If both *Coverage(s, v_d)* and *Coverage(s, v_g)* are below 90%, then v_d is considered to be incorrect. The results are summarized in Table 4.1.

From the table, we can see that the DSCC based vectorization works reasonably well. The relatively higher percentage of wrong segments for bar chart type is due to the existence of some scanned bar chart images that are noisier than others. For pie charts, more broken vectors occur due to the fact that arcs are more prone to being broken into small pieces than straight lines during DSCC construction. To reduce the number of broken vectors, more lenient rules for combination can be specified.

4.4.3 Experiment for Coordinate Line Detection

To evaluate the performance of our coordinate line detection method, we adopt the same metrics used by Zhou et al. [3]. Precision is defined as the proportion of detected axes that is actually correct. Recall measures the proportion of correctly detected axes over existing axes in the chart images. For comparison purpose, the same set of testing data in [3] was used. As all the images in the original dataset have coordinate lines, 100 more images that are pie charts and other types of line drawings that do not have coordinate lines were added to measure the methods' ability to handle false positives. 34 of the new images have rectangular frames. The newest version of Zhou's of method stated in [3] and our own method were both applied to the testing images. Then precision and recall were calculated respectively. Table 4.2 summarized the precision and recall of our method (denoted as "proposed") and Zhou's method (denoted as "Zhou").

Table 4.2. Testing results of coordinate line detection methods

Category / Method	Detected axes		Correct axes		Precision (%)		Recall (%)	
	X-axis	Y-axis	X-axis	Y-axis	X-axis	Y-axis	X-axis	Y-axis
Zhou	496	501	462	467	93.14	93.21	93.71	94.73
Proposed	478	478	473	469	98.95	98.12	95.95	95.13

From the table, we can see that the recall of our method is slightly better than that of Zhou's method. This is because both methods perform similar geometric analysis on straight lines in the images. The difference is that Zhou et al. used Hough transform to obtain line information while we use vectorization to get lines in vector format. The DSCC based vectorization method is able to join line segments with small gaps in between, while Zhou's method does not perform this task. The precision of our method is about 5% higher than that of Zhou's method. This is expected because the 34 new images added to the testing data cause more false positives for Zhou's method. On the other hand, our method perfectly avoids such false positives by taking into consideration of text blocks during candidate selection.

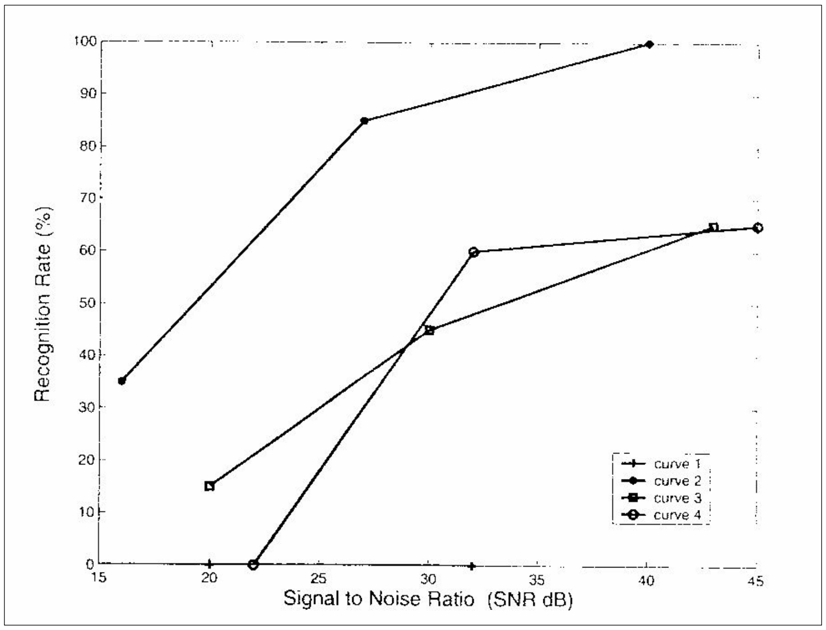


Figure 4.13. Example of broken axis lines

There are still some erroneous results given by our method. There are two causes. Firstly, some coordinate lines are not properly labeled. There are a few coordinate lines without any text label. This violates the design principles summarized in Chapter 3. Secondly, the quality of some chart images is very poor, causing a coordinate line broken into small line segments with very large gaps in between. The vectorization method fails to detect the complete line. This is a case of error propagation. One example is shown in Figure 4.13. It is noted that other straight line detection methods such as the Hough Transform also have problem dealing with broken lines.

4.4.4 Experiment for Chart Type Recognition

The performance of chart type recognition was evaluated by the result of the model matching. During model matching, all graphical symbols constructed were compared against those in the domain knowledge for each chart type and the similarity between the given image and every known chart type was calculated. The chart type returning the highest similarity value was deemed to be the type of the given image. Each of the 200 chart images was processed by the system and was assigned a type. The number of images whose type was correctly recognized is shown in Table 4.3. Most of the errors are due to the failure to recognize certain graphical symbols (such as the coordinate lines) or the failure to satisfy certain global constraints (such as the summation of angles equals to 2π).

Table 4.3. Performance of chart type recognition

Chart type	Number of images	Correctly Recognized	Accuracy (%)
2D bar	80	75	93.75
2D pie	48	44	89.58
3D pie	12	10	83.33
Line	60	51	85.00
Overall	200	180	90.00

From the table, we can see that the accuracy of chart type recognition is correlated to the regularity of each chart type. The method works the best for 2D bar charts because such charts only contain straight lines and data components are regular rectangles. On the other hand, the recognition accuracy of 3D pie chart is lower than that of the 2D pie charts in the table, due to the fact that circular arcs are more accurately constructed than elliptic arcs because the former are symmetric and requires fewer parameters to specify while the latter are non-symmetric and requires more parameters to specify. The accuracy of line chart type recognition is lower than that of the 2D bar charts and 2D pie charts. This is because line charts do not have unique data components. The existence of grid lines sometimes fools the system to mistakenly treat a line chart as a bar chart, such as the one in Figure 4.14.

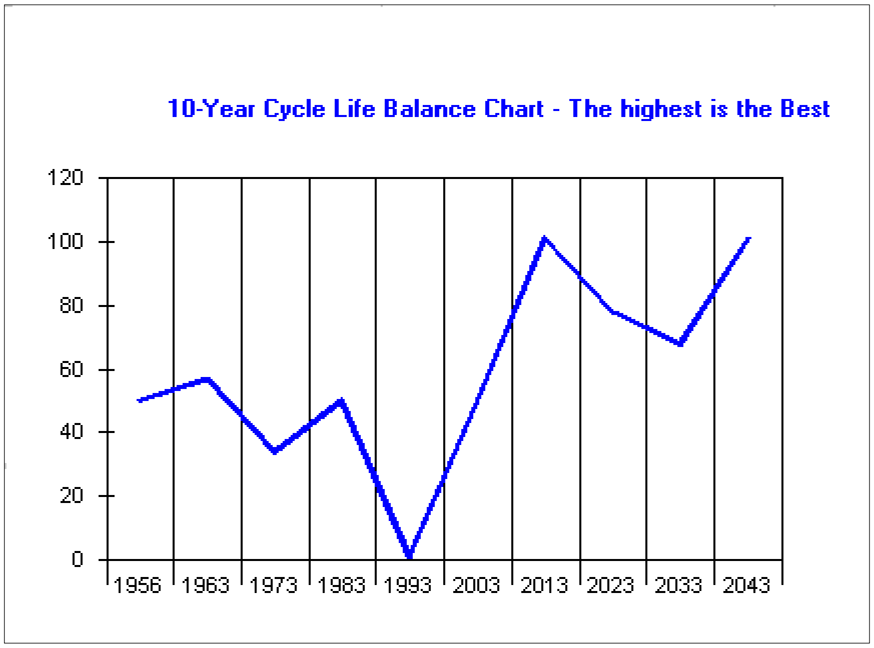


Figure 4.14. Example of a line chart mistakenly treated as bar chart

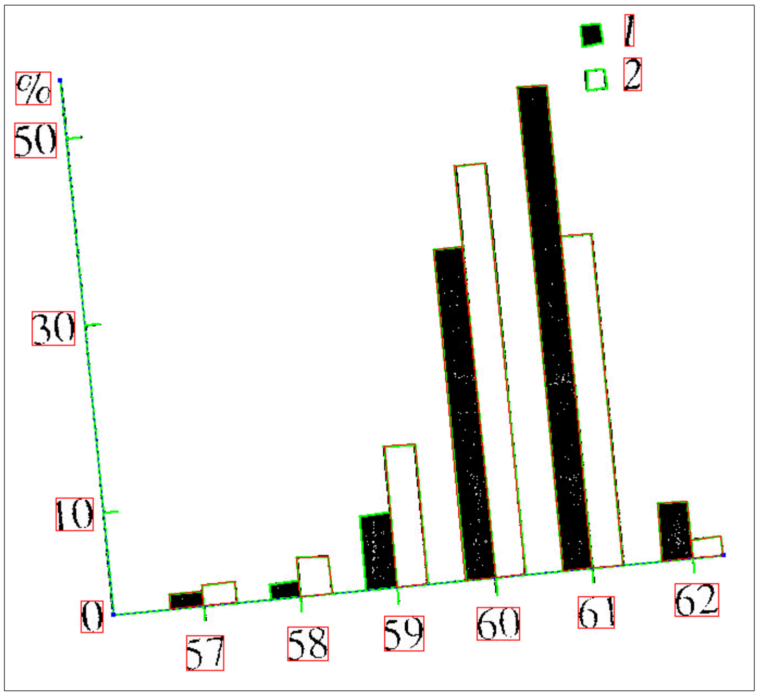


Figure 4.15. Sample 2D bar chart with skew angle

Some of the chart images are skewed with significant angle, such as the example shown in Figure 4.15. As the graphical symbol construction and the text grouping methods are all rotation invariant, the existence of skew angle does not affect the recognition of axis lines, data components and text blocks. Thus the type of the chart can still be correctly recognized. The boxes in Figure 4.15 represents text blocks formed.

4.4.5 Experiment for Data Component Recognition

The performance of data component recognition is also measured. The ground truth data for the 200 testing images specifies the number of bars for bar charts and the number of wedges for 2D pie charts and 3D pie charts. For each chart image in the dataset, data components were constructed from the vectorized lines and arcs after coordinate line detection, regardless of the chart type which was still unknown during graphical symbol construction. After all the images are processed, the total number of data components for each kind was calculated. The performance metrics are recall and precision. Precision is the number of correctly detected data components divided by the total number of data components detected. Recall is the number of correctly detected data components divided by the number of data components in the ground truth. The results are summarized in Table 4.4.

Table 4.4 Performance of data component recognition

	In the ground truth	Detected	Correctly detected	Precision (%)	Recall (%)
Bars	1719	1521	1512	99.41	87.96
2D Wedges	351	328	322	98.17	91.74
3D wedges	50	41	41	95.56	82.0

A common cause of missing data components is due to the poor image quality, which affects the results of vectorization as well. If a straight line or an arc is broken into more than one segment with large gap in between, then the data component this line or arc belongs to cannot be properly constructed. This is again a side effect of error propagation. Besides this, many missing bars are due to occlusion. This happens when a bar chart contains more than one data series and the bars in one data series is occluded by the bars in another data series. Such occlusion violates the constraints used for bar construction. An example is shown in Figure 4.16. Recognition of occluded graphical symbol presents a challenging problem. One possible solution is to generate hypotheses based on the portion of the graphical symbol that is visible with a probability that it matches some defined symbols in the domain knowledge.

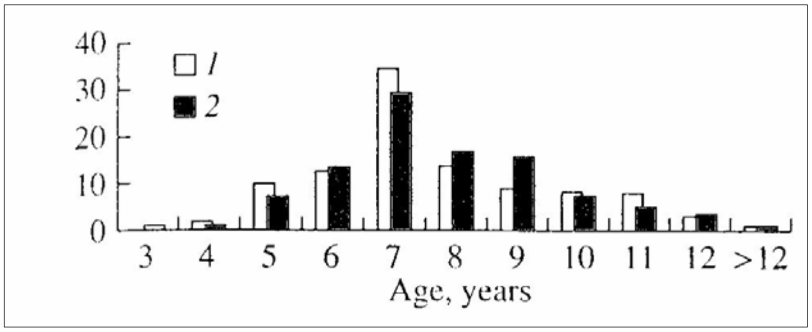


Figure 4.16. Sample 2D bar chart with bar occlusion

Another cause of missing bars is the failure of coordinate line detection. Some false positives of bars occur in line charts where grid lines are wrongly treated as bars with the same height. In some extreme cases, the arc of a wedge is so short that it is recognized as a straight line segment by the program. This happens for both 2D and 3D wedges. False positives of 2D wedges are the 3D wedges whose two straight sides have similar length and the major and minor radii are similar as well. Such wedges also lower the recall of 3D wedge recognition.

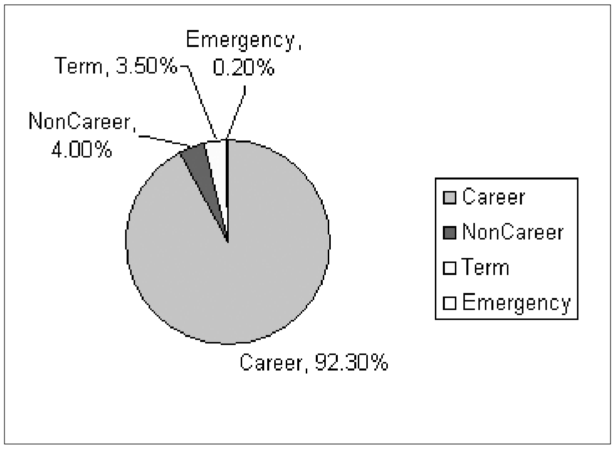


Figure 4.17. Sample pie chart with very small percentage

There are extreme cases where the percentage of a category is very small. In this case, the corresponding wedge in the pie is so narrow that it is hardly recognizable. For example, category "Emergency" in Figure 4.17 only has 0.2% and the appearance of the wedge in the pie is actually two straight lines touching each other, to form a straight line that is thicker than others. The system fails to detect the wedge in this extreme case.

4.4.6. Experiment for Learning Based Data Component Recognition and Chart Classification

To test the ability of the system to handle new types of chart, 10 doughnut chart images were added into the dataset. The experiment was carried out in 20 test runs. During each test run, a number of images were randomly chosen from each chart type to form the set of training images I_{train} and the remaining images became the testing images I_{test}. During training process, one chart type was learnt at a time and the CF values were stored. During matching, formula (4.12) was applied and the chart type returning the highest similarity value was assign to the testing image. Due to the space limit, only the average accuracy of chart classification for the 20 runs is presented in Table 4.5. In each test run, the accuracy is calculated as the percentage of testing images that were correctly classified.

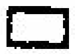
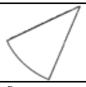
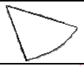
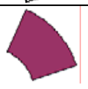
Table 4.5. Summary of classification results

No. of I_{train} per run	Type	No. of I_{test} per run	Average Accuracy (%)
3	2D bar	77	88.81
	2D pie	45	89.33
	3D pie	9	91.11
	Line	57	14.04
	Doughnut	7	100
5	2D bar	75	95.00
	2D pie	43	89.19
	3D pie	7	95.71
	Line	55	3.91
	Doughnut	5	100

From Table 4.5, we can see that the accuracy of chart image classification is reasonably good for all chart types except line chart. This is because of the assumption we made in the introduction section that data are represented using shapes, which is not true for line chart where data are actually represented using x-y plots. Although some shapes appear occasionally, none of them is closely correlated to the type. The correlation factor of most shapes for line chart is zero, and as a result, the similarity value calculated is also zero, causing the system to fail to recognize the correct type for line chart images. When all the similarity values are too low for an input image, the type of the image will be “unidentified”.

During the training process, the system also identified the shapes with the highest CF value. These shapes are the best candidates to be the representation of data component for each chart type, and they are summarized in Table 4.6. The first three values in the feature vector show the number of edges for each edge type. For example, a data component in 2D bar chart has 4 straight line edges, 0 circular arc edges and 0 elliptic arc edges; while a data component in 2D pie chart has 2 straight line edges and 1 circular arc edge. The fourth value in the feature vector is a sequence among edges reflecting how the edges are ordered (denoting a straight line as 1, a circular arc as 2 and an elliptic arc as 3). Rotation is taken care of here, thus the order 131 is the same as 311. The last two values in the feature vector show the number of parallel edges and number of symmetric axes.

Table 4.6. Data component identified for chart types

Chart type	Feature vector	Sample shape
2D bar	<4,0,0, 1111, 2, 2>	
2D pie	<2,1,0, 121, 0, 1>	
3D pie	<2,0,1, 131, 0, 0>	
Doughnut	<2,2,0, 1212, 1, 1>	

Another output of the system is the degree of neighborhood among shapes, calculated by formula (4.13). One restriction is that the shapes to be considered must have non-zero CF value, meaning that they must appear in all positive examples. With this restriction, we only found one combination of shapes whose *DoN* > 0 for all test runs: <2,0,1, 131, 0, 0> and <2,0,2, 1331, 2, 0>. This is a typical combination of shapes in 3D pie charts, such as shape no.1 and no. 5, or shape no. 3 and no. 7 in Figure 4.10(b). For the 2D charts, no common combination of shapes was found. This is expected, since data are represented using single shape in these 2D charts.

Chapter 5

Chart Interpretation

Through a perceptual test, Zhou et al. claimed that graphical information and textual information can be processed comparatively independently at the intermediate vision level, which focuses on chart recognition [3]. However, textual information and graphical information are both essential for capturing higher level information. With either one missing, the interpretation of a chart is not achievable.

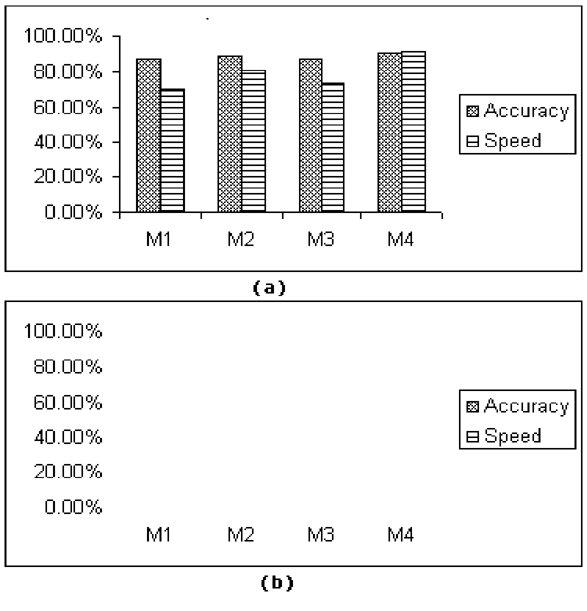


Figure 5.1. A perceptual test on a chart conducted by Zhou et al. (a) A testing chart image. (b) An image without the stage area of the testing image. All the ten subjects can categorize image (b) as a chart without the knowledge of image (a) within 2 seconds. (Reproduced from [3])

Let us illustrate the issue by looking at the example given by Zhou et al. in Figure 5.1. Although by removing the graphical part of the bar chart in Figure 5.1(a), all subjects can still correctly categorize Figure 5.1(b) as a chart, it is impossible for subjects to reveal the semantic information in the chart. The textual part only presents a sequence of percentages, a sequence of labels and two legends. On the other hand, the graphical part only contains a horizontal line at the bottom, a vertical line on the left, and a sequence of rectangles attached to the horizontal line. What would the scale of each axis be? What is the label of each bar? What is the value represented by each bar? The answers to these questions all remain unknown until the textual and graphical information contained in the images are connected with each other.

Text components are grouped into blocks and then are recognized making the use of OCR. On the other hand, the graphical components are segmented and form high level symbols, as illustrated in the previous chapter. The association of text and graphics examines both structural and semantic correspondence between these two types of information and it allows us to capture the semantic meaning carried by chart images in a more complete way. The combined information is then used to perform chart image interpretation. The result is saved in both XML format and plain text format, to be used by different applications later.

5.1 Text/Graphics Association

As Tombre et al. already pointed out, associating textual information with graphics is an important step in the semantic labeling of graphics [32]. However in the past, although research activities on both graphics recognition and text recognition were continuously growing, very few of them actually did attempt to associate the two kinds of information together. Kasturi et al. developed a system to interpret various components in a line drawing, including text strings [33]. However the text components were not recognized. Joseph and Pridmore presented an experimental system for mechanical engineering drawing interpretation [34]. Like Kasturi's, the system had no provision for the text recognition. Lamiroy et al. have conducted experiments to analyze the role of the text components in cutaway diagrams [35, 36]. The result reported was limited to identifying the relationship among drawings, their indices and the legends. Furthermore, in their work the graphics layer is completely discarded so the text and graphics association was still in preliminary stage. Thus our work is undoubtedly a novel attempt in the document image analysis research area.

It is worth to note that there are works on auto-annotation, which also involves associating text with images, such as [37-39]. The text used either comes from the text accompanying the image [40, 41], or are automatically assigned after image categorization. In the auto-annotation works, text can be used to describe the characteristic of an object (unary constraint), or the spatial relationship among objects or parts of an object (binary constraint), or even the whole picture. It is also possible to use the text contained in the image directly, such as text in video frames but this requires text segmentation and OCR process to obtain the textual information [42-45]. Text/graphics association for chart image recognition is similar to text/graphics association for auto-annotation in the sense that both works involve finding the correspondence between textual information and image information. In auto-annotation work, the image information can be feature information from the whole image, region information or object information. In our case, image information mainly refers to graphical symbols constructed. The two works are different from each other from the following aspects:

- In most auto-annotation works, the electronic text that is machine readable is required. In our case, however, text is presented as a part of the image, i.e. it is in the raster format. Text blocks need to be extracted and recognized before being associated with graphics.
- In document image analysis task, besides the actual content of the text, the role of the text is also important because it allows the system to provide syntactic information of the document. This also applies to chart recognition. In chart recognition, the logical role of each text block needs to be determined further. This is, however, untouched in auto-annotation works because the role of the text is indexing and is describing the image after the association task.
- In auto-annotation works, after being associated together, textual information is used for indexing and retrieval. In our case, the textual information is further used to get the semantic information of the chart image, such as calculation of the data values etc.

The differences discussed above are summarized in Table 5.1.

Table 5.1. Differences between auto-annotation and chart recognition

	Auto-annotation	**Chart recognition**
Source of text	Mainly outside image	Inside image
Format of text	Electronic	Image
OCR	May be needed	Necessary
Logical role of text	Not required	Required
Further manipulation of text	Maybe	Yes

5.1.1 Problem Formulation

The task of associating text blocks with graphical symbols is modeled as a problem of classifying text blocks into different roles in a given chart. In other words, we need to find out which text block corresponds to which graphical symbol and what kind of role does the text block play in the chart. After going through a collection of charts, we have identified 11 text roles in charts in total, which are summarized in Table 5.2. As one can see from the table, most text blocks are attached with certain graphical symbols, such as the x-axis label etc., while some text blocks play a global role, such as the title of the whole chart etc.

Table 5.2. Major roles of text in charts

Block label	Role in charts	Type of role
CTTL	Title of the chart	Descriptive
XTTL	Title of x-axis	Descriptive
XLAB	Label along x-axis	Implicative
XUNT	Unit of x-axis	Implicative
YTTL	Title of y-axis	Descriptive
YLAB	Label along y-axis	Implicative
YUNT	Unit of y-axis	Implicative
DVAL	Data value	Descriptive
DLAB	Data label	Descriptive
LGND	Legend name	Descriptive
OTHR	Other description	Descriptive

Furthermore, the 11 roles of text are the delineated into two major types, as shown in the last column in Table 1. First of all, a text block can be used to describe a graphical symbol or the whole chart. Such text blocks are called *descriptive* blocks. An example is the data label *DLAB*, which describes each data entry plotted in the chart. On the other hand, some text blocks do not provide any description. Rather, they join with the graphical symbols to imply information that is not directly presented. Such text blocks are called *implicative* blocks. One example is the labels along y-axis *YLAB*, which normally are numbers. The numbers themselves do not carry semantic meaning, but are used to specify the range of data values.

5.1.2 The Proposed Solution

The existing approaches to assign labels to text blocks are mostly based on locational features [3, 88]. The location based rules assumes that certain type of text is always placed in certain location in a chart respectively. For example, the labels along x-axis are always below the x-axis line. However the assumption is quite fragile. First of all, the way text is placed in a chart can be customized. For example, the legends can appear within the plot area or outside the plot area. Secondly, some chart types do not have strictly structured layout. For example, a pie chart can have a number of wedges and the labels for the wedges are placed around them. It is infeasible to say that data labels must appear in certain location in this case. Thus rule based approaches are unsuitable, unless all situations can be exhaustively enumerated.

A better alternative is to train the system to learning how to assign labels to text block automatically. In this way, the rules can be obtained through training with realistic examples. Even if the rules do not cover some incoming new cases, the system can still be re-trained to update the rules automatically.

Based on the information about the text blocks and graphical symbols, five features are defined for training the association rules. Three of these features capture explicit locational relationship between text blocks and graphical symbols, one feature makes use of the implicit characteristics of a text block itself, and the last one represents the global positional information. We hypothesize that the text block and graphical symbol to be associated together are nearest neighbor of each other. This hypothesis is based on the argument raised by Larkin and Simon [46] stating that information in diagrammatic representation is organized by location. This hypothesis can significantly reduce the amount of information to be processed by the classifier since only the nearest symbol is examined. The details of the features are as below.

- **Distance between a text block and the nearest graphical symbol.** The value of this feature is real. The distance between a text block T_i and a graphical symbol G_j is defined as:

$$D(T_i, G_j) = \min D(E_{il}, E_{jk}), \forall E_{il} \in T_i, \forall E_{jk} \in G_j \tag{5.1}$$

where E_{il} is one of the four edges of the bounding box of T_i, and E_{jk} is one of the edges of graphical symbol G_j. Every graphical symbol G_j can be treated as a composition of edges. Function $D(E_{il}, E_{jk})$ is a basic geometric function that calculates the shortest distance between two edges. There is a special case where the text box is inside the graphical symbol (a 2D shape in this case), and the distance is set to zero. This is detected by checking whether the center of bounding box is inside G_j. By calculating the distance between a text block and all existing graphical symbols, the symbol that is nearest to the text block is identified. Only the distance between the text block to the nearest graphical symbol is stored.

If we directly use absolute distance as the value of this feature, then it may be affected by the size of the image because larger images tend to result in longer distances. To remove this effect, we normalize the coordinates of every point $P(x, y)$ by dividing x by image width W and dividing y by image height H. After normalization, both x and y falls into the interval [0, 1), and the distance calculated is independent of the image size.

- **Type of the graphical symbol that is nearest to a given text block.** This feature is nominal. Although the set of graphical symbols depends on the type of charts, there are some commonly used symbols. These symbols form the universal set of graphical symbols and each element in the set is given a label. For example, graphical symbols may be labeled as "X-AXIS", "Y-AXIS", "BAR", "WEDGE" etc. The value of this feature is actually determined together with the first feature. Sometimes a text box is inside a graphical symbol, then that symbol is treated as the nearest to the text box.
- **Relative position between a text block and a graphical symbol.** This feature is nominal. If we use the center C of a graphical symbol as the reference point, the angle between the center of the text bounding box and the center of the symbol can be any value from $[0, 2\pi)$. In our work, we quantize the angles into 8 intervals. Then the relative position between a text box and a graphical symbol can be represented as one of the 8 labels (*T = top, B = bottom, L = left, R = right, TL = top-left, TR = top-right, BL = bottom-left, BR = bottom-right*). Again this feature is only calculated between the text block and its nearest neighbor. If the text box is inside the graphical symbol, then the relative position is *NIL*.
- **String checks.** We also use a string parser to check if a given text string can be interpreted as an integer or a floating point number. The purpose is to find out whether the text indeed directly represents values. The parser is written as a Boolean function *isNumber()* which returns true when the text string can be parsed to integer or floating point number, and false otherwise. This feature is calculated implicitly for any given text block.
- **Centricity of a text block.** This feature is calculated to measure how close a text block

is to a bisector of the whole image. It is a global feature and its value is real. The value is normalized onto the interval [0, 1], where 0 means the text block lies on the bisector and 1 means the text block is farthest from the bisector. The calculations are performed for both the horizontal and vertical bisectors, so there are actually two values. These two values are further stored separately in the format of a feature vector. This feature is useful to classify text blocks with global roles, such as chart title etc.

The next step is to use a machine learning algorithm to learn a set of association rules based upon the training examples. A set of training images are collected from the internet or scanned documents. Text blocks were extracted from these images and were manually assigned a block label. Every labeled text block forms a training example. The feature vector contains value of the five features defined and the correct class that the text block belongs to, for example <0.047420, Y_AXIS, L, true, 0.23, 0.78, YLAB>. The learning algorithm used by our system is C4.5 [47], a decision tree learner used in many machine learning tasks. C4.5 is the extension of the ID3 algorithm and it allows continuous attributes such as the distance feature used in our work.

During actual classification, the same set of features is calculated from the text blocks extracted from testing images. The trained classification rules are then applied to assign the corresponding label to each text block. Figure 5.2 shows an example of association results. As the association is a two-way process, the content of a text block is also attached to the corresponding graphical symbol, for future interpretation purpose.

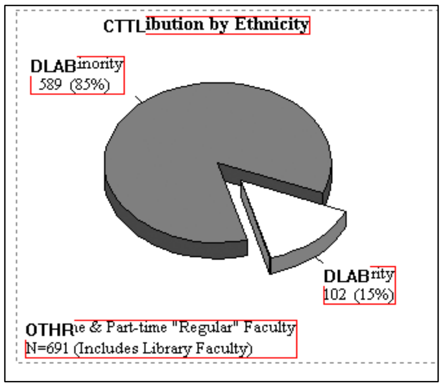


Figure 5.2. Example of text/graphics association

5.2 Extraction of Tabular Data

To achieve some high-level understanding, the textual and graphical information are combined together to further imply information that is not directly presented. One such information is the data values, as chart is a visualization tool to present tabular data. The core information of the tabular data consists of data label plus data value. After associating text with graphics, the system further performs interpretation to obtain both the data label and the data value. There are two rules for finding the data labels.

Rule 1: If the data label directly appears as a *DLAB* text block in the chart, then it is directly associated with the corresponding graphical symbol representing the data component using our method.

Rule 2: If no text block is classified as data label, then labels attached to the axis are aligned with data components in the x-y axis area to assign the data label. The alignment rule is based on the relative position between axis labels and data components.

To obtain the value for each data entry, similar rules are applied:

Rule 1: If the data value appears as a *DVAL* text block, then it is directly associated with a data component using our method and the value is immediately available.

Rule 2: If no text block is classified as its data value, then the value needs to be calculated. The calculation depends on the domain knowledge of each chart type about which attribute of a data component should be looked at. For example, in bar charts we look at the height of each bar shape but in pie charts we look at the angle of each wedge instead. Without the domain knowledge, the calculation cannot be carried out. The type of the value (integer or float) calculated should agree with the type of the axis label, if there is any.

Note that the calculation in Rule 2 is based on the assumption that the scale of the axis line is linear. However for some data plots, non-linear scale has been applied to the axes. In this case, direct calculation of data value based on the scale labels and visual attribute returns incorrect result. This case cannot be handled in the current version of the system, and it is worth to explore further in the future.

5.3 The Generation of Chart Description

Following the interpretation step, descriptions of the chart are generated. To facilitate other applications that may require different representations of information, the system generates two different types of description: an XML format description and a natural language description.

```
<?xml version="1.0"?><!--Chart_description.xml-->
<?xml-stylesheet type="text/xsl" href="Chart.xsl"?>
<!DOCTYPE Chart [
<!ELEMENT chart (type, title, x_axis, y_axis, data_set )>
<!ELEMENT type ( #PCDATA ) >
<!ELEMENT title ( #PCDATA ) >
<!ELEMENT x_axis (axis_title, labels, unit)>
<!ELEMENT y_axis (axis_title, labels, unit)>
<!ELEMENT x_axis_title ( #PCDATA )>
<!ELEMENT y_axis_title ( #PCDATA )>
<!ELEMENT labels ( label+ )>
<!ELEMENT label ( #PCDATA )>
<!ELEMENT unit ( #PCDATA )>
<!ELEMENT data_set ( data+ )>
<!ELEMENT data (label, value )>
<!ELEMENT value ( #PCDATA )>              ]>
<chart>
<type>2D Bar Chart</type>
<title>Free-swimming males</title>
<x_axis><axis_title>Time of day</axis_title>
<labels><label>5</label>
          ……
        <label>1930</label></labels></x_axis>
<y_axis><axis_title>Rate of foraging</axis_title>
<labels><label>0</label>
          ……
         <label>8</label></labels></y_axis>
<data_set>
<data><label>5 to 7</label><value>2.32</value></data>
<data><label>7 to 9</label><value>8.36</value></data>
           ……
<data><label>17 to 1930</label><value>3.86</value></data>
</data_set></chart>
```

Figure 5.3. Sample illustrations of XML description generation

5.3.1 The Generation of XML Description

The XML format description is used to represent the information contained in the given chart image in a tabular form. The hierarchical XML format is defined as follows.

At the top level, the tag <chart> is used. It contains the following parts:

- <type>: the type of the chart.
- <title>: the title of the chart if it exists.

- <x_axis> and <y_axis>: the existence of x-y axes depends on the type of the chart. If they do exist, <axis_title> shows title of each axis. <labels> contains a set of <label> attached to each axis. <unit> specifies the unit used by each axis.
- <data_set>, the tabular data obtained from the chart image. Each data entry has a <label> and a <value>.

Figure 5.3 shows part of the XML format description generated for a 2D bar chart image. The XML description presents the chart information in a tabular form, which makes it convenient for some applications such as the query processing and the re-construction of the chart etc. Note that the DTD part of the XML file is independent of the chart types, thus some fields that are invalid for some chart types have been given a value of "NIL" during generation.

5.3.2 The Generation of Natural Language Description

Some applications, on the other hand, works the best on text in its natural language format. An example is the question answering that is continuously being explored in the information retrieval (IR) and information extraction (IE) research community. To be able to apply existing question answering techniques, it is required that the result of chart interpretation should be presented in a natural language form as well. In other words, the description should consist of a set of natural language (NL) sentences. To do so, a set of templates is defined. The templates are type independent, and they cover all the components in the original chart. Figure 5.4(a) lists some of the templates defined. The detailed labels and values used to fill in the templates are obtained through the text/graphics association process and the interpretation process. After all the sentences are generated, they are combined to form a single paragraph, such as the one in Figure 5.4(b).

- Figure <f_no> contains a <c_type> chart with the title <c_title>.
- The data contained are <x_title> versus <y_title>.
- Data entry <d_label> has a value of <d_value> <y_unit>.

(a) Sample templates for NL sentence generation

Figure 1 contains a line chart with no title. The title of the x-axis is year. Data entry 1982 has a value of 1080 fatalities. Data entry 1983 has a value of 1156 fatalities. Data entry 1984 has a value of 1250 fatalities. Data entry 1985 has a value of 1181 fatalities. Data entry 1986 has a value of 1163 fatalities. Data entry 1987 has a value of 1393 fatalities. Data 1988 has value of 1590 fatalities.

(b) Sample NL sentences generation for the same chart

Figure 5.4. Illustrations of NL description generation

5.4 Experiments and Discussions

The same set of 200 chart images used in the experiments in Chapter 4 has been used to test the performance of chart interpretation module. As the evaluation of the output of the description generation part is very subjective, quantitative performance evaluation was only carried out for the text/graphics association step.

5.4.1 Experiment for Text/graphics Association

To evaluate the performance of the text/graphics association module in the system, all the text blocks in the chart images in the dataset were manually assigned the correct block label beforehand. Such assignment was included in the ground truth data. There are 1222 text blocks in the bar charts, 789 text blocks in the line charts and 246 text blocks in the pie charts. All images were passed to the system for extracting text blocks and graphical symbols automatically. Then a feature vector was calculated for each text block identified by the system and used for training and testing. 10-fold cross validation was used to measure the accuracy of the classifier trained by the C4.5 decision tree learner. The precision and recall of each of the 11 text block classes for each chart type has been shown in Table 5.3. “Precision” is defined as the number of text blocks correctly assigned a label divided by the total number of text blocks as assigned the same label. On the other hand, “recall” is defined as the number of text blocks correctly assigned a label divided by the number of text blocks having the same label in the ground truth.

Table 5.3. Text block classification results

	Bar chart		Pie chart		Line chart	
Block label	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Precision (%)	Recall (%)
CTTL	53.6	93.8	100	83.33	70	87.5
XTTL	65.2	78.9	-	-	84.2	84.2
XLAB	96.4	98.3	-	-	91.1	95.9
XUNT	-	-	-	-	-	-
YTTL	61.5	72.7	-	-	-	-
YLAB	100	95.5	-	-	95.1	94.5
YUNT	85.7	75.0	-	-	-	-
DVAL	-	-	-	-	-	-
DLAB	-	-	79.1	95.6	-	-
LGND	93.3	77.8	85.5	59.4	-	-
OTHR	33.3	9.1	-	-	-	-

5.4.2 Discussions

From the table we can see that some text class do not appear in the images of a particular chart type. For example, pie charts in the current dataset mainly have three types of text: chart title (CTTL), data labels (DLAB) and legends (LGND). Some text class never appears in some chart types. Text classes related to coordinate lines will not appear in charts without coordinate lines. However, as the dataset is not general enough, some possible situations are not covered here. For example, data values may be specifically marked in bar charts.

Since this is the first attempt to our knowledge for the text/graphics association problem in this domain, there is no comparative study available in order to judge whether the precision and recall achieved are good or not. We can only say that for most classes, the precision and recall are at a reasonable level. The poor precision and recall of the class OTHR for bar charts is due to the common confusion among chart titles, axis titles and other text descriptions. In many situations, all the descriptions about the chart are placed right below the chart title, sometimes near the title of the Y-axis. Thus it is very easy to treat such descriptions as the chart title of axis title. On the other hand, if the title of an axis is close to the top or bottom of the image and is near the center, the axis title may also be mistakenly treated as the chart title. Such confusion is very difficult to avoid, even for human beings in some cases. Take Figure 5.5 as an example, the string "Number of oocytes, %" is placed on top of the y-axis and towards the center of the chart. It is treated as chart title instead of the title of the y-axis by the system.

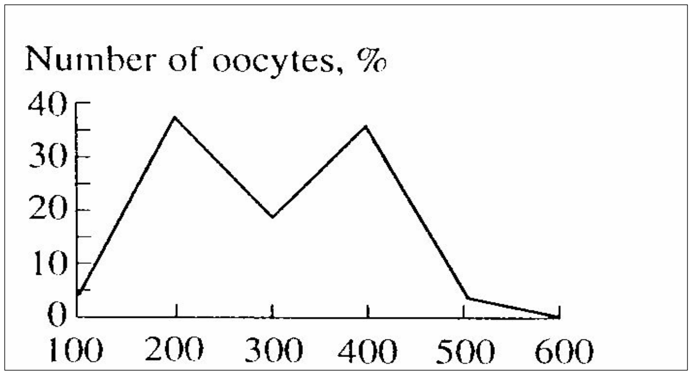


Figure 5.5. Confusion between axis title and chart title

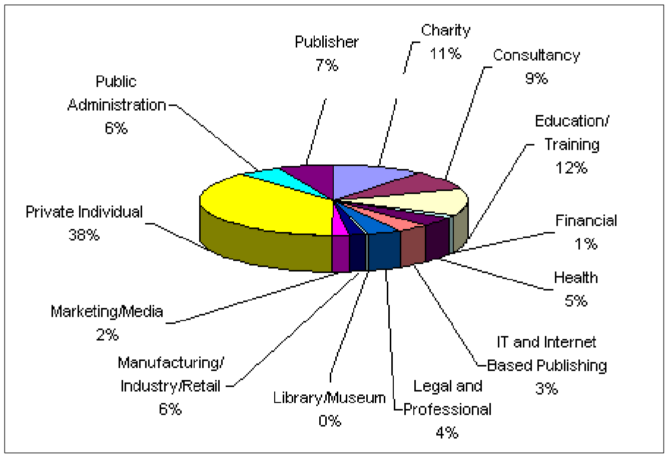


Figure 5.6. Sample pie chart with many wedges

In some cases, there are so many data entries in a pie chart that the wedges become crowded and there is not enough space to place the labels near the corresponding wedges. The solution during generation is to add extension lines to the wedges that link them to their labels. This case is not dealt with by the proposed text/graphics association method as extension lines are not recognized by the graphical symbol construction step. If the extension lines are included in the set of graphical symbols to be recognized, however, the method is still applicable in this situation.

Chapter 6

Applications

There is a Chinese proverb “a picture is worth a thousand words.” It accurately reflects the fact that pictures carry significant amount of information. This is also true for charts that are visual tools to present trends or patterns in data in a graphical way. However in the past, the lack of recognition tools for charts in the imaged form became an obstacle to information retrieval and extraction. Chart recognition and interpretation methods in this dissertation extract both syntactic information and semantic information from scientific chart images, which has been previously missing for some existing applications, such as the document analysis systems and the information extraction systems. Thus the supplementary information extracted from charts surely helps to improve the performance of these systems. In this chapter, two applications are discussed as case studies to illustrate how the output mentioned in the previous chapters can be used by these applications to enhance their performance.

6.1 Case Study One: Supplement to OCR System

Traditional OCR systems focus on segmenting and recognizing the text from input document images [48-50]. There are mainly two parts of output generated by a traditional OCR system: the recognized text in the electronic form and the layout information of the original document. For other objects in the document, such as figures and other drawings, the OCR system simply segments them out from the original document page. Thus in the output of existing OCR systems, the figures and drawings remain as images. This makes sense when there is no good technique to correctly recognize these figures and drawings, thus the safe way is to keep them untouched and leave it for the end user to manually recognize them.

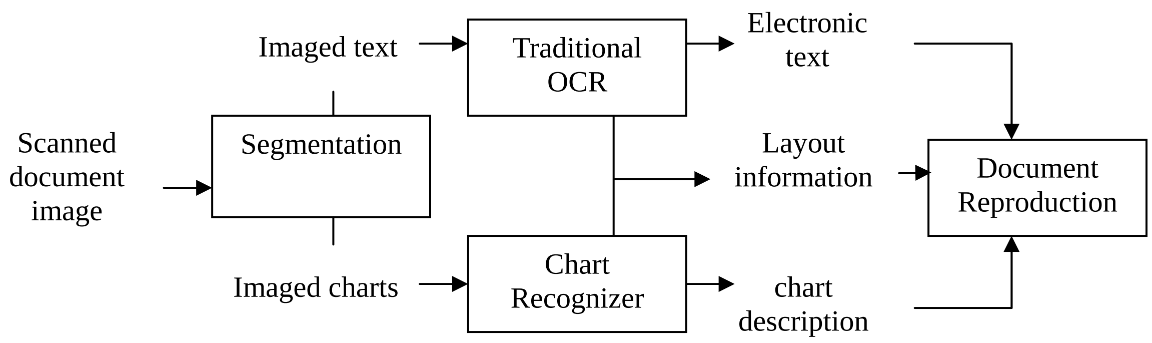


Figure 6.1. Augmentation of the proposed system with traditional OCR system.

Although our system does not aim to provide a general solution to recognize all the objects left out by existing OCR systems, it can be integrated with an OCR system to recognize the charts in the documents. Furthermore, the information carried by the XML description can be used to easily re-construct the same chart image in an electronic form. The re-constructed chart image, together with the electronic text and the layout information, can be used to reproduce the original document in a more complete form, following the steps in Figure 6.1.

A key step for the augmentation of the chart recognition with OCR is to find the location in a document page where the re-constructed chart image to be inserted. This is important because one of the desired features of an OCR system is that both structural layout and logical layout of the original document should be preserved as much as possible. To handle this issue, there are two ways to do it. Typical OCR techniques define a document element as either a text paragraph or a figure, and determine the reading order among these elements [51]. If such a reading order is available, then the elements before and after a chart image are known. The chart image is re-constructed and placed between the two elements. Otherwise, the system makes use of the figure caption below (or above in some cases) the chart image and searches for its first appearance in the text. The paragraph containing such a figure caption is considered as the element before the chart image, and the following paragraph will be treated as the element after the chart image.

Let us use an example to illustrate the steps. A sample document image is shown in Figure 6.2. The image is a scanned journal page selected from the University of Washington document image database I. The bounding box indicates the chart area detected using figure segmentation method. The chart in the bounding box is fed into the chart recognition and interpretation system, while the remaining part of the image is fed into an OCR system for recognition. The OCR system used here is the Omnipage Scansoft SDK version 12.0, which is a commercial OCR development kit. It is a typical OCR system that allows both the zoning information and the recognized text in electronic form to be extracted. As page segmentation is performed before text recognition, the process is controlled in a way such that the text recognition result from each zone is stored separately and is indicated by the zone identifier automatically assigned using a three-digit number. The zoning information returned by the OCR system is shown in Figure 6.3. The recognized text for the first two zones is shown in Figure 6.4. The output of chart recognition and interpretation is shown in Figure 6.5.

Based on the zoning information, the reading order can be determined and it specifies that the chart image is the second element in the page, after the figure caption and before any other text in the page. Thus in the reproduced document, the information captured from the chart image is placed immediately after the figure caption (ZONE_ID = 000), followed by other text. Note that there are two kinds of chart descriptions available: the XML format description and the natural language description. The choice of the format depends on how the document is to be re-constructed. If the re-constructed document is plain text, then the NL description is chosen. On the other hand, if the re-constructed document needs to contain figures and drawings, then the XML description can be used to generate charts to be placed in the same page.

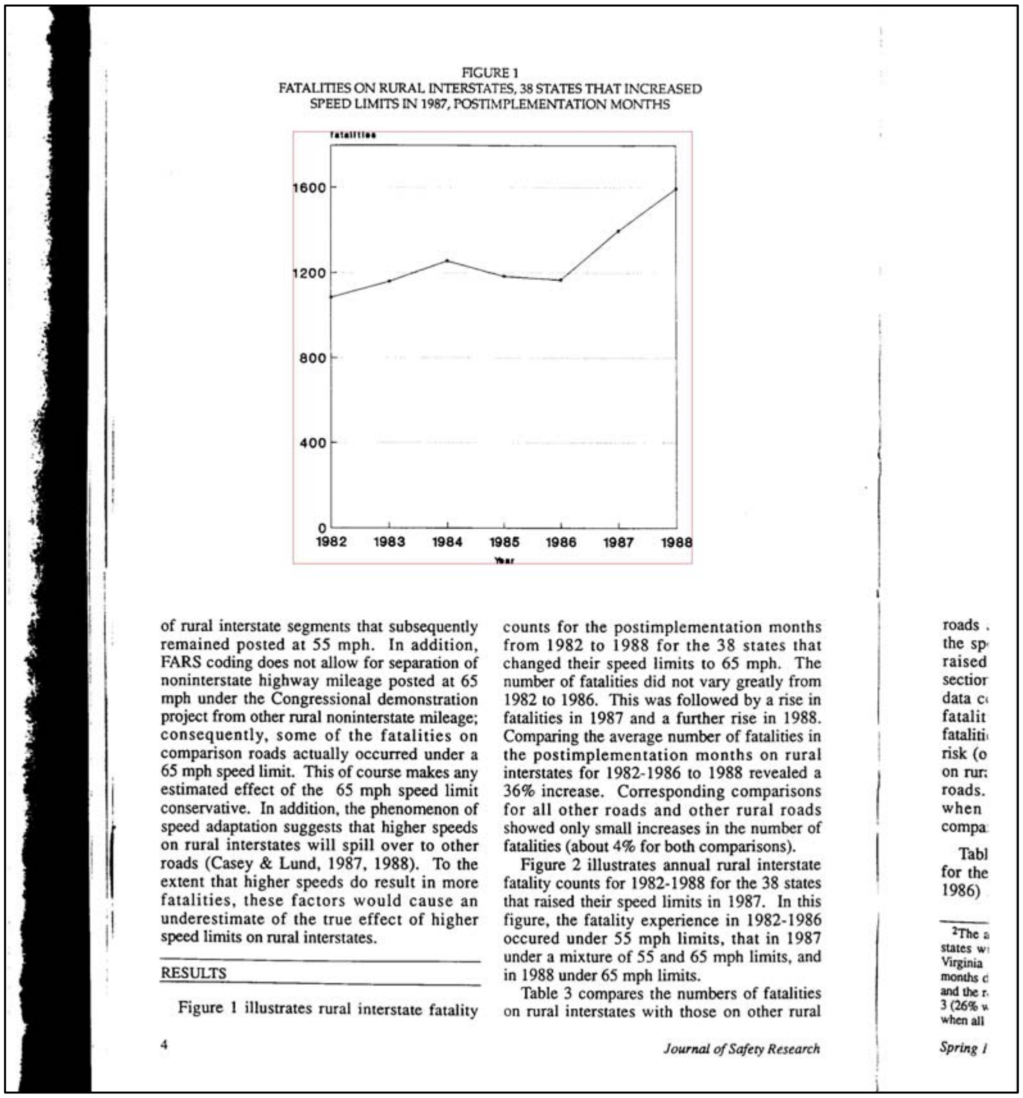
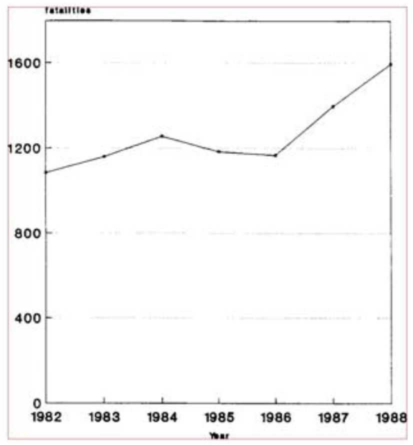


FIGURE 1
FATALITIES ON RURAL INTERSTATES, 38 STATES THAT INCREASED SPEED LIMITS IN 1987, POSTIMPLEMENTATION MONTHS



of rural interstate segments that subsequently remained posted at 55 mph. In addition, FARS coding does not allow for separation of noninterstate highway mileage posted at 65 mph under the Congressional demonstration project from other rural noninterstate mileage; consequently, some of the fatalities on comparison roads actually occurred under a 65 mph speed limit. This of course makes any estimated effect of the 65 mph speed limit conservative. In addition, the phenomenon of speed adaptation suggests that higher speeds on rural interstates will spill over to other roads (Casey & Lund, 1987, 1988). To the extent that higher speeds do result in more fatalities, these factors would cause an underestimate of the true effect of higher speed limits on rural interstates.

RESULTS

Figure 1 illustrates rural interstate fatality counts for the postimplementation months from 1982 to 1988 for the 38 states that changed their speed limits to 65 mph. The number of fatalities did not vary greatly from 1982 to 1986. This was followed by a rise in fatalities in 1987 and a further rise in 1988. Comparing the average number of fatalities in the postimplementation months on rural interstates for 1982-1986 to 1988 revealed a 36% increase. Corresponding comparisons for all other roads and other rural roads showed only small increases in the number of fatalities (about 4% for both comparisons).

Figure 2 illustrates annual rural interstate fatality counts for 1982-1988 for the 38 states that raised their speed limits in 1987. In this figure, the fatality experience in 1982-1986 occured under 55 mph limits, that in 1987 under a mixture of 55 and 65 mph limits, and in 1988 under 65 mph limits.

Table 3 compares the numbers of fatalities on rural interstates with those on other rural

roads
the sp
raised
sectior
data c
fatalit
fatalit
risk (o
on rur
roads.
when
compa

Tabl
for the
1986)

[2]The
states w
Virginia
months
and the r
3 (26%
when all

Figure 6.2. A sample scanned document containing a line chart. The boxed area is the chart area identified by the system.

```
ZONE_ID       = 000
CORNER_ONE_RC = 668 616
CORNER_TWO_RC = 1812 756

ZONE_ID       = 001
CORNER_ONE_RC = 676 772
CORNER_TWO_RC = 1848 1924

ZONE_ID       = 002
CORNER_ONE_RC = 372 2012
CORNER_TWO_RC = 1228 2856

ZONE_ID       = 003
CORNER_ONE_RC = 364 2864
CORNER_TWO_RC = 1228 2900

            … … … …
```

Figure 6.3. Zoning information returned by OCR system

```
ZONE_ID     = 000

FIGURE 1
FATALITIES  ON  RURAL  INTERSTATES,  38  STATES  THAT  INCREASED  SPEED
LIMITS IN 1987, POSTIMPLEMENTATION MONTHS
```

```
ZONE_ID     = 002

of rural interstate segments that subsequently remained posted at 55
mph. In addition, FARS coding does not allow for separation of
noninterstate highway mileage posted at 65 mph under the
Congressional demonstration project from other rural noninterstate
mileage; consequently, some of the fatalities on comparison roads
actually occurred under a 65 mph speed limit. This of course makes
any estimated effect of the 65 mph speed limit conservative. In
addition, the phenomenon of speed adaptation suggests that higher
speeds on rural interstates will spill over to other roads (Casey &
Lund, 1987, 1988). To the extent that higher speeds do result in more
fatalities, these factors would cause an underestimate of the true
effect of higher speed limits on rural interstates.
```

Figure 6.4. Text recognized by OCR system

```
<chart>
<type>2D Line Chart</type>
<title>-</title>
<x_axis><axis_title>Year</axis_title>
            <labels><label>1982</label>
                        ……
                        <label>1988</label></labels></x_axis>
<y_axis><axis_title>-</axis_title>
            <axis_unit>fatalities</axis_unit>
            <labels><label>0</label>
                        ……
                        <label>1600</label></labels></y_axis>
<data_set>
<data><label>1982</label><value>1080</value></data>
<data><label>1983</label><value>1156</value></data>
              ……
<data><label>1988</label><value>1590</value></data>
</data_set>
```

(a) the XML description for the chart in Figure 6.2

Figure 1 contains a line chart with no title. The title of the x-axis is year. Data entry 1982 has a value of 1080 fatalities. Data entry 1983 has a value of 1156 fatalities. Data entry 1984 has a value of 1250 fatalities. Data entry 1985 has a value of 1181 fatalities. Data entry 1986 has a value of 1163 fatalities. Data entry 1987 has a value of 1393 fatalities. Data 1988 has value of 1590 fatalities.

(b) NL sentences generation for the chart in Figure 6.2

Figure 6.5. Result of chart recognition and interpretation

6.2 Case Study Two: Enriching Information for A Question Answering System

Question answering techniques in the current literature mainly focus on text-based documents, by extracting sentence level answers to factoid questions [52] or definitional questions [53, 54]. For a review of the question answering systems, please refer to [55]. In recent years, information retrieval research community has given more and more attention to retrieving information from non-text sources such as images or videos. Thus the question answering techniques is also expected to be extended to make use of non-text sources. Charts are a good example of such non-text sources.

For documents containing charts, the information carried by the charts may not be fully reflected in the text. Thus for questions related to the chart images, traditional question answering methods are not able to return an answer. In this situation, the result obtained by our system becomes very helpful because it brings additional information from the chart images.

Table 6.1. Set of basic queries to the chart images

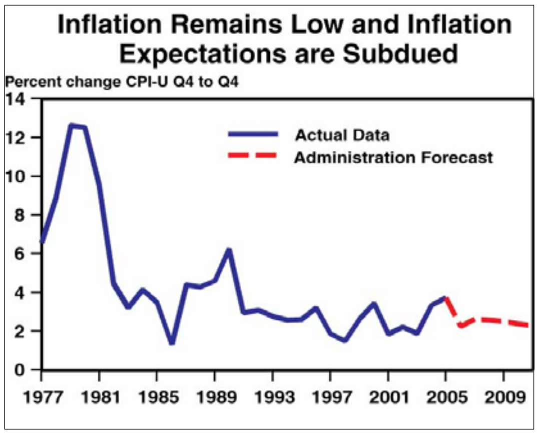
Query Type	**Return Type**	**Description**
Max.	Value or label	Maximum among all existing values.
Min.	Value or label	Minimum among all values.
Avg.	Value or label	Average value of all values selected.
Between	Value or label	Between two values directly specified by user, or between two components specified by user using their labels.
Greater than	Value or label	Data components whose value is greater than the value directly specified by user or the value of the data component specified by user.
Less than	Value or label	Data components whose value is less than the value directly specified by user or the value of the data component specified by user.
Equal to	Value or label	Data components whose value is equal to the value directly specified by user or the value of the data component specified by user.
Difference between	Value	The absolute difference between the values of two data components

6.2.1 Answering Query-like Questions

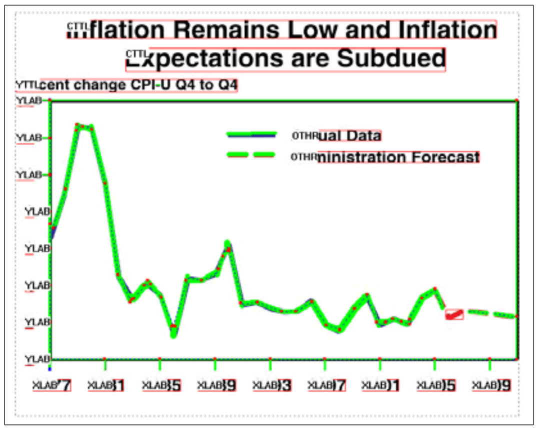
One type of questions purely refers to the chart images, and the answer cannot be obtained in a straightforward way. For example for the chart image in Figure 6.2, questions like *"From when to when does the number of fatalities decrease?"* or *"What is the maximum number of fatalities among all years?"* can be asked. There is no immediate information in the text that provides answers to these questions. These questions are actually data queries, except that the questions are in natural language form. Processing query-like questions is different from the two kinds of QA problems mentioned. It requires two major steps: translating a natural language question to query, and processing the query to generate the answer.

A method for translating natural language sentences to structured queries was presented in [56]. As natural language processing is not the focus of this dissertation, a simplified translation method is used here as an illustration. The method is based on keyword matching in the given question sentence to generate the most likely query. A vocabulary of frequently used keywords is built for this purpose, containing keywords such as "which", "where", "maximum", "minimum", "between" etc. As the questions on a chart can be treated as domain specific in the sense that the number of abstract classes in a chart is limited, the method works well to some extent. For example, *"What is the maximum number of fatalities among all years?"* is translated into the query *"Select D_value Max"*.

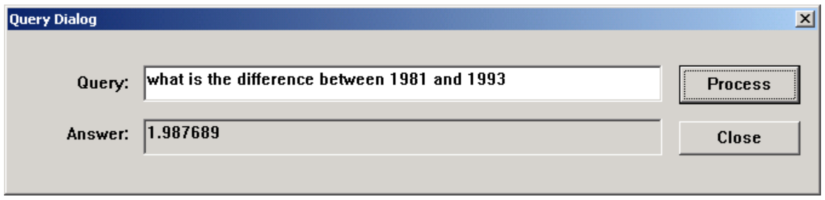
For query processing, a set of basic queries that are frequently raised by users is defined, which are shown in Table 6.1. More complicated queries can be handled by transforming them into a combination of basic queries. Some examples are shown in Figure 6.6.



(a) A sample line chart



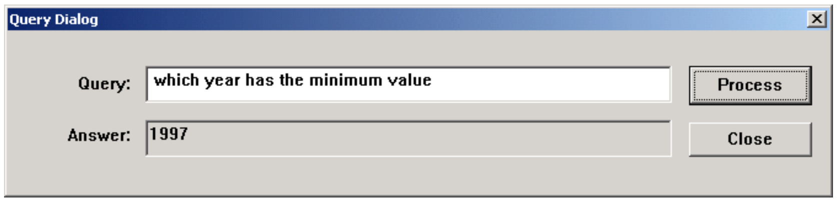
(b) Result of recognition and interpretation


Query Dialog

Query: what is the difference between 1981 and 1993

Answer: 1.987689

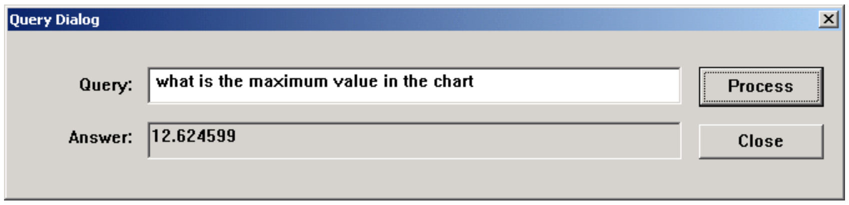
(c) Answering difference query


Query Dialog

Query: which year has the minimum value

Answer: 1997

(d) Answering min. query that returns a label


Query Dialog

Query: what is the maximum value in the chart

Answer: 12.624599

(e) Answering max. query that returns a value

Figure 6.6. Examples of answering query like questions

6.2.2 Answering Natural Language Questions

Based on our survey among a group of human testers, some questions related to the chart images are factoid questions, which try to find out facts from the chart images. For example for the chart image in Figure 6.2, a question asked is *“How many fatalities were there in the year 1984?”* For this type of questions, the original sentences in the text do not contain the require information. However, the natural language description as generated by our system indeed does reveal such information. Thus it can be used to find out the answer. To facilitate the question answering process, the NL description is inserted into the document as an additional part of the text. The insertion point is determined based on the idea described in the previous section. As the NL description contains natural language sentences, it can be handled by most traditional QA techniques.

6.2.3 Experiments on A Question Answering System

We have conducted a small experiment to test how much the system output helps with the question answering process. Ten scanned document pages have been selected from the University of Washington document database I as the testing images. Human testers were asked to give 5 factoid questions and 5 query-like questions for each document page. To involve the NL description of chart images in the question answer process, one of the 5 factoid questions was required to be related to the chart image. We used the question answering system developed by Cui et al. [53] to process both the questions and the documents to find out the answer to each question. Cui’s work uses probabilistic lexico-syntactic pattern matching, also known as *soft pattern matching*, and the result is a ranked list of sentences. In our experiment, only the sentence at the top of the ranked list (i.e. with the highest matching score) was considered as the correct answer. On the other hand, the query-like questions were also parsed using the simplified query translation method and the resulting queries were processed by the system to generate answers. The performance evaluation of the question answering is shown in Table 6.2.

Table 6.2. Comparison of question answering performance with and without information provided by chart recognition

Type of question	No. of questions	Correctness	
		Without chart information	With chart information
Factoid	50	38 (76%)	45 (92%)
Query-like	50	-	42 (84%)

From the table, it can be seen that the original textual information is insufficient to handle all the factoid questions related to the charts, as expected. With the NL description added, the performance of the question answering system is improved. The query-like questions are handled equally well, with most of the errors due to the failure to parse a sentence into the correct query.

Chapter 7

Generation of Ground Truthed Dataset

The construction of ground truth and performance evaluation has been recognized as an important factor in advancing research in various fields. George Nagy addressed the importance of "application-oriented benchmarking" in each research area in document image recognition [57]. Ground truth datasets that are both well established and publicly accessible are needed to evaluate and to compare the performance of different image recognition and analysis systems. As research on scientific chart recognition and understanding is a relatively young field, there is no well established public dataset with ground truth that is specifically established for the purpose of evaluating chart recognition systems. With the creation of such a dataset, more attention can be drawn from researchers that might be interested in this relatively new area. The dataset should have the following desired features:

- The dataset should contain sufficient number of chart images, in order to facilitate the testing of the efficiency of a system working on a large scale of images.

- The dataset should include both synthetic images and real-life images. Synthetic images are easier to generate to a large scale, while the real-life images are essential to present real-life effects.
- The chart images in the dataset should cover most commonly used chart types in order to maintain a good variety set of the test images.
- The ground truth data should contain details in multiple aspects, so that the dataset can be used to evaluate recognition systems in various prospects of real life encounters.

7.1. Automatic Ground Truthing versus Semi-automatic Ground Truthing

Based on whether human effort is involved, ground truthing tools mainly fall into two categories: automatic and semi-automatic. Most ground truthing systems reported in the literature are semi-automatic. A semi-automatic ground truthing system either involves human correction following automatic processing steps [58-60], or consists of a mixture of auto-processing steps and human inputs [61, 62]. The semi-automatic approach has certain advantages. First of all, a semi-automatic system can extract ground truth data from a wide range of images with complex layout and varying types, as long as the basic processing functions are available to support the extraction. Secondly, a semi-automatic system is good to extract ground truth from real life images, as human inputs or corrections can overcome the error raised from noise and distortions. As a result, the images in a dataset ground truthed with a semi-automatic system have a good variety of types and contain real life noise and distortions. On the other hand, there are also drawbacks of the semi-automatic approach. Firstly, the process is not very efficient as it involves human effort during the process. As a result, it will be either very time consuming or very labour intensive to form a large data collection. Secondly, human verification and correction at low-level still leave certain chance to introduce inaccurate ground truth data. For example, the start point and end point of the vectorized lines may be a few pixels from the true end-points. Although the error is insignificant for most of the time, it is undesired as what we are looking for is ground “truth”.

On the other hand, there are also fully automatic ground truthing systems, such as [63, 64]. An automatic ground truthing system usually makes the use of existing document/graphics generation packages to create datasets and captures intermediate results as the ground truth. Through literature review, we found out that automatic ground truthing is used when the targeted ground truth data only require high level details, such as the number of cells in a table or the font type of the text string. If low-level details are to be included, such as the boundary lines of a cell in a table or the bounding box of a character, then the semi-automatic approach seems to be a better choice unless such details are directly available. A typical automatic ground truthing system is computationally efficient and thus has been good for generation of dataset with a large scale. Furthermore, the ground truth data obtained through automatic process are highly accurate. But the automatic approach also has some drawbacks. Firstly, the amount of ground truth data that can be automatically obtained is restricted, and the low-level details may not be accessible. Secondly, if the system relies on a certain graphics generation package, then the dataset created only reflects the characteristics of that package, resulting in lack of variety in the dataset created. Last but not least, the system produces synthetic images which do not contain real-life noises and effects. To alleviate this drawback, a degradation module such as [65] is needed to improve the situation by introducing deformations, distortions and noise to the images produced.

Based on the above observations, we adopt both approaches for generating the public ground truthed chart image dataset. While the automatic approach is used to generate synthetic images with ground truth, the semi-automatic approach is used for extracting ground truth from real-life images. In the following sections, we are going to summarize the two systems developed by us for creating ground truthed chart image dataset. The final dataset will also be described.

7.2. Ground Truth of Scientific Chart Images

The ground truth data for scientific chart image has been in their existence on four levels: pixel level, vector level, text level and chart level. The term "level" is used here to indicate different information granularity of the ground truth data. The order of granularity among various kinds of information in the chart image is shown as the following hierarchy:

Pixel < vector and text < chart component

In the following subsections, we will discuss their significance, essential attributes and the availability of ground truth data at each of the four levels.

7.2.1 Pixel level ground truth

Pixel level ground truth is useful especially for the evaluation of graphics recognition system. It can be used to evaluate the processing capability (robustness) of image analysis algorithms [66]. The ground truth is basically the original clean image, and the actual image for testing is the degraded image. For synthetic images, a clean version is available and the pixel values can be used as pixel level ground truth data. However, the images collected from web or scanned in already contain noise and distortions, thus the original pixel values are not available. As the availability of ground truth for this level is not guaranteed, our system will not include it in the final output, though the image used for ground truth generation is still included.

7.2.2 Vector level ground truth

Vector level ground truth is the line information in the images, or more precisely the attribute values of the straight line segments and arcs that form the lines. The essential attributes of the straight line segments and arcs are the endpoints and the line width. So far as the arcs are concerned, the position of the center and the radius are also needed. Using these attribute values, the performance of vectorization algorithms can then be assessed. For both synthetic and real chart images, the vector level ground truth data can be obtained. During image synthesis, the vector information used for drawing the image is stored directly. When a real-life image is processed, fully automatic extraction of line information is possible using vectorization algorithms. However, human effort is needed here to manually correct the results to produce the ground truth. Since higher level symbols (in our case the chart components) are constructed from lines and arcs, vector level ground truth data not only serve as a standard to evaluate line detection algorithms, but also help to generate higher level ground truth data.

7.2.3 Text level ground truth

We have adopted the traditional representation of text level ground truth data, which consists of text zoning information and the electronic text content. A text zone is represented by a rectangular box, and it is also an indication of the text location. In our system, we treat each phrase as a whole block and locate the bounding box for it. There are two reasons for doing so. Firstly, human effort can be reduced by avoiding specifying the bounding boxes for each individual word during ground truthing. Secondly, the assignment of logical role to a text block is only meaningful at the phrase level. Take the chart title as an example. A typical chart title may contain multiple words and the group of words has only one logical role in the chart image. If the text zones are to be used to measure the segmentation capability of OCR systems, then one of the obvious adjustment here is to apply an automatic text segmentation algorithm (such as the x-y cut algorithm) to further locate the bounding box for each word in a text block.

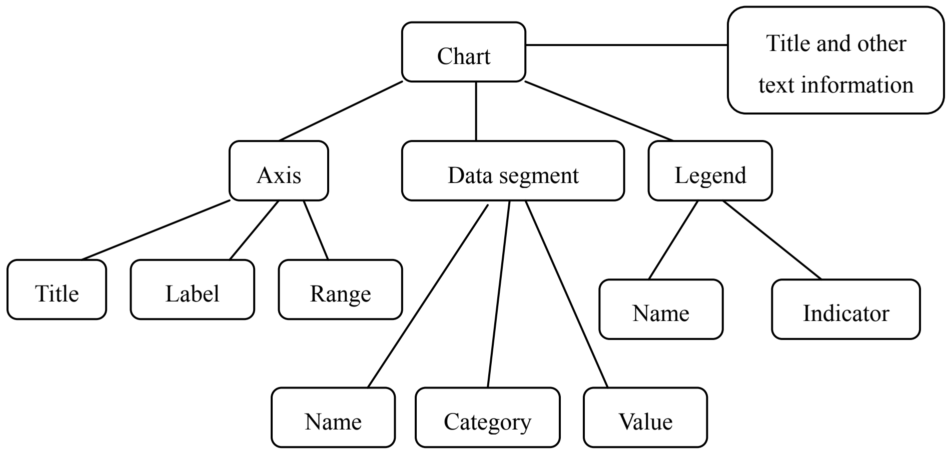


Figure 7.1. Essential components in a chart image

7.2.4 Chart level ground truth

A chart image has various components and features, but only a subset of them are essential for the revealing of the semantic information in the chart. They are summarized in Figure 7.1.

The title of a chart may not always be available. If it is available, then it does provide contextual information about the chart, together with other textual information in the same chart. The axes only exist for some chart types, such as bar chart or line chart etc. Besides the position of each axis, axis title, labels along the axis and the axis range are also important for capturing complete axis information. If there are more than one data series presented in the chart, then the legend information is used to distinguish among data series. Legend information includes legend name and legend indicator. Data segments represent data value in different forms for different chart types. For example, there are bars in bar chart, pies in pie chart etc. So in the ground truth data, we not only present the name and value of each data segment, but also specify its form. In case there are more than one data series, the category each data segment belongs to is also recored.

7.3. The Semi-automatic Approach

To obtain ground truth from real-life images, a semi-automatic system has been developed. The system does most of the job automatically while user interactions are required during the ground truth generation process. Since a typical chart image contains both text and graphics, we generate ground truth data for both kinds of information so that they can be used for evaluating both text recognition and graphics recognition. Furthermore, since the ultimate goal of chart image recognition is to understand the logical role of the chart components and to extract the data values carried by the chart, ground truth for chart components and data values are also generated.

Figure 7.2 shows the major hierarchical tasks in the proposed system. Pre-processing is performed to the input image first, including text/graphics separation, edge detection, vectorization and text grouping. Then ground truth data from text level and vector level are generated in two different modules. In the next step, data from text level and vector level are combined to generate chart level ground truth. Dashed arrows in the figure indicate that user interaction is required in the modules. Finally, the ground truth data generated for one chart image are stored into an XML document.

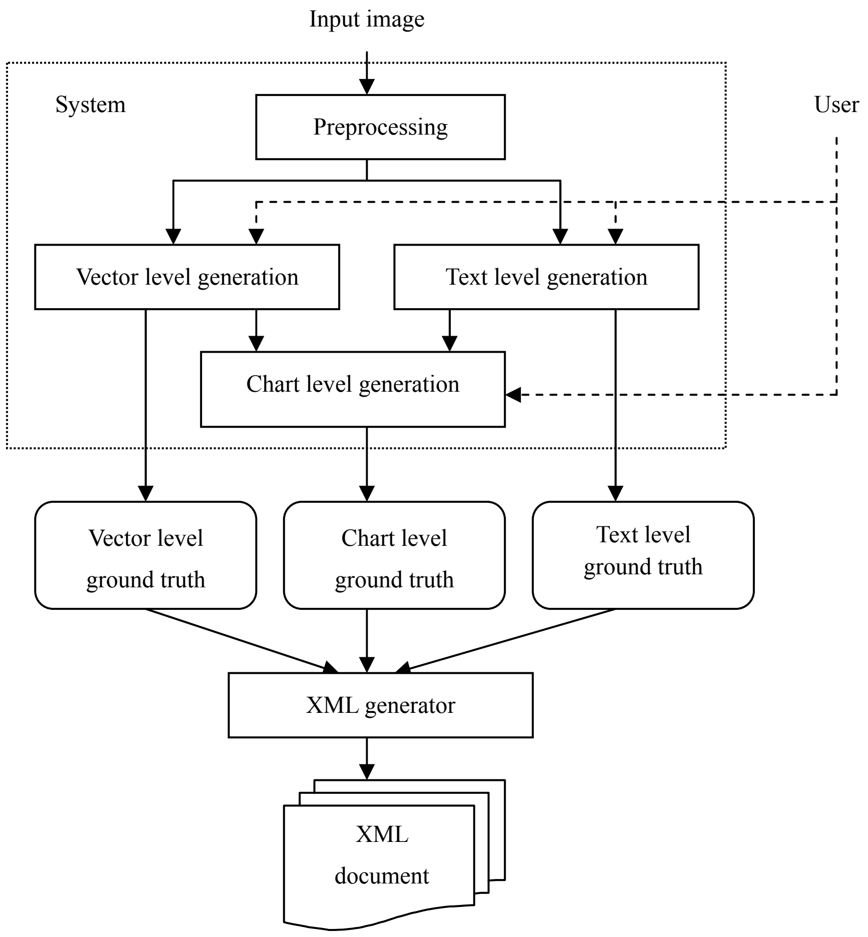


Figure 7.2. Major hierarchical tasks in a semi-automatic system

7.3.1 System preprocessing

There are several steps in the preprocessing stage:

- Text/graphics separation. Textual information and graphical information are separated in this step using connected component filtering. A series of thresholds are applied to differentiate text components from graphical components. Most of the text components can be separated from graphics successfully. Characters touching graphics cannot be separated in this case, but the problem can be partially solved later by finding user specified text regions. The text components are binarized and stored in a separate text image, which will be passed as input to text blocks construction step. The graphical components are kept in the original image and will be further analyzed to find the line information.

- Edge detection. Since all vectorization methods require a binary image, an edge map is constructed in this step. To effectively identify the edges, the system needs to be given the maximum allowed edge thickness. Edge detection is done by calculating intensity differential among neighboring pixels, followed by gap filling between left edge and right edge.
- Text block construction. Text block construction is based on the method described in [67] to the text components found previously. The system automatically calculates the text block candidates, after which the user can then refine the result by deciding whether to further split a block or merge some blocks.
- Vectorization. The purpose of vectorization is to detect line information, or more precisely information of the straight lines and arcs. Here we use the DSCC based vectorization methods introduced in Chapter 4 to construct straight lines and arcs respectively. The results are stored in vector form. The vector of straight line contains starting point, ending point and line width. The content of the vector of arc is similar, except that the arc centre is also stored.

- Locating feature points. The feature points include endpoints of straight line segments and arcs, touching points of two lines, cross points and corner points, as illustrated in Figure 7.3(a) to (d). The point sets are calculated by the system automatically, and will be used as a basis for adjusting the user specified points. If there is more than one feature point near the user selected location, then the nearest feature point will be chosen as the final point.

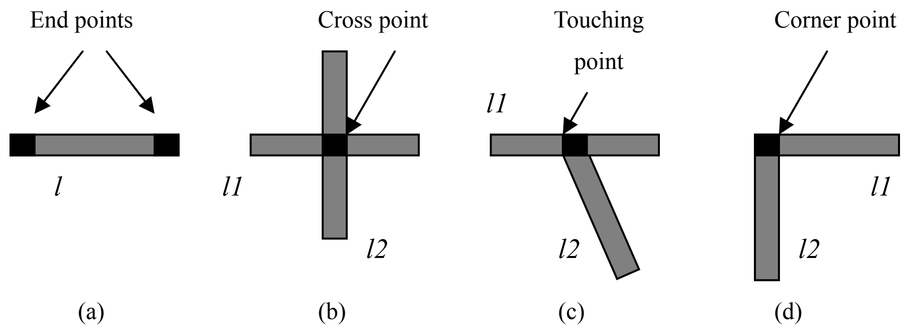


Figure 7.3(a)-(d). Illustration of the set of feature points.

7.3.2 Vector level ground truth generation

As the vectors of straight lines and arcs are already available, the task of the user is to verify the correctness and accuracy of the vectorization result. The vectors are drawn on the original image and the user can manually adjust the endpoints of a vector if it is too long, too short or outside the original line. After the user verifies and corrects all the vectors, the information stored in the vectors is then saved as the vector level ground truth data. Furthermore, the vector information is also passed on for chart level ground truth generation.

7.3.3 Text level ground truth generation

User adjustment can be performed similarly to text block candidates automatically identified by the system, by refining the boundaries of each candidate. If the text block candidate contains multiple text blocks, the user can manually specify the cutting point. On the other hand, if several text block candidates belong to the same text block, the user can also group them and form a larger block. For electronic text, current system relies on manual input to guarantee the correctness of the text content. The main reason is that currently we have not built an OCR module in our system. However, this is not an expensive approach since usually the amount of text in a chart image is not large. To further improve the efficiency of our system, we can add the OCR module into the system, as part of the future work.

After all text blocks are fixed and their contents are input, the information is the saved as text level ground truth. The information will also be used when chart level ground truth is generated.

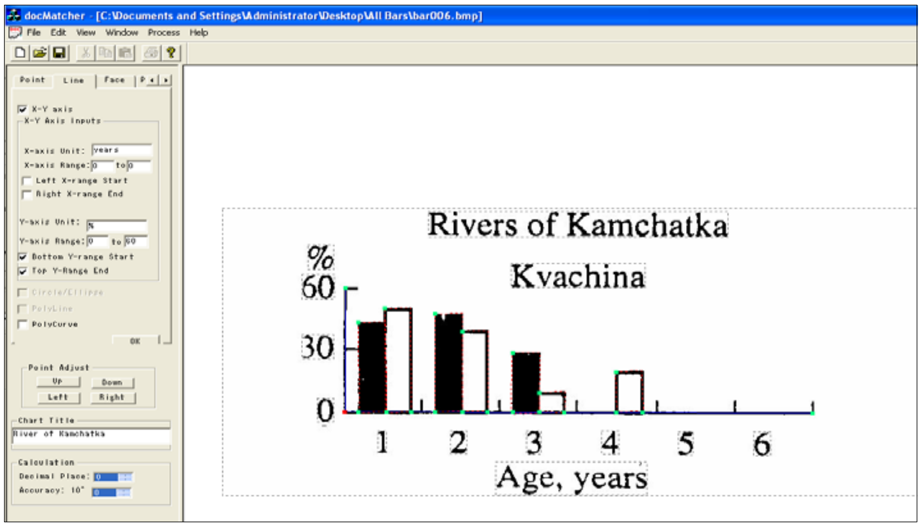
7.3.4 Chart level ground truth generation

As we mentioned before, chart level ground truth contains the information of a set of essential chart components. Obtaining such information may not be straightforward. Unless the process of chart image generation can be reversed, then the system has to rely on heuristic rules and user interaction to identify the exact position and attributes of the chart components and obtain their values.

To find out the graphical chart components, the user just needs to indicate the rough position of the critical points for each component, and then the system will automatically find the precise position by finding the best feature point within a predefined range. If the feature point provided by the system is wrong, the user can still manually adjust the position of the point.

To find the textual chart components, the user needs to manually specify the correspondence between a text block and its logical role. It is difficult to automate this step, because the text/graphics correspondence is still being studied and no general solution is found yet.

To obtain the data values, one way is to generate chart images based on synthetic data. In this way, the original data values are available for comparison with the extracted data values. For scanned chart images, the original data values may not be available, thus they need to be calculated based on the information available in the images. There are two cases: if the data values exist in the image in the format of text objects, then the user can input them into the system. If the data values are not directly given, then the system will calculate them and the user needs to verify and make any corrections if there is a large error.



docMatcher - [C:\Documents and Settings\Administrator\Desktop\All Bars\bar006.bmp]

[✓] X-Y axis

X-Y Axis Inputs

X-axis Unit: years

X-axis Range: 0 to 0

[ ] Left X-range Start

[ ] Right X-range End

Y-axis Unit: %

Y-axis Range: 0 to 60

[✓] Bottom Y-range Start

[✓] Top Y-Range End

[ ] Circle/Ellipse

[ ] PolyLine

[ ] PolyCurve

Chart Title: River of Kamchatka

Calculation

Decimal Place: 0

Accuracy: 10^ 0

Figure 7.4. A snapshot of the system interface.

Figure 7.4. shows a snapshot of the system interface. The input image is placed at the center of the image panel, surrounded by dash boundaries. The tool panels are on the left of the window. The green dots in the input image indicate the feature points used to specify the x-y axis and the bar components. The red dot is the origin of the coordinate system. The snapshot also shows the bounding boxes of all text blocks. The content and logic role of each text block can be specified in the tool panel.

7.4. The Automatic Approach

An automatic system has been developed to synthesize chart images and generate ground truth data during the process of image synthesis. The major steps in the automatic system are shown in Figure 7.5.

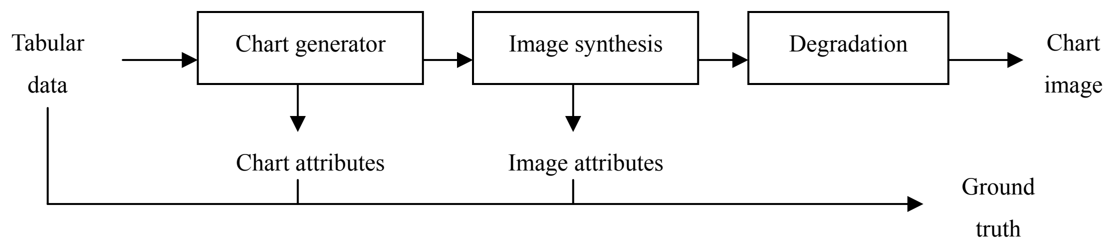


Figure 7.5. Automatic ground truthing scheme

7.4.1 Chart Generation

Randomly generated tabula data (label plus value) is used as the basis for chart generation. The data is passed into a chart generator to create a chart of a certain type chosen by the user. The current version of the system generates four common types of chart: 2D bar chart, 3D bar chart, 2D pie chart and 3D pie chart. Each chart type consists of a set of essential components, which can be further decomposed into text entities and regular graphical entities. Each graphical entity is represented as a combination of graphical primitives following geometric constraints. To draw a generated chart as an image, the drawing functions in the Windows GDI+ library are called to draw the graphical primitives such as line segments and arcs. The thickness of a line or an arc can be specified by user. GDI+ library also provides functions to render text strings in an image and estimate the bounding box of each text string. Figure 7.6 illustrates how a chart is decomposed and converted into an image. The existence of axis is type-dependent. If a chart type does not require an axis, such as a pie chart, then the system does not include it. As drawing 3D charts is more complicated than drawing 2D charts, in our approach the following steps are carried out:

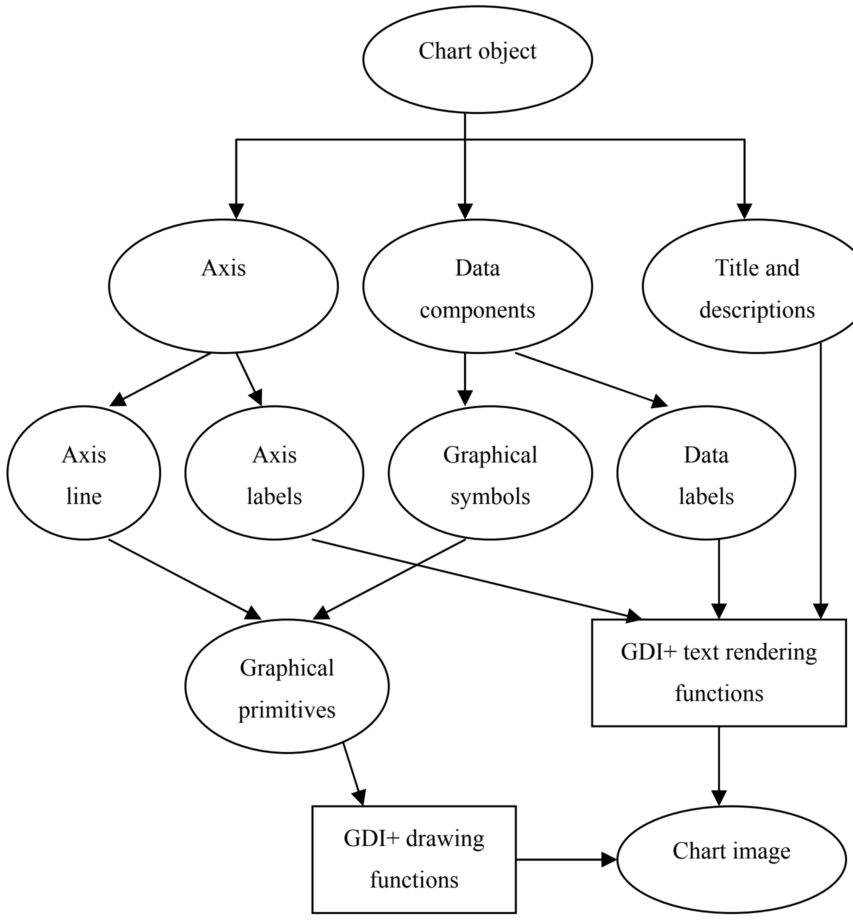


Figure 7.6. Drawing chart image using GDI+ functions

Step 1: Draw a 2D version of the chart.

Step 2: Construct 3D chart based on the 2D version, using geometric transformations. To draw a 3D bar chart from its 2D version, translation is applied on the point sets to create the cubical effects. To draw a 3D pie chart from the 2D version, perspective distortion and translation are both applied to covert a circular arc into elliptic arc.

7.4.2 The Degradation Module

For each chart generated, a clean synthetic image is created through rasterization. The degradation module is applied on the clean image to add less-than-ideal effects to simulate real-life image quality. Our degradation module is based on the degradation model proposed by Baird [65]. The original model listed 10 parameters. Considering the problem domain we are dealing with, we only adopt a subset of them. As listed in Table 7.1, the parameters included in our degradation module are used to perform the following tasks: rotation (skew angle), shearing, edge distortion, Gaussian noise and motion blur.

Table 7.1. Overview of the parameters in the degradation module

Parameter	Data Type	Range	Meaning
β	Real	(-π, π)	Skew angle, measured in degrees
λ	Real	[-1, 1]	Horizontal shearing factor
L	Integer	[0, 10]	Degree of edge distortion
v	Integer	[0, 5]	Radius of motion blur
θ	Real	(-π, π)	Angle of motion blur, measured in degrees
σ	Real	[0, 50]	Degree of Gaussian noise

1. **Rotation.** Rotation is a deformation operation. The whole chart is rotated to add a skew angle to the image. For each pixel *(x, y)* in the image plane, if the skew angle is β, then the new pixel location *(x', y')* in vector form can be calculated as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7.1}$$

2. **Shearing.** Shearing is a common deformation type that changes the shape of a geometric object. The shearing process requires one parameter, the shearing factor *λ* = *cot α*, and a pixel *(x, y)* will be mapped to the new location:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \cot\alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{7.2}$$

3. **Edge distortion.** In real-life, distortions are very likely to occur along the edges of lines or regions, mainly due to the reproduction process such as scanning or faxing etc. In order to simulate edge distortion, we adopt a convolution method based on [68], with the modification that besides pixel-adding in the original method, pixel-reduction is also performed. Here pixel-adding means a pixel change from fore-ground color to background color and pixel-reduction means the opposite. A parameter *L* here is used to controls the degree of edge distortion.

4. **Motion blur.** Motion blur most often occurs during a camera-based capturing process. The modelling of motion blur is based on [69]. Let *f(x, y)* be the input image, and *H(x, y)* be the blurring function. With two parameters *v* = the level of motion blur and *θ* = the angle of the motion blur, the blurred image *g(x, y)* is generated as:

$$g(x,y) = \sum_{n=1}^{width} \sum_{m=1}^{height} f(x-n, y-m) H(n,m) \tag{7.3}$$

where:

$$H(x,y) = \begin{cases} \dfrac{1}{2v+1}, if\ 0 \le x \le (2v+1)\cos\theta\ and\ 0 \le y \le (2v+1)\sin\theta \\ 0, otherwise \end{cases} \tag{7.4}$$

5. **Gaussian noise.** Gaussian noise models the thermal noise in electronic imaging systems. To generate Gaussian noise, the crucial step is to obtain a Gaussian (normal) distribution, a random variant with its probability density function as:

$$p(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{7.5}$$

Here we use an algorithm called *ran0* [70] to realize the polar method [71] for obtaining a standard normal variable *X0*. To add Gaussian noise, each pixel G_{ij} in the original image is added with a value *σX0*. *σ* is a parameter that controls the level of noise.

7.4.3 Generating Ground Truth Data

The initial tabular data become the semantic level ground truth. The vector information of the lines recorded during drawing process becomes the vector level ground truth. The text strings and their bounding boxes form the text level ground truth. The chart entities created during chart generation are recorded to form another part of the chart level ground truth. An additional part of the ground truth includes in this synthesis process the parameters used by the degradation module. This part of information, however, was not obtainable using the semi-automatic approach.

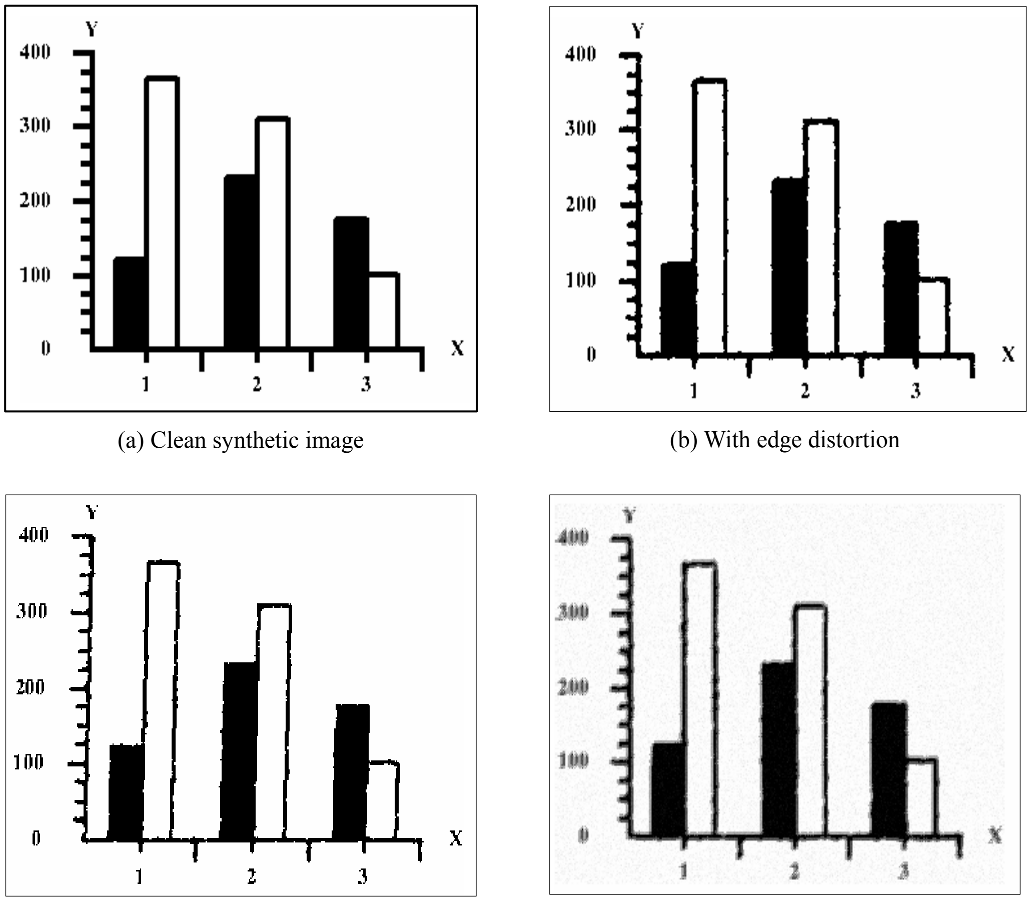
7.5. The Ground Truth Dataset

7.5.1 Dataset Description

The final dataset contains two subsets from the two works we have done: a collection of real-life images and a collection of synthetic images. For the real-life collection, 200 images were collected and the corresponding ground truth data were extracted using the semi-automatic system. In the synthetic collection produced using the automatic system, 400 clean images were created for each of the four chart types. For a clean image, one of the eight different combinations of degradation effects was added to create a noisy version. Examples of synthetic image and degraded images are shown in Figure 7.7. Thus the synthetic collection contains 1600 clean chart images and 1600 degraded images, with ground truth data in XML format. By combining the two collections, the final dataset is formed, with a total of 3400 chart images and their corresponding ground truth data. Some statistics about the complete dataset are shown in Table 7.2.

Table 7.2. The final data set

	Real			Synthetic		
Char Type	Scanned	Downloaded	Total	Clean	Noisy	Total
Bar chart	61	19	80	800	800	1600
Pie chart	-	60	60	800	800	1600
Line chart	14	46	60	-	-	-
Total	75	125	200	1600	1600	3200



(a) Clean synthetic image

(b) With edge distortion

(c) With horizontal shearing and edge distortion

(d) With edge distortion, Motion blur and Gaussian noise

Figure 7.7. Sample synthetic image and degradation effects

7.5.2 Discussions

Using the semi-automatic system, since the original input image is noisy, the lines in the image have distorted edges, which may cause trouble in finding the correct line width. In the vectorization step, the system calculates the width of a line by taking the average width of all small segments in the line. To guarantee the accuracy of the line width detected, the user needs to manually verify it and adjust the line width to its most suitable value if necessary.

We must point out that there are some special characters in the XML specification that cannot be displayed properly, such as “&” and “<” and “>”. Thus to guarantee the completeness of information in the text level ground truth data, we changed these special characters to “and”, “less_than” and “greater_than”. But then the character set does not match the original set, so we also include a plain text version of the ground truth so that all characters are available, including the special characters.

One of our assumptions is that the lines in the chart components are solid lines. Although in most cases the assumption is valid, there are some exceptions. For example, dash line may be used to connect the data points in a line chart. And sometimes the axes also appear as dashed lines. Thus to overcome this weakness, a dashed line detection algorithm should be implemented and added to the vectorization process.

In automatic ground truth generation, there is a trade off between the complexity of the implementation and the level of details to be kept in the ground truth data. If only tabular data are required, then the generation process is very simple: use a graphical package to create electronic charts and then convert it into image format. However, the ground truth will only be useful when evaluating a chart interpretation system that returns tabular data. Besides the tabular data itself, other metrics are also relevant and important to the performance evaluation of a system that deals with chart images, including the accuracy of graphical symbol construction, the accuracy of text segmentation and recognition etc. Thus to provide measurement for these metrics, the ground truth should be more enriched to include low-level information about graphical symbols, text bounding boxes and text strings, etc. As mentioned in Section 7.1, the low-level information is not directly obtainable from commercial graphical packages. Thus to obtain such information, we chose to implement our own functions for drawing and recording.

The accuracy of the automatically generated ground truth data is relatively higher than those generated using the semi-automatic system. However, some ground truth data may still be slightly erroneous. More specifically, the bounding box returned by the GDI+ function Graphics.*MeasureString()* does not reflect the true bounding box of a text string, due to the limitation of the way GDI+ computes the width of the text using hinting and anti-aliasing. The bounding box returned by the current implementation is a bit wider than the truth bounding box. The problem may be solved in the new version of the system, using alternative ways of measuring the width of text strings.

The current version of the automatic system only takes the major chart components into consideration, including: chart axes, data components, titles and labels etc. Although these are the essential components for interpreting a chart, there are other important components to be included. For example, legends are very important in a chart with multiple data series. Grid lines may also be included because they are very often used in real-life charts. Besides, the random text generation unit in the current system only generates very simple text strings such as numeric strings etc. Random alphabetic labels or even sentence based descriptions should be generated. The points mentioned above will be covered as our future work.

7.5.3 Distribution of the Dataset

The dataset is distributed online. A website has been built up to store the dataset with ground truth data under the main website of the Center for Information Mining and Extraction (CHIME), which is publicly accessible for free through the URL:

http://www.comp.nus.edu.sg/labs/chime/da/chart_dataset/dataset.html

The website provides a brief introduction of the dataset, including the number of images in each individual portion of the dataset, the links to the actual images and ground truth data, and relevant papers and other external materials. Figure 7.8 shows the main composition of the website distributing the dataset. We believe that by making this dataset freely accessible to the public, research activities in this area can be promoted in the near future.

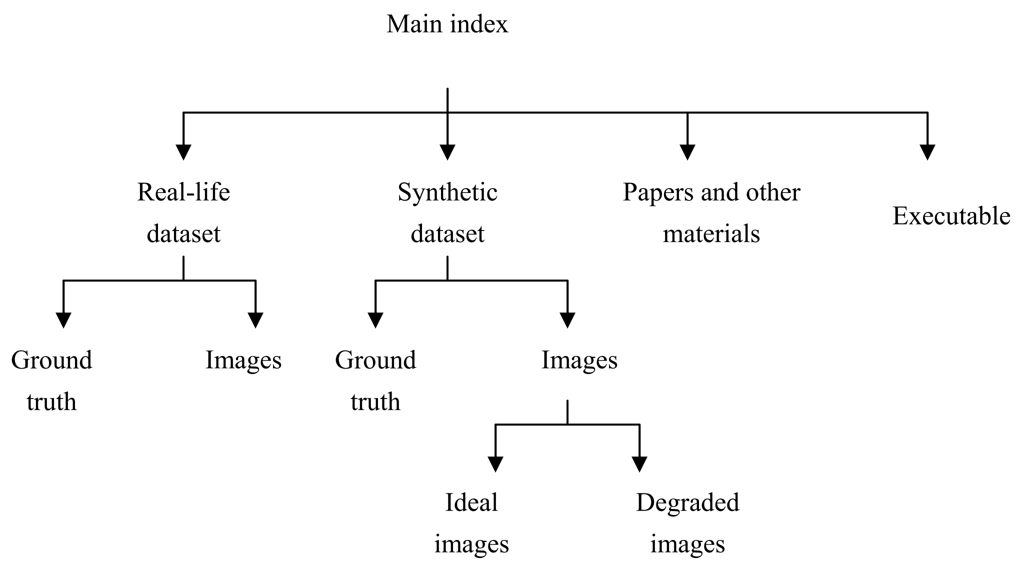


Figure 7.8. Structure of the website containing the dataset

Chapter 8

Conclusion and Future Directions

8.1 Summary of Contributions

In this dissertation, we have investigated a relatively new research problem of scientific chart image recognition and interpretation. We summarized important design principles in chart generation, which are fundamental guidelines for designing a general paradigm for recognizing and interpreting chart images. We further examined detailed problems and solutions from three aspects, namely, the chart classification and recognition, chart interpretation and its applications, and ground truth dataset generation. Corresponding to the objectives and targets listed in Chapter 1, our contributions in these three aspects are summarized as follows.

- **Chart classification and recognition:** Both graphics recognition and text recognition have been investigated. For graphics recognition, we proposed a vectorization method for extracting graphical primitives based on edge information. The graphical primitives are straight lines, circular arcs and elliptic arcs. To extract chart components, we proposed a novel approach for coordinate line detection in chart images based on both textual and graphical information. The method outperforms existing methods in reducing false positives. We applied domain knowledge to build hierarchical chart models for both chart type recognition and chart component recognition. Chart model for four commonly used chart types were specified: 2D bar chart, 2D pie chart, 3D pie chart and line chart. We also investigated machine learning based approach in chart classification using graphical symbols. The work on vectorization has been published in [76]. The work on model based chart recognition was reported in [72, 75]. The work on learning based chart classification was published in [78].

- **Chart interpretation:** We proposed a machine learning based method for associating text blocks and graphical symbols in a chart image for capturing a chart's structural information and complete semantic information. Before our work, there was no existing method to achieve the association of text and graphics in images. We studied the interpretation of different chart types and the extraction of tabular data using the combined textual and graphical information. The result of chart interpretation is stored in both XML format and natural language format for different applications to use. We further explored how chart interpretation can be applied on existing techniques. Two techniques were investigated: optical character recognition (OCR) and question answering (QA). Experiments show that our system provides enriched information to these techniques and improves their performance. The work of chart interpretation and its applications was published in [73, 75].
- **Generation of ground truth dataset:** We defined multi-level ground truth data in the context of chart image recognition and interpretation. The ground truth format at each level was specified. We proposed a semi-automatic ground truthing system for

extraction of ground truth data from real-life chart images. We also designed and implemented an automatic system for synthesizing a large number of chart images and generating corresponding ground truth data. The generated dataset with ground truth data are made publicly available for other researchers. The semi-automatic ground truthing system was reported in [74]. The automatic system for chart synthesis and ground truthing generation was reported in [77].

Comparing to the previous works, especially Zhou's work [3], this dissertation achieves the following breakthroughs and improvements.

- **Broadening chart types to be recognized and interpreted.** In Zhou's work, most methods are developed specifically for bar charts only, except for the coordinate line detection that works for all 2D and 3D chart types with coordinate lines. In this dissertation, more commonly used chart types are handled, including bar chart, pie chart and line chart. Furthermore, the features, methods and domain knowledge are designed to be type independent and general enough to handle multiple chart types.
- **Enriching the types of text labels in the charts to be handled.** In Zhou's work, only axis labels, axis titles and figure titles are defined for the text blocks in the charts. A new method is introduced here to train the system to learn to assign text labels based on a new set of features that tightly relates text blocks and graphical symbols with more complete label classes defined. Hence our works are much more general.
- **Enriching knowledge sources for chart recognition as well as interpretation.** When the number of chart types increases, the domain knowledge that specifies the chart

models is expanded accordingly. This includes expanding the set of graphical symbols to be constructed, the set of text labels to specified, and various spatial and syntactic constraints to be added. The enriched information is introduced into the chart models defined.

- **Tighter integration of text and graphics information for complete interpretation of an input chart.** In Zhou's work, the integration of text and graphics is limited and the two types of information are almost processed independently. In this dissertation, the integration is achieved on both syntactic level and semantic level, to recognize the logical roles of the text blocks, and to perform complete interpretation of the chart content. Such integration also helps with detection of graphical entities such as the coordinate lines.

8.2 Limitations of the Current System

Within the time frame given, major aspects in chart image recognition and interpretation were covered in the dissertation. However the current system still has a number of limitations. This section summarizes all the limitations of the system, although some of them were already mentioned in the previous chapters.

- **Inability to handle dashed lines or dotted lines.** The vectorization method used at the moment only handles solid lines and curves. Thus graphical symbols consist of dashed lines or dotted lines are not recognized.

- **Inability to handle 3-dimensional charts.** Although 3D pie chart is one of the chart types included in the dataset, it is just a simple case of 3-dimensional charts. When data is plotted in a 3-dimensional chart, the graphical representation is much more complex.
- **Inability to handle charts that do not use shapes or lines to represent data.** Some charts use dot plots, or color-coding to represent data. Such representations are not considered in the current system.
- **Inability to perform more advanced data calculation.** The system performs direct calculation of data values assuming linear scale is adopted by the axis. If data value is plotted based on a non-linear scale, then the value calculated by the system is incorrect.
- **Limited preprocessing steps.** In real life, a scanned image may contain shading effect and noise. The current system does not include shading removal and noise removal in the preprocessing part. Instead it assumes such operations are carried out before the image is input to the system.

8.3 Future Directions

To further improve the system to overcome the limitations mentioned, and to increase the generality and robustness of the system, the following issues can be further explored in the near future.

- **Exploration into more complex graphical symbols and a larger variety of chart types.** To recognize graphical symbols with dashed lines or dotted lines, dashed line detection method should be added into the vectorization process. At the moment, the domain knowledge in the system allows it to recognize and interpret 2D bar chart, 2D line chart, 2D pie chart and 3D pie chart. It does not handle other types of charts such as radar charts, area charts etc. Handling of 3-dimensional charts is also quite limited. By adding corresponding symbols and constraints into the domain knowledge, the system can be expanded to handle new chart types. To handle 3D charts, more advanced graphical symbol construction methods need to be used and geometric constraints in the 3D space are to be applied.
- **More intelligent interpretation for data recovery.** The solution of the problem of non-linear data calculation requires the detection of the scale type of the axis lines, which is again the result of integrating text with graphics. Scale type becomes a new type of text label associated with an axis line, besides axis title, axis label and axis unit etc.
- **More interactive text recognition and graphics recognition.** The association of textual information with graphical information certainly builds up a link between the two kinds of information. However, such linkage is only available after the recognition of each kind of information is done. Can the text blocks and graphical symbols be associated before text recognition and graphical recognition? Can the recognition result of one type of information helps to improve the recognition rate of the other type of

information? These are questions to be answered in the future.

- **Avoiding error propagation.** It is mentioned in several places that error propagation is a significant factor that affects the performance of the steps in the system. For an image of poor quality, vectorization returns imperfect graphical primitives, which further affects the result of graphical symbol construction. If some graphical symbols are not properly constructed or even become missing, both chart recognition and chart interpretation are also affected. Error propagation is the result of the sequential steps in the design of current chart recognition and interpretation paradigm. Using domain knowledge as reinforcement or using a probabilistic model can be expected to reduce such effect.
- **Involvement of more image processing techniques.** As mentioned, the current preprocessing steps are quite simple. If we expect to build a more integrated system, then some more image processing steps need to be added in, such as noise removal, shading detection and correction.

Appendix A

Multiple Instance Learning

The multiple-instance learning model was formalized by Dietterich et al. in 1997 [79]. The authors also developed algorithms for dealing with the drug activity prediction problem using the model. This work was followed by Long and Tan [80] who gave a high-degree polynomial PAC bound for the number of examples needed to learn in the multiple-instance learning model. Then Auer proposed a more efficient algorithm [81], and Blum and Kalai showed that learning from multiple-instance examples is reducible to PAC-learning with two sided noise and to the Statistical Query model [82]. Maron and P’erez proposed a framework called “Diverse Density” for solving multiple instance learning problems in various domains [83-85].

A.1 Motivation and Problem Formulation

One of the drawbacks of applying the supervised learning model is that it is not always possible for a teacher to provide labeled examples for training. Multiple-instance learning provides a new way of modeling the teacher’s weakness. Instead of receiving a set of instances which are labeled positive or negative, the learner receives a set of *bags* that are labeled as positive or negative. Each bag contains many instances. A bag is labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one instance in it which is positive. From a collection of labeled bags, the learner tries to induce a concept that will label individual instances correctly. This problem is harder than even noisy supervised learning since the ratio of negative to positive instances in a positively-labeled bag (the noise ratio) can be arbitrarily high. In the next section, the Diverse Density framework proposed by Maron and P'erez for solving multiple instance learning [83] is summarized for reference purpose.

A.2 The Maximum Diverse Density Algorithm

In this section, a probabilistic measure of Diverse Density is derived. A positive bags is denoted as B_i^+, the jth point in that bag as B_{ij}^+, and the value of the kth feature as B_{ijk}^+. Likewise, B_{ij}^- represents a negative point. Assuming for now that the true concept is a single point t, we can find it by maximizing $\Pr(x = t \mid B_1^+, \dots B_n^+, B_1^-, \dots B_m^-)$ over all points x in feature space. If we use Bayes' rule and an uninformative prior over the concept location, this is equivalent to maximizing the likelihood $\Pr(B_1^+, \dots B_n^+, B_1^-, \dots B_m^- \mid x = t)$. By making the additional assumption that the bags are conditionally independent given the target concept t, the best hypothesis is $\arg\max_x \Pi_i \Pr(B_i^+ \mid x = t) \Pi_i \Pr(B_i^- \mid x = t)$. Using Bayes' rule once more (and again assuming a uniform prior over concept location), this is equivalent to:

$$\arg\max_x \Pi_i \Pr(x = t \mid B_i^+)\Pi_i \Pr(x = t \mid B_i^-) \qquad \text{(A.1)}$$

This is a general definition of maximum Diverse Density, but we need to define the terms in the products to instantiate it. One possibility is a noisy-or model: the probability that not all points missed the target is $\Pr(x = t \mid B_i^+) = \Pr(x = t \mid B_{i1}^+, B_{i2}^+, \ldots) = 1 - \Pi_j (1 - \Pr(x = t \mid B_{ij}^+))$ and likewise $\Pr(x = t \mid B_i^-) = \Pi_j (1 - \Pr(x = t \mid B_{ij}^-))$. The causal probability of an individual instance on a potential target is modeled as related to the distance between them, namely, $\Pr(x = t \mid B_{ij}) = \exp(-\left\| B_{ij} - x \right\|^2)$. Intuitively, if one of the instances in a positive bag is close to *x*, then $\Pr(x = t \mid B_i^+)$ is high. Likewise, if every positive bag has an instance close to *x* and no negative bags are close to *x*, then *x* will have high Diverse Density. Diverse Density at an intersection of *n* bags is exponentially higher than it is at an intersection of *n – 1* bags, yet all it takes is one well placed negative instance to drive the Diverse Density down.

The Euclidean distance metric used to measure "closeness" depends on the features that describe the instances. It is likely that some of the features are irrelevant, or that some should be weighted to be more important than others. Luckily, we can use the same framework to find not only the best location in feature space, but also the best weighting of the features. Once again, we find the best scaling of the individual features by finding the scalings that maximize Diverse Density. The algorithm returns both a location x and a scaling vector s, where $\left\| B_{ij} - x \right\|^2 = \sum_k s_k^2 (B_{ijk} - x_k)^2$.

Note that the assumption that all bags intersect at a single point is not necessary. We can assume more complicated concepts, such as for example a disjunctive concept $t_a \vee t_b$. In this case, we maximize over a pair of locations x_a and x_b, and define $\Pr(x_a = t_a \vee x_b = t_b \mid B_{ij}) = \max_{x_a, x_b}(\Pr(x_a = t_a \mid B_{ij}), \Pr(x_b = t_b \mid B_{ij}))$.

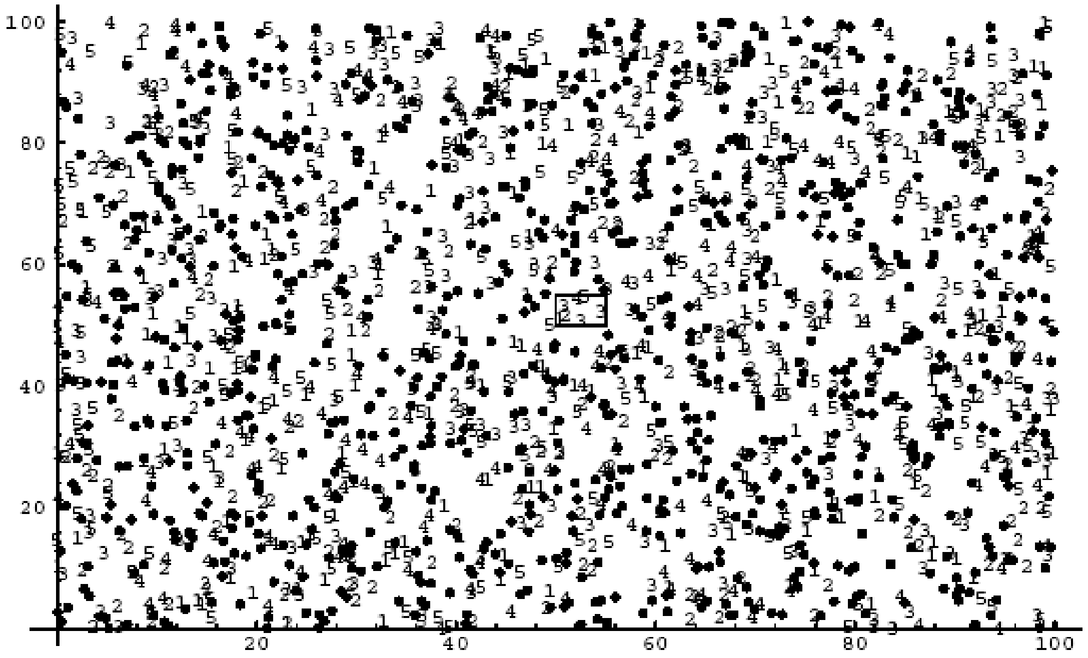
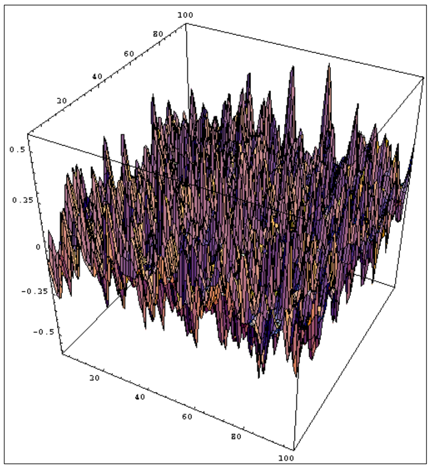


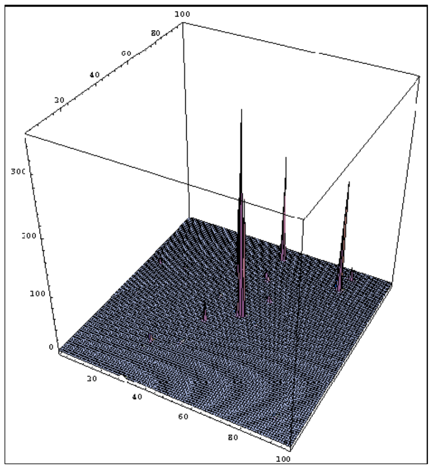
Figure A.1. Negative and positive bags drawn from the same distribution, but labeled according to their intersection with the middle square. Negative instances are dots, positive are numbers. The square contains at least one instance from every positive bag and no negatives.

To further illustrate the concept of Diverse Density, an artificial data set is created. In the data set, there are 5 positive and 5 negative bags, each with 50 instances. Each instance was chosen uniformly at randomly from a [0, 100] ×[0, 100] $\in R^2$ domain, and the concept was a 5×5 square in the middle of the domain. A bag was labeled positive if at least one of its instances fell within the square, and negative if none did, as shown in

Figure A.1. The square in the middle contains at least one instance from every positive bag and no negative instances. This is a difficult data set because both positive and negative bags are drawn from the same distribution. They only differ in a small area of the domain.



(a) Surface using regular density



(b) Surface using Diverse Density

Figure A.2. Density surfaces over the example data of Figure A.1

Using regular density (adding up the contribution of every positive bag and subtracting negative bags; this is roughly what a supervised learning algorithm such as nearest neighbor performs), we can plot the density surface across the domain. Figure A.2(a) shows this surface for the data set in Figure 2, and it is clear that finding the peak (a candidate hypothesis) is difficult. However, when we plot the Diverse Density surface (using the noisy-or model) in Figure A.2(b), it is easy to pick out the global maximum which is within the desired concept. The other major peaks in Figure A.2(b) are the result of a chance concentration of instances from different bags. With a bit more bad luck, one of those peaks could have eclipsed the one in the middle. However, the chance of this decreases as the number of bags (training examples) increases.

One remaining issue is how to find the maximum Diverse Density. In general, we are searching for an arbitrary density landscape and the number of local maxima and size of the search space could prohibit any efficient exploration. In the current situation, gradient ascent with multiple starting points has been used. This has worked out successfully in every test case because we know what starting points to use. The maximum Diverse Density peak is made of contributions from some set of positive points. If we start an ascent from every positive point, one of them is likely to be closest to the maximum, contribute the most to it and have a climb directly to it. While this heuristic is sensible for maximizing with respect to location, maximizing with respect to scaling of feature weights may still lead to local maxima.

Please refer to [83] for more detailed discussions on multiple instance learning and its applications.

Bibliography

[1]. E.R. Tufte. *The visual display of quantitative information*, Cheshire, CT, Graphics Press, 1985.

[2]. B. Twersky, J. Zacks, P. Lee and J. Heiser., "Lines, blobs, crosses and arrows: diagrammatic communication with schematic figures", M. Anderson, P. Cheng and V. Haarslev, editors, *Theory and Application of Diagrams*, *Lecture Notes on Computer Science 1889*, pages: 221-231, Springer-Verlag, 2000.

[3]. Y. Zhou. *Chart Recognition and Interpretation in Document Images*, Ph.D. Dissertation, School of Computing, National University of Singapore, 2003.

[4]. L. O'Gorman and R. Kasturi, *Document Image Analysis*, IEEE Computer Society Press, 1995.

[5]. G. Nagy, "Twenty years of Document Image Analysis in PAMI", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, No. 1, pp. 38-62, 2000.

[6]. R. P. Futrelle, "A Framework For Understanding Graphics In Technical Documents", *Expert Systems in Government Symposium*: IEEE Computer Society, pp. 386-390, 1985.

[7]. R. P. Futrelle, I. A. Kakadiaris, J. Alexander, C. M. Carriero, N. Nikolakis, J. M. Futrelle, "Understanding diagrams in technical documents", *IEEE Computer*, vol. 25, pp. 75-78, 1992.

[8]. R. P. Futrelle and N. Nikolakis, "Efficient Analysis of Complex Diagrams using

Constraint-Based Parsing", *ICDAR-95*, *Intl. Conf. on Document Analysis & Recognition,* Montreal, Canada, pp. 782-790, 1995.

[9]. R. P. Futrelle, "Summarization of Diagrams in Documents". I. Mani & M. Maybury (Eds.), *Advances in Automated Text Summarization,* Cambridge, MA: MIT Press, pp. 403-421, 1999.

[10]. R. P. Futrelle, M. Shao, C. Cieslik and A. E. Grimes, "Extraction, layout analysis and classification of diagrams in PDF documents", *Intl. Conf. Document Analysis & Recognition*. Edinburgh, Scotland, pp. 1007-1014, 2003.

[11]. M. Shao and R. P. Futrelle, "Recognition and Classification of Figures in PDF Documents", W. Liu and J. Lladós (Eds.): Selected papers from *Workshop on Graphics Recognition, GREC 2005, LNCS 3926,* Springer, pp. 231-242, 2006.

[12]. S. Carberry, S. Elzer, N. Green, K. McCoy and D. Chester, "Understanding Information Graphics: A Discourse-Level Problem", *Proceedings of SigDial*, pp. 1-12, 2003.

[13]. S. Elzer, S. Carberry, I. Zukerman, D. Chester, N. Green and S. Demir, "A Probabilistic Framework for Recognizing Intention in Information Graphics", *Proceedings of the Nineteenth International Conference on Artificial Intelligence (IJCAI-05),* 2005.

[14]. S. Carberry, S. Elzer, N. Green, K. McCoy and D. Chester, "Extending Document Summarization to Information Graphics", *Proceedings of the ACL Workshop on Text Summarization*, 2004.

[15]. D. Chester and S. Elzer, “Getting Computers to See Information Graphics so Users Do Not Have to”, *15th International Symposium on Methodologies for Intelligent Systems*, *Lecture Notes on Artificial Intelligence 3488*, pp. 660–668, 2005.

[16]. Y. P. Zhou and C. L. Tan, “Learning-based scientific chart recognition”, *4th IAPR International Workshop on Graphics Recognition*, *GREC2001*, pp. 482-492, 2001.

[17]. Y. P. Zhou and C. L. Tan, “Coordinate systems reconstruction for graphical documents by hough-feature clustering and geometric analysis”, *International Conference on Pattern Recognition, ICPR 2004,* Cambridge, UK, 23-26 Aug 2004.

[18]. Y. P. Zhou and C. L. Tan, “Hough technique for bar charts detection and recognition in document images”, *International Conference on Image Processing, ICIP 2000*, pp. 494-497, 2000.

[19]. N. Yokokura and T. Watanabe, “Layout-Based Approach for extracting constructive elements of bar-charts”, *Graphics recognition: algorithms and systems*, *International Workshop on Graphics Recognition*, *GREC'97*, pp. 163-174, 1997.

[20]. W. S. Cleveland, *The elements of graphing data*, Chapman and Hall, New York, 1994.

[21]. A. Wallgren, B. Wallgren, R. Persson, U. Jorner and J. A. Haaland, *Graphing Statistics & Data,* SAGE Publications, California, USA, 1996.

[22]. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Education, 2002.

[23]. K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch, “Text/Graphics Separation Revisited”, *5th International Workshop on Document Analysis System*, pp. 200-211, 2002.

[24]. W. Liu and D. Dori, “Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, pp. 202-215, 1999.

[25]. P. L. Rosin and G. A.West, “Segmentation of Edges into Lines and Arcs”, *Image and Vision Computing*, vol. 7, no. 2, pp. 109–114, May 1989.

[26]. D. Dori and W. Liu, “Incremental Arc Segmentation Algorithm and Its Evaluation”, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, pp. 424-431, 1998.

[27]. J. Song, F. Su, J. Chen, C. Tai and S. Cai, “Line Net Global Vectorization: an Algorithm and Its Performance Evaluation”, *International Conference on Computer Vision and Pattern Recognition 2000*, pp. 383-388, 2000.

[28]. P. Dosch, G. Masini and K. Tombre, "Improving arc detection in graphics recognition", *Proc. of the 15th Int. Conf. on Pattern Recognition,* vol. 2, pp. 243-246, 2000.

[29]. Y. F. Zheng, C. S. Liu, X. Q. Ding and S. Y. Pan, "A Form Frame-Line Detection Algorithm Based on Directional Single-Connected Chain", *Journal of Software*, vol. 13, pp. 790-796, 2002.

[30]. A. Fitzgibbon, M. Pilu and R. B. Fisher, “Directed Least Square Fitting of Ellipses”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, pp. 476-480, 1999.

[31]. B. Yuan and C. L. Tan, “A Multi-level Component Grouping Algorithm and Its Applications”, *8th International Conference on Document Analysis and Recognition, ICDAR’05*, pp. 1178-1181, 2005.

[32]. K. Tombre and B. Lamiroy, “Graphics recognition - From re-engineering to retrieval”, *Proc. of 7th ICDAR*, Edinburgh, Scotland, UK, pp. 148-155, August 2003.

[33]. R. Kasturi, S. T. Bow, W. El-Masri, Y. Shah, J. R. Gattiker and U. B. Mokate, “A System for Interpretation of Line Drawings”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no. 10, pp. 978-992, Oct. 1990.

[34]. S. H. Joseph and T. P. Pridmore, “Knowledge-Directed Interpretation of Mechanical Engineering Drawings”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 928-940, Sept. 1992.

[35]. B. Lamiroy, L. Najman, R. Ehrard, C. Louis, F. Quelain, N. Rouyer and N. Zeghache, “Scan-to-XML for Vector graphics: an Experimental Setup for Intelligent Browsable Document Generation”, *Proc. 4th IAPR International Workshop on Graphics Recognition,* Kingston, Ontario, Canada, pp. 312-325, Sept. 2001.

[36]. E. Valveny and B. Lamiroy, “Scan-to-XML: Automatic Generation of Browsable Technical Documents”, *Proc. of 16th International Conference on Pattern Recognition*, Quebec, Canada, pp. 188-191, Aug. 2002.

[37]. K. Barnard and D. A. Forsyth, “Learning the semantics of words and pictures”, In *International Conference on Computer Vision*, vol. 2, pp. 408–415, 2001.

[38]. K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. I. Jordan, “Matching Words and Pictures”, *Journal of Machine Learning Research*, vol. 3, pp. 1107-1135, 2003.

[39]. K. Barnard, P. Duygulu, and D. Forsyth, “Recognition as Translating Images into Text”, *Internet Imaging IX, Electronic Imaging*, 2003.

[40]. R. K. Srihari and D. T. Burhans, “Visual semantics: Extracting visual information from text accompanying pictures”, *American Association on Artificial Intelligence, AAAI’94*, Seattle, WA, 1994.

[41]. R. Chopra and R. K. Srihari, “Control Structures for Incorporating Picture-Specific Context in Image Interpretation”, *Proc. of International Joint Conference on Artificial Intelligence, IJCAI’95,* , Montreal, Canada, pp. 50-55.

[42]. Yi Zhang and Tat-Seng Chua, “Detection of Text Captions in Compressed domain Video”, *Proceedings of ACM Multimedia’2000 Workshops (Multimedia Information Retrieval),* California, USA. November, 2000, pp. 201-204.

[43]. J. C. Shim, C. Dorai and R. Bolle, “Automatic Text Extraction from Video for Content-Based Annotation and Retrieval”, *14th International Conference on*

Pattern Recognition, ICPR'98, vol. 1, pp. 618-611, 1998.

[44]. Jain, A. K., Yu, B., “Automatic Text Location in Images and Video Frames”, *Pattern Recognition*, vol. 31, no. 12, pp. 2055-2076, 1998.

[45]. Antonacopoulos, A., Karatzas, D., “An Anthropocentric Approach to Text Extraction from WWW Images”, *4th IAPR Workshop on Document Analysis Systems, DAS2000*, Rio de Janeiro, pp. 515-526, 2000.

[46]. Larkin J. H, Simon H. A, “Why a Diagram is (sometimes) Worth Ten Thousand Words”, *Cognitive Science*, Vol. 11v, no. 1, pp. 65-100, 1987.

[47]. R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[48]. Fujisawa H., Nakano Y. and Kurino K., “Segmentation methods for character recognition: From segmentation to document structure analysis”, *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1079-1091, 1992.

[49]. Jairo Rocha, Theo Pavlidis, “Character Recognition Without Segmentation”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, pp. 903-909, 1995.

[50]. M. Shridhar and A. Badreldin, “High Accuracy Character Recognition using Fourier and Topological Descriptors”, *Pattern Recognition*, vol. 17, pp. 515-524, 1984.

[51]. T. M. Breuel, Layout Analysis based on Text Line Segment Hypotheses, *DLIA’03*, Edinburgh, Scotland, August, 2003.

[52]. D. Ravichandran and E. Hovy, Learning Surface Text Patterns for a Question Answering System, *Proc. of ACL'02, Philadelphia*, pp. 41-47, July 2002.

[53]. H. Cui, M.-Y. Kan and T. S. Chua, Unsupervised Learning of Soft Patterns for Definitional Question Answering, *Proc. of the Thirteenth World Wide Web conference (WWW 2004)*, New York, May 17-22, pp. 90-99, 2004.

[54]. J. Xu, R. M. Weischedel and A. Licuanan, Evaluation of an extraction-based approach to answering definitional questions, *Proc. of Special Interest Group on Information Retrieval, SIGIR '04*, Sheffield, UK, pp. 418-424, 2004.

[55]. L. Gillard, P. Bellot, M. El-Bèze, Evaluations of Question Answering and Evaluations of the Evaluation, *The fifth Int. Conf. on Language Resources and Evaluation, LREC 2006*, Genoa, Italy, 24-26 May 2006.

[56]. R. J. Kate and R. J. Mooney, Using String-Kernels for Learning Semantic Parsers, *Proc. of the Joint 21st Int. Conf. on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-2006)*, Australia, July, pp. 913-920, 2006.

[57]. G. Nagy, "Twenty years of Document Image Analysis in PAMI", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 38-62, 2000.

[58]. R. M. Haralick et al., "UW English document image database I: A database of document images for OCR research", *UW CD-ROM*, 1995.

[59]. R. M. Haralick et al., “UW-II English/Japanese Document Image Database: A Database of Document Images for OCR Research”, http://www.science.uva.nl/research/dlia/datasets/uwash2.html

[60]. I. Phillips, “Users’ reference manual”. CD-ROM, UW-III Document Image Database-III, 1995.

[61]. S. Yacoub, V. Saxena, and S. Sami, “PerfectDoc: A Ground Truthing Environment for Complex Documents”, *8th Int. Conf. on Document Analysis and Recognition*, vol. 1, pp. 452-456, 2005.

[62]. M. Suzuki, S. Suzuki and A. Nomura, “A Ground-Truthed Mathematical Character and Symbol Image Database”, *8th Int. Conf. on Document Analysis and Recognition*, vol. 2, pp. 675-679, 2005.

[63]. Y. Wang, R. M. Haralick, I. T. Phillips, “Automatic Table Ground Truth Generation and a Background-Analysis-Based Table Structure Extraction Method”, *International Conference on Document Analysis and Recognition, ICDAR 2001*, pp. 528-532, 2001.

[64]. G. Zi, D. Doermann, “Document Image Ground Truth Generation from Electronic Text”, *17th International Conference on Pattern Recognition, ICPR’04*, vol. 2, pp. 663-666, 2004.

[65]. H. S. Baird, “Document Image Defect Models”, *Proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition*, Murray Hill, NJ, 1990. Reprinted in

H. S. Baird, H. Bunke and K. Yamamoto, *Structured Document Image Analysis*, Springer-Verlag: New York. pp 546-556.

[66]. W. Liu, D. Dori, “A protocol for performance evaluation of line detection algorithms”, *Machine Vision and Applications*, vol. 9, pp. 240-250, 1997.

[67]. B. Yuan and C. L. Tan, “A Multi-level Component Grouping Algorithm and Its Applications”, *8th International Conference on Document Analysis and Recognition, ICDAR’05*, pp. 1178-1181, 2005.

[68] J. Zhai, W. Y. Liu, D. Dori, and Q. Li, “A Line Drawings Degradation Model for Performance Characterization”, *Proceedings of 7th International Conference on Document Analysis and Recognition*, Edinburgh, Scotland, 2003.

[69] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, Second Edition, Addison-Wesley Publishing Company, 1987.

[70] H. P. William, A. T. Saul, T. V. William and P. F. Brian, *Numerical recipes in C++: The Art of Scientific Computing*, Cambridge University Press, New York, 2002.

[71] S. M. Ross, *A Course in Simulation*, Macmillan Publishing Company, New York, 1990.

[72]. W. Huang, C. L. Tan and W. K. Leow, "Model based chart image recognition", *International Workshop on Graphics Recognition, GREC2003*, 30-31 July, Barcelona, Spain, 2003.

[73]. W. Huang, C. L. Tan and W. K. Leow, "Associating Text and Graphics for Scientific

Chart Understanding", *International Conference on Document Analysis and Recognition 2005*, *ICDAR’05*, Seoul, Korea, 2005.

[74]. L. Yang, W. Huang and C. L. Tan, "Semi-automatic ground truth generation for chart image recognition", *7th IAPR Workshop on Document Analysis Systems, DAS’06*, New Zealand, 13-15 Feb 2006.

[75]. W. Huang and C. L. Tan, "A System for Understanding Imaged Infographics and Its Applications", *ACM Symposium on Document Engineering, DocEng 2007*, 28-31 August, Winnipeg, Canada, 2007.

[76]. R. Liu, W. Huang and C. L. Tan, "Extraction of Vectorized Graphical Information from Scientific Chart Images", *International Conference on Document Analysis and Recognition, ICDAR 2007*, 23-26 Sept, Curitiba, Brazil, 2007.

[77]. W. Huang, C. L. Tan and J. Zhao, "Generating Ground Truthed Dataset of Chart Images: Automatic or Semi-automatic?", *International Workshop of Graphics Recognition*, September, Curitiba, Brazil, 2007.

[78]. W. Huang, S. Zong and C. L. Tan, “Chart image classification using multiple-instance learning”, *IEEE Workshop on Applications of Computer Vision, WACV’07*, Feb 21st-22nd, Austin, Texas, USA, 2007.

[79]. T. G. Dietterich, R. H. Lathrop, and T. Lozano-P´erez, “Solving the Multiple-Instance Problem with Axis-Parallel Rectangles”, *Artificial Intelligence*

Journal, vol. 89, 1997.

[80]. P. M. Long and L. Tan, “PAC-learning axis alligned rectangles with respect to product distributions from multiple-instance examples”, In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[81]. P. Auer, “On Learning from Multi-Instance Examples: Empirical Evaluation of a theoretical Approach”, *NeuroCOLT Technical Report Series*, NC-TR-97-025, March 1997.

[82]. A. Blum and A. Kalai, “A Note on Learning from Multiple-Instance Examples”, *Machine Learning*, Vol. 30, No. 1, January, pp. 23 – 30, 1998.

[83]. O. Maron, T. L. P´erez, “A framework for multiple-instance learning”, *Advances in Neural Information Processing Systems*, vol. 10, pp. 570-576, 1997.

[84]. O. Maron and A. L. Ratan, “Multiple-instance learning for natural scene classification”, *Proc. 15th Int. Conf. on Machine Learning*, pp. 341-349, 1998.

[85]. A. L. Ratan, O. Maron, W. E. L. Grimson and T. L. P´erez, “A framework for learning query concepts in image classification”, *International Conference on Computer Vision and Pattern Recognition*, pp. 423-429, 1999.

[86]. M. M. Syslo, “An efficient cycle vector space algorithm for listing all cycles of a planar graph”, *SIAM Journal on Computing*, vol. 10, no. 4, pp. 797-808, 1981.

[87]. A. Ferreira and M. Fonseca and J. Jorge, “Polygon Detection from a Set of Lines”, *Proceedings of 12 Encontro Portugu es de Computacao Grafica (12th EPCG),* Porto, Portugal, pp. 159-162, 2003.

[88]. W. Browuer, S. Kataria and S. Das, “Segregating and Extracting Overlapping Data Points in Two-dimensional Plots”, *ACM Joint Conference on Digital Libraries*, Pittsburgh, Pennsylvania, USA, pp. 276-279, 2008.

[89]. R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena. “Geometric layout analysis techniques for document image understanding: a review”, *Technical report, IRST*, Trento, Italy, 1998.

[90]. http://en.wikipedia.org/wiki/Chart