Honors Year Project Report

Creation of Synthetic Chart Image Database with Ground Truth

By

Zhao Jiuzhou

Department of Computer Science

School of Computing

National University of Singapore

2005/2006

Honors Year Project Report

Creation of Synthetic Chart Image Database with Ground Truth

By

Zhao Jiuzhou

Department of Computer Science

School of Computing

National University of Singapore

2005/2006

Project No.: H063230

Advisor: A/P Tan Chew Lim

Deliverables:

Report: 1 Volume

Program: 1 CD

Abstract

Large amount of chart images with their ground truth are essential in training and testing chart understanding systems. However, manually retrieving ground truth from real-life chart images is a time consuming and error-prone process. In this project, we propose a procedure that makes ground truthing a fast and reliable task by generating synthesized chart images and automatically obtaining their ground truth. We systematically study the hierarchical structures of chart images, and define 4 levels of ground truth to be recorded. To make synthetic images more helpful in training and testing, we design our own parameterized image degradation model which simulates some commonly seen real-world image defect types. By applying such degradation model, synthesized noise-free images can be further degraded with precise controls. Based on above ideas, we build a software package which can be used to generate chart image ground truth databases with large sizes in a short time, this will greatly facilitate researchers in training and testing their chart understanding systems.

Subject Descriptor:

I.3.3 Picture/Image Generation

I.7.5 Document Capture

H.5.2 User Interfaces

Keywords:

Ground Truth, Scientific Chart Generation, Image Degradation Model

Implementation Software and Hardware:

Intel Pentium 4 CPU 1.60G, Microsoft Windows XP SP2,

Microsoft Visual C++ 6.0, Microsoft Platform SDK 2003

Acknowledgement

I want to express my sincere gratitude to my supervisor A/P Tan Chew Lim. Thanks him for giving me the opportunity to work on this project, as well as his guidance, support and patience with me. It is my great honor and happiness to work under the supervision of his.

I also would like to thank Mr. Huang Wei Hua, for his great help and many valuable suggestions throughout the course of my project.

Thanks my parents' support during my undergraduate study.

Thanks life, I am enjoying it.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Scientific chart has the advantage of representing numerical data in a clear and comprehensive manner; it is widely used in various kinds of documents. Therefore, in recent years scientific chart analysis and recognition draws increasingly more and more attentions to researchers. One critical issue in this research area is the need for large quantity of scientific chart images together with their ground truth values. By *ground truth*, we mean the correct information of every detail in chart images. Take a bar chart for example: a typical bar chart contains axes that are formed by lines, rectangle regions called bars, textual information, etc; the ground truth of a bar chart includes correct information about attributes of the geometric objects such as width and length of lines, values interpreted by different bars and attributes of the text such as content and location of each word, etc (detailed scientific chart ground truth will be discussed in chapter 3, this example is just for a brief illustration). Ground truth has significant usages in training, testing and evaluating document understanding systems/algorithms.

However, there is not much study on how to create ground truth database of scientific chart images. Since ground truth will be used in training and testing, the quantity of ground truth data shall be large in size, the quality in terms of accuracy must be high enough. The usual method of getting a chart image's ground truth is through human labeling, but this is quite labor intensive and proven to be too slow a process to get enough amount of ground truths. What's worse, manually labeling is prone to errors, and the inaccurate ground truth will degrade the recognition system's performance. Hence a fast and reliable method of getting scientific chart images' ground truth is needed. One good choice is to use synthetic chart images and their ground truths.

1.1.1 Synthetic Data vs. Real-life Data

The advantages of using synthetic data include rapid generation process, low cost and high accuracy. However, since the ultimate goal for recognition system is to handle real-life data, one key issue is how useful the synthetic data is. (Dennis and Phillips 99) and (Baird 2000) addressed this issue and pointed out that a system's performance is largely dependent on the size and quality of the training data, no matter whether the data is synthesized or obtained from real life. From this point of view, synthetic data is helpful due to the virtues of its large volume in quantity and high accuracy, which the real-life data is lack of. Synthetic data and real-life data are complementary to each other, using synthetic data along with real-life data is a major trend in document analysis research. In fact, even synthetic data alone has been extensively used by researchers in training and testing recognition systems. Some of the work on using synthetic data alone to successfully improve character recognition systems' performances can be found in (Cano, Perez-Cortes, Arlandis, and Llobet 2002), and some typical work on using only synthetic data for testing and evaluating recognition systems can be found in (Ho, Baird 1995) and (Jenkins 1993).

1.1.2 Synthetic Data Degradation Modeling

Recognition system's ability of handling ideal images well enough does not guarantee the system will also survive when facing corrupted images. Real-life images are often inevitably corrupted from image acquisition, transmission and storage stages. It is necessary to artificially degrade ideal synthetic images (a process called degradation), by simulating the various types of corruptions (such as noise) that images may have in the real world. Quantitatively specified degradation models can lead to more meaningful testing and evaluating strategies, with two obvious benefits listed below: (1) we are able to explicitly compare performances of different document recognition systems/algorithms by using the same degradation level(s); (2) as stated in (Kanungo, Haralick and Phillips 1993), if we evaluate a system/algorithm for certain continuous degradation levels, we can locate the break-down points of the system/algorithm simply by observing the changes of performances, hence the weaknesses of the system/algorithm can be more easily identified. Since it is relatively costly to collect real-life data and retrieve their

ground truth, real-life data sets are always tend to be small in size; this deficiency may cause inadequate image degradation types, so systems which are trained or tested using real-life data sets may not perform well when facing unseen data. Degradation models, however, can help to reduce such bias because they are able to cover a wider range of image defects and thus making recognition systems more robust.

1.2 Project Contribution

In this project, we developed an automatic scientific chart image ground truth generation system and we apply different kinds of degradations to the ideal synthetic chart images to simulate the effect of real-world image degradation types. The contributions of this project can be summarized as follows:

- we build a software package that is able to rapidly generate ideal scientific chart images currently including bar chart (2D, 3D) and pie chart (2D, 3D); at the same time, ground truth associated with each chart image is obtained automatically and with almost 100% accuracy. This software package also implements our own image degradation model; ideal images can be further turned into degraded images with precisely controls. This software will facilitate researchers in training and testing their recognition systems/algorithms by providing them a convenient way of generating large amount of ground truth data with trustworthy accuracy.
- we study commonly seen real-world image degradation types systematically, and use those degradation types to define our own parameterized degradation model, which can be applied to the ideal chart images to produce real-like degraded images.
- Based on our idea and implementation, a synthetic chart image dataset with ground truth can be built easily. The size of the dataset is fully dependent on the users, because we make our system automatically generate chart image ground truth, users can just create as many chart images as they like in a short time. The diversities of different degraded chart images can be decided by users in order to fit them own intentions. The quality of the dataset is also guaranteed: (1) the ground truth has very high accuracy; (2) variations from ideal images which

simulate real-world image degradation types can make recognition systems/algorithms more robust.

1.3 Report Organization

The rest of the report is organized as follows. Chapter 2 gives a review of related work. Chapter 3 talks about our system's overall structure and gives the detailed definition of chart ground truth. Chapter 4 and chapter 5 further elaborate our system by looking inside on the ideal chart image synthesizing and image degradation techniques used. Chapter 6 discusses the evaluation of our system and shows some images that our system produced. Chapter 7 gives a conclusion of the whole project and highlights possible future work.

Chapter 2

Related Work

There are few studies on ground truth generation for synthetic chart image in the literature, since in the past few decades the focal point of document analysis research was on processing textual information. Numerous techniques for generating text ground truth have been proposed, although they can not directly be used in our work, some of their ideas give enlightenments to our research.

All the synthetic data ground truth generation process can be generalized as two steps: synthetic data generation followed by applying degradation model. Synthetic data generation produces ideal synthetic data; degradation model produces variations of the ideal data. While generating synthetic data is almost a matter of implementation issue, the document degradation model has been extensively researched in the past few years.

The first and fundamental study of document degradation model is (Baird 1990). In his paper Baird defined the role of degradation model to be used to simulate the less-than-ideal properties of real-life document images. He listed some commonly seen image degradation types occurred in real life:

Image Degradations	
Image Deformation	Image Corruption
1. Geometric Transformation: • translation • rotation • scaling • stretching 2. Time-dependent Distortion: e.g. due to biomechanics of handwriting	1. Imperfections due to printing, scanning, faxing, etc • noise • blurring • coarsening • thinning and thickening • jittering • perspective distortion

Table 2.1 Image Degradation Types

Baird also enumerated some attributes that a degradation model should have:

- *Parameterization*: it is better to have a degradation model expressed as a function of several parameters, since by this way, the level of degradation can be quantitatively measured and controlled.
- *Completeness*: this means how many types of degradation in real-life images can be simulated by the degradation model. When a system is trained using models with a larger coverage of degradation types, the system will have a higher accuracy estimates to unseen real images; similarly when a system is tested using models with a larger coverage of degradation types, the weaknesses of the system can be more easily spotted.
- *Calibration*: this means to what extent the degradation model can simulate the real-life images. The higher similarity between simulated image and real image, the better performance training or testing by using the synthetic image will achieve.

Baird thus proposed a 10-parameter model for character degradation; each parameter represented the level of a degradation type such as those mentioned in table 1. This model is later used extensively by researchers in testing and evaluating OCR system.

Besides textual information, document images always contain graphical information such as diagrams and figures. (Zhai, Liu, Dori, Li 2003) proposed line drawings degradation model which simulate 4 types of common degradations for engineering drawings: Gaussian noise which simulates the electronic noise in electronic equipment; high frequency noise which always occurs on the edge of lines because the ink would spread on rough paper; hard pencil noise due to the unevenness of paper's surface; motion blur noise due to the movement of the paper during scanning.

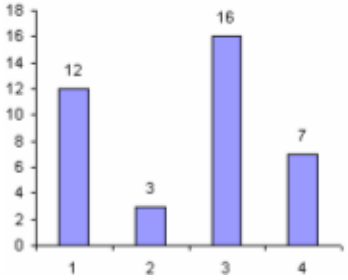
In this project, we will design our own degradation model, and apply it to the generated synthetic chart images. Next chapter discusses the overall structure of our chart ground truth generation system.

Chapter 3

System Overall Structure

3.1 Chart Image Ground Truth

Before illustrating the overall structure of our system, we first talk about the chart image ground truth that our system will produce. As shown in figure 3.1, a chart image can be viewed hierarchically from different levels (or perspectives): at the highest level, since a chart image is used to represent certain numeric data, each component of the chart is assigned some concrete meaning, for instance, every bar in the bar chart represents how much value an entity has. We human beings always interpret chart images from this level, that is, when we see, for instance, a bar chart, what we observe is what entities are being represented, how much value each entity has, what is the trend of the data, etc. Human beings normally will not be interested in information like what is the width in pixel of every bar, how long is the *X* axis; at a lower level, the chart image can be decomposed to components such as bars, pie slices etc. Since each component is certain kind of geometric shape, we can view a chart as an appropriate organizing of shapes. The shape can be further decomposed to geometric primitives like lines and arcs, hence at this level a chart can be viewed as group of lines or arcs. Finally, the lowest level is the pixel information, because fundamentally pixel is the basic unit of every image being displayed, any line or arc is essentially composed by pixels. In fact computers "view" a chart image from this level.



	Level	*Description*
Highest ↑	Semantic level	Actual meaning associated with chart
	Component level	Components in the chart such as bars in bar chart
	Vector level	Lines; Arcs
Lowest	Pixel level	Pixel information about the chart image

Figure 3.1 View a chart image from different levels

The ultimate goal of a chart understanding system is to understand input chart images from the semantic level. Usually the input is at pixel level (e.g. an input chart image in bitmap format), and in other words, chart understanding process is a procedure that changes the "computer view" (pixel level understanding) all the way up to the "human view" (semantic level understanding). When a chart understanding system goes through such a process, information of the input chart on the above 4 levels will be retrieved, since in order to move to a higher level understanding, the system must know information on the lower level(s). Therefore our system provides the chart image ground truth for the above 4 levels.

3.1.1 Pixel Level Ground Truth

Pixel level ground truth records the color information for every pixel in the ideal chart image. Our system records the pixel level ground truth by keeping a copy of the original ideal chart image. The usage of the pixel level ground truth is: some document recognition systems/algorithms perform well when tested using ideal image, but adding little perturbation to the input will reduce the performances tremendously. Testing recognition systems/algorithms using artificially corrupted images, and at the same time comparing the results obtained with those getting from testing by ideal images, will help researchers to identify the pitfalls and strengths of their systems/algorithms.

3.1.2 Vector Level Ground Truth

Vector level ground truth includes attributes about line (straight line) segments and arc segments in a chart image. Lines and arcs are essential primitives for constructing more complicated object. Detecting lines and arcs, also called vectorization, is a necessary step for a chart recognition system to understand higher level components in a chart. Vector level ground truth provides a standard to measure how well the performance a vectorization process can achieve.

A line segment is determined by its two vertices and its width, so we will store the attributes of a line as:

<*starting point'X, starting point'Y, ending point's X, ending point's Y, line width*>;

For arc segment, besides the above attributes, the center of the ellipse (we treat circle as a special case of ellipse) which contains the arc and the length of semi-major and semi minor of that ellipse should also be recorded. So the attributes of an arc is:
<starting point'X, starting point'Y, ending point's X, ending point's Y, center point'X, center point's Y, semi-major length, semi-minor length, line width>;

3.1.3 Component Level Ground Truth

We define a chart image's components as the higher level (than lines and arcs) objects in the chart, which are always comprised by primitives like lines or arcs or their combinations. For instance, the chart in figure 3.1 is formed by a coordinate system: an *X* axis, a *Y* axis, and four rectangles. Every component in a chart image is a certain geometric object. Some geometric objects are regular, such as rectangle, sector; some are irregular, such as *X* axis which is a longer line segment with several shorter line segments (ticks) on it. Chart understanding system should be able to recognize various chart components, since this can help the system decide the information like the chart type etc. In (Huang, Tan, Leow 2003), chart components are used to construct chart models, in this way, different chart models will have distinguishable features since the basic object shapes that constitute every model are different in nature; so when a new chart image is given, the components it contains will be detected and matched with existing model in order to recognize the chart. In fact human beings recognize different chart types by simply identifying chart components: when a person sees a chart with several bars and *X*, *Y* axes, he will say it a bar chart, rather than a pie chart or line chart. Therefore it is necessary to provide chart image's component level ground truth.

3.1.4 Semantic Level Ground Truth

To generate semantic level ground truth, the textual information in the chart image will be used to associate meanings with chart components and the chart itself. Again taking the chart in figure 3.1 as an example, when the text is taken into account, we will know the first bar represents entity 1, which has value 12; second bar represents entity 2, which has value 3, etc; and we will also know the scale on *X* axis is 2, the minimal and maximal values it can represent are 0 and 18 respectively. By providing semantic level ground

truth, we can evaluate how well a chart understanding system really understands chart images. Following the convention in OCR research, we use a bounding box which bounds the text to represent the location of text, and content of text is also recorded.

3.2 System Illustration

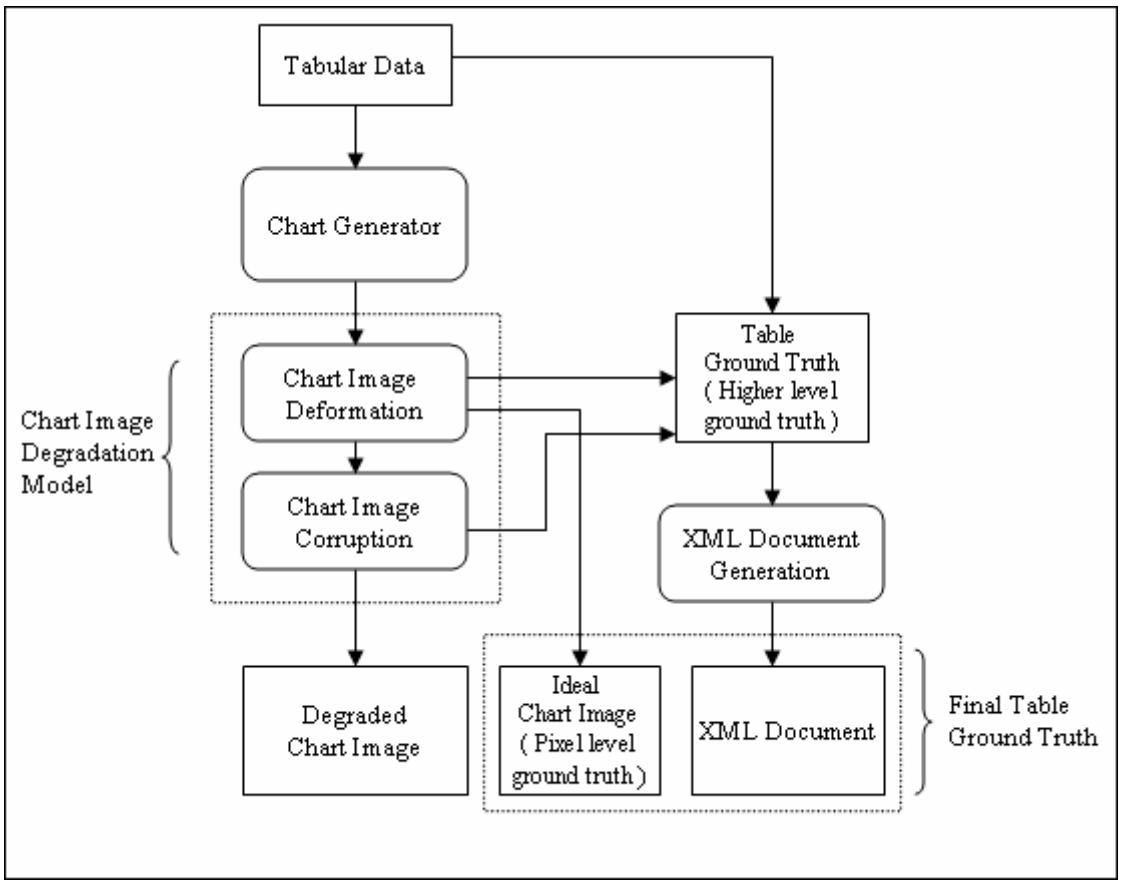


Figure 3.2 Chart ground truth generation system's overall structure

Figure 3.2 gives an overview of our system for synthetic chart ground truth generation. The collected numerical data which to be represented by chart is firstly given to *Chart Generator*. The task of *Chart Generator* is to create a chart of a certain type to represent the data given to it. *Chart Generator* will give appropriate geometrical information to every chart component such as location and size. Every drawing primitive such as line, arc etc is totally determined by *Chart Generator*, hence the ground truth associated with the chart can be obtained accurately. However, at this moment the ground truth is not stored, because in the image degradation model, *Chart Image Deformation* process will

introduce geometric transformations to the chart image, which will cause the vector level ground truth, such as position of lines and arcs, of the chart to change.

After going through the *Chart Image Deformation* process, the geometric attributes of every chart component is then fixed, so the vector level ground truth of the chart no longer changes. We thus record chart ground truth in this stage. We have used a convention here: although chart image deformation is part of our image degradation model, we still call images which have only undergone deformation as ideal images; because until that moment, the deformed image is still "clean" - no corruption such as noise and blur, etc has been added; only after the image corruption process been applied then we call the it as "degraded". So in this stage we store the deformed ideal image as the representative of pixel level chart ground truth; higher level ground truths are stored separately, later they will be converted to machine-readable XML format by the *XML Document Generation* process.

Lastly *Chart Image Corruption* process will be applied to the previously deformed ideal image. This process mainly approximates the physics of the printing and imaging process in the real world. Our degradation model is parameterized, the parameters of image corruptions and image deformations will also be recorded as ground truth for the degraded images. After image corruption process, the chart image is further degraded, and the final degraded image together with the chart image ground truth including a copy of ideal chart image (pixel level ground truth) and XML file (higher levels ground truth) are the final output of our system.

In this project, our major interest is the synthetic chart images themselves and their degraded variants. Since numerous researches on optical character recognition and character degradation have been done in the past, we will not put particularly focus on the character information in the chart. So when a synthetic chart is created, minimal textual information is generated for the chart such as data labels and scales on *X* axis.

Chapter 4

Synthetic Chart Image Generation

There are numerous chart synthesizing tools available for generating chart images, perhaps the most widely used one is the Microsoft Excel chart generator. Besides the chart image itself, in order to obtain the ground truth data, additional information on every detail in the chart image is also needed, such as vector level and component level information mentioned in section 3.1. However, such information is not available in commonly used chart synthesizing software. Therefore, we design our own routine, which can produce not only chart images but also their ground truths.

Currently our system can construct bar charts (2D, 3D) and pie charts (2D, 3D). The process of chart construction can be viewed as assigning appropriate values to several data structures which compose necessary components of the chart. As stated in section 3.1 of chapter 3, a chart's structure is hierarchically, figure 4.1 further elaborates on this. Here in figure 4.1 we use general term like "Bar Chart" to represent both 2D bar chart and 3D bar chart. 2D chart is relatively easier to create, so we mainly talk about how to get 3D chart from 2D chart by utilizing simple affine transformations.

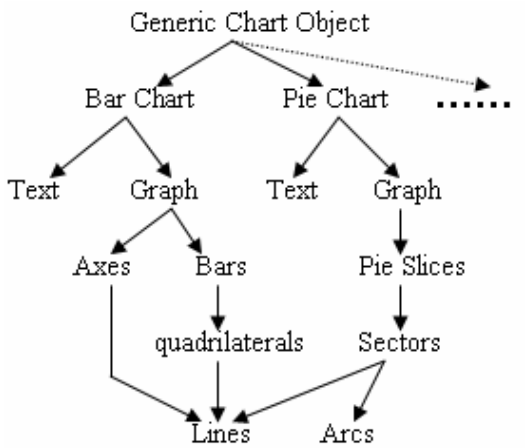


Figure 4.1 Decompose bar chart and pie chart

4.1 Drawing 3D Shapes in the 2D Space

When 3D shapes are displayed on the screen or on the paper, it can be seen as proper organizations of 2D primitives which give viewers three-dimensional look-like feelings. Because a screen or a paper's surface is just two dimensional, every object in such two dimensional space is in nature a 2D object, but by placing certain 2D objects together, human's visual system can interpret them as a single 3D object. One instance is given in figure 4.2.

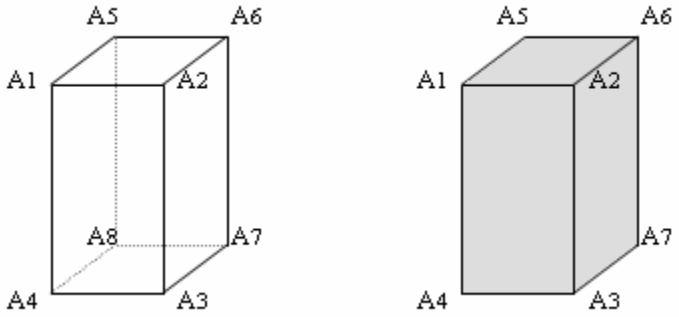


Figure 4.2 Cubes are displayed in a two-dimensional space

In figure 4.2, the left hand side is the skeleton of a cube, which is constructed by connecting its 8 vertices. The displaying environment is a two-dimensional surface, every vertex from A_1 to A_8 and every line segment formed between vertices, are 2D geometric objects; but by grouping those 12 line segments in the manner as shown above, what we will observe is a 3D look-like cube. The right hand side is the same cube having its surfaces filled with a color. Thus this cube can be viewed as composed by 3 two-dimensional quadrilaterals: $A_1A_2A_3A_4$, $A_1A_2A_6A_5$, and $A_2A_3A_7A_6$.

Since 2D primitives are adequate to compose 3D objects, we will purely draw 3D shapes on the 2D plane, and later in chapter 5 when we apply deformations on 3D shapes, we also just manipulate their 2D components.

4.2 From 2D Bar Chart to 3D Bar Chart

In a 3D bar chart, the bars become cubes and they are placed in a 3D look-like space. If we let the 8 vertices of a cube be in 2 groups: 4 are the vertices of the front rectangle

which appears “nearer” to the viewer; the other 4 are the vertices of the back rectangle which appears “further” to the viewer. Since the cube is drawn in 2D space, we can view those 4 back vertices are obtained by applying translation operation in both *X, Y* direction on the 4 front vertices. Based on this, we have following steps to create a 3D bar chart:

Step 1: create a 2D bar chart to represent the source data

Step 2: for every component of the 2D bar chart, apply a predefined translation operation on all the component’s vertices in both *X, Y* direction. If the component is a bar, then we get 4 new vertices, together with the bar’s original 4 vertices, a cube can be formed; if the component is an axis, then the 2 new vertices together with the axis’ original 2 vertices, forms a quadrilateral. As shown in figure 4.3, we need 3 quadrilaterals to form a 3D look-like space where bars would be placed, step 2 will produce two such quadrilaterals, namely *A* and *B* in figure 4.3, the last quadrilateral *C* can be automatically determined.

Step 3: adjust every cube’s thickness and position, allow it to be placed properly inside the 3D look-like space formed in step 2

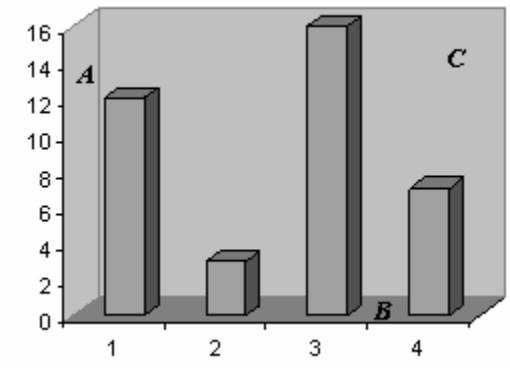


Figure 4.3 Quadrilateral *A*, *B* and *C* form a 3D look-like space where bars would be placed

4.3 From 2D Pie Chart to 3D Pie Chart

A 2D pie chart consists of circular sectors which can all together form a circle. While a 3D pie chart consists of 3D elliptical sectors which together form a cylinder. We call them elliptical sectors because the upper and lower sides of the cylinder formed by those sectors are ellipse. A 3D elliptical sector has 6 vertices with 3 of them locating on the

upper plane of the sector and the other 3 locating on the lower plane of the sector. Similarly, we can view the lower 3 vertices as the result of being applied translation only in *Y* direction on the upper 3 vertices. So in order to get a 3D elliptical sector, we must first have a 2D elliptical sector.

When converting a circular sector to an elliptical sector, the original sweep angle of the circular sector must be re-calculated. If this step is ignored and original sweep angle unchanged, a wrong percentage of the elliptical sector will be caused. As shown in figure 4.4, suppose the length of the ellipse's semi-major is *a*, length of its semi-minor is *b*, sweep angle of original circular sector is φ, sweep angle of corresponding elliptical sector is θ. The relationship between φ and θ can be found as follows:

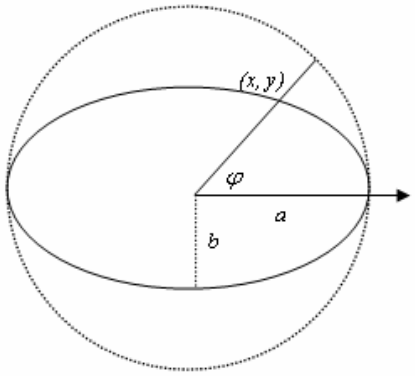


Figure 4.4 Converting a circular angle to elliptical angle

The parametric form of the ellipse is:

$x = a\ cos(t)$

$y = b\ sin(t)$

where *t* can be any value. So in the circular sector we have

$$tan(\varphi) = \frac{y}{x} = \frac{b\sin(t)}{a\cos(t)} = \frac{b}{a}tan(t)$$

we then let t equal to θ, which is the corresponding elliptical.

$$\theta = t = tan^{-1}(\frac{a}{b}\ tan(\varphi)) \qquad (4.1)$$

Besides the angle, the 2 arc vertices of the original circular sector should also be re-calculated. The 2 arc vertices are the intersections between a sector's arc and its two radiuses, we call them starting vertex and ending vertex that the sector's arc. We define the starting angle and ending angle as the angles between the *X* axis and the starting vertex and ending vertex, respectively. By using formula *(4.1)*, new starting angle and ending angle will be calculated for the elliptical sector, therefore the corresponding starting vertex and ending vertex of the elliptical sector can be obtained.

We use following steps to construct 3D pie chart from 2D pie chart:

Step 1: create a 2D pie chart to represent the source data

Step 2: from step 1 we get several 2D circular sectors, and convert every 2D circular sectors to its corresponding 2D elliptical sectors

Step 3: for every 2D elliptical sectors, apply a predefined translation operation in *Y* direction on its 3 vertices (2 arc vertices + 1 center point), then we get the 3 vertices of the lower plane of the 3D elliptical sector, the old 2D elliptical sector is the upper plane of the 3D elliptical sector.

Chapter 5

Chart Image Degradation Model

5.1 Chart Image Deformation

Chart image deformation alters geometric attributes of a chart image, which usually simulates the global degradations on chart images. Such global degradation effects normally involve *rotation* and *shearing*.

5.1.1 Rotation

When a chart image is scanned or photocopied, if the chart image is not being placed appropriately, then the output chart image is not displayed horizontally, i.e. the chart image becomes skewed. This error is usually caused by human beings in the scanning or photocopying process. Rotation is a very common degradation type because one major source for real-life chart images is scanned or photocopied images. Figure 5.1 contains a real-life bar chart image which is slightly skewed.

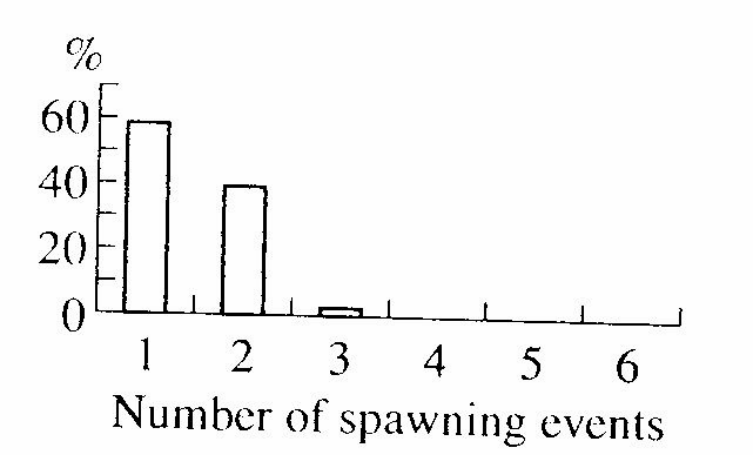


Figure 5.1 A scanned chart image which is slightly skewed

Since our system draws chart images in the 2D plane, we will model the rotation in the *X*-*Y* coordinates: suppose a point (*x*, *y*) is to be rotated with respect to the original point by an angle β, and (*x'*, *y'*) denotes the rotated point. We further assume the line segment

from the original to point (x, y) forms an angle α, and the length of that line segment is r, such as shown in figure 5.2.

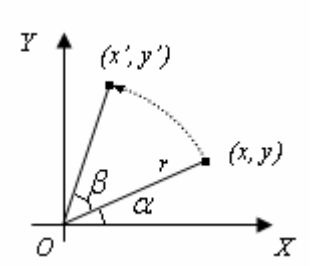


Figure 5.2 Rotate a point in the X-Y coordinates

so we have

$$x' = r\ cos(\alpha + \beta) = r \cos\alpha \cos\beta - r \sin\alpha \sin\beta \qquad (5.1)$$

$$y' = r\ sin(\alpha + \beta) = r \cos\alpha \sin\beta + r \sin\alpha \cos\beta \qquad (5.2)$$

since

$$x = r \cos\alpha,\ y = r \sin\alpha$$

equation *(5.1)* and *(5.2)* can be converted to

$$x' = x \cos\beta - y \sin\beta$$

$$y' = x \sin\beta + y \cos\beta$$

in vector form, we can write the above equations as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad (5.3)$$

Thus when we apply rotation to a geometric object, we can use the above rotation matrix to find the new position of the rotated object. For instance, if a line segment is rotated by some angle, since a line segment's position is determined by its two vertices, we can use the rotation matrix to find the new values of the two vertices, and re-draw the line segment in that new position. The rotated angle β is in the range of (-180, 180), measured in degrees.

5.1.2 Shearing

Shearing is also a common degradation type existing in real-life chart images. Shearing will usually change the shape of a geometric object. Take the chart in figure 5.3 for example, this bar chart has been slightly sheared, such that the whole chart is inclined to

the right, and each bar in this chart thus becomes the shape more close to a parallelogram rather than a strict rectangle.

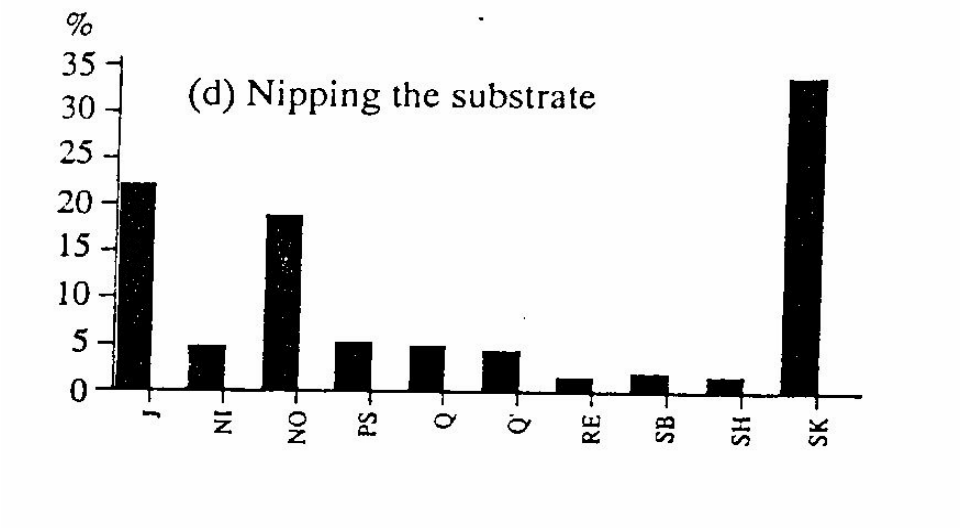


Figure 5.3 A real-life chart image which is slightly sheared

We can use a shearing matrix to model the shearing effect. Since horizontal shearing is most commonly seen, we only model horizontal shearing. The derivation of shearing matrix is as follows: suppose a point (x, y) is sheared to a new location (x', y') by a distance r, as shown in figure 5.4.

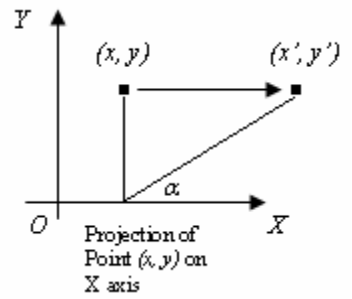


Figure 5.4 Shear a point along the X direction

We have

$x' = x + r$ and

$y' = y$

Since

$r = y \cot \alpha$

where α is the angle between *X* axis and the line segment formed by point (*x'*, *y'*) and the projection of point (*x*, *y*) on X axis.

So in vector form we get the shearing matrix:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & \cot\alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad (5.4)$$

We call $cot\alpha$ the horizontal shearing factor. The value and sign of $cot\alpha$ determine to what extent and the direction of the shearing. However, if the absolute value of $cot\alpha$ is too large, the absolute value of *x'* (which equals to $|x + y\ cot\alpha|$) will also become too large, this leads to severely yet unrealistically shearing effect. Therefore we restrict the values that $cot\alpha$ can take are in the range [-1, 1], that is, maximal shearing effect is achieved when $x' = x + y$ or $x' = x - y$. Usually in real-life images, only slightly shearing effect exists, so small values for $cot\alpha$ are preferred.

5.1.3 Deforming 3D Objects

Section 5.1.1 and 5.1.2 mentioned deformations in the two-dimensional space, which are also applicable to 3D objects. As discussed in section 4.1, every 3D shape is constructed by several 2D primitives. Hence just by manipulating 2D primitives, the deformation effects on 3D shapes can also be achieved.

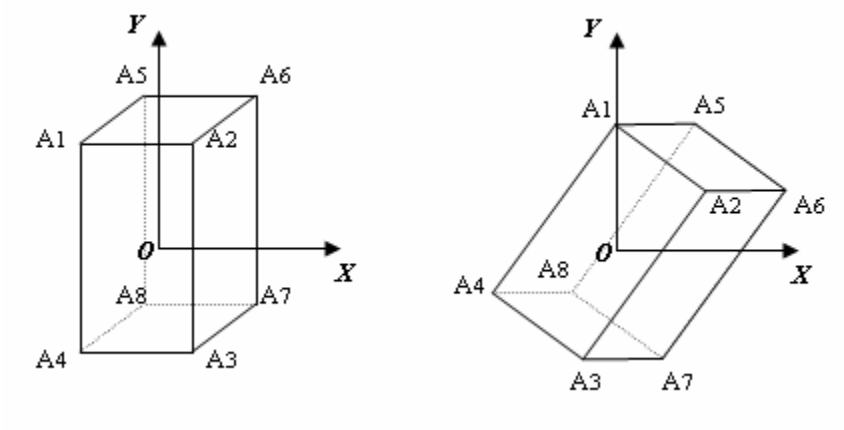


Figure 5.5 Rotate a 3D look-like cube in the 2D space

In figure 5.5, a 3D cube is rotated in the *X*-*Y* coordinates. Since 8 points (vertices) determine a cube, they are first rotated, and then line segments are drawn between rotated vertices to form a rotated 3D cube. Rotating a point in the *X*-*Y* coordinates is described in section 5.1.1. Shearing 3D objects is similar.

In our discussion so far we demonstrated our ideas using a traditional coordinate system with *X* axis pointing rightward and *Y* axis pointing upward. However, in most imaging systems, *Y* axis' direction is defined to be downward. We implement our system following this convention. The formulas we derived above are still applicable, the only difference is the orientation which just affects the signs of angles. For example, in our implementation, we define clock-wise be the positive direction of rotation, while in traditional coordinate system clock-wise is a negative direction of rotation.

5.2 Chart Image Corruption

As mentioned in section 3.2, image deformation process still produces ideal images, in the sense that each pixel still conveys the correct information. Chart image corruption process, however, will alter certain pixels' value, thus making the chart image truly "degraded", or corrupted. Hence after this process, certain correct information in the ideal images will be obscured or distorted. In this section we introduce three kinds of image corruption effects: edge distortion, motion blur and Gaussian noise.

5.2.1 Edge Distortion

In real-life images, distortions are very likely to be found along the edges of lines or regions. Figure 5.6 contains part of the magnified chart image shown in figure 5.1. We can see that the edges of bars' are not perfect lines, but being distorted. Such distortion effect always goes with image reproducing processes like scanning, photocopying and faxing etc. For instance, most scanners use a CCD array to scan document, which is controlled by a stepper motor, usually the stepper motor is not ideal, so mechanical side

effects such as vibration is unavoidable, such vibration during scanning process may cause distortions on the image components' boundaries due to pixel mis-alignment.

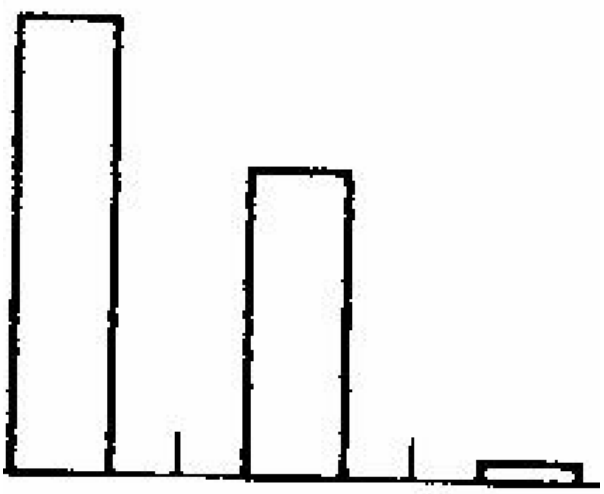


Figure 5.6 Part of the magnified chart image in figure 5.1

(Zhai et al, 2003) proposed a method to generate high frequency noise along the edges of lines, by using convolution. Our edge distortion algorithm is based on their idea but modifications are made to suit our own purpose: their high frequency noise models situations when lines are drawn or printed on a rough paper, the ink may spread out from the drawing region to non-drawing region, thus making the edges become un-smooth (i.e. some background pixels which are near the edges become the edge color); our purpose is to model pixel mis-alignment which is usually occurred during image reproducing processes such as scanning and faxing. As shown in figure 5.6, edge distortion can be generalized as 2 categories: pixel increasing and pixel decreasing along the edges, these two phenomenons are shown detailedly in figure 5.7.

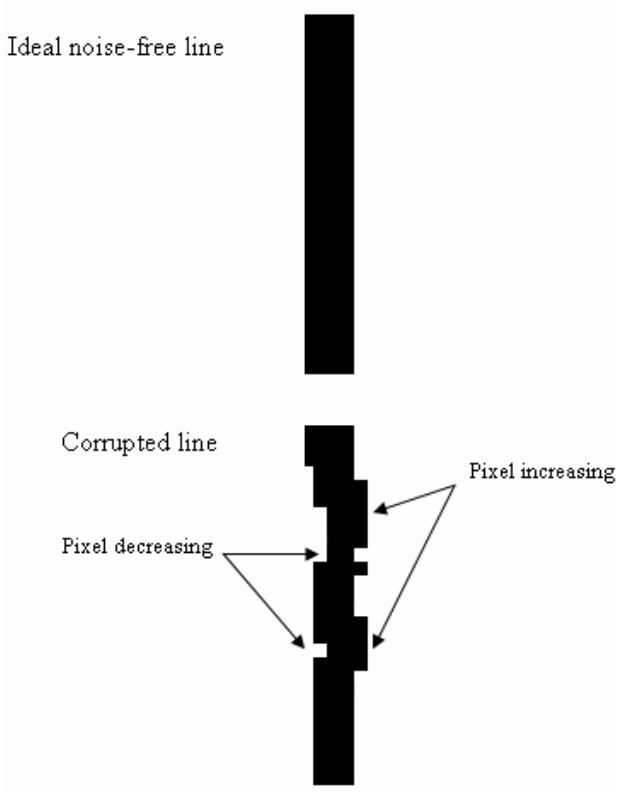


Figure 5.7 Edge distortion types, the lines are magnified

Figure 5.8 shows an algorithm to model the edge distortion effect which is modified from (Zhai et al, 2003)'s high frequency noise generation procedure. Generally speaking, edge distortion in the real-life images is a non-deterministic process, that is, there is no rigid pattern that the distortion will follow. Hence we also model the distortion using a randomly manner.

To model randomly pixel increasing and pixel decreasing along boundaries, 2 problems need to be solved: *(1)* locating edges of a line; *(2)* deciding whether to expand an edge pixel or shrink it, choice is made randomly since edge distortion is a random process. From *step 2* in figure 5.8 we can see that, *t* is the average value after convolving an area in the image which has current pixel as the area's center; only when current pixel is near a boundary of different colors, such as an edge of regions, did the *t* become a different enough value comparing to the original color value. In *step 3*, condition $| t - G_{ij} | > \delta$ measures the difference between the calculated value for current pixel and its original value. δ is always set to a small integer. If the difference is larger than the threshold δ,

Let G denote the set of pixels in a chart image.
Let *RAND( )* denote a random number generator which generates real numbers uniformly distributed between 0 and 1.
Let *L_MAX* be a predefined value.
Let L be a threshold value which is in the range of [0, *L_MAX*].

Make a copy of the pixel set G, use H to denote such a copy

for i = 1 to chart image's height
  for j = 1 to chart image's width
    Let R = *RAND( )* * *L_MAX*.
    if $R < L$ then
      Step 1: Create a convolution kernel K with size of $w = 3$, each value in kernel K is a randomly generated integer ranging from 1 to 1000
      Step 2: Calculate the integer value t using the following formula:

$$t = \frac{\sum_{m=-w/2}^{(w-1)/2} \sum_{n=-w/2}^{(w-1)/2} Kpq G'mn}{\sum_{m=-w/2}^{(w-1)/2} \sum_{n=-w/2}^{(w-1)/2} Kpq}$$

      where $p = 1+m+w/2$, $q = 1+n+w/2$; G' is the area of pixel in G which has G_{ij} as its center and with size of w, G'_{mn} is the (m, n)th item in G'
      Step 3: if $|t - G_{ij}|$ > some predefined integer threshold value δ, then randomly assign H_{ij} to the chart image's background color or edge color with equal probability
      end if
    end if
  end for
end for
Update every pixel value in G with the corresponding pixel value in H
Use a 3 by 3 median filter to convolve G, output the final result

Figure 5.8 Algorithm to simulate edge distortion effect

we then know the current pixel is near the edges of some region, in which case, we will alter the current pixel value to either the background color or edge color, i.e. the assignment is done randomly. After the whole chart image has been convolved, pixels near edges will have either the background color or edge color, giving an impression of smashing edges. The last stage is to apply a 3 by 3 median filter. By the characteristics of median filter, when it is being applied, the interior regions of shapes will not be changed because an interior region is filled with the same color value. Since edges have been

smashed, i.e. pixels on the edges or near the edges might be the edge color, or might be the background color, when a median filter is convolved along the edges, the percentages of edge color pixels in all the pixels covered by the median convolution kernel (in our case, 9 pixels are covered each time) are non-deterministic, therefore, the center pixel in the median convolution kernel will be assigned background color or edge color randomly. After convolving the whole chart image, edge distortion effect is achieved. The result of above process is shown in figure 5.9.

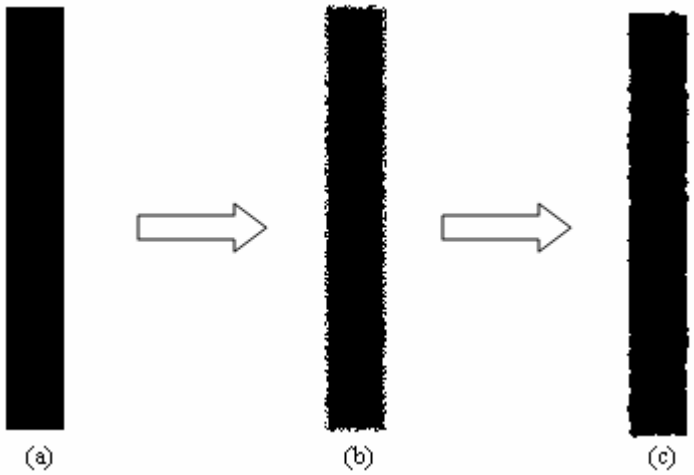


Figure 5.9 (a) ideal image, (b) edge get smashed (c) final output, after applying median filter

In the algorithm in figure 5.8, *L* controls the degree of edge distortion, if *L* is large, the edges get more severely smashed, hence more distortion after median filtering; *L = 0* means there is no distortion, *L = L_MAX* means largest distortion; *L_MAX* takes the value of 10, so $L \in [0, 10]$, figure 5.10 shows some different levels of edge distortions that our system produced. As mentioned earlier, our study does not put the emphasis on degradation model of optical characters, so the smashing procedure and the median filter are not applied to the textual information in the chart image. For specialized optical character degradation modeling, please refer to (Baird 1990, Baird 2000).

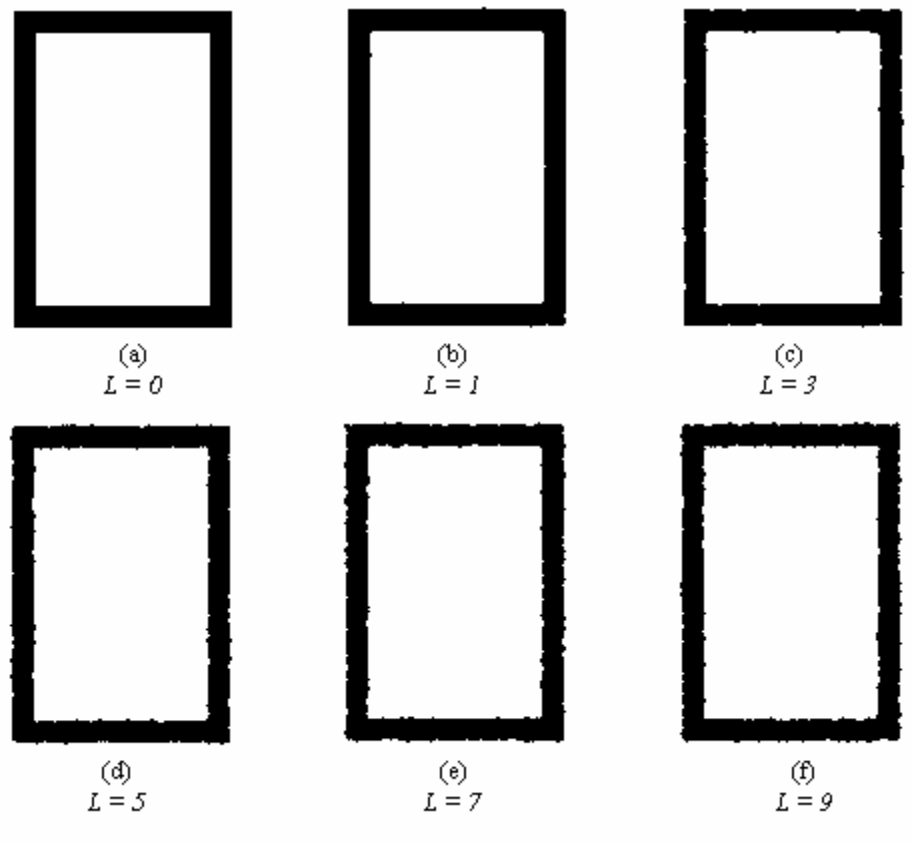


Figure 5.10 Some different distortion levels, L = 0 means no distortion

5.2.2 Motion Blur

Motion blur is caused by the relative motion between the camera and the document during capturing process. It's also one of the major sources of image degradation types, especially in the processes when documents are recorded by photographing or scanning. We adopt a well-known degradation model (Gonzalez, Wintz 1987) to model the motion blurring process, as well as noise, which will be discussed in section 5.2.3. This degradation model is shown in figure 5.11.

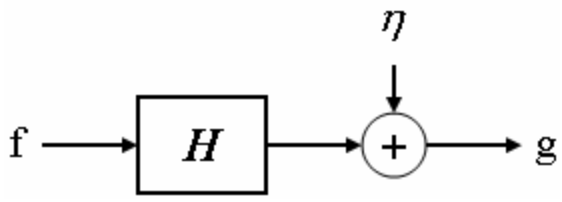


Figure 5.11 Degradation model for motion blur and noise

And mathematically we have

$g(x, y) = f(x, y)^{**}H(x, y) + \eta(x, y)$ *(5.5)*

where $g(x, y)$ is the observed image, $f(x, y)$ is the input image, $H(x, y)$ is the blurring function or called blurring point spread function which can represent to what extent a point source has been smeared by the image process, ** denotes the convolution operation, and $\eta(x, y)$ is the noise function.

For this part, let's assume $\eta(x, y) = 0$ first. Since a chart image contains pixels that form a 2D array with a *width* and a *height*, the blurred image can be obtained by the following formula:

$$g(x, y) = \sum_{n=1}^{width} \sum_{m=1}^{height} f(x-m, y-n)H(n, m) \qquad (5.6)$$

We model the situations when an object moves at a constant velocity V, and further suppose the exposure time is unit time, moving direction forms an angle θ with the horizon, S is the distance that the object has traveled during the exposure time. In this case, the blurring point spread function in *(5.6)* is defined as:

$$H(x, y) = \begin{cases} 1/S, when 0 \le |x| \le S * \cos\theta, 0 \le |y| \le S * \sin\theta \\ 0, otherwise \end{cases} \qquad (5.7)$$

Since exposure time is a unit time,

$S = V * exposure\ time = V$

Because we are using $H(x, y)$ to convolve the input chart images, odd values for V are preferred. Let $V = 2v + 1$, $v \in [0, 5]$, so equation *(5.7)* can be further represented as:

$$H(x, y) = \begin{cases} 1/(2v+1), when 0 \le |x| \le (2v+1) * \cos\theta, 0 \le |y| \le (2v+1) * \sin\theta \\ 0, otherwise \end{cases} \qquad (5.8)$$

In equation (5.8), v denotes the level of the motion blur, $v = 0$ means no motion blur is added; $\theta \in$ (-180, 180), measured in degrees, denotes the angle of the motion blur; as stated in 5.1.3, clock-wise rotating θ results in an angle of positive value. Figure 5.12 gives 2 motion blur effects our system produced.

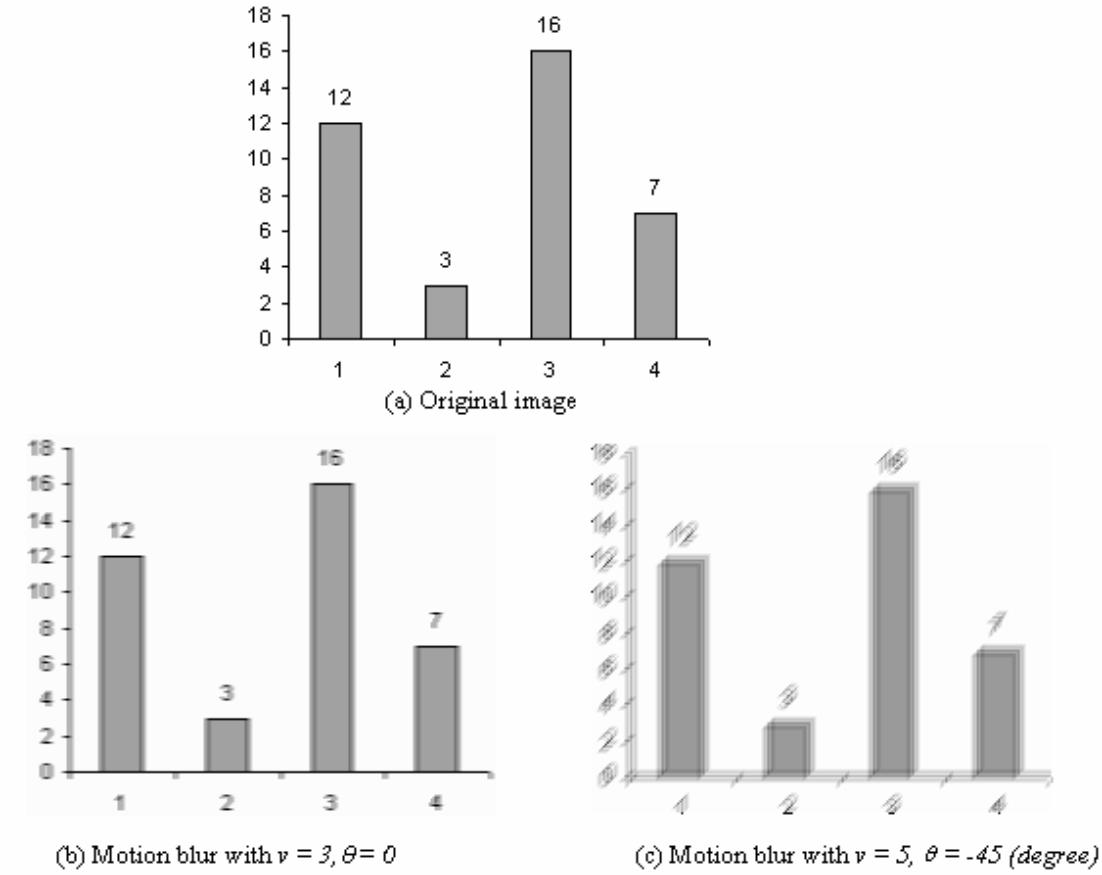


Figure 5.12 Clean chart image and its motion blurred versions

5.2.3 Gaussian Noise

Electronic devices such as camera, printer, scanner etc are playing important roles in image acquisitions. As pointed out in (Pratt, 1991), thermal noise is the most common noise source in electronic imaging systems which is caused by electrons' randomly fluctuations, and additive Gaussian stochastic process is a very good model of thermal noise. Hence we let the term $\eta(x, y)$ in formula *(5.5)* represent an additive Gaussian noise.

The core issue is to obtain a Gaussian (normal) distribution. For the purpose of keeping the average brightness in the input chart image unchanged, we let the mean μ of the distribution be 0, and therefore, what we need to find is a random variant with its probability density function as:

$$p(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2} \qquad (5.9)$$

We use the polar method (Ross 1990) in figure 5.13 to generate a standard normal distribution, for complete proof of polar method please refers to (Ross 1990). Since polar method generates a normal distribution based on uniformly distributed values between 0 and 1, the quality of the generated standard normal fully depends on the quality of the uniform distribution. A good uniform distribution has a longer period and less correlation among successive numbers, in order to achieve this and not sacrifice much computational time, we adopted the algorithm called *ran0* in (William, Saul, William, Brian 2003).

Step 1: Generate two independent random numbers $U1$ and $U2 \in [0, 1]$

Step 2: Let $V1 = 2U1-1$, $V2 = 2U2-1$, $S = V1^2 + V2^2$

Step 3: Check: if $S \geq 1$ then return to Step 1;

Step 4: Check: if $S = 0$, let $X = 0$, $Y = 0$. Otherwise let

$$X = \sqrt{\frac{-2\log S}{S}}\, V1, \qquad Y = \sqrt{\frac{-2\log S}{S}}\, V2$$

then X and Y are the standard normally distributed variants.

Figure 5.13 Polar method to generate standard normal variables

Polar method returns a standard normal variable, say, X_0. Relate X_0 to X in *(5.9)* we have:

$$X_0 = \frac{(X-\mu)}{\sigma} = \frac{X}{\sigma} \;\rightarrow\; X = \sigma\, X_0 \qquad (5.10)$$

Hence for each pixel G_{ij} in the original image, we update G_{ij}'s value by adding a value $\sigma\, X_0$ to G_{ij}'s old value, and if the sum is greater than 255, set it to 255, likewise if the sum is smaller than 0, set it to 0. By this way, the Gaussian noise is added to the original

image. σ controls the levels of Gaussian noise. Larger/smaller the σ will be, more/less Gaussian noise will be added. However, if σ is too large, the image will be unnecessarily distorted, hence we set the upper bound of σ be 50 which is adequate to model thermal noise; although in *(5.9)*, theoretically σ can not be 0, in *(5.10)*, $\sigma = 0$ means no Gaussian noise will be added. Practically we allow σ can be 0, therefore $\sigma \in [0, 50]$. Figure 5.14 shows Gaussian noise at level 8 and level 20.

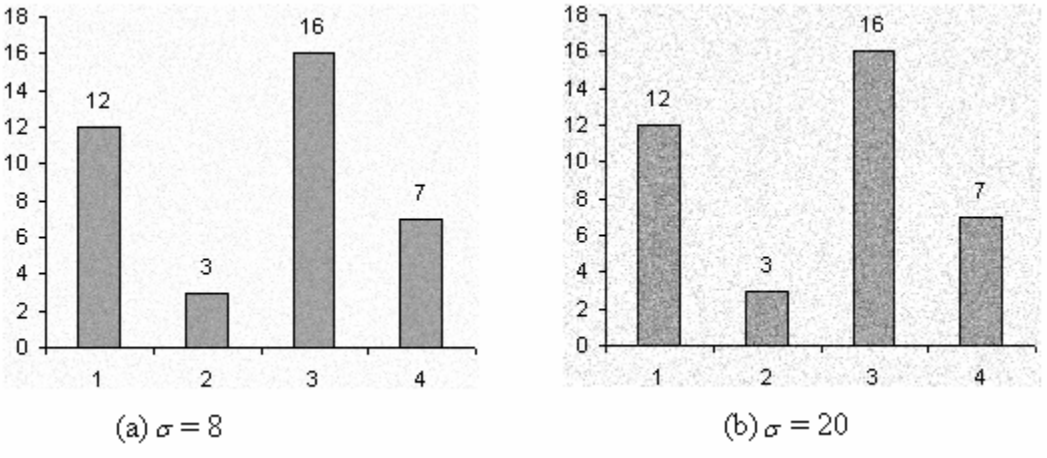


Figure 5.14 The effect of Gaussian noise

5.3 Summarization of Degradation Model

In this section, we have developed a chart image degradation model with parameters: $\theta_r, \lambda, L, v, \theta_b, \sigma$, a summarization of those parameters is given in table 5.1.

parameter	*type*	*range*	*meaning*
θr	real value	(-180, 180)	rotation angle, measured in degrees
λ	real value	[-1, 1]	horizontal shearing factor
L	integer value	[0, 10]	degree of edge distortion
v	integer value	[0, 5]	radius of motion blur
θb	real value	(-180, 180)	angle of motion blur, measured in degrees
σ	real value	[0, 50]	degree of Gaussian noise

Table 5.1 Parameters of our degradation model

The effects of above degradation types can be combinatory, we define the sequence of the degradation types being applied as: first, rotation and shearing, which still produce a noise free image; second, edge distortion; third, motion blurring and lastly Gaussian noise.

Chapter 6

Results

6.1 System Evaluation

The major strength of our system is to automatically generate chart image's ground truth. During the generating process human will only do some trivial jobs such as selecting which type of charts to produce, deciding to what extent a chart image gets degraded and choosing file names a chart image and its ground truth to be stored with. The system itself will automatically find the ground truth associated with chart images. This is because when generating a chart image, our system first decides all the attributes of the chart, which are indeed the ground truth data for it, and then use those attributes to render the chart image. This procedure is just the converse with labeling ground truth from existing images. Our system frees humans from such tedious and error-prone manually labeling, thus will make the ground truthing task fast and reliable.

Another strength is that the parameterized degradation model enables us to quantitatively measure degradation types, then it is possible to make comparisons between different chart recognition systems/algorithms. Our system is able to create a systematic set of degradations for an ideal image, such that we can have chart images from slightly degraded to severely degraded, forming a continuum of degradation levels, testing using such sets will enable us to find the break-down point of a given chart recognition system/algorithm.

Since chart objects are highly modular. As discussed in previous sections, higher level components can be repeatedly decomposed to lower level components. The chart itself is at the highest level, it contains number of smaller components. Smaller components are used by different types of charts. For example, bar chart, line chart and high-low chart all have a coordinate system. 3D charts can be viewed as transformations of their corresponding 2D fellows. In our design of the system, we catch the above relationships

among different charts and different components, therefore, modularity and reusability of the code are realized, making our system extensible to include new chart types and add new functionalities.

6.2 User Interface

Figure 6.1 gives the snapshot of our system's user interface, the white area is the drawing place where the images are displayed, and from here users can get feedback on the appearance of a degraded image. Functionalities of other parts are labeled in the figure. A complete walk through of the system can be found in Appendix A.

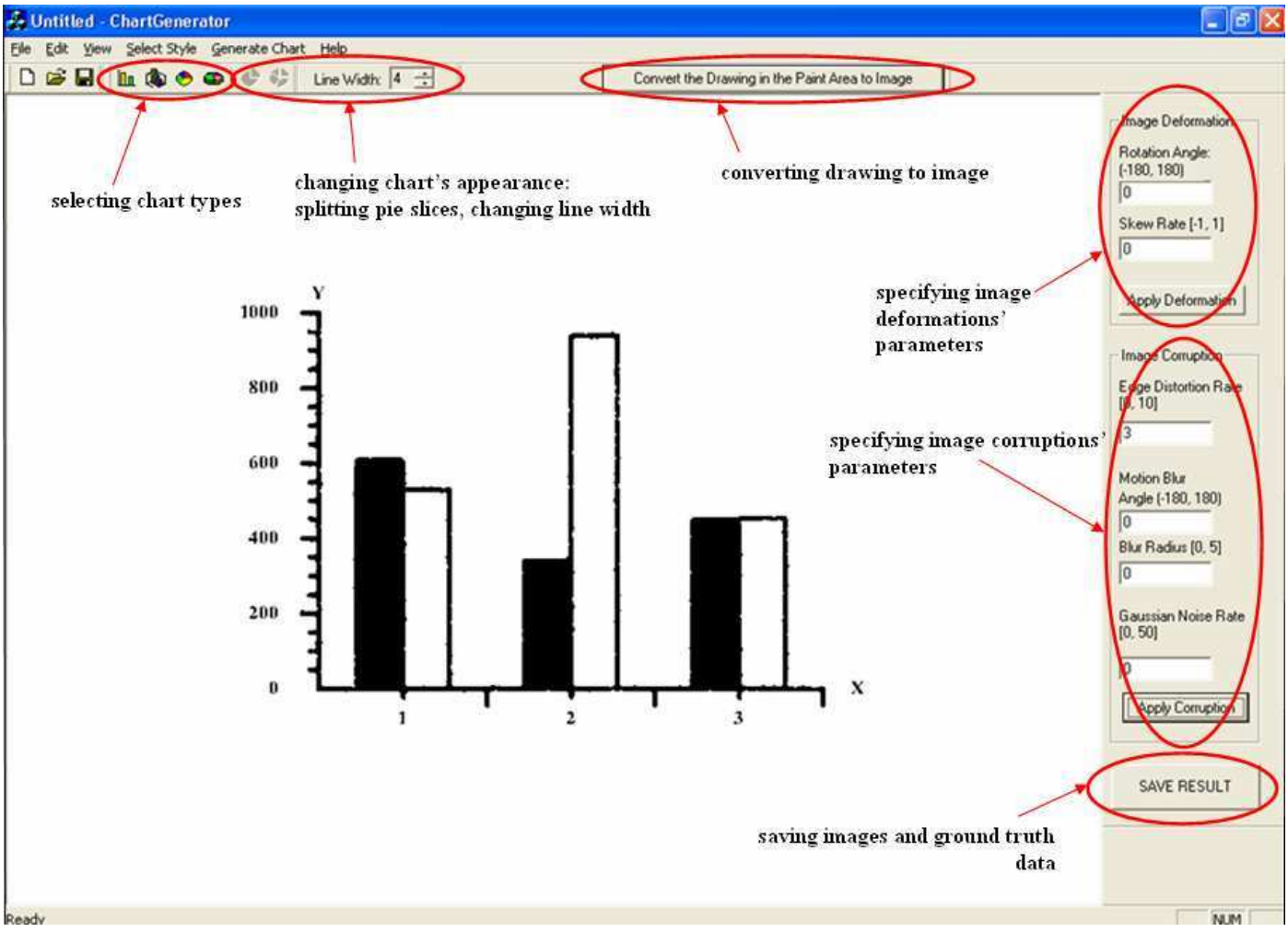


Figure 6.1 Screen capture of the system's user interface

6.3 Sample Images

In this section we show some sample images our system generated. Non-zero degradation parameters are labeled under each image and the original ideal image are shown for comparison with its degraded versions. Readers can judge from their own experience to what extent the synthesized degraded images resemble real-life images.

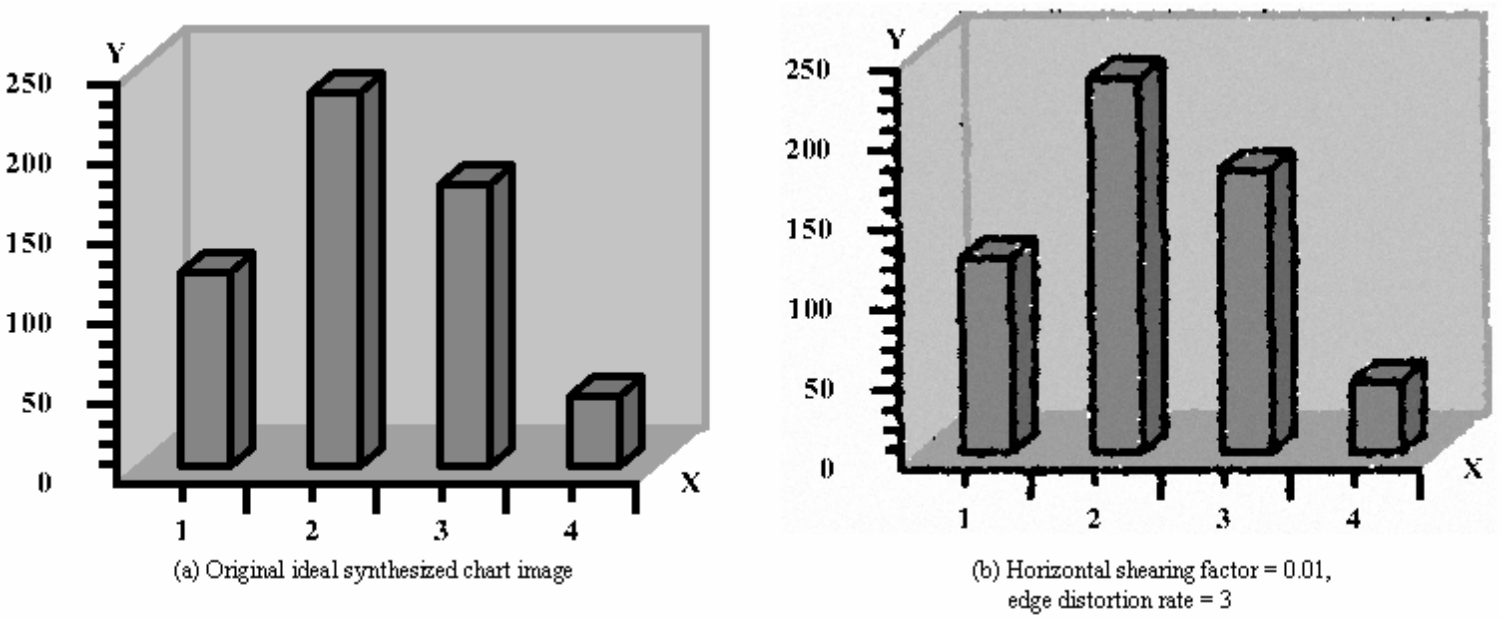
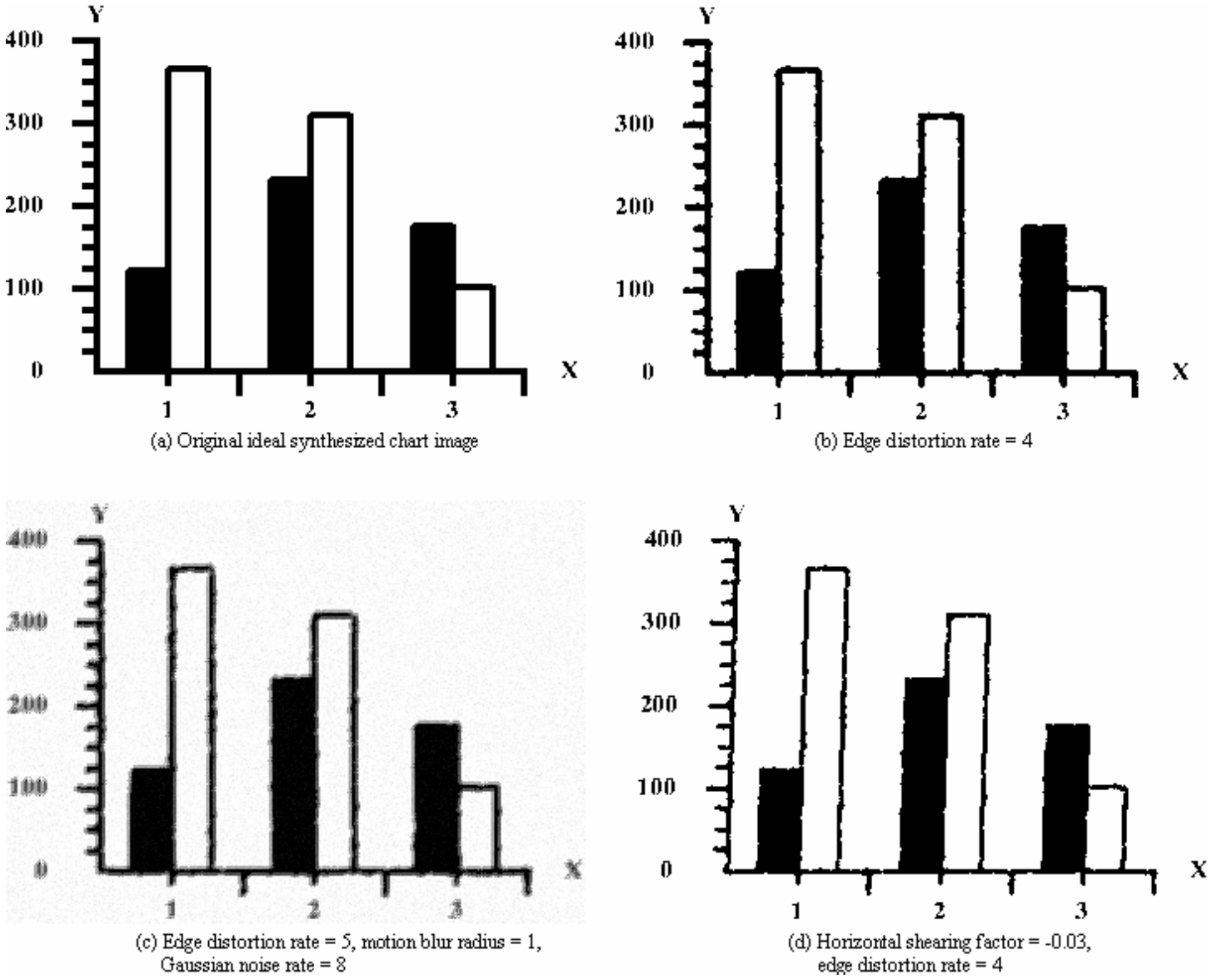


Figure 6.2 A 3D bar chart and its degradation version



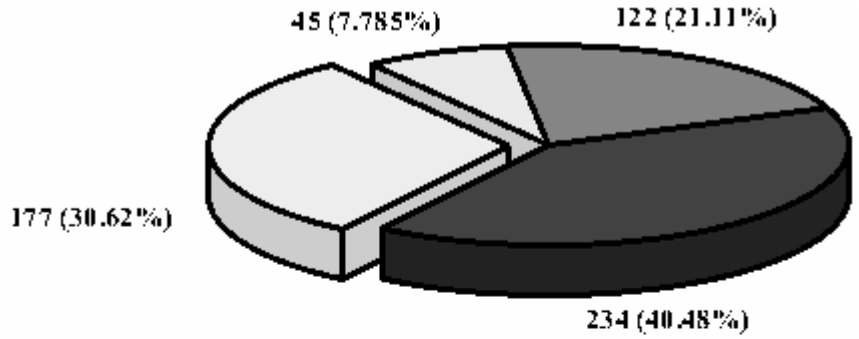
(a) Original ideal synthesized chart image

(b) Edge distortion rate = 4

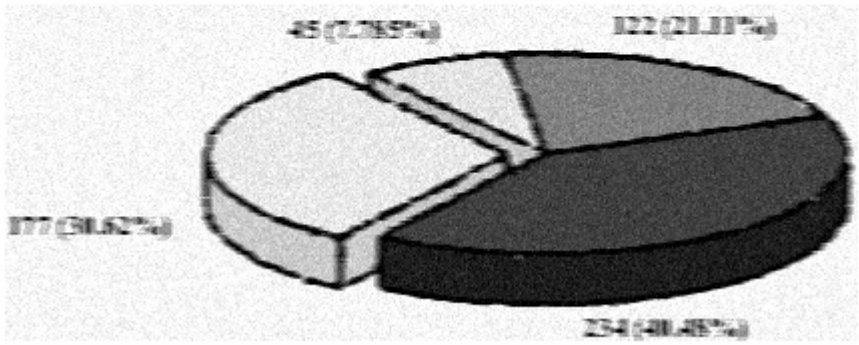
(c) Edge distortion rate = 5, motion blur radius = 1, Gaussian noise rate = 8

(d) Horizontal shearing factor = -0.03, edge distortion rate = 4

Figure 6.3 A 2D bar chart and its degradation versions

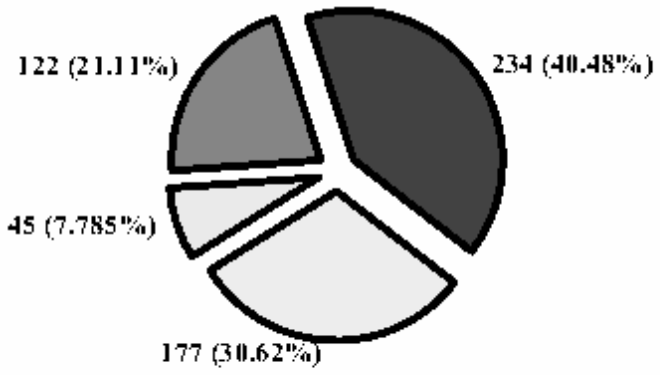


(a) Original ideal synthesized chart image

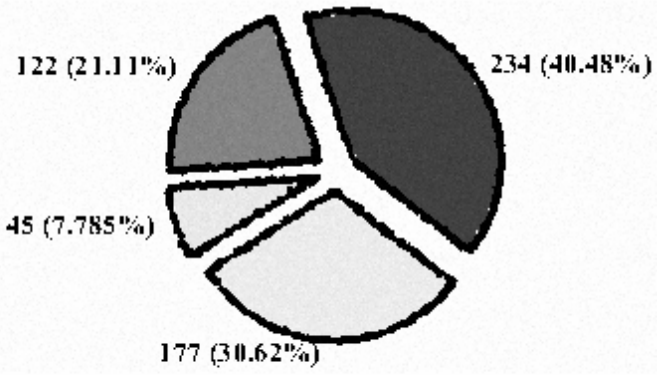


(b) edge distortion rate = 4, motion blur radius = 2,
Gaussian noise rate = 11

Figure 6.4 A 3D pie chart and its degradation version



(a) Original ideal synthesized chart image



(b) Gaussian noise rate = 4,
edge distortion rate = 5

Figure 6.5 A 2D pie chart and its degradation version

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this project, we have designed and implemented an software package to create synthesized chart images and automatically obtain their ground truth. We investigate some chart image degradation types which commonly occur in the real-world, and come out with a parameterized model which simulates degradation effects including rotating, shearing, edge (boundary) distortion, motion blurring and Gaussian noise. We applied our degradation model on generated ideal noise-free images to produce various levels of degraded chart images. Our software package can be used to create synthetic chart image ground truth datasets in large scales.

7.2 Future Work

Some future improvements of this project include:

- Currently our system is only able to generate bar charts (2D, 3D) and pie charts (2D, 3D), more chart types can be studied and introduced into the system in the future; besides, more functionalities can be added to our chart generation process to give more alternatives to the charts' attributes.
- Our image degradation model can be further enriched, as more physical devices' characteristics are studied; other sources of image degradations in real-world such as degradation caused by chemical natures of writing materials are also worth exploring.
- Performance measure using the ground truth data will be defined. There are works on line and arc detection algorithms' evaluations which can be used to define performance measure on vector level information. Performance measures from other levels need to be studied.
- Two important issues in degradation modeling are model validation and parameter estimation. Validation and parameter estimation themselves can be yet

another new research topics. Validation of a model means given a degraded image and its ideal prototype, how likely it is to duplicate the degraded image by applying the degradation model on the ideal image. Parameter estimation means given a degraded image and its ideal prototype, estimate the parameters of degradation. Since we haven't addressed these 2 problems in this project, future researches on these 2 topics are still needed.

Reference

Baird, H.S. (1990). "Document Image Defect Models". In 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

Baird, H.S. (2000). "The State of the Art of Document Image Degradation Modeling". In Proceedings of 4th IAPR International Workshop on Document Analysis Systems, Rio de Janeiro, Brazil 2000, pp. 1-16.

Cano, J., Perez-Cortes, J., Arlandis, J., Llobt, R. (2002). "Training Set Expansion in Handwritten Character Recognition". In Proceedings of 9th Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Windsor, Ontario, Canada 2002, pp. 548-556

Dennis, S. and Phillips, I. (1999). "Ground Truthing: Real or Synthetic Data – a Panel Discussion". At 5th International Conference on Document Analysis and Recognition, Bangalore, India, September, 1999, pp. 20-22.

Gonzalez, R.C. and Wintz, P. (1987) "Digital Image Processing" Second Edition, Addison-Wesley Publishing Company, 1987.

Ho, T. K. and Baird, H. S. (1995) "Evaluation of OCR Accuracy Using Synthetic Data", In Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada 1995, pp. 413-422.

Huang, W.H., Tan, C.L. and Leow W.K.(2003) "Model Based Chart Image Recognition", International Workshop on Graphics Recognition, GREC2003, 30-31 July 2003, Barcelona, Spain.

Jenkins, F. "The Use of Synthesized Images to Evaluate the Performace of OCR devices and algorithms", Master's Thesis, University of Nevada, Las Vegas, August, 1999

Kanungo, T., Haralick R.M. and Phillips I. (1993). “Global and Local Document Degradation Models”. In Proceedings of 2nd International Conference on Document Analysis and Recognition, Tsukuba, Japan, October 1993, pp. 730 – 734

Pratt, W.K. (1991) “Digital image processing” Second Edition, John Wiley & Sons, Inc. New York, 1991

Ross, S.M. (1990) “A Course in Simulation” Macmillan Publishing Company, New York, 1990

William, H.P., Saul, A.T., William, T.V., Brian, P.F. (2002) “Numerical recipes in C++: The Art of Scientific Computing”, Cambridge University Press, New York, 2002

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

Appendix A

Steps in Using the Chart Ground Truth Generation System

Step 1. Create a chart image by clicking a button shown below, the attributes of the chart can be altered after the chart has been created. Possible image deformations may be specified in this step.

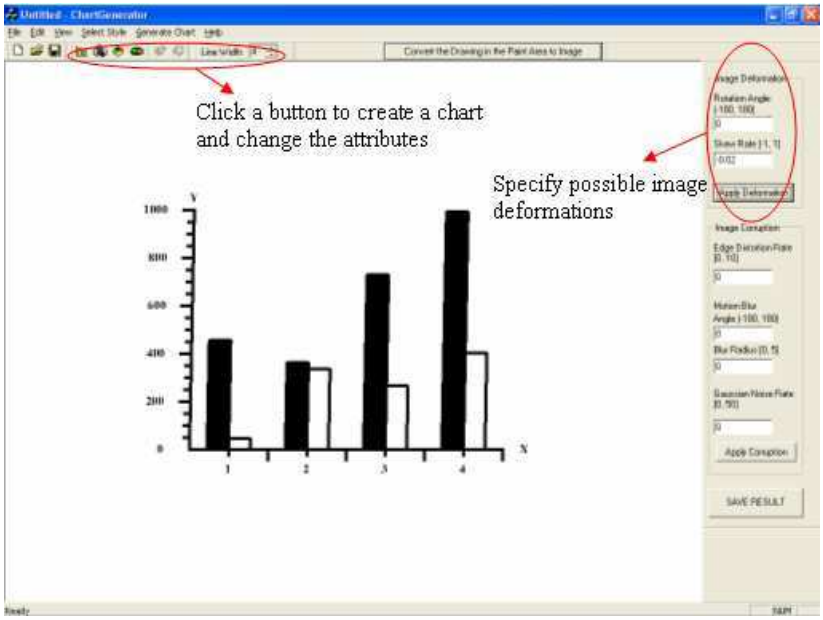


Figure A.1 (Step 1) Create a chart drawing and change its attributes

Step 2. Press down the left button of the mouse and select an area in the drawing panel which contains the chart, then click the button shown below to convert the selected area into an image.

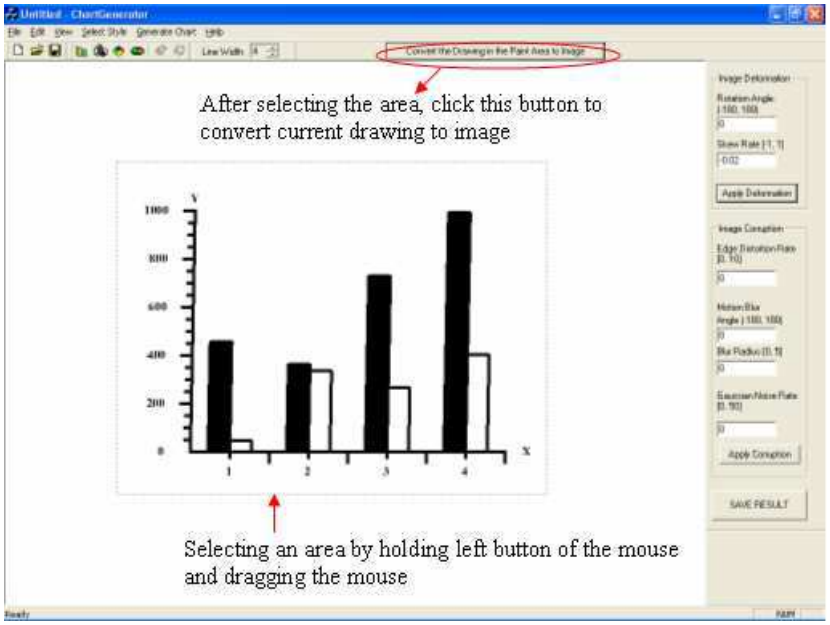


Figure A.2 (Step 2) Convert the chart drawing to image

Step 3. Specify possible image distortion types including edge distortion, motion blur and Gaussian noise, to make the original ideal image look like real-life image.

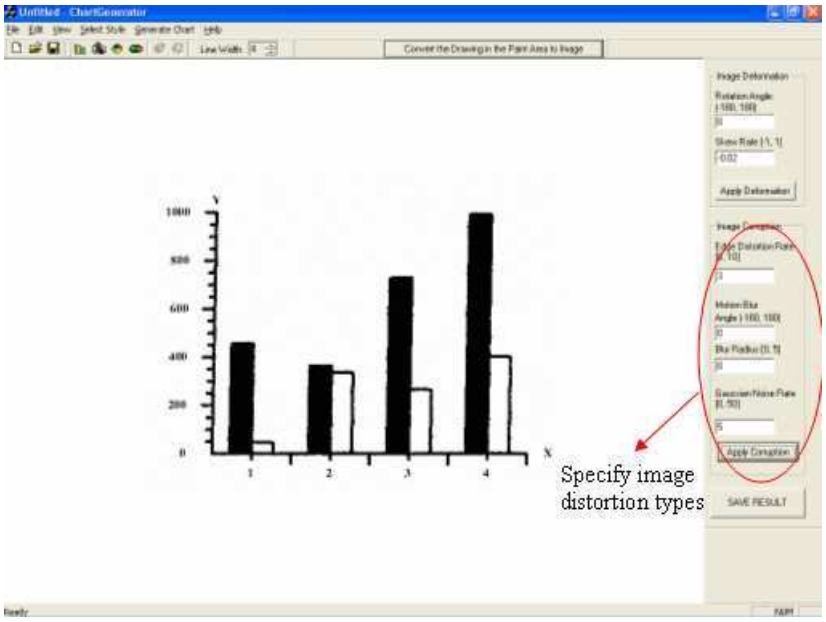


Figure A.3 (Step 3) Distort the ideal chart image

Step 4. Click the “SAVE RESULT” button shown below, a file dialog will prompt out to ask the users to choose file name and path to store the result. The original clean image, its degraded version and the ground truth data associate with the degraded image will be stored.

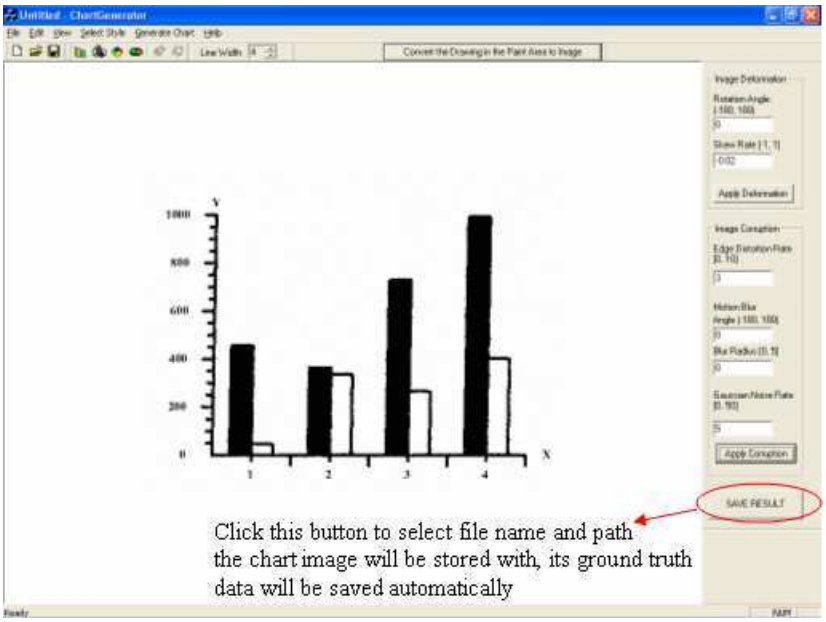


Figure A.4 (Step 4) Save the final image and its ground truth data