SHAPE-BASED IMAGE RETRIEVAL IN ICONIC IMAGE DATABASES

By

CHAN Yuk Ming

A Dissertation

submitted in partial fulfillment of the requirements

for the Degree of Master of Philosophy

Division of Computer Science and Engineering

The Chinese University of Hong Hong

June 1999

Abstract

Image retrieval based on image content has become a hot topic in the fields of image processing and computer vision. In this research, we focus on the problem of shape-based image retrieval in trademark database and Chinese cursive script character image database.

There are two major problems about the current approaches to shape-based image retrieval. First, researchers usually focus on using a single feature, e.g., Fourier descriptors, invariant moments or Zernike moments, without combining them for possible better results. Second, even if they combine the shape features, the weights assigned to them are often determined with an ad hoc procedure. Hence, we propose to group different shape features together and suggest a supervised learning technique using genetic algorithm (GA) to determine a suitable weighting factor for each individual feature for image retrieval. Based on the proposed scheme, we have built two prototype systems. The systems achieve both the desired accuracy and efficiency.

First, we handle the problem of shape-based image retrieval in trademark database. Trademark image retrieval is becoming an important application for logo registry, verification, and design. The number of trademarks is over one million now and is growing rapidly. Therefore, the task of registering a new trademark becomes difficult without inadvertent infringement of copyright of an existing trademark. In our system, we group different shape features including invariant moments, eccentricity, Fourier descriptors of approximated boundary, and circularity of approximated boundary together and we use the proposed supervised learning technique to determine the weights of the features. A trademark database with 1400 monochromatic images was tested. The retrieved images by our system using the learned dissimilarity function agreed well with those obtained by human subjects. Besides, our system was robust on retrieving rotated, scaled, mirrored, deformed, and hand drawn images. Each query took only 2.3 seconds on average.

Second, a Chinese cursive script character image retrieval system has been built. Several shape features including invariant moments, edge directions, and eccentricity are used to represent a character image. The original, thinned, and normalized images are used to extract the features. The weights for different shape features in the dissimilarity function are found by the proposed GA. A database of 1400 monochromatic images was tested. From the experimental results, our proposed system was robust on retrieving scaled and deformed images. Each query took 11.7 seconds on average.

摘要:

基於內容的圖像檢索技術，正成爲圖像處理及計算機視覺領域的一個活躍的研究課題。在這份研究報告中，我們集中討論了於商標及中國草書字體圖像資料庫內，實現基於形狀的圖像檢索問題。

在現有的基於形狀圖像檢索方法中，主要存在兩大問題，分別爲：第一、研究人員經常使用單一特徵，例如：傅立葉描述子，不變式矩，沙朗矩，而忽視了不同特徵的結合使用，這種結合將有機會獲得更好的檢索結果；第二、即使研究工作者試圖把不同的特徵組合起來，每個特徵的加權因子都只是以某個特定方式來確定的。所以，在本文中，我們建議在組合不同形狀特徵的同時，使用一個以遺傳算法爲基礎的監督學習技術，來計算每個形狀特徵最合適的加權因素，從而達到更好的圖像檢索效果。根據以上建議，我們建立了兩個原型系統，這兩個系統都能達到理想的準確度及效率。

第一個原型系統主要是爲解決商標資料庫中形狀圖樣檢索的問題而設計。商標圖形檢索在商標註冊、核實及設計中已逐漸成爲重要的運用。現時全球商標總數目已超越一百萬，而且不斷激增。故此，註冊新商標時需避免違反現有商標之版權也變得日益困難。在我們的系統中，我們把不同的形狀特徵如：不變式矩，離心率，邊界估計的傅立葉描述子及邊界估計的環狀描述組合起來，與此同時，亦使用監督學習技術來確定每個形狀特徵的重要性。我們對一個儲存了1400個黑白圖案的商標的資料庫作出測試，實驗結果表明，使用這個以學習來確定不相似性函數參數的檢索系統，檢索出來的結果與人類主觀觀察的結果的相符合。除此以外，對於旋轉變換了的，比例變換了的，反像的，變形的和手繪的圖像，這個系統能穩健地作出檢索。每次檢索的時間平均亦只需約2.3秒。

第二個原型系統則爲中國草書字體圖像檢索的問題而設計。每個字體都以其不變式矩，邊沿方向及離心率來表示。我們以其原始的，細化的和規格化的影像來折取其特徵。在不相似性函數中，不同特徵之比重則以我們所提議的遺傳算法來計算。我們以一個擁有1400個單色字體圖像的資料庫做實驗，實驗結果表明，對於比例變換了的和變形的圖像，這個系統能穩健地作出檢索。每次檢索平均需時約11.7秒。

Acknowledgement

This thesis could not have been done without the help of few individuals who deserve thanks and credit.

First of all, I owe a great deal of thanks to my supervisor, Prof. Irwin King, for his sharp insight, expertise, and enthusiasm. I thank him for the guidance, encouragement, and emotional support during the last two years. The comments of my examining committee, Prof. Ada Fu, Prof. Pheng-Ann Heng, and Prof. Mark Liao, have been extremely helpful to me in bringing this report to its finished form.

Moreover, many thanks go to my friends. I am especially grateful to Winnie Lo, Po-Shan Kam, Chi-Man Lam, Dick Lau, and Xue-Qun Li for the care they showed me, the support they gave me, and all the fun that we shared.

Finally, I am grateful to my parents and sisters for their love, support, and patience during the past two years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the age of multimedia information, digital images play an important role for storing, representing, and describing information in variety of domains like satellite images, medical images, fingerprints, art collections, trademarks, and etc. However, the increasing use of images is causing problems because the technology for organizing and searching images based on their content is still in its infancy.

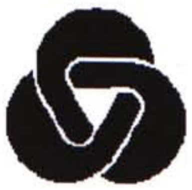
Traditionally, textual features, like filenames, captions, and keywords have been used to annotate and retrieve images. However, there are mainly three problems with these methods.

1. **Human Intervention:** Human intervention is required to describe and tag the contents of the images. As a result, it is not only time-consuming but also potentially error-prone.

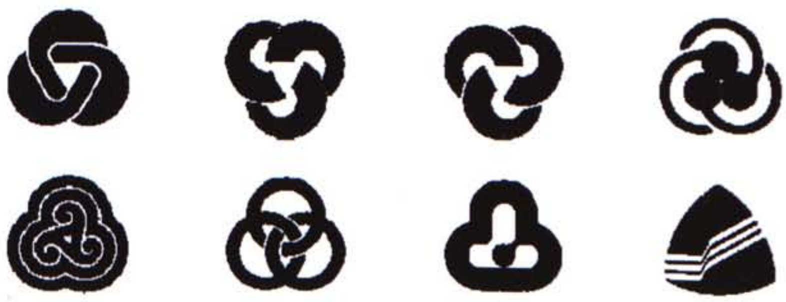
2. **Lack of Standards and Lack of Descriptive Power:** As the size of the image databases grow, the use of keywords becomes very complex and they are usually inadequate to represent the image content because the keywords used are subjective and not unique. Different users may use different words to describe a multimedia data object for retrieval. Even if preselected keywords are used, it is still hard to depict the object clearly and precisely.

3. **Linguistic Barriers:** If the image database is to be shared globally in the world, then the linguistic barriers will make the use of keywords ineffective.

In order to tackle the problems, content-based image retrieval has been proposed in the past few decades. It lets users to specify queries by features such as color, shape, and texture. Figure 1.1 shows an example of image retrieval based on shape features. Given the query image, the system retrieved the similar ones from the database. One may notice that it is very difficult to raise the query in terms of keywords. This shows that it is more effective and easier to use image content for retrieval.



(a)



(b)

Figure 1.1: An example of image retrieval by image content. (a) Query. (b) Top 8 retrievals based on shape features.

1.1 Content-based Image Retrieval

Due to the problems stated above, image retrieval based on visual features is more desirable than text-based retrieval in many applications. Many content-based image retrieval systems have been developed in the past few years, for example, Query By Image Content (QBIC) [50, 13, 15, 1], Photobook [54], VisualSEEk [58], ART MUSEUM [31, 22], CORE [66], Montage [35, 36, 39, 40], and etc. Features like color, shape, and texture are usually used to represent an image for retrieval. Most of the existing content-based image retrieval systems[1] have the system structure as shown in Figure 1.2. The features of the images are extracted in advance and then stored along with the images in the database. When a user raises a query, the features of the query image are first extracted. The features extracted are then matched with the features in the database. A set of images are retrieved and sorted based on the degree of similarity calculated in the matching process.

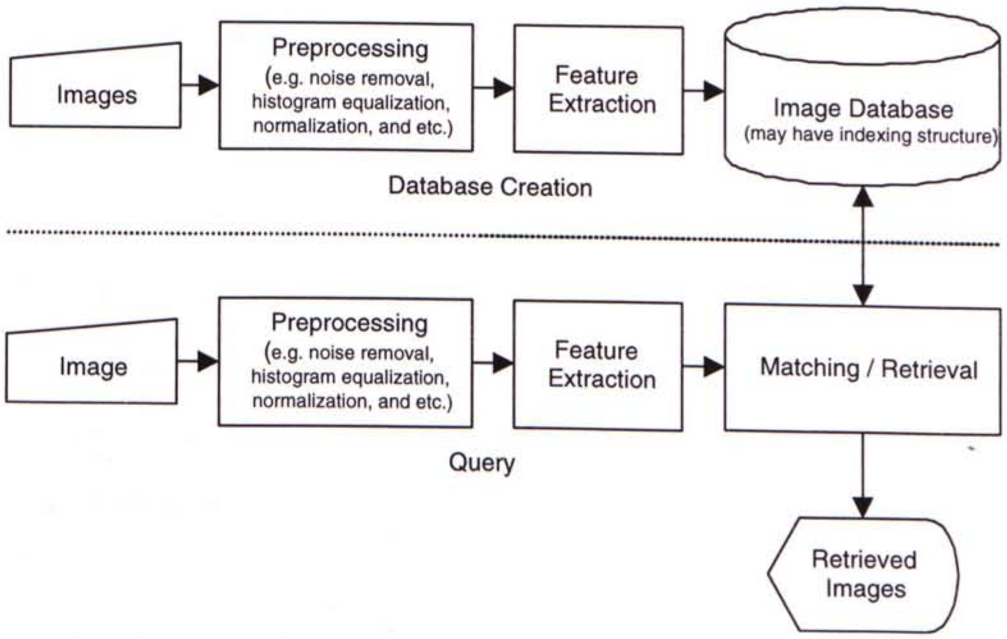

Images
Preprocessing
(e.g. noise removal, histogram equalization, normalization, and etc.)
Feature Extraction
Image Database
(may have indexing structure)
Database Creation
Image
Preprocessing
(e.g. noise removal, histogram equalization, normalization, and etc.)
Feature Extraction
Matching / Retrieval
Query
Retrieved Images

Figure 1.2: A typical content-based image retrieval model.

[1]There may be other modules like services supporting module or relevance feedback module.

1.2 Designing a Shape-based Image Retrieval System

In this research, we would like to investigate the problem of image retrieval based on shape features. In fact, shape similarity has been proven to be a difficult problem [48, 49] in vision applications. The problem remains difficult in content-based image retrieval applications where similarity depends on the particular application.

In designing a shape-based image retrieval system, there are three primary issues that we need to consider. They are *shape representation*, *similarity measurement*, and *retrieval method.*

1. **Shape Representation:** Shape representation is one of the central issues in designing a shape-based image retrieval system. If we want a system to distinguish objects of different types, we must first decide which parameters of the objects will be measured. The particular parameters that are measured are called the features.

   Good features should have four characteristics [5]:

   (a) **Discrimination.** Features should take on significantly different values for objects belonging to different classes.

   (b) **Reliability.** Features should take on similar values for all objects of the same class.

   (c) **Independence.** The various features used should be uncorrelated with each other. While highly correlated features might be combined to reduce noise sensitivity, they generally should not be used as separate features.

   (d) **Small Numbers.** The complexity of an image retrieval system increases rapidly with the number of features used. Adding more features that are noisy with existing features may degrade the performance of the system.

Sometimes, the representation scheme has to satisfy the invariance properties like scaling, rotation, and translation invariance. Sometimes, it does not need to satisfy them. For example, for a trademark image retrieval system, a scaling, rotation, and translation invariant representation scheme should be used. However, for a photograph retrieval system, a user may want to retrieve a picture with an object at certain position. A translation invariant representation may not be suitable in such situation. One has to choose a suitable shape representation scheme to suit a specific application.

2. **Similarity Measure** Given a shape representation scheme, the degree of similarity of any two shapes can be calculated by using a similarity measuring metric. Metric is commonly defined as follow:

   Definition 1.1 (Metric) *A metric [51] on a set A is a function d on $A \times A \rightarrow \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers, which satisfies the following conditions for all $(x, y) \in A \times A$ and $z \in A$:*

   (a) $d(x, y) \geq 0$

   (b) $d(x, y) = 0$ *iff* $x = y$

   (c) $d(x, y) = d(y, x)$

   (d) $d(x, y) \leq d(x, z) + d(z, y)$

   L_2-norm (Euclidean distance) is one of the most common metric distance functions and it is defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{1.1}$$

   In the above definition, it is assumed that the feature values are numerical. In case of binary features or symbolic features, maybe we need to transform them into some numerical representations before calculating the similarity. A similarity measure is not only necessary that the computation method should be efficient, but also the degree of similarity calculated should produce visual ranking of shapes which is similar to human perception.

3. **Retrieval Method:** Given a representation scheme and its similarity measure, we then have to determine whether an indexing structure can be used or not. Some researchers have proposed to use indexing trees, e.g., R-tree [20], R+-tree [57], R*-tree [2], Quad-tree [14], k-d tree [3], or VP-tree [67], to increase the speed of searching. Some have proposed to use filtering approaches [6, 30] in which, in the first stage, simple and easily computable shape features are used to quickly browse through the database to generate a moderate number of plausible retrievals and in the second stage, the candidates from the first stage are screened using a more accurate measuring method to discard spurious matches. The target is that a linear search in the database can be avoided. A good design for retrieval is important especially for a real-time application.

1.3 Information on Trademark

1.3.1 What is a Trademark?

A trademark is either a word, phrase, symbol or design, or combination of words, phrases, symbols or designs, which identifies and distinguishes the source of the goods or services of one party from those of others. A service mark is the same as a trademark except that it identifies and distinguishes the source of a service rather than a product [63]. Most of the trademarks are an abstract drawing of an animal, or a natural object like sun, moon or etc. The total number of registered trademarks and service marks is over a million in U.S. alone and the number is continuing to grow, therefore, the task of registering a new trademark becomes difficult without inadvertent infringement of copyright of an existing trademark.

1.3.2 Search for Conflicting Trademarks

Currently, the trademarks are classified manually by grouping into several similar shapes according to a specific class order, followed by performing the matching process manually by the operator. The problem is that Trademark Registry has to ensure a new trademark to be registered is sufficiently different from those existing trademarks to avoid confusion of the public.

Owing to the large intake of applications, there is a time lag of about 10-14 months before the Registry could issue a report advising the applicant or his authorized agent of the outcome or result of search and examination [27]. Therefore, there is a great demand for an automatic trademark retrieval system based on shape features.

According to Intellectual Property Laws of Hong Kong, a trademark shall not be registered if:

1. the trademark is similar to an earlier trademark;

2. the goods or services for which the application for registration is made are identical or similar to those for which the earlier trademark is protected; and

3. the use of the trademark in relation to those goods or services is likely to cause confusion on the part of the public.

In case of trademarks consisting of symbols and designs, the issue of similarity is primarily based on visually similarity. In the trademark context, "distinctive" means unique enough to reasonably serve as an identifier of a product in the market place.

1.3.3 Research Scope

A trademark is a complex pattern, consisting of various shapes and text. Trademarks can be divided into three main types as shown in Figure 1.3. A word-in mark is a trademark that contains only characters or words in the mark. A device-mark contains graphical or figurative elements only. A composite-mark consists of characters or words and graphical elements. For the types of word-in mark and composite-mark, text annotation is adequate for the retrieval from the database [34]. Since the trademarks are registered in binary image form, color does not play a useful role in distinguishing between various marks. Our system focuses on retrieving device-mark types only based on shape.

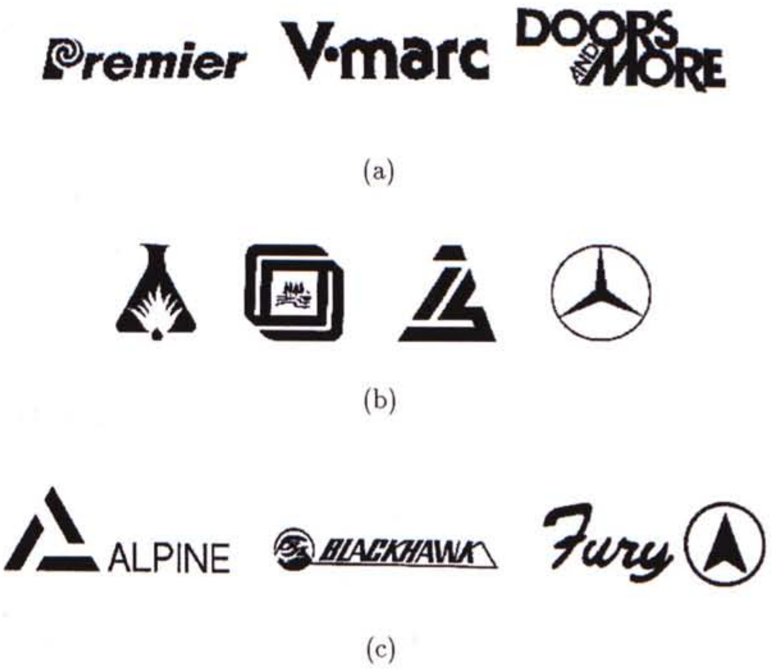


Figure 1.3: Types of trademarks. (a) Word-in marks. (b) Device-marks. (c) Composite-marks.

Our target is to build an automatic, efficient and accurate trademark retrieval system based on shape. When the operator enters a new mark to the system, he/she can make sure that no identical trademark is already in the system in disregard of its orientation angle or scale. Furthermore, even if there is no identical one, he/she may want to see how similar shaped ones are already in the system. The search process should be robust to noise in image or minor variations or deformations in its shape.

1.4 Information on Chinese Cursive Script Character

Chinese calligraphy is rare and invaluable in the history of Chinese civilization. Graphically, it is an abstract art and has a rich variety of form and design. From a practical point of view, it is a kind of written language. There are five styles of Chinese calligraphy. Among them, the cursive script style most closely approaches abstract art. Figure 1.4 shows a page of Chinese cursive style calligraphy [41]. The features shared by all cursive styles are a simplified structure, running together of strokes, rapidly written and a low level of legibility. In this research, we try to build a system for Chinese cursive script character image retrieval.

1.5 Problem Definition

In this research, we would like to build a trademark image retrieval system and a Chinese cursive script character image retrieval system. A trademark logo is registered in the binary form at the Trademark Registry and a cursive script character image is also a binary image; therefore, shape features are used in the systems. In the two systems, we use two different sets of features to represent the images. We will discuss more in Chapter 5 and Chapter 6.

When designing an image retrieval system, use of a single image attribute for retrieval may lack sufficient discriminatory information; therefore, the results may not be satisfactory especially for large databases. In order to increase the

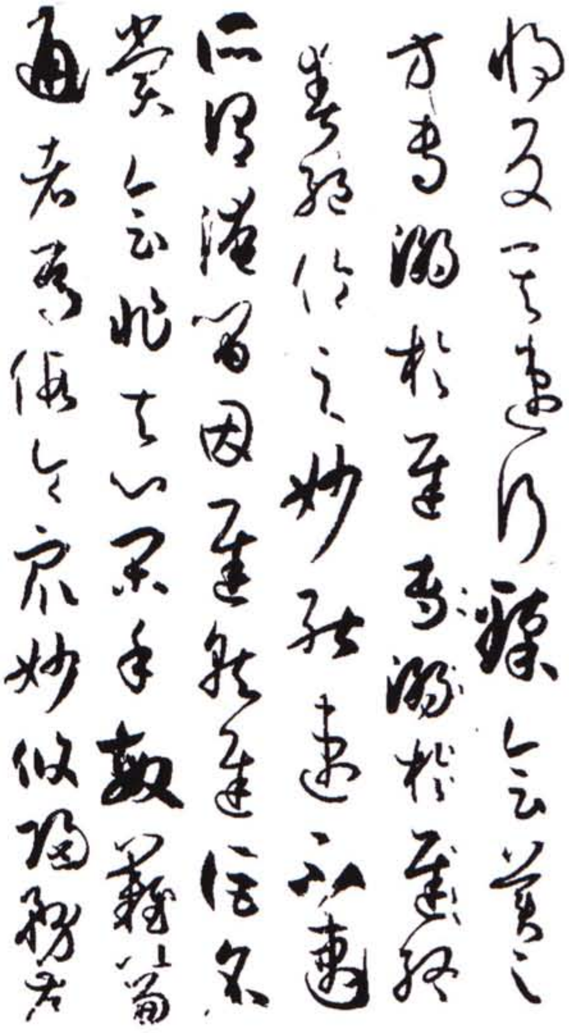


Figure 1.4: A page of cursive script style Chinese calligraphy

accuracy of the retrievals, it is often necessary to integrate the results obtained from the query based on individual features. However, the importance of various features are often difficult to determine. To tackle these shortcomings, we propose to integrate several shape features and suggest a technique to find the weighting factors of the dissimilarity function for image retrieval. The proposed technique has been applied to a trademark database with 1400 images as well as a Chinese cursive script character database with 1400 images. Appendix A and Appendix B show some representative images in our trademark image database and Chinese cursive script character image database, respectively.

The problems that we are facing are:

1. to find a way to integrate different shape features so as to increase the accuracy of image retrieval.

2. to determine the weighting factors of different shape features in the dissimilarity function for image retrieval.

3. to find a set of effective features to represent a trademark image and build a trademark retrieval system.

4. to find a set of effective features to represent a Chinese cursive script character image and build a cursive script character image retrieval system.

1.6 Contributions

The main contributions of this work for solving the defined problems are summarized as follows:

1. **Integration of Shape Features:** We proposed to integrate the results of different shape-based retrievals by combining the associated dissimilarity values. A supervised learning technique using genetic algorithm (GA) for finding the weights in the dissimilarity function has been proposed [8, 7]. The proposed method can be used to assign suitable weights for different image features for image retrieval.

2. **Trademark Image Retrieval System:** A prototype system for trademark retrieval based on shape features has been built [8]. This system uses features including invariant moments, Fourier descriptors of approximated boundary, eccentricity and circularity. The features can be extracted by the system

automatically. The weighting factors for different shape features are found by the proposed GA.

3. **Chinese Cursive Script Character Image Retrieval System:** A prototype system for Chinese cursive script character image retrieval has been developed [7]. Several shape features including histogram of edge directions, invariant moments and eccentricities are used to increase the accuracy and robustness of the system. Shape features of the original, thinned, and linearly normalized image are used. The weights for different shape features are found by the proposed GA.

4. **Shape Feature Extraction Toolbox for Matlab V5.3:** A toolbox for shape feature extraction has been built. The toolbox can be used to extract various shape features, for example, eccentricity, rectangularity, circularity, Fourier descriptors, curvature scale space image, Hu's seven moments, and etc, from binary images. The toolbox is attached in Appendix C.

Experiments have been conducted and the results show that:

1. The weights found by our proposed GA improved the accuracy of retrieval.

2. Our trademark image retrieval system was robust under deformed, rotated, scaled, mirrored, and hand drawn queries. In addition, the trademarks retrieved by our system agreed well with human perception. Each query took 2.3 seconds on average.

3. Our cursive script character image retrieval system was robust under deformed and scaled queries. Each query took 11.7 seconds on average.

1.7 Thesis Organization

The rest of the thesis is organized as follows. A literature survey on several shape-based image retrieval systems is reviewed in Chapter 2. The background information for shape representation and matching is covered in Chapter 3. Chapter 4 describes our proposed genetic algorithm for weight assignment in dissimilarity function for image retrieval. Chapter 5 presents our trademark image retrieval system and some experimental results. Our Chinese cursive script character image retrieval system and its experimental results are discussed in Chapter 6. Finally, conclusion and some ideas for future research are presented in Chapter 7.

Chapter 2

Literature Review

There were various shape-based image retrieval systems proposed in the past few years. Considerable works have been done on shape representation, matching, and similarity measuring problems. In this chapter, several shape-based image retrieval systems developed in the past few years are reviewed. This chapter is then summarized with Table 2.1.

2.1 Trademark Retrieval using QBIC Technology

The QBIC (Query By Image Content) system [50, 15, 13, 1] accepts queries using color percentages, color layout, texture, and shape. It serves as an image database filter so as to reduce the search complexity for the users.

QBIC uses the geometric features to represent the shape of an object. These include area, circularity, eccentricity, major axis orientation, and invariant moments. The features are computed from binary images representing individual objects. The distance between two images is computed by using a weighted Euclidean distance with the inverse of feature variances for normalization. For example, the distance between image i and j is defined as

$$d_{ij} = \sum_k \frac{(k_i - k_j)^2}{\sigma_k^2} \,, \tag{2.1}$$

where k is a feature.

Based on the QBIC technology, a trademark retrieval system [26] on the basis of shape features is developed. For each trademark, the operator segments the important boundary shape components of it for feature extraction. Figure 2.1 shows an example of the representative components. The shape features are then stored in the database. The user can query by using one representative component as shown in Figure 2.2. The results are then displayed in the order of similarity. Up to now, there is no report for the performance of the system. A demonstration trademark search system can be found at [26].

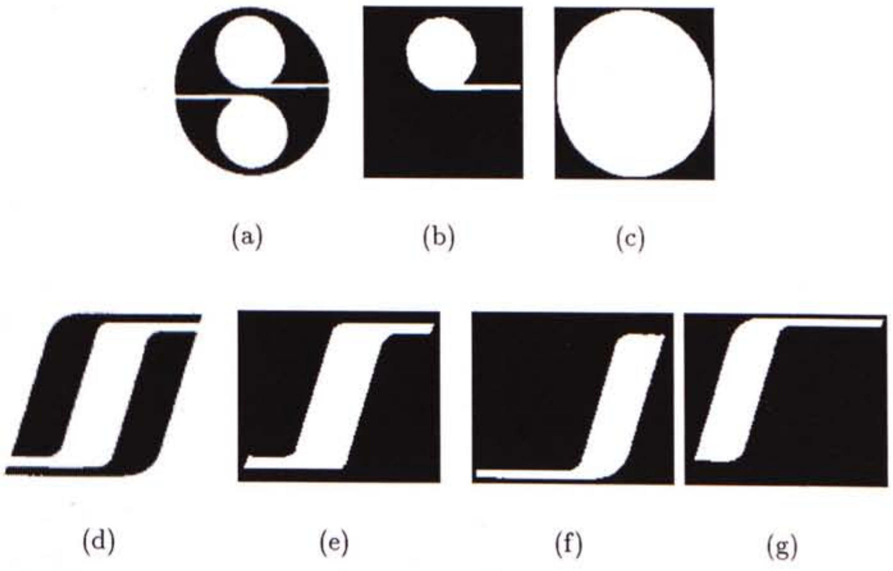
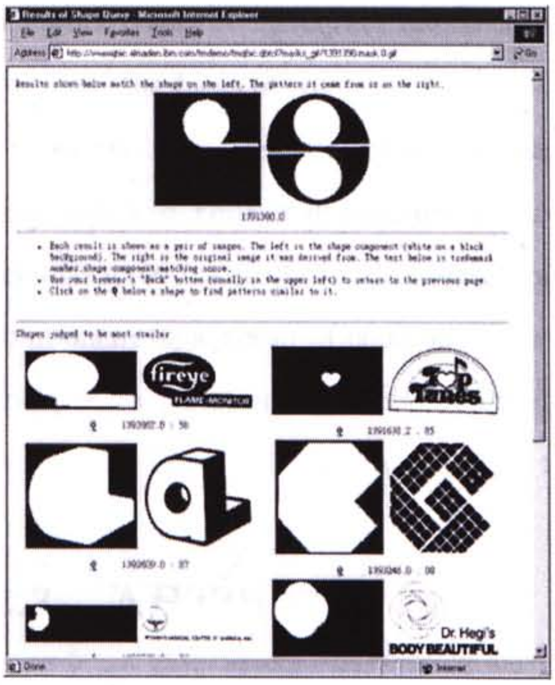


Figure 2.1: Boundary shape components are used to represent a trademark. (a) and (d) are two original images; (b), (c), (e), (f) and (g) are the corresponding representative boundary shapes.



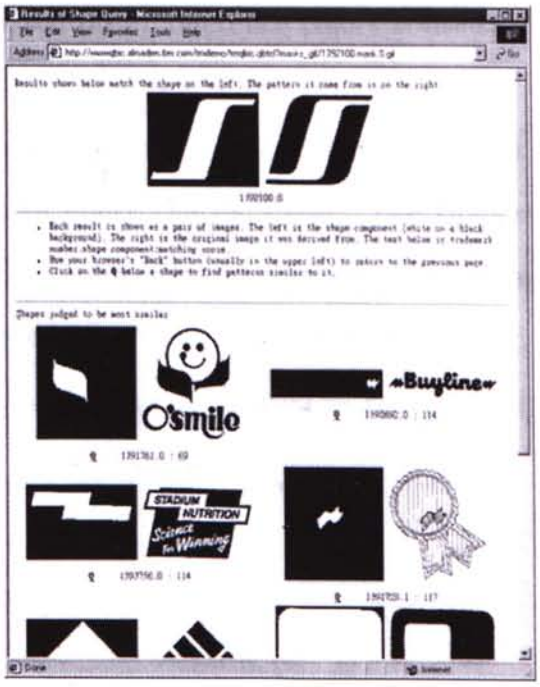
Results of Shape Query - Microsoft Internet Explorer

Results shown below match the shape on the left. The pattern it came from is on the right.

1391390.0

- Each result is shown as a pair of images. The left is the shape component (white on a black background). The right is the original image it was derived from. The text below is trademark number.shape component:matching score.
- Use your browser's "Back" button (usually in the upper left) to return to the previous page.
- Click on the below a shape to find patterns similar to it.

Shapes judged to be most similar



Results of Shape Query - Microsoft Internet Explorer

Results shown below match the shape on the left. The pattern it came from is on the right.

1392100.0

- Each result is shown as a pair of images. The left is the shape component (white on a black background). The right is the original image it was derived from. The text below is trademark number.shape component:matching score.
- Use your browser's "Back" button (usually in the upper left) to return to the previous page.
- Click on the below a shape to find patterns similar to it.

Shapes judged to be most similar

(a)

(b)

Figure 2.2: Trademark queries using a representative component in the QBIC system.

2.2 STAR

STAR (System for Trademark Archival and Retrieval) [65, 38] uses a combination of color and shape features for retrieval purposes. The color of an image is represented in terms of the R, G and B color components and the shape is represented in terms of combination of outline-based features (sketch of the images) and region based features (objects in an image). The features used to describe both the color and shape in an image are non-information preserving which means the features cannot be used to reconstruct the image. Though these features cannot be used to reconstruct the image, they are useful as approximate indicators of shape.

For the color components of a trademark, a histogram of reference colors is used to represent the colors in an image. These reference colors are selected manually and the color space is divided to represent bins corresponding to each reference color. A histogram intersection based on 27 reference colors is used to query on the basis of color.

For the shape features of a trademark, multiple features including invariant moments and Fourier descriptors are used to query the database. The shapes are extracted manually by the user through an interactive user interface. Once a component or a group of components is selected by the user, they are extracted and the image features are computed and stored. An integrated retrieval based on the two features is used to query on the basis of shape.

2.3 ARTISAN

The ARTISAN (Automatic Retrieval of Trademark Images by Shape ANalysis) [12, 11] project aims at developing and evaluating a prototype shape retrieval system for trademark images consisting of abstract geometric designs.

Given an image, the system first extracts the region boundaries and approximates them by straight-line and circular-arc segments. After removing the anomalies caused by noise in boundary representations, it groups the region boundaries into families and constructs envelopes for proximal boundary families. Then the global shape features including aspect ratio, circularity, transparency, relative area, right-angleness, sharpness, complexity, directedness and straightness are extracted and stored in the database. They claimed that the entire segmentation and feature extraction process is automatic, requiring no human intervention at any stage.

The users can raise a query by selecting an image and runtime search parameters. The system then extracts appropriate shape features from the query image and computes appropriate similarity scores between the query and the stored images by shape feature matching. The most similar images are then displayed on the screen. From their report, we cannot find how they integrate the results of retrievals using different shape features.

The ARTISAN system was tested with 10745 images. From their report [11], the average retrieval effectiveness scores (mean and standard deviation) for 24 queries were that normalized recall $R_n = 0.90$ and normalized precision $P_n = 0.63$. Although the current ARTISAN prototype does not yet offer reliable enough performance, they are working on enhancing the retrieval effectiveness.

2.4 Trademark Retrieval using a Visually Salient Feature

Kim *et. al.* [34, 33] presented a method for trademark retrieval based on shape feature. Zernike or pseudo-Zernike moments of an image are used as a feature set. To retrieve similarity shaped trademarks, they introduced the concept of a "visually salient feature" that dominantly affects the global shape of the trademarks. The visually salient feature is determined by the probabilistic distribution model of a trademark database.

They tested their system with 3000 trademarks. All trademark images were preprocessed by binary thresholding and normalized to the size of 100 by 100 pixels. 90 Zernike moment features are used to represent a trademark in their database. The average precision was 0.65 and the average recall rate of the original trademarks among the top 30 candidates queried by noisy or deformed trademarks was 0.925.

The advantages of using most salient feature for trademark retrieval are fast and robust. The similar trademarks can be retrieved quickly and the method is robust to the minor transformation of shape. However, the authors do not expect their system to handle a database of more than 3000 trademarks with the most salient feature only and they suggested using additional shape features to handle more diverse and complex shapes.

2.5 Trademark Recognition using Closed Contours

Peng and Chen [53] proposed to use closed contours for trademark recognition. Trademarks are first decomposed into complete sets of elementary closed contours, each of which is then encoded as an angle-code string according to chain-code information. A two-step string matching algorithm is then employed to compute similarities between closed contours. Finally, the maximum and average terms of the similarities between contours are integrated into the whole trademark similarity measure.

Their method to encode closed contours is insensitive to translation, rotation and scale variations. Moreover, it can cope with the special resemblances such as inclusion, mirror and partial resemblances. A database of 250 trademarks was tested in their experiment. Each trademark was scanned into a 128×128 pixels binary image. 35 trademarks were tested and the average retrieval accuracy is 82.5%.

2.6 Trademark Retrieval using a Two Stage Hierarchy

Jain and Vailaya [30] proposed a method for trademark retrieval using a two-stage hierarchy. In the first stage, simple and easily computable shape features including edge direction histogram and invariant moments are used to quickly browse through the database to generate a moderate number of plausible retrievals when a query is presented. In the second stage, the candidates from the first stage are screened using a deformable template matching process to discard spurious matches. They also proposed to use only the gross details in the image by using filled-in query image features. Figure 2.3 presents several images and their corresponding filled-in images.

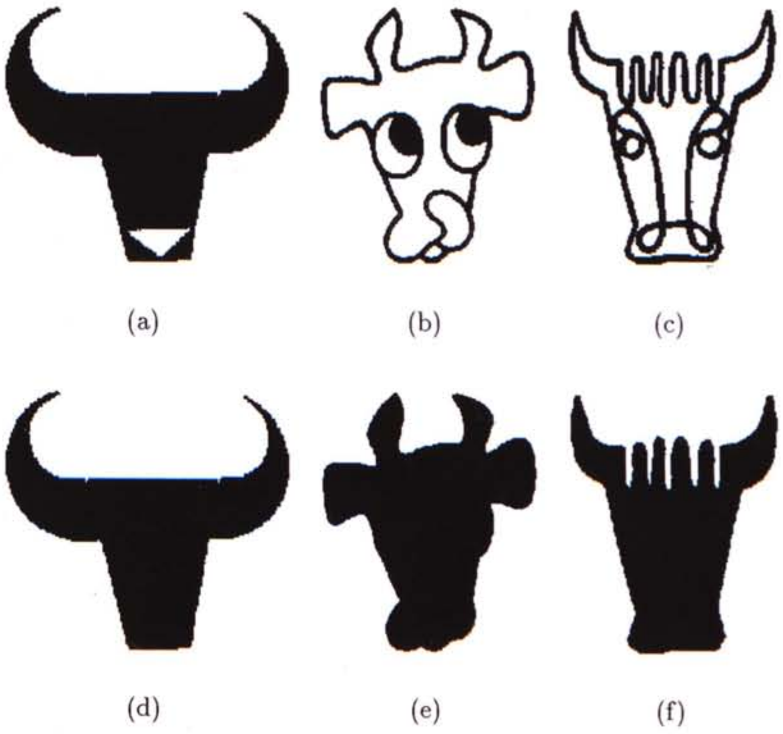


Figure 2.3: Image filling as a heuristic for trademark retrieval. (a), (b) and (c) are the original images; (d), (e) and (f) are the corresponding filled-in images.

In their system, the results of retrieval are integrated by combining the associated dissimilarity values. The integrated dissimilarity index S_t is defined as

$$S_t = \frac{w_e \times S_e + w_m \times S_m}{w_e + w_m}, \tag{2.2}$$

where w_e and w_m are the weights assigned to the edge direction-based similarity and the invariant moment-based similarity, respectively, and S_e and S_m are the dissimilarity values for edge direction-based retrieval and invariant moment-based retrieval, respectively. In their testing, they used equal weights ($w_e = w_m = 1$).

Their methods were tested with a database of 1100 images. In the first experiment, 5 images were used as queries to tested the system. The recall rate of rotated, scaled and noised queries were 98.5%, 100%, and 100% respectively among the top 20 matches. In the second experiment, the recall and precision rates for 3 query images were tested for the image filling heuristic and they found that retrievals based on image filling had higher accuracy and precision rates.

2.7 Logo Matching using Negative Shape Features

Soffer and Samet [60, 59] proposed a method to represent and match logos based on positive and negative shape features. Negative shape features are features that represent an object that consists of several components enclosed in a simple geometric structure based on its interior with the components considered as holes. A rectangular border around each logo that is the minimum bounding box of image with 4 pixels is added to each side. Figure 2.4 shows an example of positive and negative components. Several shape descriptors are used to represent a shape. They include first invariant moment, circularity, eccentricity, rectangularity, hole area ratio (HAR), horizontal gap ratio (HGR), and vertical gap ratio (VGR). They did not mention the method of integrating different shape features for calculating the similarity in their report.

Three methods were used to compute the similarity in their experiments. The first method does not add a border around the logos and compares logos using all the components for each logo. The second method chooses one component to represent each logo. The representative component is selected by the user, and it can be a negative component. For the third method, a border is added around each logo and comparison is done by using all the components for each logo. The similarity between two logos is computed based on the set of feature vectors using

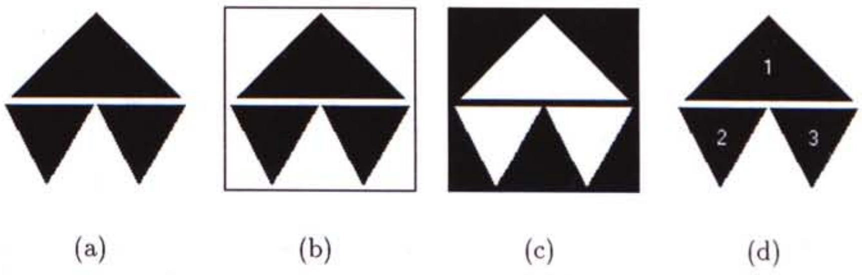


Figure 2.4: An example of positive and negative components: (a) original logo; (b) logo with border; (c) negative components; (d) positive components.

the multi-component similarity measure.

The methods were tested with a database of 130 logos. They reported that using positive and negative shape features is very effective for logo similarity matching than using one component or no border representation methods.

2.8 Chapter Summary

Although many of the systems provided examples of system outputs, recall and precision in their reports, their evaluation methods are different from one another; the database sizes and the database images are not the same; and the test cases are not the same as well. Therefore, it is difficult to compare the effectiveness of these different approaches to the problem of image retrieval. In Table 2.1, we try to summarize the characteristics and performance of the systems.

Table 2.1: Summary of the characteristics of different shape-based trademark retrieval systems.

System	**Feature(s) Used**	**Multiple (M) / Single (S) Component(s)**	**Automatic Feature Extraction**	**Setup / Results**
Trademark Retrieval using QBIC technology	Area, circularity, eccentricity and invariant moments	M	No	1000 images
STAR	Color histogram, invariant moments, Fourier descriptors	M	No	100 images
ARTISAN	Aspect ratio, circularity, transparency, relative area, right-angleness, sharpness, complexity, directedness and straightness	M	Yes	10475 images; 24 queries; normalized recall = 0.90, normalized precision = 0.63
Trademark Retrieval using a Visually Salient Feature	Zernike or pseudo Zernike moments	S	Yes	3000 images; 90 Zernike moment features used; average recall = 0.925; average precision = 0.65
Trademark Shape Recognition using Closed Contours	Angle-code string according to chain-code information of closed contours	M	Yes	250 images; 35 spurious trademarks are tested; retrieval accuracy = 0.825
Trademark Retrieval using a Two Stage Hierarchy	Invariant Moments, edge direction histogram	S	Yes	1100 images; recall for rotated, scaled and noised queries were 98.5%, 100%, 100%, respectively among the top 20 matches
Logo Matching using Negative Shape Features	Positive and negative shape features include invariant moments, circularity, eccentricity, rectangularity, horizontal gap ratio (HGR), vertical gap ratio (VGR), and hole area ratio (HAR)	M	No	130 images; using positive and negative shape features is more effective than using either of them.

Chapter 3

Background on Shape Representation and Matching

In this chapter, we deal with the shape representation methods of *binary* images, that is to say images where the pixel is either black or white. This is because our target is to retrieve binary images from an image database.

Given a binary image, the information of it can be stored in a more compact form. For example, the boundary pixels of an object contain all of its information, and hence, it is sufficient to store the boundary pixels of an object. The boundary pixels here can be used as a feature set for representing the image. Obviously, this is a much more compact representation of a binary image. However, if there are many objects inside an image, the representation method can be more complex.

Feature extraction is the process to extract features in a content-based image retrieval system. Users may want to retrieve database objects similar to a query in terms of some kinds of features. Therefore, the useful features of an object is usually extracted in advance and stored in the database as a feature vector. Feature extraction is defined as follow:

Definition 3.1 (Feature Extraction) *Given an image I and a set of feature* *parameters* $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^{n}$*, a feature extraction function* f *is defined as*

$$f : I \times \boldsymbol{\theta} \rightarrow \mathcal{R}^d \ , \tag{3.1}$$

which extracts a real-valued d-dimensional feature vector.

Many features can be used in content-based image retrieval, such as color, sketch, texture and shape. In this research, we focus on the problem of shape-based image retrieval. Some of the shape features, shape representation methods and matching methods including some simple geometric features, Fourier descriptors, chain code, invariant moments, Zernike moments, edge direction histogram, and Curvature Scale Space (CSS) representation are reviewed in this chapter. This chapter is then summarized with Table 3.1.

3.1 Simple Geometric Features

There are some simple geometric features that can be used to represent shapes. For example, circularity, rectangularity, hole area ratio (HAR), horizontal gap ratio (HGR), vertical gap ratio (VGR), central moments, major axis orientation and eccentricity are all very common to be used.

3.1.1 Circularity

Idea

Circularity reflects the complexity of the boundary of an object. Given a shape, circularity [28] is defined as

$$c = \frac{p^2}{A} \ , \tag{3.2}$$

where A is the area of the shape and p is the perimeter of the shape.

Discussion

Generally, this measurement shows large values for elongated objects and it is roughly correlated to the subjective concept of complexity of the boundary. For example, circularity is minimum and equals 4π for a circle. Circularity is 16 for a square and is $12\sqrt{3}$ for an equilateral triangle. Circularity is invariant to translation, rotation, and scaling. Thus, it is useful to distinguish objects independent to their orientation, position, and size. The area A and perimeter p of an object can be found in $O(WH)$, where W and H are the width and the height of a frame buffer, respectively. Therefore, the computational complexity is $O(WH)$ and the storage complexity of this representation method is $O(1)$. This method is indexable.

3.1.2 Rectangularity

Idea

Given an object, rectangularity R [5, 59] is defined as

$$R = \frac{A}{A_R}. \tag{3.3}$$

where A is the area of the object and A_R is the area of its minimum enclosing rectangle.

Discussion

It represents how well an object fills its minimum enclosing rectangle. This parameter takes on a maximum value of 1.0 for rectangular objects. The computational complexity is $O(WH)$, where W and H are the width and the height of a frame buffer, respectively. This feature is invariant to rotation, scaling and translation. It is indexable and the storage complexity of this representation method is $O(1)$.

3.1.3 Hole Area Ratio

Idea

Given an image, hole area ratio HAR [59] is defined as

$$HAR = \frac{A_h}{A}\,, \tag{3.4}$$

where A is the area of the object in the image and A_h is the total area of all holes in the object.

Discussion

Hole area ratio is effective in discriminating between symbols that have big holes and symbols with small holes. This feature is scale, rotation, and translation invariant. The computational complexity is $O(WH)$, where W and H are the width and the height of a frame buffer, respectively. The storage complexity of this representation method is $O(1)$. This feature is indexable.

3.1.4 Horizontal Gap Ratio

Horizontal gap ratio HGR [59] is defined as

$$HGR = \frac{HG^2}{A}\,, \tag{3.5}$$

where A is the area of the component in the image and HG is the number of horizontal gaps in the component (i.e. the number of pixels in the object whose right neighbor is a hole).

Discussion

This feature discriminates among symbols based on the shape of the holes themselves. This feature is scale and translation invariant but it is not rotation invariant. The computational complexity is $O(WH)$, where W and H are the width and the height of a frame buffer, respectively. The storage complexity of this representation method is $O(1)$. This feature is indexable.

3.1.5 Vertical Gap Ratio

Vertical gap ratio VGR [59] is defined as

$$VGR = \frac{VG^2}{A} , \tag{3.6}$$

where A is the area of the component in the image and VG is the number of vertical gaps in the component (i.e. the number of pixels in the object whose bottom neighbor is a hole).

Discussion

This feature discriminates among symbols based on the shape of the holes themselves. It has the same characteristics as the horizontal gap ratio.

3.1.6 Central Moments

Idea

Given a binary image $f(x, y)$, the moments of order $(p + q)$ of it are defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) , \tag{3.7}$$

where $p, q = 0, 1, 2, ...$

The central moments [18] of $f(x, y)$ are defined as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) , \tag{3.8}$$

where $\bar{x} = m_{10}/m_{00}$, $\bar{y} = m_{01}/m_{00}$.

Discussion

The central moments μ_{pq} can be expressed in terms of the moments m_{pq} [24]. The central moments are computed using the center of gravity as the origin; therefore, they are invariant to translation. In the case where the area of the objects is normalized to a constant c, they are scale invariant. However, they are not rotation invariant. For each shape, only the computed moment will be stored and thus the storage requirement is $O(1)$ and the computational complexity of the shape moment is $O(WH)$, where W and H are the width and the height of a frame buffer, respectively. The shape moment is just a number and hence it can be easily indexed for efficient retrieval. However, the discrimination power of a single moment is not good because it is not unique. Two different shapes may have the same moment value. Therefore, a large set of moments is usually required to distinguish similar shapes.

3.1.7 Major Axis Orientation

Idea

The orientation [28] of an object is defined as the angle between the x axis and the axis around which the object can be rotated with minimum inertia. The object is most elongated in this direction. This angle is given by

$$\theta = \frac{1}{2}\, tan^{-1} \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\ , \tag{3.9}$$

where μ_{pq} is the central moment of order $(p+q)$ of an image which is defined in Eq. 3.8.

Discussion

Orientation is useful for image retrieval with a particular angle. It can be calculated in $O(WH)$, where W and H are the width and the height of a frame buffer, respectively and it is indexable. It is invariant to translation and scaling.

3.1.8 Eccentricity

Idea

The ratio of the major axis to the minor axes is called eccentricity [28] and is defined as

$$\epsilon = \frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}{(\mu_{20} + \mu_{02})^2}, \tag{3.10}$$

where μ_{pq} is the central moment of order $(p+q)$ of an image which is defined in Eq. 3.8.

Discussion

The value of eccentricity ranges from 0 to 1. It is zero for a circular object and one for a line-shaped object. Eccentricity is invariant to translation, rotation, and scaling. Eccentricity can be calculated in $O(WH)$, where W and H are the width and the height of a frame buffer, respectively, and it can be indexed.

3.2 Fourier Descriptors

Idea

Fourier descriptors are complex coefficients of the Fourier series expansion of waveforms [68, 18, 55]. Given a shape in the Cartesian plane, the boundary points are re-sampled to obtain an l-point closed boundary as shown in Figure 3.1. Each coordinate pair can be treated as a complex number $z = x + jy$. The coefficients of Discrete Fourier Transform (DFT) of this complex vector,

$$z_u = \frac{1}{l} \sum_{m=0}^{l-1} z_m e^{-\frac{2\pi jmu}{l}} \tag{3.11}$$

are known as the Fourier descriptors of the boundary. The first coefficient gives the centroid of the object's boundary. With increasing index, the Fourier descriptors give increasingly fine details of the boundary.

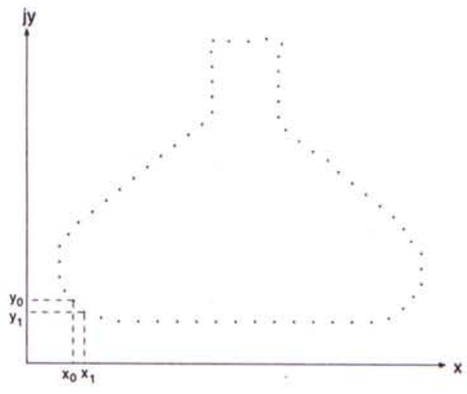


Figure 3.1: A re-sampled boundary and its representation as a complex sequence. The points (x_0, y_0) and (x_1, y_1) are (arbitrarily) the first two points in the sequence.

Discussion

This representation has the great advantage that it reduces a 2-D problem to a 1-D problem. Besides, the first few Fourier descriptors can be used to capture the gross essence of a boundary. Thus these coefficients carry shape information and can be used as the basis for differentiating between distinct boundary shapes. This method is sensitive to sampling error and l is usually set to 64 or 128 for representing a boundary shape.

The Fourier descriptors are invariant to the starting point of sampling, rotation, scaling as well as reflection. Except for the first coefficient which gives the centroid of the object, all Fourier descriptors are translation invariant. The storage complexity of this method is $O(l)$ where l is the number of points in the normalized planer curve. The computational complexity is $O(l \log(l))$ by using the Fast Fourier Transform (FFT) algorithm and this method is indexable.

3.3 Chain Codes

Idea

Chain codes [16, 18] can be used to represent a boundary of a shape by a connected sequence of straight-line segments of specified length and direction and has been used in several works such as [52, 61]. This representation is based on the 4- or 8- connectivity of the segments. The chain code of a digitized shape is generated by using the turning directions of its boundary pixels. The direction of a shape is coded by using a numbering scheme like the ones shown in Figure 3.2. Figure 3.3 illustrates the computation of chain code.

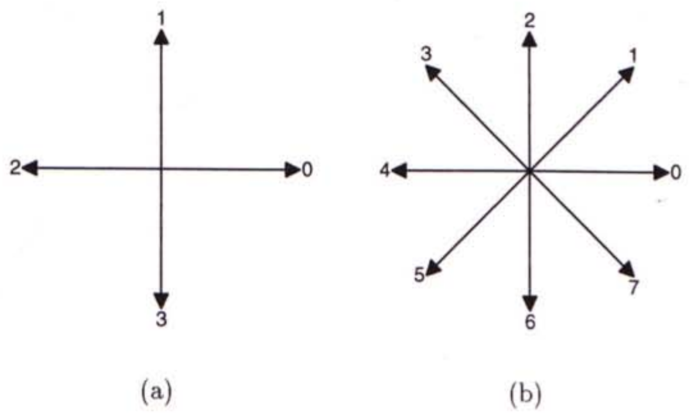
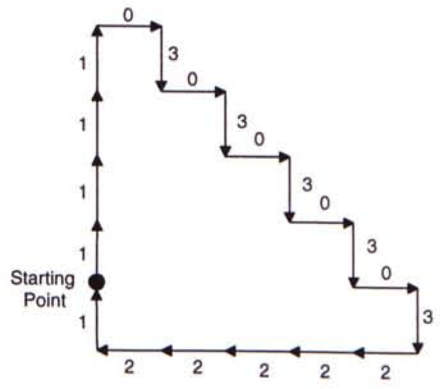


Figure 3.2: Chain code numbering schemes: (a) Directions for 4-directional chain code. (b) Directions for 8-directional chain code.



Chain Code:	11110303030303222221
Normalized Chain Code:	03030303032222211111
Difference:	33131313133000030000

Figure 3.3: An example of the 4-directional chain code computation.

Discussion

The chain code of a boundary depends on the starting point. The code can be normalized by the following way: A chain code generated can be treated as a circular sequence of direction numbers; therefore, the starting point can be redefined so that the resulting sequence of numbers forms an integer of minimum magnitude. Alternatively, the chain code can be normalized for rotation by using the first difference of the chain code instead of the code itself. The difference is obtained simply by counting (counterclockwise) the number of directions that separate two adjacent elements of the code (See Figure 3.3).

The matching of shapes is performed by substring matching of the chain code. A two-step string-matching procedure [37] can be used to determine the similarity between two chain codes. The similarity s proposed by [37] is defined as:

$$s = \frac{c}{m}\,, \tag{3.12}$$

where c is the maximum number of the common angle codes in the two closed contours and m is the maximum length of the two strings.

This method is translation and rotation invariant. Scale invariance can be achieved by normalization of the original shapes. The storage complexity of this method is $O(k)$ where k is the number of boundary points of the digitized shape. The computational complexity is also $O(k)$. This method is indexable since the representations are strings.

3.4 Seven Invariant Moments

Idea

From the second-order moments and third-order moments, a set of seven invariant moments has been derived [24]:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} && (3.13)\\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 && (3.14)\\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{21} - \eta_{03})^2 && (3.15)\\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} - \eta_{03})^2 && (3.16)\\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (3.17)\\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\
&\quad 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) && (3.18)\\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (3.19)
\end{aligned}$$

where η_{pq} are normalized central moments defined by

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(p+q+2)/2}} \qquad (3.20)$$

for $p + q = 2, 3, \ldots$, and μ_{pq} is the central moment of order $(p + q)$ of an image which is defined in Eq. 3.8.

Discussion

These features are invariant under rotation, scale, translation, and reflection of images and have been widely used in a number of applications [10, 30, 25] due to their invariance properties. The storage complexity of this representation method is $O(1)$ and the computational complexity is $O(WH)$ where W and H are the width and the height of a frame buffer respectively. This method is indexable.

3.5 Zernike Moments

Idea

Zernike moments are complex orthogonal moments whose magnitude has rotational invariant property [33, 32]. Since they are orthogonal, they are better than other moments, like central moments that are non-orthogonal, in terms of information redundancy.

Zernike moments are defined inside the unit circle and the radial polynomial vector $R_{nm}(\rho)$ is defined as

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}\rho^{n-2s} , \tag{3.21}$$

where n=0, 1, 2, ..., ∞, $|m| \leq n$, and $n - |m|$ is even. Then the two-dimensional Zernike moment of order $(n+m)$ of an image, $f(\rho, \theta)$, in polar coordinate is defined as

$$A_{nm} = \frac{n+1}{\pi} \sum_{\rho} \sum_{\theta} [V_{nm}(\rho,\theta)]^* f(\rho,\theta), \quad s.t. \quad \rho \leq 1 , \tag{3.22}$$

where $V_{nm}(\rho, \theta)$ is a Zernike basis polynomial defined as

$$V_{nm}(\rho,\theta) = R_{nm}(\rho) e^{-jm\theta}. \tag{3.23}$$

Discussion

The order of Zernike moments is determined by two parameters; n and m. Each order of Zernike moment is computed by multiplying the radial polynomial to the phase term.

Zernike moments are scale, rotation, and translation invariant. For each order of Zernike moment, the storage complexity is $O(1)$[1] and the computational complexity is $O(WH)$, where W and H are the width and height of a frame buffer respectively, by the area based method using a lookup-table [33].

[1]Usually, a large set of moments are required to distinguish similar shapes.

3.6 Edge Direction Histogram

Idea

A histogram of edge directions [29, 64] can be used to represent the global information of a shape. The edge direction information can be obtained by using the Canny edge operator [4]. Figure 3.4 shows an example of shape representation using edge directions.

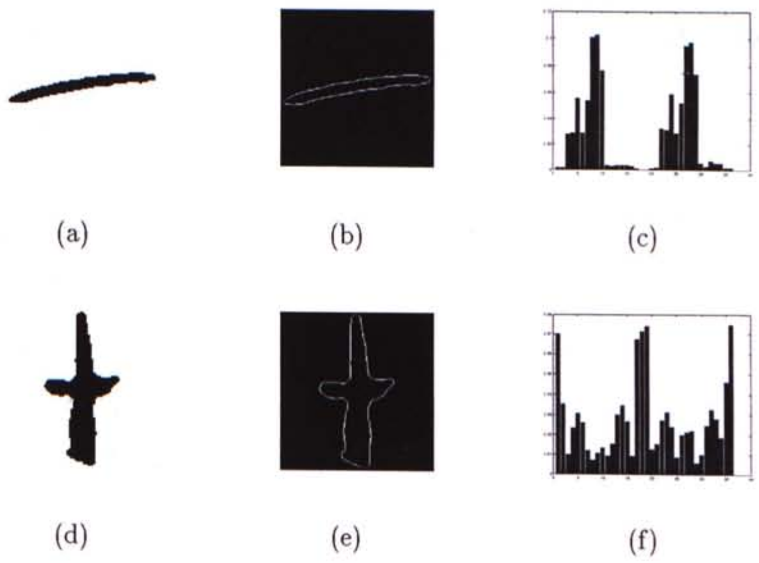


Figure 3.4: An example of shape representation using edge directions. (a) and (d) Original images. (b) and (e) Edge images. (c) and (f) Histograms of edge directions.

Discussion

Although the use of edge directions captures the general shape information and the histogram is invariant to translation of images, two totally different images may yield similar edge histograms. Scale invariance can be achieved by normalizing the histograms with respect to the number of edge points in the image. Moreover, the histogram is also not rotation invariant. A rotation of the image shifts each of the edge directions by the amount of rotation and due to the quantization effects in binning the edge directions, it may also affect the bin membership. We will discuss more on the problems encountered in Chapter 5.

The storage requirement of this method is $O(b)$, where b is the number of bins in the histogram and the computational complexity is $O(WH)$ where W is the width of a frame buffer and H is the height of the frame buffer. This representation method is indexable.

3.7 Curvature Scale Space Representation

Idea

A Curvature Scale Space (CSS) representation is a multi-scale organization of the curvature zero-crossing points of a planar curve, which is proposed by Mokhtarian and Mackworth [47, 46, 44, 42, 45]. The CSS representation of a planar curve T represents the curve uniquely modulo scaling and a rigid motion. To compute it, T is first parameterized by the arc length parameter u:

$$T(u) = (x(u), y(u)). \tag{3.24}$$

The polygon is sampled to obtain a new list of points such that all distances between points adjacent on the list are equal on the original polygon. An evolved version T_σ of T can then be computed. T_σ is defined by:

$$T_\sigma = (X(u,\sigma), Y(u,\sigma)) \tag{3.25}$$

where

$$X(u,\sigma) = x(u) * g(u,\sigma) \tag{3.26}$$

$$Y(u,\sigma) = y(u) * g(u,\sigma) \tag{3.27}$$

where $*$ is the convolution operator and $g(u, \sigma)$ denotes a 1-D Gaussian kernel of width σ. As σ increases, the shape of T_σ changes. This process of generating ordered sequences of curves is referred to as the evolution of T. Figure 3.5 shows an example of curve evolution for increasing values of σ.

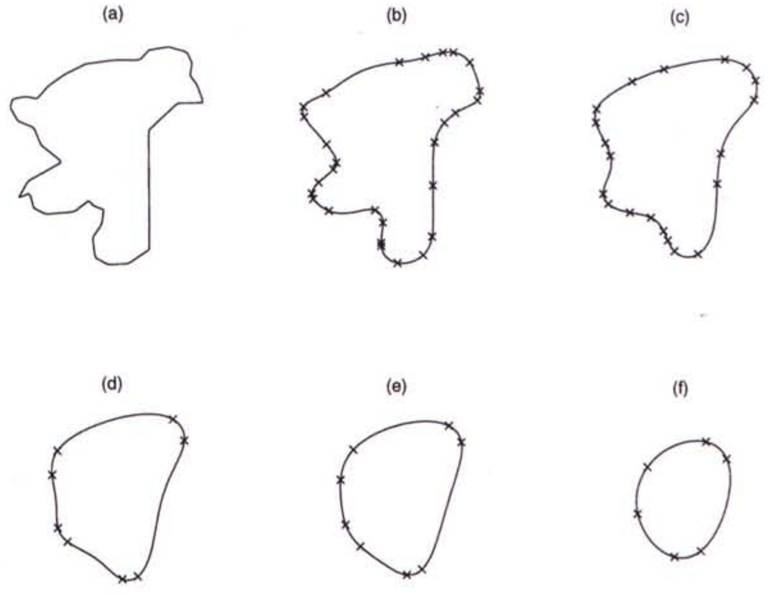


Figure 3.5: Curve evolution with zero-crossing points: (a) original curve; (b) σ=10; (c) σ=20; (d) σ=30; (e) σ=40; (f) σ=70.

When σ becomes sufficiently high, T_σ will be a convex curve with no curvature zero crossings. The process can then be terminated and the resulting points can be mapped to the (u, σ) plane. The result is a binary image called a CSS image as shown in Figure 3.6. The horizontal axis in this image represents the normalized arc length u, and the vertical axis represents σ, the width of the Gaussian kernel.

Discussion

The features of the CSS image used for matching are the maxima of the CSS contours. Since a change in the starting point of a CSS representation causes a circular shift in the CSS image, matching the highest maxima of the CSS image [45] is not a straightforward process. Figure 3.7 demonstrates the shifting property of the CSS image. The summation of the Euclidean distances between the matched

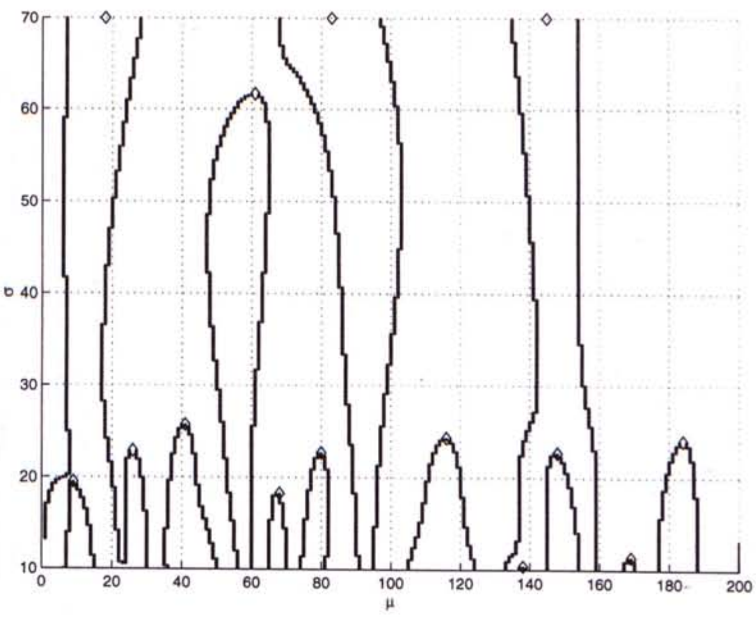


Figure 3.6: A curvature scale space (CSS) image and its maxima.

pairs of maxima is defined to be the dissimilarity value between two CSS images. The time complexity for matching is $O(m)$, where m is the number of maxima for representing a shape in a CSS image. A two-stage approach for CSS image matching has been proposed [6]. In the first stage, the σ vector is used for fast screening and a detailed matching only applies to the potential matches from the first stage.

The time complexity of calculating the CSS image is $O(lg\sigma_c)$, where l is the number of points in the normalized planar curve, g is the number of points in the Gaussian kernel representation, and σ_c is the maximum σ value in curve evolution. The storage complexity is $O(m)$ where m is the number of maxima for representing a shape in a CSS image, which is usually less than 20. This method is not indexable because of the circular shift property of the CSS representation.

3.8 Chapter Summary

Shape representation is one of the most important issues in designing a shape-based image retrieval system. One has to choose some good features to represent a

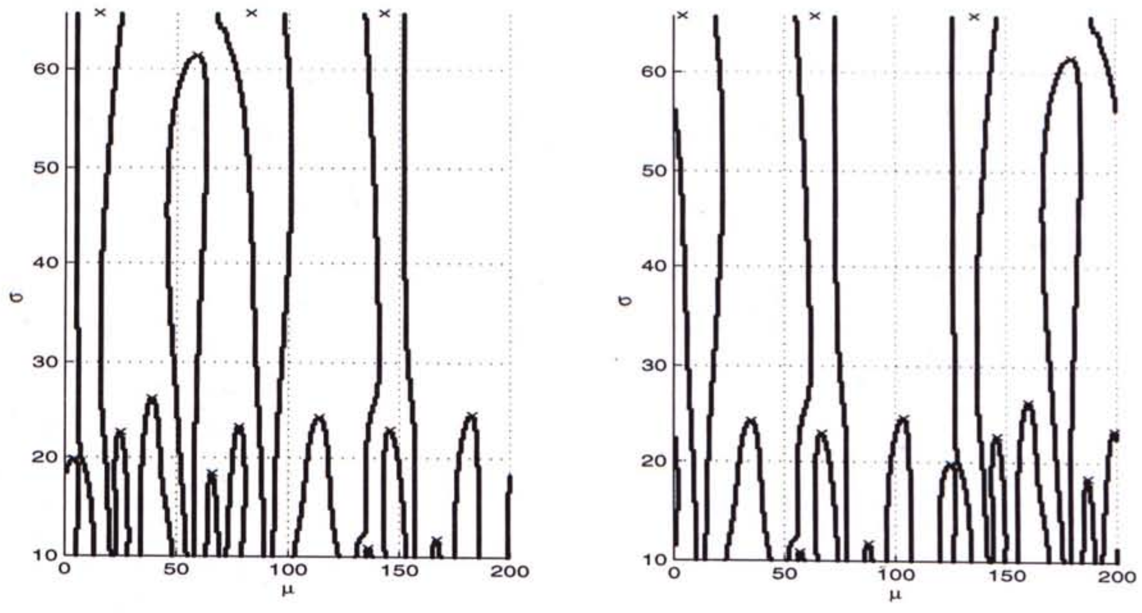


Figure 3.7: A change in the starting point of the CSS representation only causes a circular shift in the CSS image.

shape such that the system is efficient, accurate and robust. In Table 3.1, various properties of different representation methods are summarized.

Table 3.1: Summary of the characteristics of different representation methods.

Feature	**Translation Invariant**	**Scale Invariant**	**Orientation Invariant**	**Storage Requirement**	**Computational Complexity**	**Global (G) / Local (L)**	**Indexable**
Circularity	Yes	Yes	Yes	$O(1)$	$O(WH)$	G	Yes
Rectangularity	Yes	Yes	Yes	$O(1)$	$O(WH)$	G	Yes
Hole Area Ratio (HAR)	Yes	Yes	Yes	$O(1)$	$O(WH)$	G	Yes
Horizontal Gap Ratio (HGR)	Yes	Yes	No	$O(1)$	$O(WH)$	G	Yes
Vertical Gap Ratio (VGR)	Yes	Yes	No	$O(1)$	$O(WH)$	G	Yes
Central Moments	Yes	No†	No	$O(1)^{\S}$	$O(WH)$	G	Yes
Major Axis Orientation	Yes	Yes	N/A	$O(1)$	$O(WH)$	G	Yes
Eccentricity	Yes	Yes	Yes	$O(1)$	$O(WH)$	G	Yes
Fourier Descriptors	Yes	Yes	Yes	$O(l)$	$O(l\log(l))$	L	Yes
Chain Codes	Yes	No†	No‡	$O(k)$	$O(k)$	L	Yes
Invariant Moments	Yes	Yes	Yes	$O(1)$	$O(WH)$	G	Yes
Zernike Moments	Yes	Yes	Yes	$O(1)^{\S}$	$O(WH)$	G	Yes
Edge Direction Histogram	Yes	Yes	No	$O(b)$	$O(WH)$	G	Yes
CSS Representation	Yes	Yes	Yes	$O(m)$	$O(lg\sigma_c)$	L	No

b is the number of bins in the edge direction histogram.
k is the number of boundary pixels of a digitized shape.
n is the number of vertices of polygon.
m is the number of maxima in a CSS image.
l is the number of points in the normalized planar curve.
g is the number of points in the Gaussian kernel representation.
W is the width of a frame buffer.
H is the height of a frame buffer.
σ_c is the maximum σ value in curve evolution.
† Scale invariance can be achieved by normalization of the original shapes.
‡ Yes, if the chain code is normalized.
§ for each order of moment.

Chapter 4

Genetic Algorithm for Weight Assignment

In this chapter, we introduce a method for weight assignment by integrating features for image retrieval [8]. Given a set of training examples, our method helps to determine a set of suitable weights in the dissimilarity function based on a supervised learning method using genetic algorithm. After determining the weights, they can be used in the retrieval system afterward.

4.1 Genetic Algorithm (GA)

Genetic algorithm (GA) [23, 19, 9] has been become an important tool in machine learning and function optimization. The metaphor underlying GAs is natural selection. To solve a learning task, a design task, or an optimization task, a GA maintains a population of "organisms" (bits strings) and probabilistically modifies the population, seeking a near-optimal solution to the given task. Beneficial changes to parents are combined in their offspring, and a GA is a control structure that adapts to the problem being solved through syntactic operations on bit strings.

4.1.1 Basic Idea

GAs are algorithms that used to deal with optimization problems. Given a function F on a given search space S, their goal is to find optimum. In an initialization step, a set of points of the search space S (called a population of individuals), is drawn at random. An abstract view of the algorithm is

Algorithm 4.1 Genetic Algorithm

```
1  $t = 0$
2  generate initial population $P(t)$
3  evaluate $P(t)$
4  repeat
5      $t = t + 1$
6      select $P(t)$ from $P(t-1)$
7      crossover $P(t)$                ▷ with replacement
8      mutate $P(t)$                   ▷ with replacement
9      evaluate $P(t)$
10 until termination criterion is achieved
```

The GA iterates over the following 4 steps until a satisfactory optimum is reached.

1. **Evaluation** The function F is computed for each individual in the population, ordering the population from the worst to the best.

2. **Selection** Depending on the values of the individuals, select subsets called *reproducing sets* from the population. Best individuals have more chance to be selected than poor ones.

3. **Reproduction** from each reproducing set, generate some number of new individuals using various genetic operators like crossover and mutation.

4. **Replacement** A new population is generated by replacing some of the individuals of the old population by the new ones.

4.1.2 Genetic Operators

Genetic operators provide the basic searching mechanism of the GA. The operators are used to create new solutions based on existing solutions in the population.

Simple Crossover

Let x and y be two n-dimensional vectors denoting chromosomes (parents) from a population P. Simple crossover generates a random number r from a uniform distribution from 1 to m and creates two new individuals, x' and y', according to Eq. 4.1 and Eq. 4.2.

$$x'_i = \begin{cases} x_i, & \text{if } i < r \\ y_i, & \text{otherwise} \end{cases} \tag{4.1}$$

$$y'_i = \begin{cases} y_i, & \text{if } i < r \\ x_i, & \text{otherwise} \end{cases} \tag{4.2}$$

Figure 4.1 illustrates the crossover operator.

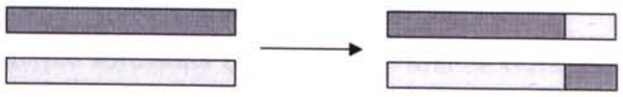


Figure 4.1: Crossover operator.

Uniform Mutation

Let a_i and b_i be the lower and upper bound, for all variables i. Uniform mutation randomly selects one variable j, and sets it equal to a uniform random number $U(a_i, b_i)$:

$$x'_i = \begin{cases} U(a_i, b_i), & \text{if } i = j \\ x_i, & \text{otherwise} \end{cases} \tag{4.3}$$

Figure 4.2 shows an example of mutation.

Figure 4.2: Mutation operator.

4.2 Why GA?

Genetic algorithm has many applications and exhibit impressive optimization capabilities when compared to other optimization techniques especially when the search space is large ($\approx 2^{300}$) and the function F is quite irregular [17].

Moreover, their scientific interest as a model of biological evolution, GA has three main technological interests:

1. They are very robust techniques able to deal with a very large class of optimization problems.

2. They are easy to program in parallel and the acceleration obtained by doing so is considerable [62].

3. They operate on several solutions simultaneously, gathering information from current search points to direct subsequent search. Their ability to maintain multiple solutions concurrently makes GAs less susceptible to the problems of local maxima and noise.

Due to the fact that GA is a powerful tool for solving optimization problems, we use it to solve the problem of weight assignment in dissimilarity function. In fact, the other more traditional methods, for example, neural networks, multiple classifiers, and etc. may also be used to solve the weight assignment problem, but the formalization of the solution may be complicated. Hence, we propose to use GA to solve the problem.

4.3 Weight Assignment Problem

In order to build a robust and accurate system, we use multiple shape features. Retrievals based on these features are integrated [8, 7] for better accuracy and higher system recall rate.

4.3.1 Integration of Image Attributes

Use of a single image attribute for retrieval may lack sufficient discriminatory information and might not be able to support large variations in image orientation and scale. For example, two different objects may have the same area; two different objects may have the same circularity. However, the chance of two different objects having the same area and the same circularity is small. In order to increase the accuracy of retrievals, integration of the results obtained from the query based on individual shape features is proposed.

The integrated dissimilarity is defined as follows:

Definition 4.1 (Integrated Dissimilarity) *Let x_f^I be a feature vector of an image I on the basis of a feature extraction function f. Then, the integrated dissimilarity function, D_t, between two images I_1 and I_2 is defined as*

$$D_t(I_1, I_2) = \sum_{i=1}^{n} D_{f_i} \frac{w_i}{\sum_{j=1}^{n} w_j} \, , \tag{4.4}$$

where f_i is a feature extraction function, D_{f_i} is the Euclidean distance between the feature vector $x_{f_i}^{I_1}$ and feature vector $x_{f_i}^{I_2}$, w_i is the weight assigned to feature vector set i and n is the number of the feature vector sets.

The problem is that it is difficult to determine the weights. Some researchers proposed to use equal weights [29, 30]. They also proposed to determine the weights on the basis of the accuracy of the individual feature-based queries. The QBIC system uses Eq. 2.1 to determine the weights. However, none of the proposed methods are determined based on human perception.

4.4 Proposed Solution

The problem of finding weights for the dissimilarity function for image retrieval and the proposed solution are discussed in this section.

4.4.1 Formalization

Definition 4.2 (Image Database) *An image database DB is defined as*

$$DB = \{I_i\}_{i=1}^{n} , \tag{4.5}$$

where I_i is an image in the database.

Definition 4.3 (Training Pair) *A training pair TP is defined as*

$$TP = (I_Q, I_M) , \tag{4.6}$$

where $I_Q \in DB$ is a query image and $I_M \in DB$ is the user defined best-matched image.

Definition 4.4 (Ranking Score Function) *Given a training pair $TP = (I_Q, I_M)$, r is the ranking of I_M for the query I_Q using D_t. The ranking score function s is defined as*

$$s : r \to \mathcal{R} . \tag{4.7}$$

The higher the ranking, the higher will be the score.

Definition 4.5 (Total Count) *Given an integrated dissimilarity function D_t with a set of weights $\boldsymbol{w}$, n training pairs, the total count $TC(\boldsymbol{w})$ is defined as*

$$TC(\boldsymbol{w}) = \sum_{i=1}^{n} s(r_n) , \tag{4.8}$$

where s is a ranking score function, n is the number of training pairs and r_n is the ranking for the query n using D_t.

Definition 4.6 (Optimal Weights) *The set of weights* $\boldsymbol{w}$ *in a dissimilarity function* D_t *is defined as* $\boldsymbol{w} = \{w_i\}_{i=1}^{n}$, *where* w_i *is the weight assigned to feature vector set* i.

The optimal weights are $\boldsymbol{w}$ *such that* $TC(\boldsymbol{w})$ *is maximized, i.e.,*

$$\arg\max_{\boldsymbol{w}} TC(\boldsymbol{w}) \ . \tag{4.9}$$

4.4.2 Proposed Genetic Algorithm

Chromosome Representation

A chromosome representation is used to describe an individual in the population of interest. A chromosome in our GA is defined as

$$c = (w_1, w_2, \ldots, w_i, \ldots, w_n), \tag{4.10}$$

where w_i is the weight assigned to feature vector set i and n is the number of feature vector sets which is the same as the number of genes in the chromosome. Note that a population P is defined as

$$P = \{c_1, c_2, \ldots, c_i, \ldots, c_{PopSize}\} \ , \tag{4.11}$$

where $PopSize$ is the number of individuals in the population and c_i is a chromosome.

Selection Function

A selection function plays a vital role in a GA because it selects individuals to reproduce successive generations. A probabilistic selection is performed based on the individual's fitness such that the better individuals have an increased chance of being selected.

The selection method, Roulette wheel, proposed by Holland [23] has been used in our implementation. The probability, P_i, for each individual is defined by

$$P[Individual\ i\ is\ chosen] = \frac{F_i}{\sum_{j=1}^{PopSize} F_j} \ , \tag{4.12}$$

where F_i is equal to the fitness of individual i. A series of N random numbers is generated and compared against the cumulative probability, $C_i = \sum_{j=1}^{i} P_j$, of the population. If $C_{i-1} < U(0,1) \leq C_i$, the individual is selected.

Genetic Operators

Simple crossover and uniform mutation described in Section 4.1.2 have been used in our implementation.

Evaluation Functions

The evaluation function is the total count $TC(\boldsymbol{w})$ defined in Definition 4.5.

Initialization and Termination

The initial population is randomly generated. The stopping criterion is a predefined maximum number of iterations, for example, 1000 iterations. Alternatively, a GA may be terminated when there is no improvement for a predefined number of iterations. For example, if the fitness of the best individual in the population does not improve for 500 iterations, the GA stops.

4.5 Chapter Summary

In this chapter, we propose a solution to the weight assignment problem for image retrieval using GA. Given a set of training examples, the GA can find a set of optimal weights for the dissimilarity function. We have used the proposed GA to determine the weights for trademark retrieval and Chinese cursive script character image retrieval. The details will be discussed in the following chapters.

Chapter 5

Shape-based Trademark Image Retrieval System

In this chapter, our shape-based trademark image retrieval system [8] is discussed. We state some problems on the existing solutions and suggest a solution to tackle the shortcomings. We propose to integrate several shape features including invariant moments, eccentricity, Fourier descriptors of approximated boundary, and circularity of approximated boundary for trademark retrieval. The weights in the dissimilarity function can be found by the supervised learning method discussed in Chapter 4. The proposed solution was tested on a database of 1400 monochromatic trademark images. The retrieval speed was fast and the similar-shaped trademark retrieval results were promising. Several experiments were conducted and the results are presented at the end of this chapter.

5.1 Problems on Existing Methods

There are several problems on the existing methods for trademark retrieval. A literature review of the existing systems can be found in Chapter 2.

5.1.1 Edge Direction Histogram

A histogram of the edge directions [29, 30] has been used in a trademark retrieval system. There are two disadvantages for this technique. First, this technique lies in the lack of discernment, because the histogram alone does not contain any information of edge location. For example, although the shapes of the images shown in Figure 5.1(a)-(c) are quite different, they have similar edge direction histograms of the edge directions as shown in Figure 5.1(d). Second, this representation method is not rotational invariant. Figure 5.2 illustrates the problem. Therefore, an edge direction histogram may not be a good choice for representing a trademark because the problem requires a set of rotational invariant features.

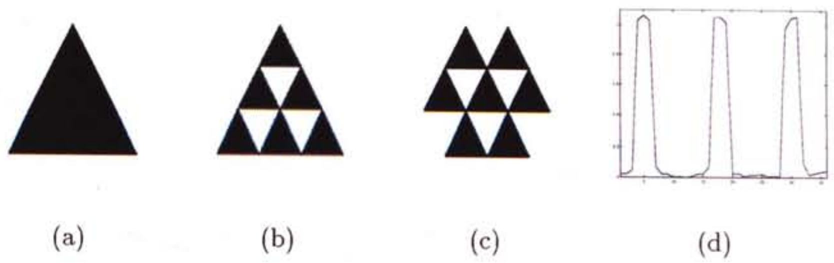


Figure 5.1: (a), (b), and (c) are examples of different shapes whose edge direction histograms are very similar as shown in (d).

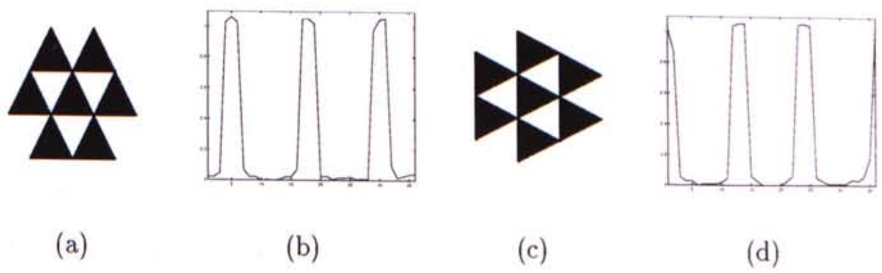


Figure 5.2: Edge direction histogram is not rotational invariant. (a) and (c) are 2 original images. (b) and (d) are their corresponding edge direction histograms.

5.1.2 Boundary Based Techniques

Boundary based techniques such as boundary matching [11, 12], chain-code [53], and Fourier descriptors [65, 38, 60] of *localized components* may not be suitable for similar-shaped trademark retrieval because the boundary shape can be totally changed when there is a small crack like an opening or an object touching its neighboring objects. For example, the shapes shown in Figure 5.3(a), (b), and (c) are very similar in human perception. However, the boundaries of these shapes, as shown in Figure 5.3(d), (e), and (f) are very different. Therefore, it is hard for a system to extract the shape features robustly and automatically. One may think that the problem can be solved by selecting the boundary manually by the operators, however, different people may also use different methods to segment an image, especially for the image consists of complex patterns.

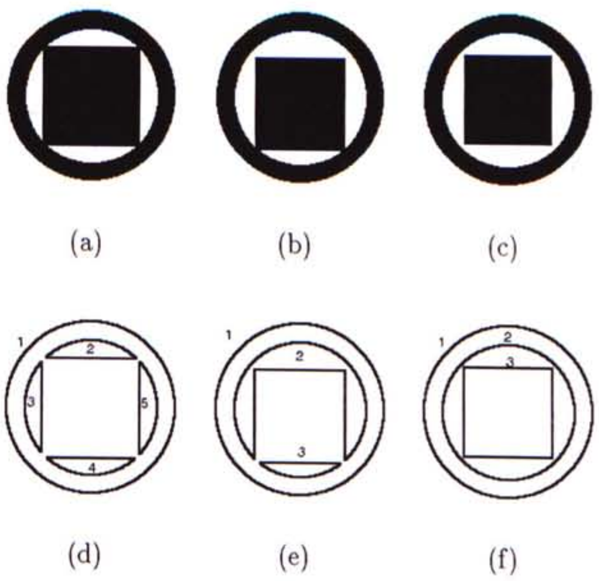


Figure 5.3: The boundaries found can be very different although they are very similar in human perception.

Furthermore, while most Fourier descriptors or curvature-based methods are based on a single boundary, a trademark consists of a complex pattern that has more than one boundary. As a result, it is difficult to define the distance between two sets of boundaries. Even if it is defined, the spatial relationship of various components will be lost.

5.2 Proposed Solution

The selection of a set of effective features plays an important role in a content-based image retrieval system. In our system, we use several translation, rotation, and scale invariant features to represent a trademark. The features can be extracted efficiently and automatically without user intervention. The features are integrated using a sum of weighted Euclidean distances. The weights in the distance function were found by the GA suggested in Chapter 4. Figure 5.4 shows a model of the proposed system.

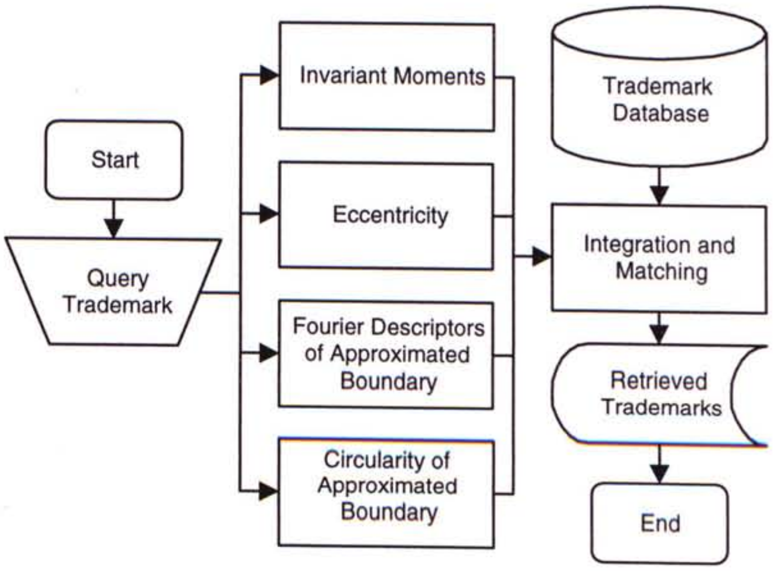


Figure 5.4: The proposed shape-based trademark retrieval system.

5.2.1 Image Preprocessing

Image preprocessing is essential before extracting the shape features. The images preprocessed by binary thresholding are separated into the background and the object. Pixels with gray levels larger than the predefined threshold are assigned gray level 1 (i.e. white), while the others are assigned gray level 0 (i.e. black). Noise in the binary images is usually in the form of speckles. The noise of the image is removed by convolving with a median filter with the size of 3×3. After that, the resolution of the image are normalized to 111×111 pixels.

5.2.2 Automatic Feature Extraction

Most of the proposed methods in solving the trademark retrieval problem need human to segment the shapes from each trademark. The method is good at search for a component in other images. However, the components are usually too local and cannot represent the whole trademark as an example shown in Figure 5.5. Therefore, instead of using shape features of components to represent a trademark, we propose to use some global features that can be extracted automatically by the system.

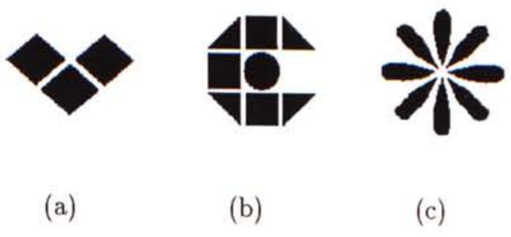


Figure 5.5: It may be too local to represent a trademark with its components like the circles, triangles, squares, and etc.

We use the following features to represent the shapes of an image.

- **Invariant Moments:** The global shape information is represented in terms of seven invariant moments [24] that are invariant under rotation, translation, and scaling.

- **Eccentricity:** Eccentricity [28] is invariant under rotation, translation, scaling, and reflection.
- **Circularity of Approximated Boundary:** Circularity [28] of approximated boundary is also invariant under rotation, translation, scaling, and reflection.
- **Fourier Descriptors of Approximated Boundary:** The global image shape is captured by an approximated boundary and described using Fourier descriptors [68, 18, 55]. The feature is invariant under rotation, translation, scaling, and reflection.

5.2.3 Approximated Boundary

We propose to use an approximated boundary to capture the whole image shape. An approximated boundary is used to capture the global shape of a trademark neglecting its details. Figure 5.6 shows an example of approximated boundary extraction of a trademark.

Given a normalized trademark $f(x, y)$, it is closed by a closing operator with an increasing size of circular structuring element e until a connected component $f'(x, y)$ is found. The edges of $f'(x, y)$ are then extracted and the boundary points can be traced by using Algorithm 5.1. Dilation, erosion, and closing operators [43, 21] are defined as follows:

Definition 5.1 (Dilation) *Dilation of an object increases its geometrical area by setting the background pixels adjacent to an object's contour to the object's gray level value. Dilation is defined as the union of all vector additions of all pixels a in object A with all pixels b in the structuring element B.*

$$A \oplus B = \{t \in Z^2 : t = a + b, a \in A, b \in B\} , \tag{5.1}$$

where vector t is an element of the image space Z^2.

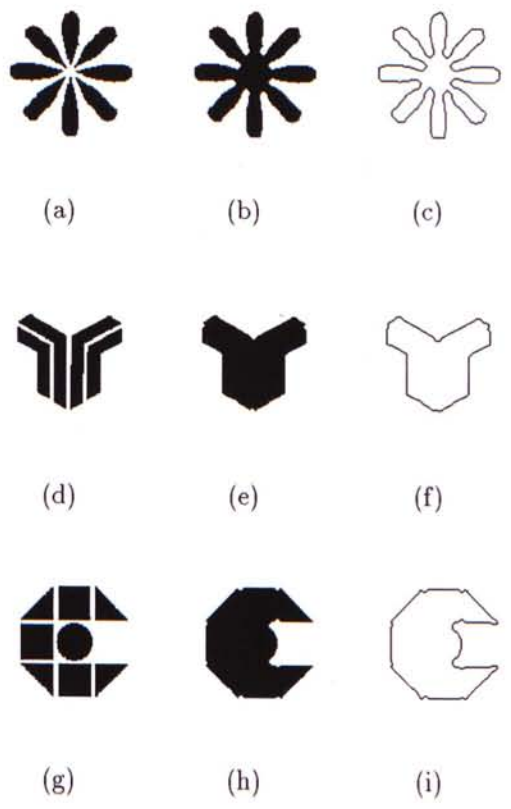


Figure 5.6: We propose to use an approximated boundary to capture the whole image shape. (a), (d), and (g) are the original images; (b), (e), and (h) are the closed image; (c), (f), and (i) are the approximated boundary extracted.

Definition 5.2 (Erosion) *Erosion of an object reduces its geometrical area by setting the contour pixels of an object to the background value. Erosion is defined as the complement of the resulting dilation of the compliment of object A with structuring element B.*

$$A \ominus B = (A^c \oplus B)^c \ . \tag{5.2}$$

Definition 5.3 (Closing) *Closing of an image is defined as the dilation of the image followed by the erosion of the dilated image.*

$$A \bullet B = (A \oplus B) \ominus B \ , \tag{5.3}$$

where A is the original image and B is structuring element.

Algorithm 5.1 $[x\ y] = tracing\_pts(is\_start, starting\_point, current\_point, image)$

```
▷ Given a binary image of edges and a starting point, the algorithm traces the boundary
  pixels and returns a list of boundary points.
$sx = current\_point.x$
$sy = current\_point.y$
$tmpx = []$
$tmpy = []$
if ($starting\_point == current\_point$) && (not $is\_start$)
    $is\_stop = 1$                ▷ A closed boundary is found. is_stop is a static variable.
else if ($image[sx, sy]\ != \ 1$)
    $x = []$                      ▷ The pixel is not black in color.
    $y = []$
else
    $image[sx, sy] = 0$
    for $i = 1$ to 8              ▷ Eight directions.
        switch $i$
            case 1 $next.x = sx + 1,\ \ next.y = sy$
            case 2 $next.x = sx,\ \ next.y = sy - 1$
            case 3 $next.x = sx - 1,\ \ next.y = sy$
            case 4 $next.x = sx,\ \ next.y = sy + 1$
            case 5 $next.x = sx + 1,\ \ next.y = sy - 1$
            case 6 $next.x = sx - 1,\ \ next.y = sy - 1$
            case 7 $next.x = sx - 1,\ \ next.y = sy + 1$
            case 8 $next.x = sx + 1,\ \ next.y = sy + 1$
        end switch
        $[tmpx\ tmpy] = tracing\_pts(0, starting\_point, next, image)$    ▷ Call itself
        if ($is\_stop == 1$)
            $x = [sx\ tmpx]$      ▷ A closed boundary is found.
            $y = [sy\ tmpy]$
            break
        end if
    end for
end
return $[x\ y]$
```

5.2.4 Integration of Shape Features and Query Processing

Integration of Shape Features

The shape features are integrated using the dissimilarity function defined in Definition 4.1 with the weights learned by the GA. The integrated dissimilarity function can be rewritten as

$$\begin{aligned} D_t = {} & d_{m_1} w_{m_1} + d_{m_2} w_{m_2} + d_{m_3} w_{m_3} + d_{m_4} w_{m_4} + d_{m_5} w_{m_5} + \\ & d_{m_6} w_{m_6} + d_{m_7} w_{m_7} + d_{circularity} w_{circularity} + \\ & d_{eccentricity} w_{eccentricity} + d_{Fourier\ descriptors} w_{Fourier\ descriptors} \end{aligned} \tag{5.4}$$

where d_i is the Euclidean distance of query using feature i and w_i is the weight assigned to feature set i.

Query Processing

The extracted feature vectors of the images are stored in the database. The values of each feature are normalized by the maximum value of that particular feature, so that the values are ranged between 0 and 1. When a user raises a query, the features of the query trademark are first extracted and then normalized. The normalized features are matched linearly with the features in the database and integrated by the dissimilarity function. After that, the trademarks are displayed in the order of similarity.

5.3 Experimental Results

The evaluation of the performance of a shape-based image retrieval system is a difficult task, because shape similarity is a purely subjective matter. In order to estimate the performance of the system, several experiments were carried out to estimate both the subjective and objective criteria:

1. **Subjective Criterion:** How well can similar-shaped trademarks be retrieved in accordance with the human perception.

2. **Objective Criterion:** How well can the original trademarks be retrieved in the presence of deformation, rotation, scaling and etc.

In Experiment 1, we found the weights in the dissimilarity function using the proposed GA. In Experiment 2, 3, 4, 5, and 6, we tested the speed, precision, and recall rate of the system. In Experiment 7, we compared different integration methods for trademark retrieval. All the experiments were conducted with a database of 1400 monochromatic trademark images on an Ultra 5 Machine running Matlab V5.3. Appendix A shows some representative images in our database.

5.3.1 Experiment 1: Weight Assignment using Genetic Algorithm

In this experiment, we tested our proposed GA and found the appropriate weighting factors for various shape features. The best weights found were used in the Experiment 2, 3, 4, 5, 6, and 7.

The proposed GA was tested with the following setup. The population size $PopSize$ was 30. The maximum number of iterations was set to 1000 because according to our testing, the proposed GA always converges within 1000 iterations. There were 50 training pairs TP. The size of the trademark database DB was 1400. The values of the weights (genes) were bounded by 0 and 1. The probability of application of crossover was 0.6 and the probability of application of mutation was 0.05. The experiments were repeated 10 times with different random seeds.

The evaluation function was the total count $TC(\boldsymbol{w})$. The ranking score function s was defined as

$$s(r) = \begin{cases} \frac{k+1-r}{k}, & \text{if } r \leq k \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

where r is the ranking of the target and k is the lowest ranking that has a score. In our implementation, k was 140. Therefore, if the target is ranked in the first position, $TC(\boldsymbol{w})$ is increased by 1. If the target is ranked in the second position, $TC(\boldsymbol{w})$ is increased by $\frac{139}{140}$ and so on. We set k to be 140 because it is 10% of the total number of images in the database.

Figure 5.7 shows the best result and the worst result obtained among the 10 trials. From the result, the GA improved the accuracy of the retrievals of the training examples by changing the weights in the dissimilarity function. For the best result, it converged in 800 iterations with a total count of 44.43, which means on average, the distance function ranks the target of a query in a training example to the sixteenth position.

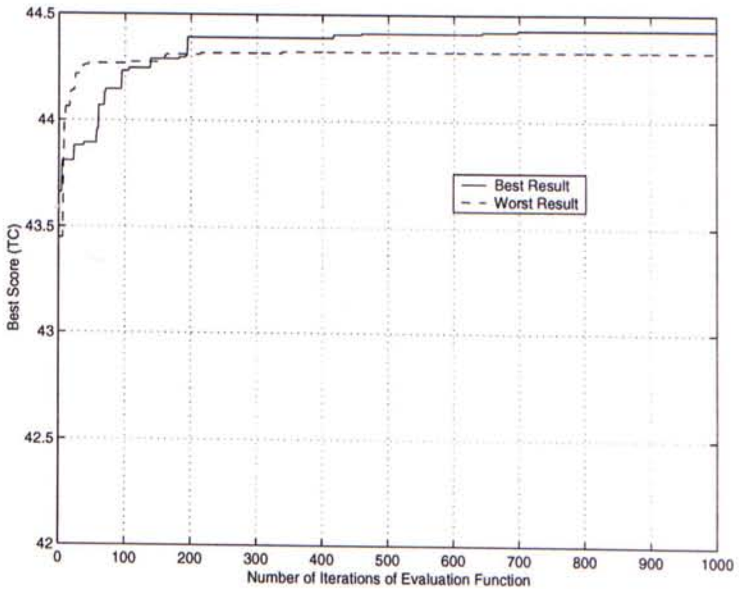
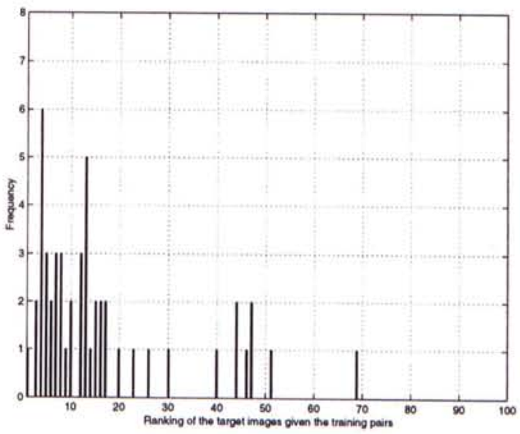


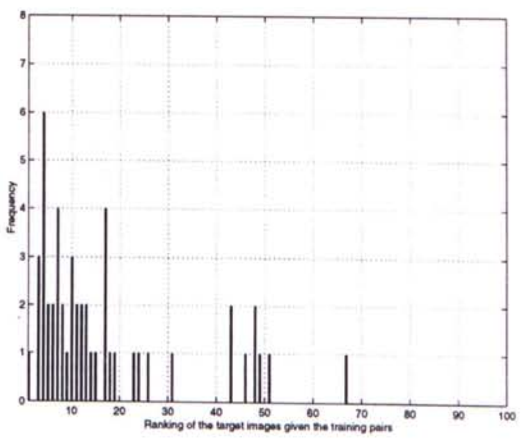
Figure 5.7: Results of Experiment 1. TC versus the number of iterations.

The performance of the weights of the best result and the worst result can be further analyzed by the ranking histograms in Figure 5.8. In fact, the two histograms look very similar. From the ranking histogram of the best result obtained among the 10 trials, 64% of the targets were ranked at the top 14 positions, 84% were ranked at the top 28 positions and 100% were ranked at the top 70 positions.

The best weights found were used as the final weights for trademark retrieval



(a)



(b)

Figure 5.8: Results of Experiment 1. (a) The ranking histogram of the best result among the 10 trials. (b) The ranking histogram of the worst result among the 10 trials.

and they are presented in Table 5.1. From the results, third invariant moment, fourth invariant moment, and fifth invariant moment have the highest weights. It means that according to the training examples, they are more important for the trademark image retrieval problem. On the other hand, circularity, eccentricity, and first invariant moment are not important, and hence, they may be discarded if someone wants to reduce the number of features for representing a trademark.

There are several parameters, like population size, crossover rate, and mutation rate that can affect the performance of the GA. The population size affects both the ultimate performance and the efficiency of the GAs. GAs generally do poorly with very small populations, because the population provides an insufficient sample size for most hyperplanes [56]. A large population is more likely to contain representatives from a large number of hyperplanes. Hence, the GAs can perform a more informed search. As a result, a large population discourages premature convergence to suboptimal solutions. On the other hand, a large population requires more evaluations per generation and hence, the rate of convergence is slow.

Table 5.1: Results of Experiment 1. Weights found for different shape features.

Feature	Weight	Percentage
Circularity	0.0210	0.5287
Eccentricity	0.0667	1.6806
First invariant moment	0.0421	1.0609
Second invariant moment	0.3965	9.9839
Third invariant moment	0.8239	20.7459
Fourth invariant moment	0.9700	24.4234
Fifth invariant moment	0.9492	23.8988
Sixth invariant moment	0.1426	3.5906
Seventh invariant moment	0.3454	8.6974
Fourier descriptors	0.2141	5.3897

The crossover rate controls the frequency with which the crossover operator is applied. In each new population, $m_c \times PopSize$ structures undergo crossover. The higher the crossover rate, the more quickly new structures are introduced into the population. If the crossover rate is too high, high-performance structures are discarded faster than selection can produce improvements. If the crossover rate is too low, the exploration rate of the search may be too slow. Mutation is a secondary search operator that increases the variability of the population. A low level of mutation serves to prevent any given bit position from remaining forever converged to a single value in the entire population. A high level of mutation yields an essentially random search.

5.3.2 Experiment 2: Speed on Feature Extraction and Retrieval

In this experiment, we tested the speed on feature extraction and image retrieval. When a query was submitted to the system, its features were computed first. Next, the similarity measure between the query and all trademarks in the database were computed. The average elapse time of 100 trials for feature extraction of the query image and database query are listed in Table 5.2.

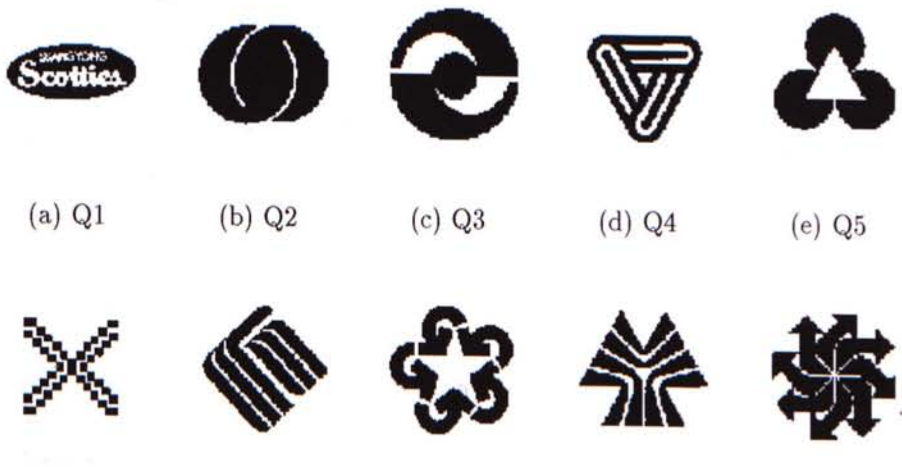
Table 5.2: Results of Experiment 2. Average elapse time for retrieving a trademark from a database of 1400 images on a Sun Ultra 5 Machine.

Feature Extraction	Database query	Total
2.24s	0.05s	2.29s

5.3.3 Experiment 3: Evaluation by Precision

In this experiment, we tested the retrieval precision of our proposed system. Several trademarks shown in Figure 5.9 were submitted as query images to the system. Retrieval precision is defined as

$$Precision = \frac{Number\ of\ relevant\ trademarks}{Number\ of\ retrieved\ trademarks} \times 100 \qquad (5.6)$$



(a) Q1 (b) Q2 (c) Q3 (d) Q4 (e) Q5

(f) Q6 (g) Q7 (h) Q8 (i) Q9 (j) Q10

Figure 5.9: Query trademarks.

This testing is rather subjective because the judgement of similarity (relevance) is a subjective issue. In order to evaluate the system, we asked a number of volunteers to determine the relevance. The precision of the queries are shown in Table 5.3, and some candidates for the relevant trademarks retrieved among the top 14 trademarks are shown in Figure 5.10. The average precision was about 64% among the top 14 candidates.

From the results obtained in Figure 5.10, similar trademarks were retrieved regardless of their orientation, size, and position because the features that we used have the rotation, scaling, and translation invariant characteristics. Besides, the trademarks retrieved agreed well with human perception.

Table 5.3: Results of Experiment 3. Precision on the basis of the dissimilarity function with the weights found by the GA. n refers to the number of retrieved image.

Query	$n = 5$ (%)	$n = 7$ (%)	$n = 14$ (%)
Q1	60%	57%	43%
Q2	80%	71%	79%
Q3	60%	57%	64%
Q4	80%	71%	64%
Q5	100%	86%	79%
Q6	100%	100%	64%
Q7	80%	71%	64%
Q8	100%	86%	64%
Q9	60%	57%	50%
Q10	100%	86%	71%
Avg.	82%	74%	64%

5.3.4 Experiment 4: Evaluation by Recall for Deformed Images

Our aim is to develop a trademark image retrieval system that is insensitive to variations on image deformation. In this experiment, we tested the behavior of our image retrieval system in the presence of the deformation transformation as shown in Figure 5.11. Similar objective evaluation criteria has also been used by Kim [34]. A set of 100 deformed trademark images as shown in Figure 5.12 was

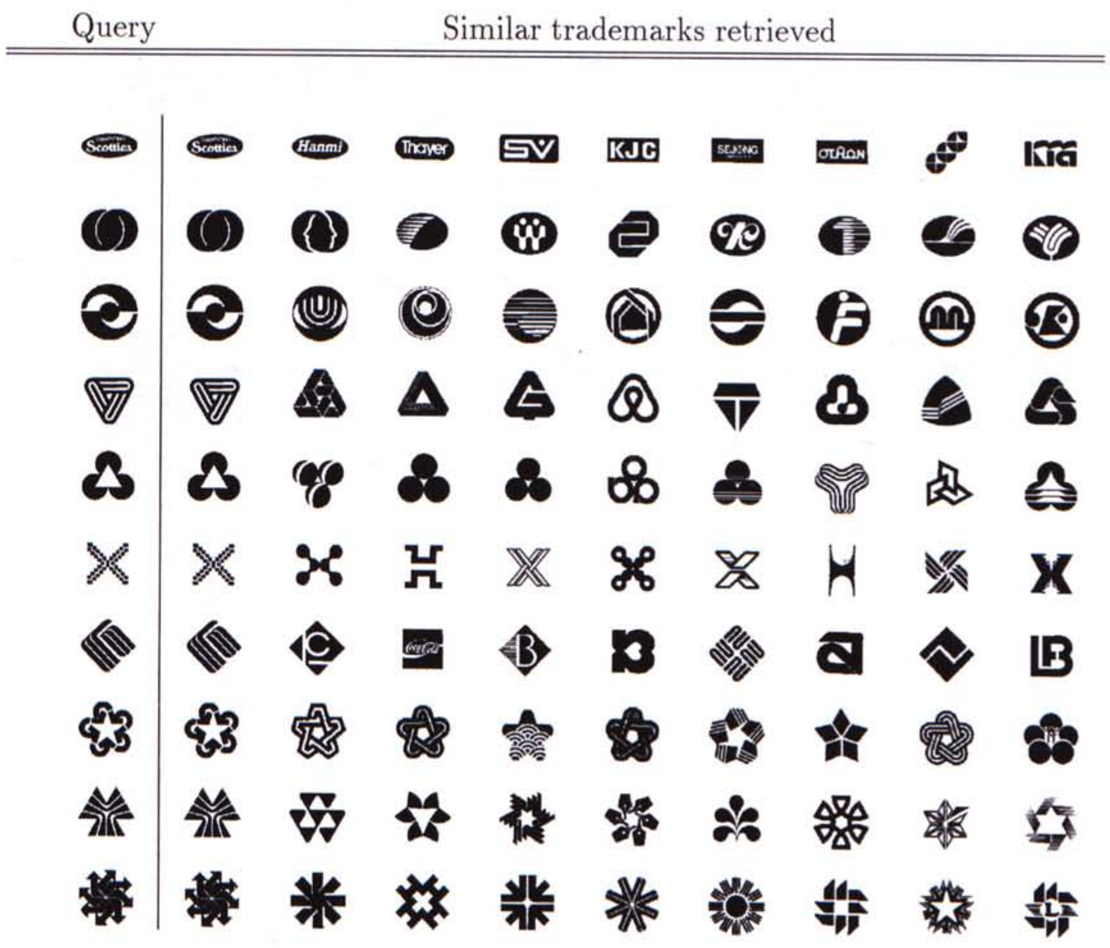


Figure 5.10: Results of similar trademarks retrieval. The first column are the query trademarks and the others are the retrieved similar trademarks.

used for performance evaluation. The deformed trademarks were submitted as query images to our trademark retrieval system to examine whether the deformed images can retrieve their original ones or not.

Table 5.4 presents the results of recall rate in the top 70 candidates queried by the deformed trademarks, where n corresponds to the position of the correct retrieval. From the results, 93% of the deformed images can recall their original ones in the top 28 candidates which is equal to 2% of the database size. The average recall rate of the original ones among the top 70 candidates queried by deformed images was 96%.

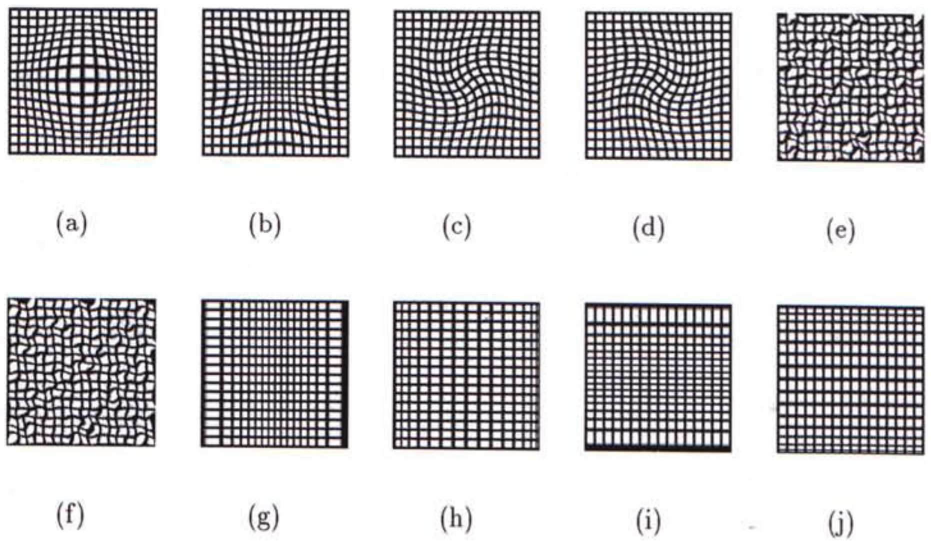


Figure 5.11: Distortions including Pinch: (a) and (b); Twirl: (c) and (d), Ripple: (e) and (f); Horizontal Extension (HE): (g) and (h); and Vertical Extension (VE): (i) and (j).

5.3.5 Experiment 5: Evaluation by Recall for Hand Drawn Query Trademarks

In this experiment, we tested the system with 100 hand drawn queries as shown in Figure 5.13. Every image was presented as the query image to examine whether the hand drawn query can retrieve their original ones or not.

From the results, we can see that the system is robust to retrieve hand drawn images. All the images can be recalled in the top 70 positions. 71% of their original images can be recalled in the top 28 positions.

5.3.6 Experiment 6: Evaluation by Recall for Rotated, Scaled and Mirrored Images

In order to test the recall rate for rotated, scaled, and mirrored images, we conducted the following experiment.

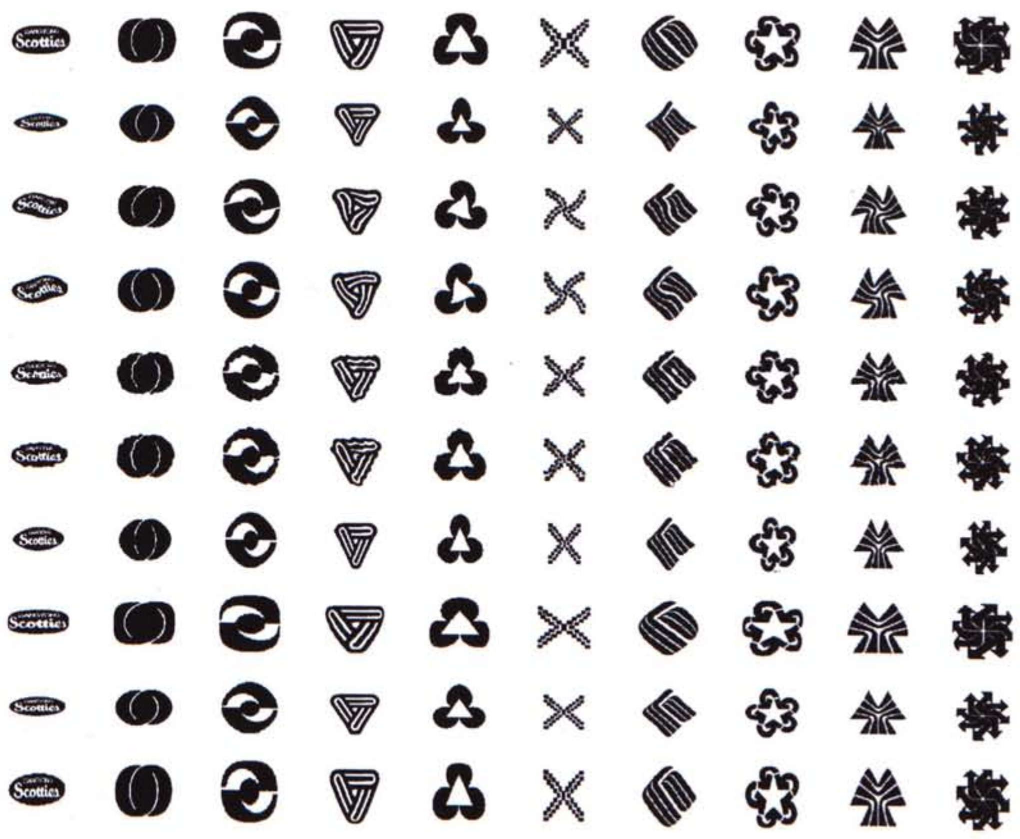


Figure 5.12: 100 queries deformed from 10 trademarks for performance evaluation.

1. **Rotated Query:** Every image in Figure 5.9 was rotated arbitrarily with the range of $\pm 180°$ and then presented as the query image. Figure 5.14 shows some examples of the query images.

2. **Scaled Query:** Every image in Figure 5.9 was scaled randomly and then presented as the query image. The scaling factors were bounded by 0.5 and 50.

3. **Mirrored Query:** Every image in Figure 5.9 was mirrored and then presented as the query image.

The experiments were repeated 100 times with rotated, scaled, and mirrored queries. Table 5.6 presents the results of the retrieval. From the results, we notice that the shape features that we used were very effective in retrieving rotated,

Table 5.4: Results of Experiment 4. Recall for deformed images. n refers to the position of the correct retrieval. For the query nature, please refer to Figure 5.11.

Query Nature	$n = 1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
(a) Pinch	20%	60%	70%	70%	80%	90%	10%
(b) Pinch	20%	60%	60%	90%	90%	100%	0%
(c) Twirl	100%	100%	100%	100%	100%	100%	0%
(d) Twirl	100%	100%	100%	100%	100%	100%	0%
(e) Ripple	60%	60%	80%	90%	100%	100%	0%
(f) Ripple	60%	70%	90%	90%	100%	100%	0%
(g) HE	40%	50%	70%	100%	100%	100%	0%
(h) HE	0%	10%	20%	70%	80%	90%	10%
(i) VE	40%	50%	70%	90%	100%	100%	0%
(j) VE	30%	60%	70%	80%	80%	80%	20%
Avg.	47%	62%	73%	88%	93%	96%	4%

Table 5.5: Results of Experiment 5. Recall for the hand drawn images. n refers to the position of the correct retrieval.

Query Nature	$n = 1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
Hand Drawn	16%	19%	33%	51%	71%	100%	0%

scaled, and mirrored images. This is expected because the features are rotation, scale, and reflection invariant.

5.3.7 Experiment 7: Comparison of Different Integration Methods

In this experiment, we compared the recall rate of different integration methods. The 100 deformed trademarks in Figure 5.12 were used as queries to recall their original ones. The integration methods include:

1. **Equal weights.** All the weights were set to 1 for all the features in the dissimilarity function. This method has been used in [30] for integrating two

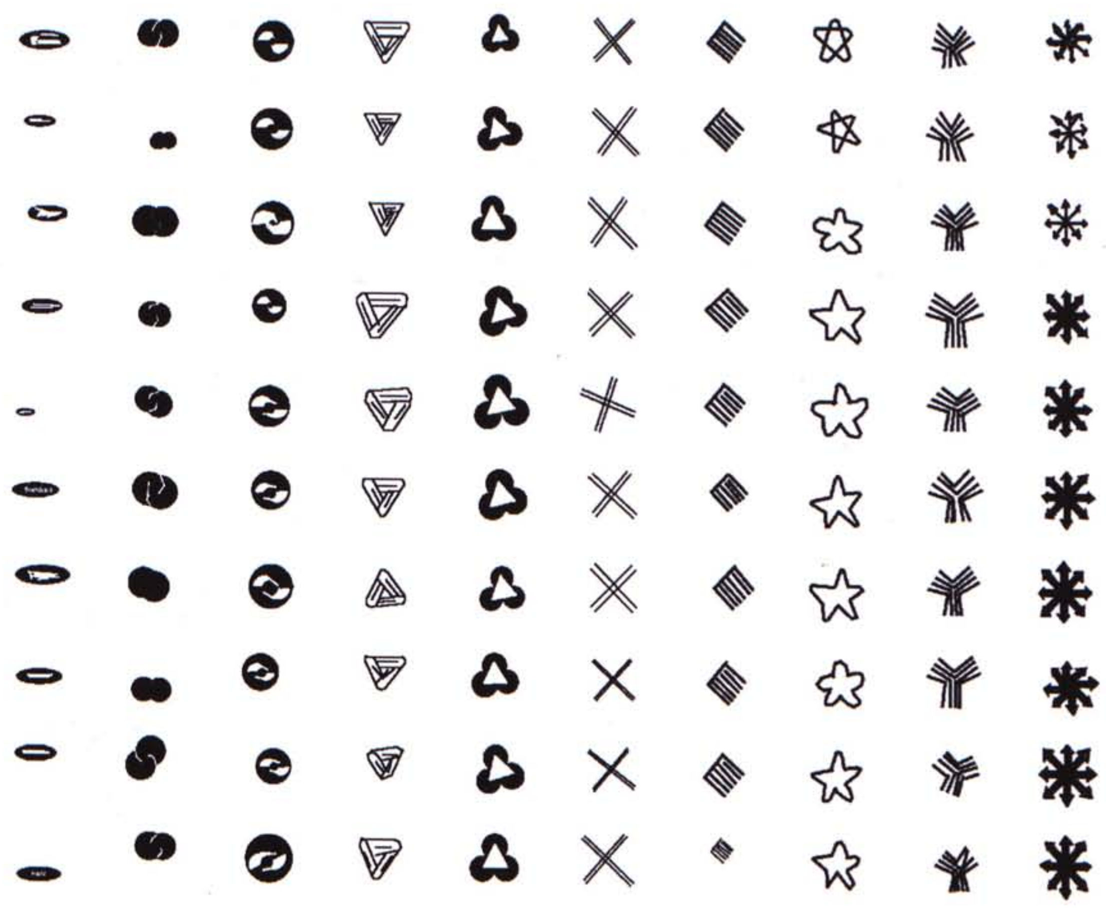


Figure 5.13: 100 hand drawn query trademarks for performance evaluation.

shape features.

2. **Weights determined by individual recall rate.** A weighted Euclidean distance were used as the distance function. The weights used are proportional to the accuracy of retrievals using individual features. The accuracy of a particular shape feature was determined by the same experimental setup as in Experiment 4. This method was proposed by [30] but they did not test its performance.

3. **Integration with the method in QBIC (See Eq. 2.1).** The distance between two images is computed using a weighted Euclidean distance with the inverse of feature variances used for normalization.

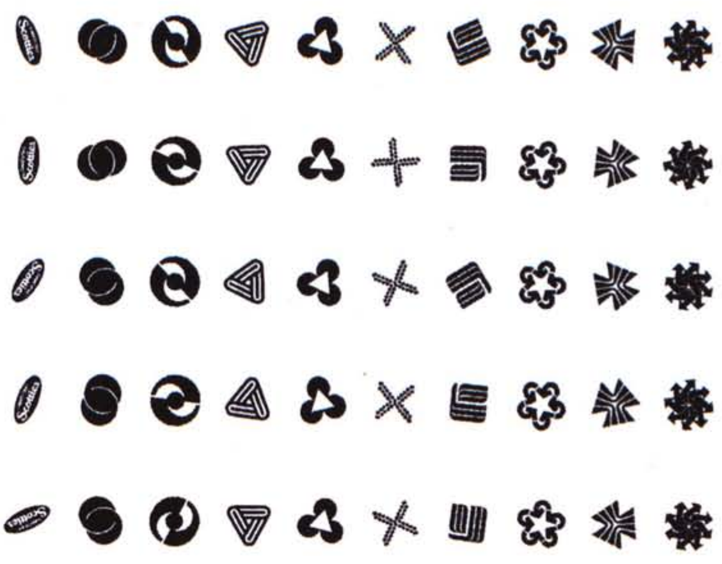


Figure 5.14: Some examples of rotated queries from 10 types of trademarks for performance evaluation.

Table 5.6: Results of Experiment 6. Recall for rotated, scaled, and mirrored images. n refers to the position of the correct retrieval.

Query Nature	$n = 1$ (%)	$n \leq 2$ (%)	Not Retrieved(%)
(a) Rotated	95%	100%	0%
(b) Scaled	100%	100%	0%
(c) Mirrored	100%	100%	0%

4. **Weights found by GA.** The distance between two images is computed using a weighted Euclidean distance with the weights found by our proposed genetic algorithm.

From Table 5.7, the dissimilarity function found by our GA has a better recall rate than that of the other methods. This is expected because the weights were optimized by the GA based on the training examples given by the users. Hence, its recall rate should be better than that of the other integration methods.

Table 5.7: Results of Experiment 7. Retrieval results on the basis of different integration methods. n refers to the position of the correct retrieval. (a) Equal weights. (b) Weights determined by individual recall rate. (c) QBIC. (d) Our system.

Query Nature	$n = 1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
(a)	43%	58%	66%	80%	82%	92%	8%
(b)	40%	55%	64%	79%	85%	91%	9%
(c)	33%	49%	62%	74%	82%	93%	7%
(d)	47%	62%	73%	88%	93%	96%	4%

5.4 Chapter Summary

In this chapter, we have discussed some problems of the existing solutions on trademark retrieval and proposed a solution by integrating several effective shape features using a sum of weighted Euclidean distances. The features that we used include invariant moments, eccentricity, Fourier descriptors of approximated boundary, and circularity of approximated boundary. The weights in the dissimilarity function were found by using the method proposed in Chapter 4. The results show that the weights found by our GA improved the recall rate. Our proposed system was robust under deformed, rotated, scaled, mirrored, and hand drawn queries. In addition, the trademarks retrieved by our system agreed well with human perception.

Chapter 6

Shape-based Chinese Cursive Script Character Image Retrieval System

In this chapter, our shape-based Chinese cursive script character image retrieval system [7] is discussed. First, we compare this problem with the trademark retrieval problem. Then, the target of this research is explained. After that, a system for cursive script character image retrieval is proposed. Experimental results are given and discussed at the end of this chapter.

6.1 Comparison to Trademark Retrieval Problem

There are several differences between the trademark retrieval problem and the cursive script character image retrieval problem based on shape.

6.1.1 Feature Selection

The goals of the two systems are different; therefore, the shape features to be used may be different. The goal of the trademark retrieval problem is to detect the infringement cases based on visual similarity. The shape features to be used in such a system should be insensitive to translation, rotation, and size variations. Moreover, mirror resemblances existing in trademarks must be considered. On the other hand, for a Chinese cursive script character retrieval system, the similarity measure must be insensitive to translation and size variations. Rotational invariance is not important because the character is usually directed upward.

6.1.2 Speed of System

For a trademark retrieval system, the speed can be slow because the tasks can be done offline. However, for a cursive script character image retrieval system, speed is an important factor.

6.1.3 Variation of Style

A trademark usually has less noise and styles. However, in handwritten characters, there exist many types of variations, such as position, size, and shape distortions due to various writing styles of human beings. For cursive style, the situation is much worse because there are many variations.

1. **Simplified Structures:** The characters often have simplified structures. Some examples are given in Figure 6.1.

2. **Jammed Strokes:** Some of the strokes in a character are usually jammed together as shown in Figure 6.2.

3. **Variation of Aspect Ratio:** The aspect ratio and the width of strokes of the same character may have a large variation. Several examples are shown

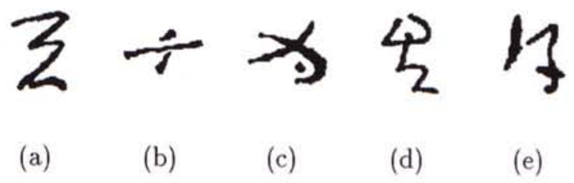


Figure 6.1: Characters with simplified structures.

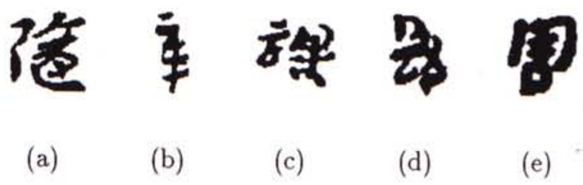


Figure 6.2: Characters with strokes jammed together.

in Figure 6.3.

4. **Many Styles:** There can be a large variety of different styles for the same character. Figure 6.4 shows 27 different cursive styles of the same character. Some of them cannot be recognized by human beings easily.

6.2 Target of the Research

Due to the reasons stated above, it is challenging to develop a Chinese cursive style calligraphy character image retrieval system based on shape. It is not our target to retrieve all the other styles of the character given a particular style of a character because it is quite impossible to achieve; on the other hand, it is our aim to find the similar images in the database given a query image.

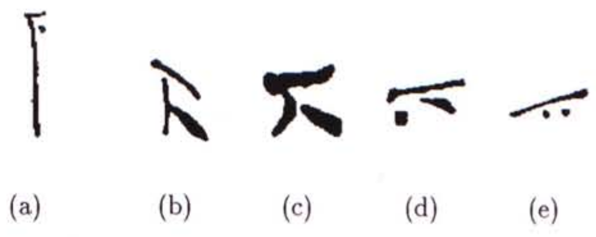


Figure 6.3: The aspect ratio and the width of strokes for the same character vary.

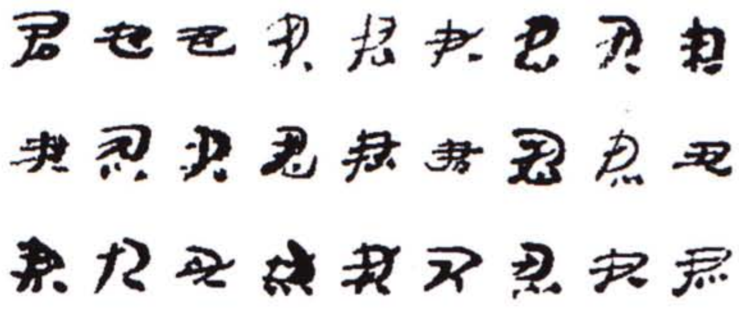


Figure 6.4: There can be a large variety of styles for the same character.

6.3 Proposed Solution

6.3.1 Image Preprocessing

The images, captured by a scanner and processed by binary thresholding, are separated into background and objects. Pixels with gray levels larger than the predefined threshold are assigned gray level 1 (i.e. white), while the others are assigned gray level 0 (i.e. black). Noise in the binary images is usually in the form of speckles. The noises of the images are removed by convolving with a median filter with the size of 3×3. After that, the images are resized to 200×200 pixels.

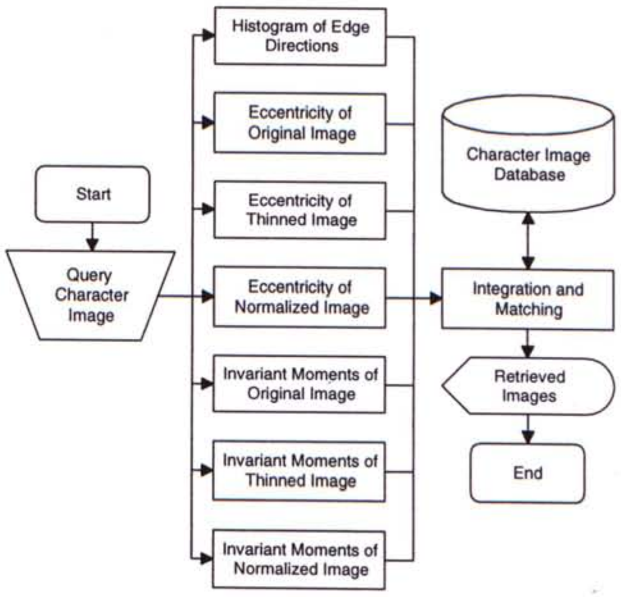

Histogram of Edge Directions
Eccentricity of Original Image
Eccentricity of Thinned Image
Eccentricity of Normalized Image
Invariant Moments of Original Image
Invariant Moments of Thinned Image
Invariant Moments of Normalized Image
Start
Query Character Image
Character Image Database
Integration and Matching
Retrieved Images
End

Figure 6.5: The proposed shape-based Chinese cursive script character image retrieval system.

6.3.2 Automatic Feature Extraction

In the proposed system, several shape features are chosen to represent a cursive script character (see Figure 6.5). They are edge direction histogram of the original image, eccentricity of the original, thinned and linearly normalized image, and the first three invariant moments of the original, thinned and linearly normalized image. The properties of edge direction histogram, eccentricity and invariant moments are discussed in Chapter 3. Thinned and normalized images are explained in the next section.

6.3.3 Thinned Image and Linearly Normalized Image

For each image, we find its thinned image and its linearly normalized image. A thinned character has a one-pixel width and a normalized character has an aspect ratio of one. Figure 6.6 shows an example of the original, thinned and linearly normalized image. We use these two heuristics to compensate for the effects of different width of strokes and different aspect ratios in characters.

Then, we store the edge directions of the original image, eccentricity of the

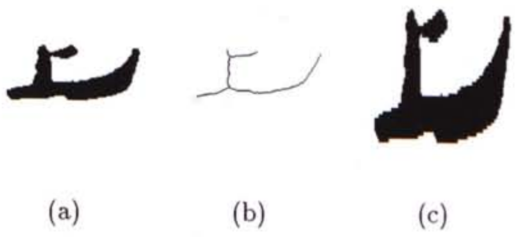


Figure 6.6: An example of thinned and normalized image. (a) Original image. (b) Thinned image. (c) Normalized image.

original, thinned and linearly normalized image, and the invariant moments of the original, thinned and linearly normalized image as a feature set for image retrieval.

6.3.4 Edge Directions

A histogram of edge directions is used to represent an image. The edge information is extracted using Canny edge operator [4] with $\sigma = 1$ and Gaussian mask of size = 9. The histogram of edge directions is represented by 36 bins, each spanning 10°. Euclidean distance metric is used to compute the dissimilarity value between two edge direction histograms. Figure 6.7 demonstrates the shape representation using edge directions.

In order to achieve invariance to scale, the histograms are normalized to the number of edge points in the image. The edge direction representation method is not rotation invariant. To reduce the effect of rotation, we smooth the histograms using the method proposed in [29]. A histogram is treated as a 1-D discrete signal. Smoothing is defined as:

$$I_s[i] = \frac{\sum_{j=i-k}^{i+k} I[j]}{2k+1}, \tag{6.1}$$

where I_s is the smoothed histogram. I is the original histogram and the parameter k determines the degree of smoothing. In our experiments, we used $k = 1$.

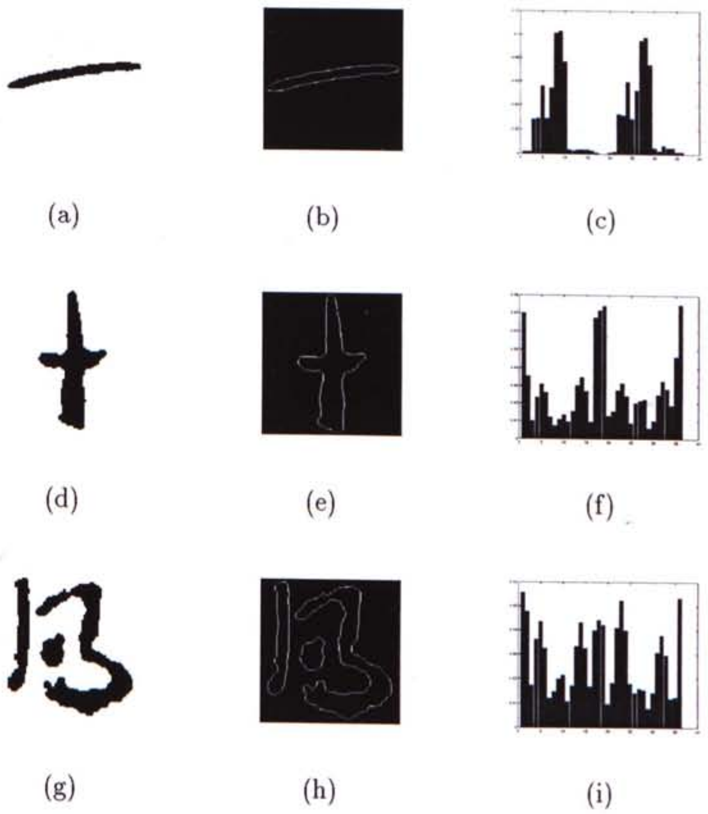


Figure 6.7: An example of shape representation using edge directions. (a), (d) and (g) Original images. (b), (e) and (h) Edge images. (c), (f) and (i) Histograms of edge directions.

6.3.5 Integration of Shape Features

The extracted feature vectors of the images are normalized by the maximum value of each corresponding feature and stored in the database. The weighting factors of the dissimilarity function are found by the proposed GA. When a user raises a query, the features of the query image are first extracted. Then, the extracted features are matched linearly with the features in the database and integrated by the dissimilarity function. After that, the images are displayed in the order of similarity.

6.4 Experimental Results

We conducted several different experiments to measure the performance of our proposed system. In Experiment 8, We found the weights in the dissimilarity function using the proposed GA. In Experiment 9, 10, and 11, we tested the speed and recall rate of the system. In Experiment 12, we compared different integration methods for retrieval. Experiments were conducted with a database of 1400 monochromatic cursive character images on an Ultra 5 Machine running Matlab V5.3. Appendix B shows some representative images in our database.

6.4.1 Experiment 8: Weight Assignment using Genetic Algorithm

In this experiment, we tested our proposed GA and found the appropriate weighting factors for various shape features. The weights found were used in the Experiment 9, 10, 11, and 12.

The proposed GA was tested with the following setup. The population size $PopSize$ was 30. The maximum number of iterations was set to 1000 because according to our testing, the proposed GA always converges within 1000 iterations. In addition, there were 50 training pairs TP. The size of the database DB was 1400. The values of the weights (genes) were bounded by 0 and 1. The probability of application of crossover was 0.6 and the probability of application of mutation was 0.05. The experiments were repeated 10 times with different random seeds. The evaluation function was the total count $TC(\boldsymbol{w})$. The ranking score function s was the same as Eq. 5.5 in Chapter 5.

Figure 6.8 shows the best result and the worst result obtained among the 10 trials. From the result, the GA improved the accuracy of the retrievals of the training examples by changing the weights in the dissimilarity function. For the best result, it converged in 500 iterations with a total count of 42.37, which means on average, the distance function ranks the target of a query in a training example to the 22nd position. The best weights found are presented in Table 6.1 and were used as the final weights for retrieval afterward.

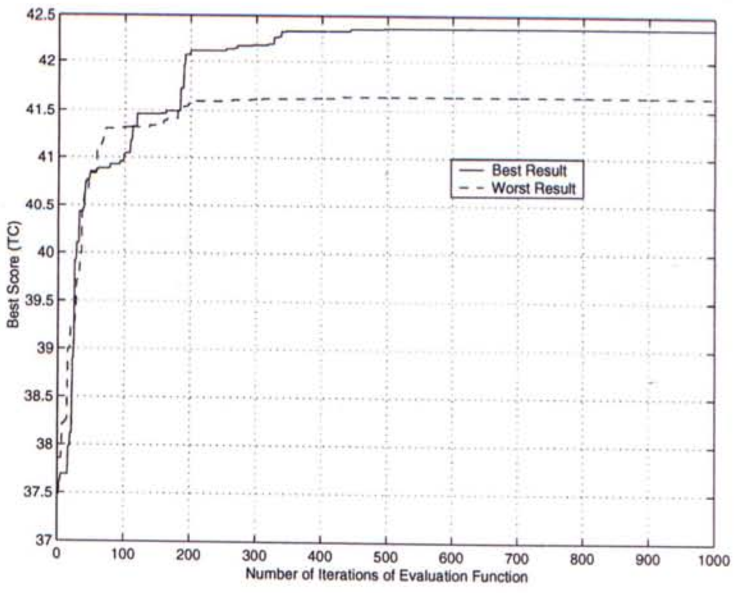


Figure 6.8: Results of Experiment 8. TC versus the number of iterations.

The performance of the distance function found can be further analyzed by the ranking histogram in Figure 6.9(a) and (b). From the histogram of the best result, 74% of the targets were ranked at the top 14 positions, 80% were ranked at the top 28 positions, 86% were ranked at the top 70 positions, and 94% were ranked at the top 140 positions. However, several targets were ranked very low. This is because images that appear to be similar need not be similar in their shape. Another reason is that the features that we used cannot model the similarity of those training examples. However, we believe that by introducing more effective features, the accuracy can be improved.

Table 6.1: Results of Experiment 8. Weights found for different shape features.

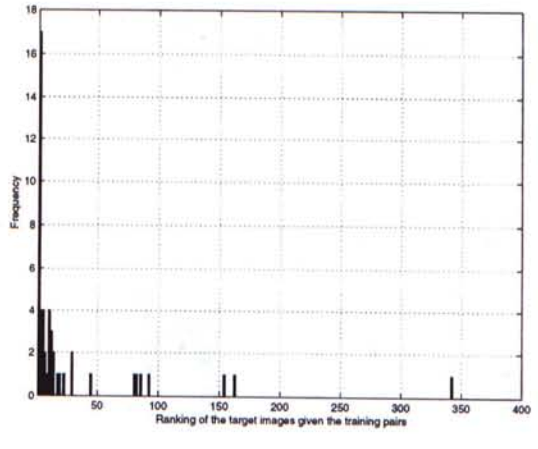
Feature	Weight	Percentage
Edge direction histogram of original image	0.0224	0.40
Eccentricity of original image	0.0433	0.77
Eccentricity of normalized image	0.1497	2.66
Eccentricity of thinned image	0.0494	0.88
First invariant moment of original image	0.3756	6.67
Second invariant moment of original image	0.9073	16.11
Third invariant moment of original image	0.9854	17.50
First invariant moment of normalized image	0.1156	2.05
Second invariant moment of normalized image	0.1051	1.87
Third invariant moment of normalized image	0.9962	17.69
First invariant moment of thinned image	0.9755	17.32
Second invariant moment of thinned image	0.0136	0.24
Third invariant moment of thinned image	0.8930	15.86

6.4.2 Experiment 9: Speed on Feature Extraction and Retrieval

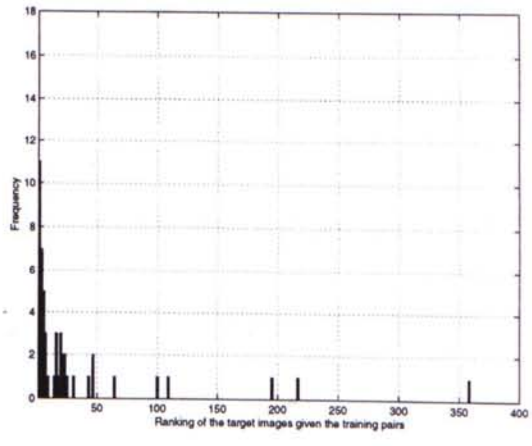
In this experiment, we tested the speed on feature extraction and image retrieval. When a query was submitted to the system, its features would be computed first. Next, the similarity measure between the query and all images in the database were computed. The average elapse time of 100 trials for feature extraction of the query image and database query are listed in Table 6.2.

Table 6.2: Results of Experiment 9. Average elapse time for retrieving an image from a database of 1400 images on a Sun Ultra 5 Machine.

Feature Extraction	Database query	Total
11.61s	0.04s	11.65s



(a)



(b)

Figure 6.9: Results of Experiment 8. (a) The ranking histogram of the best result among the 10 trials. (b) The ranking histogram of the worst result among the 10 trials.

6.4.3 Experiment 10: Evaluation by Recall for Deformed Images

Our aim is to develop a Chinese cursive script image retrieval system that is insensitive to variations on image deformation. In this experiment, we tested the behavior of our image retrieval system in the presence of the deformation transformation as shown in Figure 5.11. Several cursive character images shown in Figure 6.10 were submitted as query images to the database. Similar objective evaluation criteria has also been used by Kim [34]. A set of 100 deformed images as shown in Figure 6.11 was used for performance evaluation. The deformed images were submitted as query images to our retrieval system to examine whether the deformed images can retrieve their original ones or not.

Table 6.3 presents the results of recall rate in the top 70 candidates queried by the deformed images, where n corresponds to the position of the correct retrieval. From the results, 90% of the deformed images can recall their original ones in the top 28 candidates which is equal to 2% of the database size. The average recall

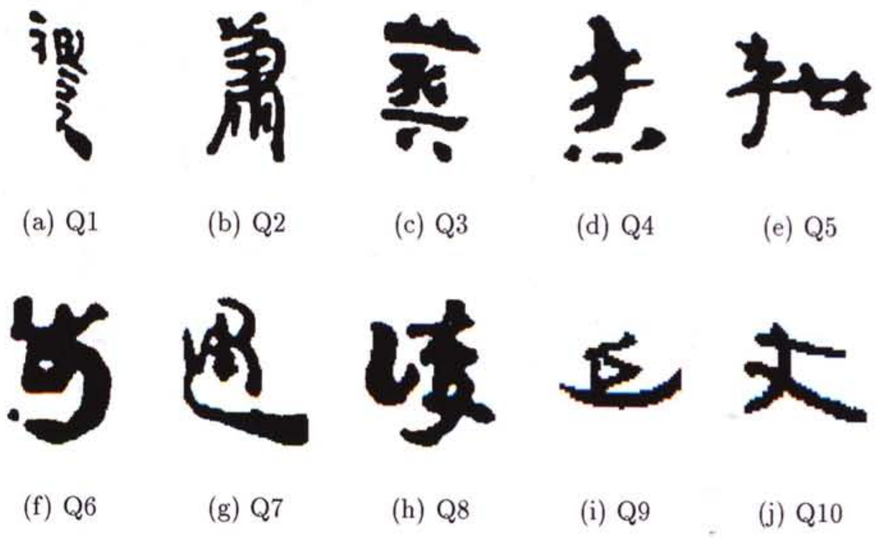


Figure 6.10: Query character images.

rate of the original ones among the top 70 candidates queried by deformed images was 94%.

6.4.4 Experiment 11: Evaluation by Recall for Rotated and Scaled Images

In order to test the recall rate for rotated and scaled images, we conducted the following experiment.

1. **Rotated Query:** Every image in Figure 6.10 was rotated arbitrarily with the range of $\pm 10^\circ$ and then presented as the query image. We only tested with the range stated above because our target is not to retrieve arbitrarily rotated images and we assume the query image to our system is directed upward.

2. **Scaled Query:** Every image in Figure 6.10 was scaled arbitrarily and then presented as the query image. The scaling factors were bounded by 0.5 and 50.

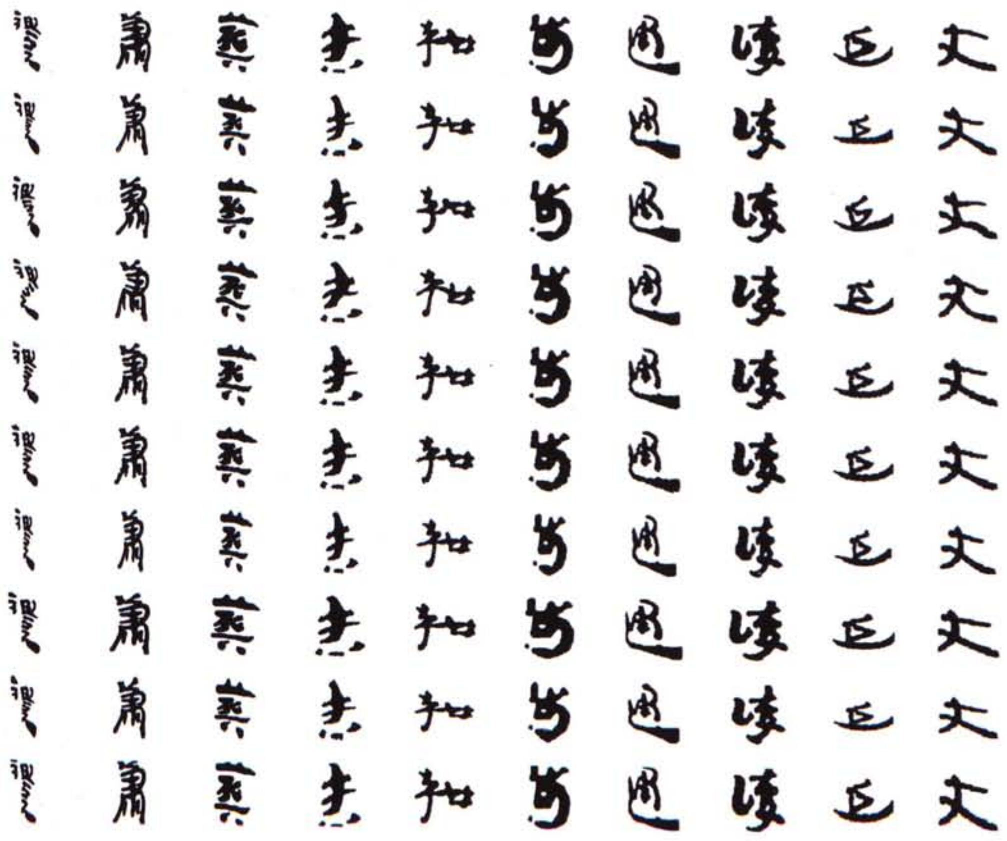


Figure 6.11: 100 queries deformed from 10 character images for performance evaluation.

The experiments were repeated with 100 rotated queries and 100 scaled queries. From the results presented in Table 6.4, we notice that the shape features were very effective in retrieving scaled images. However, the system only recalled 98% of the rotated images in the top 70 positions. This is because we used an edge direction histogram to represent an image, which is rotation sensitive. We consider not to use an edge direction histogram for representing a character image in our future development.

Table 6.3: Results of Experiment 10. Recall for deformed images. n refers to the position of the correct retrieval. For the query nature, please refer to Figure 5.11.

Query Nature	$n=1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
(a) Pinch	50%	70%	80%	90%	90%	90%	10%
(b) Pinch	40%	50%	80%	90%	90%	100%	0%
(c) Twirl	10%	20%	50%	80%	80%	90%	10%
(d) Twirl	10%	10%	40%	80%	80%	80%	20%
(e) Ripple	20%	20%	60%	80%	80%	90%	10%
(f) Ripple	20%	30%	40%	70%	80%	90%	10%
(g) HE	70%	90%	100%	100%	100%	100%	0%
(h) HE	60%	60%	90%	100%	100%	100%	0%
(i) VE	80%	100%	100%	100%	100%	100%	0%
(j) VE	70%	80%	100%	100%	100%	100%	0%
Avg.	43%	53%	74%	89%	90%	94%	6%

Table 6.4: Results of Experiment 11. Recall for rotated and scaled images. n refers to the position of the correct retrieval.

Query Nature	$n=1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
(a) Rotated	64%	70%	83%	91%	96%	98%	2%
(b) Scaled	100%	100%	100%	100%	100%	100%	0%

6.4.5 Experiment 12: Comparison of Different Integration Methods

In this experiment, we compared the recall rate of different integration methods. The 100 deformed images in Figure 6.11 were used as queries to recall their original ones. The integration methods include:

1. **Equal weights.** All the weights were set to 1 for all the features in the dissimilarity function. This method has been used in [30] for integrating two shape features.

2. **Weights determined by individual recall rate.** A weighted Euclidean distance were used as the distance function. The weights used are proportional to the accuracy of retrievals using individual features. The accuracy of a particular shape feature was determined by the same experimental setup as in Experiment 10. This method was proposed by [30] but they did not tested its performance.

3. **Integration with the method in QBIC (See Eq. 2.1).** The distance between two images is computed using a weighted Euclidean distance with the inverse of feature variances used for normalization.

4. **Weights found by GA.** The distance between two images is computed using a weighted Euclidean distance with the weights found by our proposed GA.

Table 6.5: Results of Experiment 12. Retrieval results on the basis of different integration methods. n refers to the position of the correct retrieval. (a) Equal weights. (b) Weights determined by individual recall rate. (c) QBIC. (d) Our system.

Query Nature	$n = 1$ (%)	$n \leq 2$ (%)	$n \leq 5$ (%)	$n \leq 14$ (%)	$n \leq 28$ (%)	$n \leq 70$ (%)	Not Retrieved(%)
(a)	43%	50%	62%	73%	80%	88%	12%
(b)	41%	47%	54%	69%	76%	83%	17%
(c)	51%	57%	65%	80%	86%	93%	7%
(d)	43%	53%	74%	89%	90%	94%	6%

From Table 6.5, the dissimilarity function found by our GA has a better recall rate than that of the other methods. This is expected because the weights were optimized by the GA based on the training examples given by the users. Hence, its recall rate should be better than that of the other integration methods.

6.5 Chapter Summary

In this chapter, our prototype system for Chinese cursive script character image retrieval has been discussed. Several shape features including histogram of edge directions, invariant moments, and eccentricities were used to increase the robustness of the system. We also used the heuristic of the original images, the thinned images, and the normalized images for extracting features. The weighting factors of the shape attributes in the dissimilarity function were determined by the proposed supervised learning method using GA. Experimental results show that our system was robust to retrieve scaled and deformed images. This demonstrates the power of a content-based image retrieval system that given a query image, the system retrieves all the similar images from the database. Our ultimate goal is to build a supporting tool for the calligraphers and artists to do comparative study on Chinese cursive script characters.

Chapter 7

Conclusion

Traditional text-based image retrieval systems have many limitations; therefore, content-based search has been introduced for solving the problems. In this thesis, we have addressed the problem of content-based image retrieval from iconic image databases based on shape features. This chapter summarizes the results of the research and outlines several directions for future research.

7.1 Summary

We have proposed a supervised learning technique for finding the weighting factors of the dissimilarity function using a genetic algorithm. Given a set of training examples, the algorithm assigns suitable weights for different image features. The weights found can be used for image retrieval afterward. Besides, we have developed two prototype shape-based image retrieval systems. The two systems achieve both the desired accuracy and efficiency.

The first system is a trademark image retrieval system. We propose to use an approximated boundary to capture the global shape of a trademark. Invariant moments, eccentricity, Fourier descriptors of approximated boundary, and circularity of approximated boundary are used to represent a trademark. The weights of the features in the dissimilarity function are determined by the proposed supervised learning technique. A trademark database of 1400 images has been tested. Experimental results show that the weights found by the GA improved the accuracy of retrieval and our system was robust on retrieving rotated, scaled, deformed, mirrored, and hand drawn images. Moreover, the retrieved images by our system agreed well with human perception. Each query took 2.3 seconds on average.

The second system is a Chinese cursive script character image retrieval system. We propose to use the original, thinned, and normalized images to extract the shape features including invariant moments, edge directions, and eccentricity. The weights for different shape features are found by the proposed GA. We have tested the performance of the system on a database of 1400 cursive images. The results show that the weights found by the GA improved the accuracy of retrievals. Moreover, our system was robust on retrieving scaled and deformed images. Each query took 11.7 seconds on average.

The proposed supervised learning technique is scalable for solving the problem with a database of more than 1400 images. The execution time for training grows linearly with the number of images in the database. Although it takes a long time to find the weights, the training algorithm only needs to be executed once. The weights found can be used for image retrieval afterward.

7.2 Future Research

There are several limitations in the proposed systems. A number of issues can be addressed to extend the capability of the systems in the future.

7.2.1 Limitations

1. **Learning Capabilities:** The systems currently have no learning capabilities after the GA training. Therefore, they cannot improve their accuracy

with time.

2. **Partial Matching of Objects:** The systems cannot match based on segmented objects.

3. **Indexing Structure:** The retrieval system now has no indexing structure. The retrieval time increases with the number of images linearly.

7.2.2 Future Directions

First, a future extension may be in the direction of building a system to learn through positive and negative examples of every query. Second, the systems can be extended to match on the basis of segmented objects rather than the entire image. An algorithm for comparing the difference of two sets of objects with their spatial information can be studied. Hence, the systems can allow partial matching of objects. Finally, a further step could be in the direction of building an index structure for the database.

Appendix A

A Representative Subset of Trademark Images

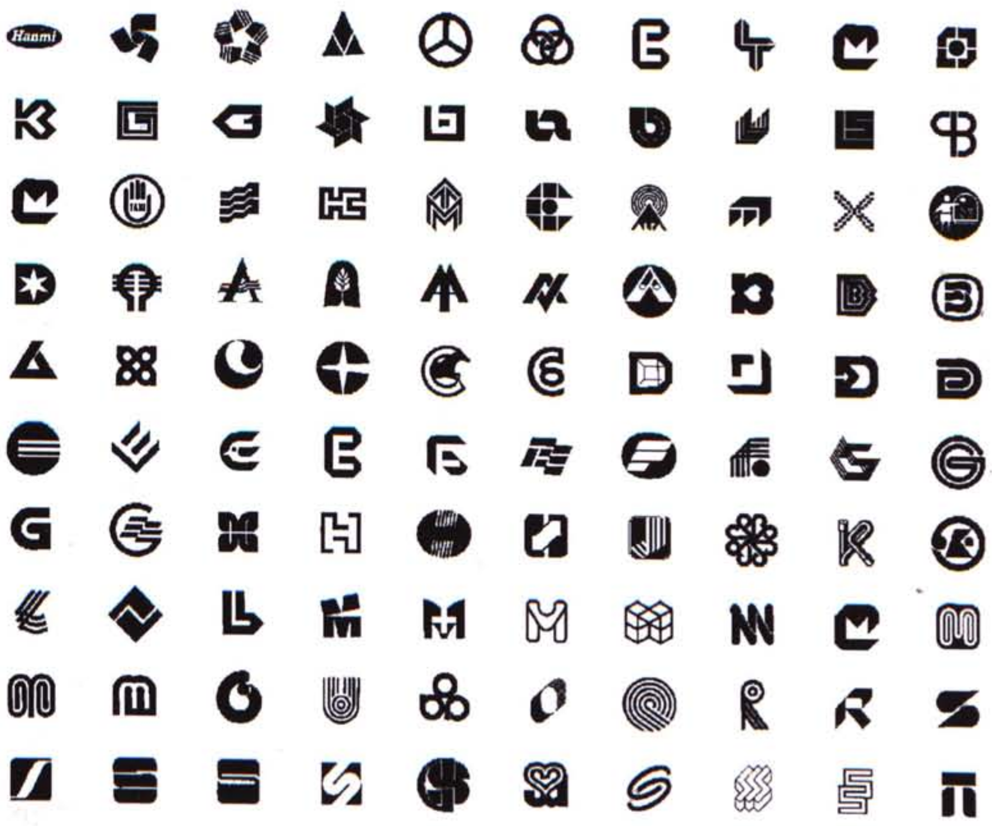


Figure A.1: Examples of trademarks present in the database.

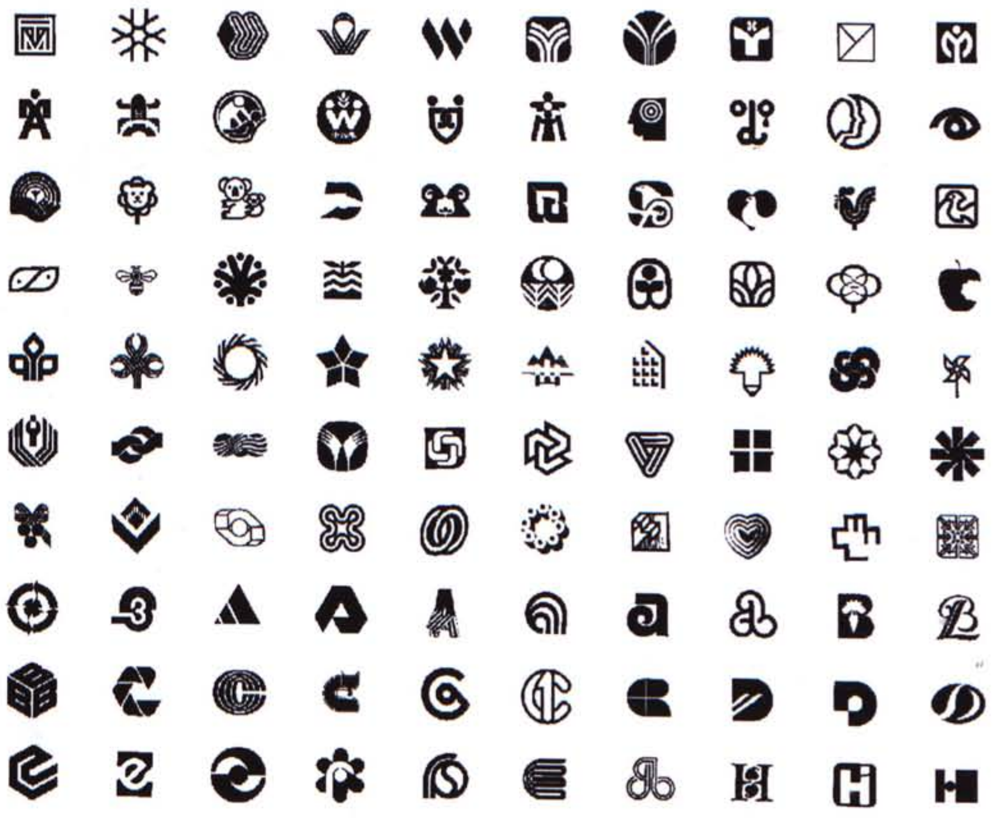


Figure A.2: Examples of trademarks present in the database.

Appendix B

A Representative Subset of Cursive Script Character Images

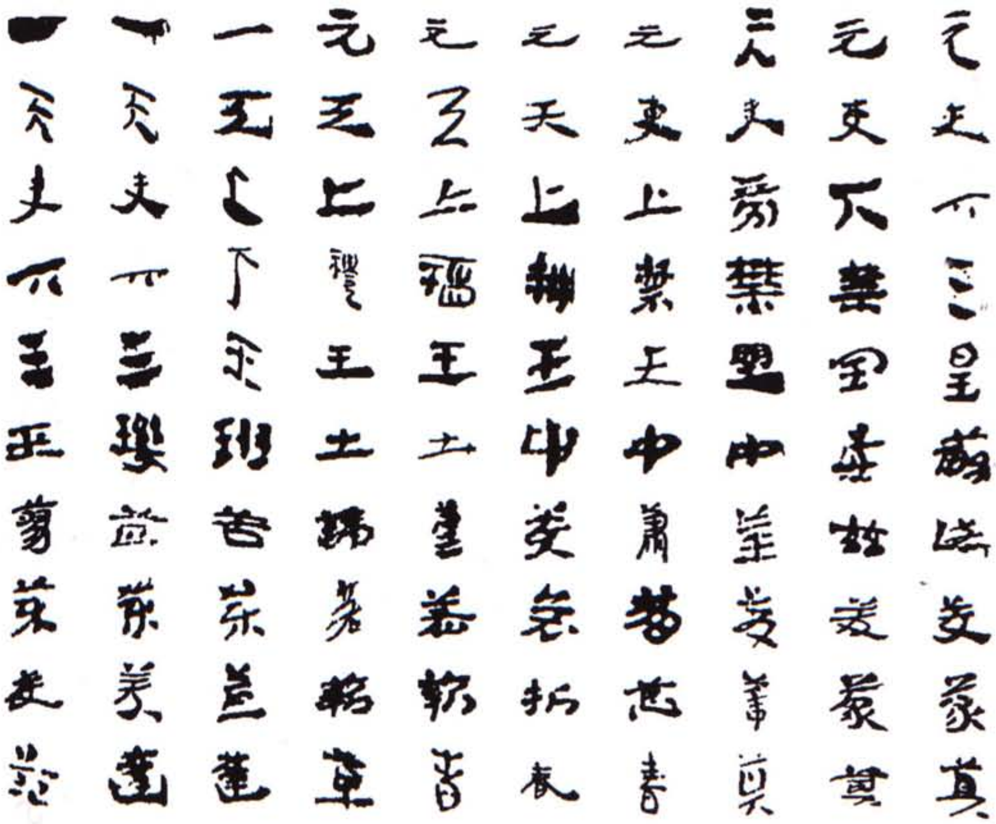


Figure B.1: Examples of Chinese cursive script characters present in the database.

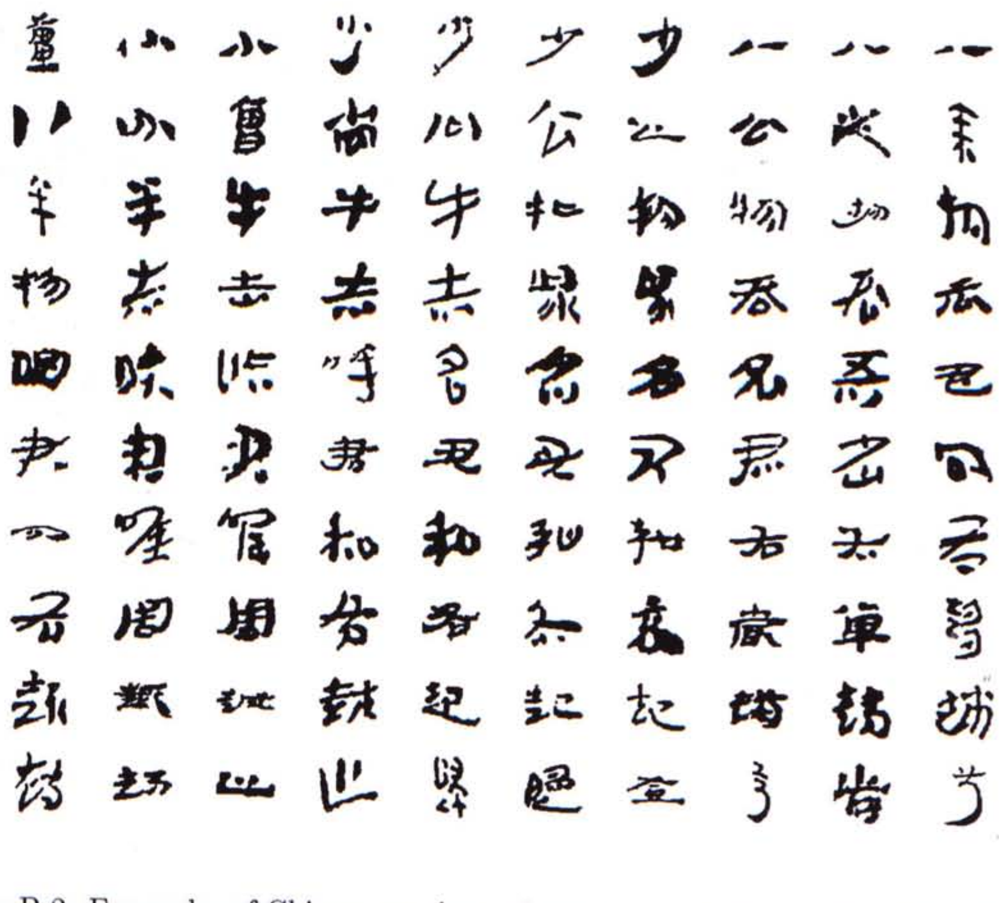


Figure B.2: Examples of Chinese cursive script characters present in the database.

Appendix C

Shape Feature Extraction Toolbox for Matlab V5.3

C.1 central_moment

```
function [m] = central_moment(fig,p,q)
% Filename: CENTRAL_MOMENT.M
%
%   Format: m=central_moment(fig,p,q)
%
%           m   - moment
%           fig - binary image
%           p,q - order p,q
%
%    Usage: Calculate the central moment of a binary image
%           fig with the order of p,q.
%
%  Example: >> f = imread('circle.tif');
%           >> m = central_moment(f, [2 0 1], [0 2 1]);
%           >> m20 = m(1)
%           >> m02 = m(2)
%           >> m11 = m(3)
%
%     Note: This function supports vector processing.
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

 fig = double(fig);
 [ny nx] = size(fig);
 [cx cy] = centroid(fig);
m = zeros(size(p));

 for i = 1:ny
   for j = 1:nx
```

```
            m = m + fig(i,j).*(i-cx).^p.*(j-cy).^q;
        end
    end
return
```

C.2 centroid

```
function [cx,cy] = centroid(fig)
% Filename: CENTROID.M
%
%   Format: [cx,cy]=centroid(fig)
%
%           cx, cy - centroid
%           fig    - binary image
%
%    Usage: Calculate the centroid of a binary image
%
%  Example: >> fig = [ 1 1 1; 1 1 1; 1 1 1]
%           >> [x,y] = centroid(fig)
%           >> fig = [ 0 0 0; 1 1 1; 1 1 0]
%           >> [x,y] = centroid(fig)
%           >> fig = imread('circle.tif');
%           >> [x,y] = centroid(fig)
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

  m00 = sum(sum(fig));
  m = moment(fig, [1 0], [0 1]);

  cx = m(1) / m00;
  cy = m(2) / m00;
return
```

C.3 cir

```
function [c]=cir(fig)
% Filename: CIR.M
%
%   Format: c=cir(fig)
%
%           c   - circularity
%           fig - binary image
%
%    Usage: Calculate the circularity of a binary shape
%           (circularity = perimeter * perimeter/area)
%
%  Example: >> cir(imread('circle.tif'))
%           >> cir(imread('rect.tif'))
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

  perimeter = bwarea(bwperim(fig,4));
```

```
    % minimize quantization error
    area = bwarea(fig) - (perimeter/2);

    c=(perimeter^2)/area;
return
```

C.4 css

```
function [peak_list,X,Y]=css(x,y,is_plot_peak)
% Filename: CSS.M
%
%   Format: [peak_list,X,Y]=css(x,y,is_plot_peak)
%
%          peak_list    - peaks
%          X            - list of points after curve evolution
%          Y            - list of points after curve evolution
%          x            - list of points (x-coordinate)
%                         (resampled to 200 pts)
%          y            - list of points (y-coordinate)
%                         (resampled to 200 pts)
%          is_plot_peak - plot the CSS image, 1 or 0
%
%    Usage: Find the shape descriptors in the Curvature
%           Scale Space (CSS) Representation
%
% Example: >> [x,y] = gen_shape(200); plot(x,y,'-x');
%          >> [a,X,Y] = css(x,y,1);
%          >> figure; plot(X,Y,'x');
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

    if length(x)~=200
       error('[x y] should be resampled to 200 points');
       break;
    end
    if length(y)~=200
       error('[x y] should be resampled to 200 points');
       break;
    end

    start_sigma=7;
    end_sigma=60;
    range=250;
    number_of_samples = 199;

    % if you want to plot the CSS representation with the peaks
    if is_plot_peak
       figure;
    end

    % store the final peak list for indexing
    peak_list=[];

    % store the current zero-crossing list
    zero_cross=[];
    len_curr=0;

    % store the previous zero-crossing list
    prev_zero_cross=[];
```

```
len_prev =0;

for i=start_sigma:1:end_sigma
  g=gen_gaussian_vector(number_of_samples, i , range);

  X=warp_conv(x,g);
  Y=warp_conv(y,g);
  Xt = my_dev(X);
  Yt = my_dev(Y);
  Xtt = my_dev(Xt);
  Ytt = my_dev(Yt);

  k=(Xt.*Ytt-Xtt.*Yt) ./ (((Xt.*Xt)+(Yt.*Yt)).^(1.5));

  sec_dev_k=my_dev(my_dev(k));
  prev_zero_cross=zero_cross;

  % Find the zero-crossings
  zero_cross=find(sec_dev_k .* ...
    [sec_dev_k(200); sec_dev_k(1:199)] < 0);

  % length of current zero_cross and prev_zero_cross
  len_b4_prev=len_prev;
  len_prev=len_curr;
  len_curr=length(zero_cross);

  % error handling rountine
  if (len_prev ~=0) & (len_curr>len_prev)
    % fprintf('Warning: Sigma=%d\n',i);
    if (i>=70)
      zero_cross =[];
      peak_list =[peak_list ; ...
        find_peak(prev_zero_cross', zero_cross', i)];
      break;
    end
  end

  if (i>=end_sigma)
    % leave the for loop
    zero_cross =[];
    peak_list =[peak_list ; ...
      find_peak(prev_zero_cross', zero_cross', i)];
    break;
  elseif (len_curr < len_prev) & (len_curr < len_b4_prev)
    % find the peak
    peak_list =[peak_list ; ...
      find_peak(prev_zero_cross', zero_cross', i)];
  end

  % if you want to plot the CSS representation
  if is_plot_peak
    hold on;
    if ~isempty(zero_cross)
      plot(zero_cross, i, '.r');
    end
  end
end

% if you want to plot the CSS representation with the peaks
if is_plot_peak
  hold on;
  if ~isempty(peak_list)
    plot(peak_list(:,2), peak_list(:,1), 'bx');
    axis([1 200 7 60]);
```

```
         hold off;
       end
     end

return

% Find the peaks given 2 list of points
 function [peak] = find_peak(a, b, sigma)

for (j=0:4)
   k=1;
   for (i=1:length(b))
      if isempty(b)
         break;
      end
      newa = a(a~=b(k)+j);
      if length(newa)~=length(a)
         a = newa;
         b = b(b~=b(k));
      else
         k=k+1;
      end
   end

   k=1;
   for (i=1:length(b))
      if isempty(b)
         break;
      end

      newa = a(a~=b(k)-j);
      if length(newa)~=length(a)
         a = newa;
         b = b(b~=b(k));
      else
         k=k+1;
      end
   end
end

a1=a;
diff1 =0;
for (i=1:2:length(a1)-1)
   diff1=diff1+abs(a1(i+1)-a1(i));
end

a2=[a(2:length(a)) 200+a(1)];
diff2 =0;
for (i=1:2:length(a2)-1)
   diff2=diff2+abs(a2(i+1)-a2(i));
end

if diff1<diff2
   k=1;
   for (i=1:2:length(a1)-1)
      peak(k,2)= ceil((a1(i)+a1(i+1))/2);
      peak(k,1)=sigma;
      k=k+1;
   end
else
   k=1;
   for (i=1:2:length(a2)-1)
      peak(k,2)= ceil((a2(i)+a2(i+1))/2);
      peak(k,1)=sigma;
      k=k+1;
    end
    if peak(length(peak(:,1)),2)>200
      peak(length(peak(:,1)),2)= peak(length(peak(:,1)),2)-200;
    end
  end


% Find the warp around diff.
function [result]=my_dev(a)
a_len = length(a);
a = a';
result =[a(2:a_len) a(1)]'-[a(a_len) a(1:a_len-1)]';


% Gaussian Vector
function [vector]=gen_gaussian_vector(number_of_element,...
  sigma,range)
vector=gaussian(linspace(-range,range,number_of_element),...
  sigma);
vector=vector-vector(1);
vector=[vector./sum(vector)]';


function [result]=gaussian(t,sigma)
result = exp(-t.*t/(2*sigma*sigma))/(sigma*sqrt(2*pi));
```

C.5 css_match

```
function [real_diff] = css_match(x,y)
% Filename: CSS_MATCH.M
%
%    Format: [diff]=ccs_match(x,y)
%
%            diff - the difference between the 2 sets of peaks
%            x, y - 2 sets of peaks
%
%     Usage: Calculate the difference of 2 CSS images.
%            Each CSS image is represented by its peaks.
%
%   Example: >> a = [9 153;10 158;24 187;44 47;60 93;60 122;60 199];
%            >> b = [9 88;10 59;24 88;44 148;60 23;60 100;60 194];
%            >> css_match(a,b)
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0


  x= sortrows(x,[1 2]);
  y= sortrows(y,[1 2]);
  len_x = length(x);
  len_y = length(y);

  real_diff=Inf;
  error_limit = 0.3*50;              % Error limit

  for j = 1:200                      % y_shift index
    x_copy = x;
    len_x_copy = len_x;
```

```
      y_copy = y;
      len_y_copy = len_y;

      curr_diff=0;

      % match the y(k,:) to x_copy(l,:)
      % here, y(k,:) is the model for matching
      % and,  x_copy(l,:) is the target for matching

      for k = 1:len_y
        for l = 1:len_x_copy
          tmp1 = calc_diff(x_copy(l,:), y(k,:));
          if tmp1 <= error_limit
            curr_diff=curr_diff+tmp1;
            for m=1:len_y_copy
              if y_copy(m,:) == y(k,:)
                y_copy = [y_copy(1:m-1, :);
                y_copy(m+1:len_y_copy , :)];
                len_y_copy = len_y_copy - 1;
                break;
              end
            end %m
            for n=1:len_x_copy
              if x_copy(n,:) == x_copy(l,:)
                x_copy = [x_copy(1:n-1, :);
                x_copy(n+1:len_x_copy , :)];
                len_x_copy = len_x_copy - 1;
                break;
              end
            end
            break
          end
        end
      end

      if ~isempty(x_copy)
        x_sum=sum(x_copy(:,1));
      else
        x_sum=0;
      end

      if ~isempty(y_copy)
        y_sum=sum(y_copy(:,1));
      else
        y_sum=0;
      end

      curr_diff = curr_diff + y_sum + x_sum;

      if curr_diff < real_diff
        real_diff = curr_diff;
      end

      y = shift_right(y,1);
    end
return

% Shift the CSS representation to the right by skip_points of points
function [shifted_m] = shift_right(m, skip_points)
  shifted_m = [m(:,1), mod(m(:,2)+skip_points,200)];
  shifted_m = sortrows(shifted_m,[1 2]);
return

% Calc the diff between 2 points in CSS representation
function [diff_ab] = calc_diff(a,b)
```

```
  diff_ab = sqrt(min(sum((a-b).^2), ...
              sq(a(:,1)-b(:,1))+sq(a(:,2)+200-b(:,2))));
  diff_ab = sqrt(min( diff_ab , ...
              sq(a(:,1)-b(:,1))+sq(b(:,2)+200-a(:,2))));
return

% Calc the square of x
function [y] = sq(x)
  y=x*x;
return
```

C.6 ecc

```
function [e] = ecc(fig)
% Filename: ECC.M
%
%    Format: e=ecc(fig)
%
%            e   - eccentricity
%            fig - binary image
%
%     Usage: Calculate the eccentricity of a binary image
%
%   Example: >> fig = [ 1 1 1; 1 1 1; 1 1 1]
%            >> ecc(fig)
%            >> fig = [ 0 0 0; 1 1 1; 1 1 0]
%            >> ecc(fig)
%            >> fig = imread('circle.tif');
%            >> ecc(fig)
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0

  u = central_moment(fig, [2 0 1], [0 2 1]);
  u20 = u(1);
  u02 = u(2);
  u11 = u(3);

  e = (((u20-u02)^2 + 4*(u11^2)) / (u20+u02)^2);
return
```

C.7 edge_directions

```
function [histogram,eout] = edge_directions(img, ...
  number_of_bin, k)
% Filename: EDGE_DIRECTIONS.M
%
%   Format: [histogram,eout]=edge_directions(img, ...
%             number_of_bin, k)
%
%           histogram     - Edge direction histogram
%           eout          - edges
%           img           - binary image
%           number_of_bin - number of bins in the histogram
%           k             - smoothing factor
%
%    Usage: find the edge direction histogram of a
%              binary image img
%
%   Example : >> a = imread('rect.tif');
%             >> [h,e] = edge_directions(a, 18, 5);
%             >> figure; plot(h);
%             >> figure; imshow(e);
%
%      Note : modified from the canny edge detection
%             program, edge.m, from Matlab 5.3
%
%    Author : David Chan
%   Address : CSE Dept, CUHK, Hong Kong
%    E-mail : ymchan@cse.cuhk.edu.hk
%
%      Date : June-1999
%   Version : 1.0

a = img;
thresh = [];
sigma = 1;
H = [];
kx = 1;
ky = 1;

% Transform to a double precision intensity image
if isa(a, 'uint8')
   a = im2double(a);
end

m = size(a,1);
n = size(a,2);
rr = 2:m-1; cc=2:n-1;

% The output edge map:
e = zeros(m,n);

% Magic numbers
GaussianDieOff = .0001;
% Used for selecting thresholds
PercentOfPixelsNotEdges = .7;
% Low thresh is this fraction of the high.
ThresholdRatio = .4;

% Design the filters - a gaussian and its derivative

pw = 1:30; % possible widths
ssq = sigma*sigma;
width = max(find(exp(-(pw.*pw)/(2*ssq))>GaussianDieOff));
if isempty(width)
   width = 1;  % the user entered a really small sigma
end
t = (-width:width);
len = 2*width+1;
% We will average values at t-.5, t, t+.5
t3 = [t-.5; t; t+.5];
% the gaussian 1-d filter
gau = sum(exp(-(t3.*t3)/(2*ssq))).'/(6*pi*ssq);
% derivative of a gaussian
dgau = (-t.* exp(-(t.*t)/(2*ssq))/ ssq).';

% Convolve the filters with the image in each direction
% The canny edge detector first requires convolutions with
% the gaussian, and then with the derivitave of a gauusian.
% I convolve the filters first and then make a call to conv2
% to do the convolution down each column.

```

```
    ra = size(a,1);
    ca = size(a,2);
    ay = 255*a;  ax = 255*a';

    h = conv(gau,dgau);
    ax = conv2(ax, h, 'same').';
    ay = conv2(ay, h, 'same');
    ang = round(atan2(ay,ax) .* 180 ./ pi);

    mag = sqrt((ax.*ax) + (ay.*ay));
    magmax = max(mag(:));
    if magmax>0
       mag = mag / magmax;     % normalize
    end

    % Select the thresholds
    [counts,x]=imhist(mag, 64);
    highThresh = min( find (cumsum( counts ) > ...
       PercentOfPixelsNotEdges*m*n)) / 64;
    lowThresh = ThresholdRatio*highThresh;
    thresh = [lowThresh highThresh];

    % The next step is to do the non-maximum supression.
    % We will accrue indices which specify ON pixels in strong edgemap
    % The array e will become the weak edge map.
    idxStrong = [];
    for dir = 1:4
       idxLocalMax = cannyFindLocalMaxima( dir,ax,ay,mag);
       idxWeak = idxLocalMax(mag(idxLocalMax) > lowThresh);
       e(idxWeak)=1;
       idxStrong = [idxStrong; idxWeak(mag(idxWeak) > highThresh)];
    end

    rstrong = rem(idxStrong-1, m)+1;
    cstrong = floor((idxStrong-1)/m)+1;
    e = bwselect(e, cstrong, rstrong, 4);
    % Thin double (or triple) pixel wide contours
    e = bwmorph(e, 'thin', 1);

    eout = e;

    edgeindex = find(eout~=0);
    edg = ang(edgeindex);

    histogram = hist(edg,number_of_bin);
    histogram = warp_conv(histogram', ones(k,1));
    histogram = histogram ./ size(edg,1) * 1;           % normalization

return

function idxLocalMax = cannyFindLocalMaxima(direction,ix,iy,mag)
%
% This sub-function helps with the non-maximum supression in the
% Canny edge detector.  The input parameters are:
%
% direction-the index of which direction the gradient is pointing,
%              read from the diagram below. direction is 1, 2, 3, or 4.
% ix        -input image filtered by derivative of gaussian along x
% iy        -input image filtered by derivative of gaussian along y
% mag       -the gradient magnitude image
%
%    there are 4 cases:
%
%                          The X marks the pixel in question, and each
%      3     2          of the quadrants for the gradient vector
%    O----O----O        fall into two cases, divided by the 45
%  4 |         | 1      degree line.  In one case the gradient
%    |         |        vector is more horizontal, and in the other
%    O    X    O        it is more vertical.  There are eight
%    |         |        divisions, but for the non-maximum supression
% (1)|         |(4)     we are only worried about 4 of them since we
%    O----O----O        use symmetric points about the center pixel.
%     (2)   (3)


  [m,n] = size(mag);

  % Find the indices of all points whose gradient (specified by the
  % vector (ix,iy)) is going in the direction we're looking at.

  if (direction==1)
      idx = find((iy<=0 & ix>-iy)  | (iy>=0 & ix<-iy));
   elseif (direction==2)
      idx = find((ix>0 & -iy>=ix)  | (ix<0 & -iy<=ix));
   elseif (direction==3)
      idx = find((ix<=0 & ix>iy) | (ix>=0 & ix<iy));
   elseif (direction==4)
      idx = find((iy<0 & ix<=iy) | (iy>0 & ix>=iy));
  end

  % Exclude the exterior pixels
  if ~isempty(idx)
      v = mod(idx,m);
      extIdx = find(v==1 | v==0 | idx<=m | (idx>(n-1)*m));
      idx(extIdx) = [];
  end

  ixv = ix(idx);
  iyv = iy(idx);
  gradmag = mag(idx);

  % Do the linear interpolations for the interior pixels
  if (direction==1)
      d = abs(iyv./ixv);
      gradmag1 = mag(idx+m).*(1-d) + mag(idx+m-1).*d;
      gradmag2 = mag(idx-m).*(1-d) + mag(idx-m+1).*d;
   elseif (direction==2)
      d = abs(ixv./iyv);
      gradmag1 = mag(idx-1).*(1-d) + mag(idx+m-1).*d;
      gradmag2 = mag(idx+1).*(1-d) + mag(idx-m+1).*d;
   elseif (direction==3)
      d = abs(ixv./iyv);
      gradmag1 = mag(idx-1).*(1-d) + mag(idx-m-1).*d;
      gradmag2 = mag(idx+1).*(1-d) + mag(idx+m+1).*d;
   elseif (direction==4)
      d = abs(iyv./ixv);
      gradmag1 = mag(idx-m).*(1-d) + mag(idx-m-1).*d;
      gradmag2 = mag(idx+m).*(1-d) + mag(idx+m+1).*d;
  end
  idxLocalMax = idx(gradmag>=gradmag1 & gradmag>=gradmag2);
return
```

C.8 fourier_d

```
function [des]=fourier_d (x,y,n)
```

```
% Filename: FOURIER_D.M
%
%   Format: [des]=fourier_d(x,y,n)
%
%           des  - the Fourier descripters
%                  (The power spectra is returned)
%             x  - a list of points (x-coordinate)
%             y  - a list of points (y-coordinate)
%             n  - resample to n points
%
%    Usage: Resample [x y] to n points and then Find the
%           Fourier Descriptors of [x y]
%
%  Example: >> x = [0.2129; 0.2137; 0.3153; 0.3734; 0.4379];
%           >> y = [0.4202; 0.1940; 0.2963; 0.2186; 0.3240];
%           >> [des] = fourier_d(x,y,32);
%
%     Note: des(1) is the power spectrum of the centroid of
%           the normalized shape which is not translation invariant.
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

  % make sure the boundary pts are in anti-clockwise direction
  if isclockwise(x,y)==1
    x = flipud(x);
    y = flipud(y);
  end

  % divide the points to n points
  [x_new,y_new] = resample_pts(x,y,n);

  des = abs( fft(x_new+y_new*i));
return
```

C.9 gen_shape

```
function [x,y] = gen_shape(n)
% Filename: GEN_SHAPE.M
%
%   Format: [x,y]=gen_shape(n)
%
%           x - x coordinate of the list of points
%           y - y coordinate of the list of points
%           n - number of points in the shape
%
%    Usage: A random shape generator
%
% Example:  >> [x,y] = gen_shape(30); plot(x,y,'-x');
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

rand('state',sum(100*clock));              % use diff. random seed

for i = 1:Inf
```

```
    x = 20;                     % Initial x,y
    y = 20;
    used_matrix = zeros(100,100);           % For checking of 'crossover'
    old_angle = 0;

    previous_x = x;                         % Previous coordinates
    previous_y = y;

    for i = 1:300                           % Max trying limit
      % get new random coordinates
      angle = ( fix(rand(1)*12)) * pi / 6;
      while ( angle == old_angle)
        angle = ( fix(rand(1)*12)) * pi / 6;
      end
      len_between_pt = rand(1) * 5 + 5;
      new_x = fix(previous_x + len_between_pt*cos( angle ));
      new_y = fix(previous_y + len_between_pt*sin( angle ));

      % get new random coordinates
      while ((new_x > 90) | (new_y > 90) | (new_x < 10) | (new_y < 10))
        angle = ( fix(rand(1)*12)) * pi / 6;
        while ( angle == old_angle)
          angle = ( fix(rand(1)*12)) * pi / 6;
        end
        len_between_pt = rand(1) * 5 + 5;
        new_x = fix(previous_x + len_between_pt*cos( angle ));
        new_y = fix(previous_y + len_between_pt*sin( angle ));
      end

      % Whether there is a crossover or not?
      [used_matrix, is_success] = ...
        fill_used_matrix(used_matrix, new_x, new_y, x(1), y(1));

      if is_success==1
        old_angle = angle;
        x = [new_x; x];
        y = [new_y; y];
        previous_x = new_x;
        previous_y = new_y;
      end

      x_len = length(x);

      tmp_matrix = used_matrix;
      [used_matrix, is_success] = ...
        fill_used_matrix(used_matrix, x(x_len), y(x_len), x(1), y(1));

      if (is_success==1) & (i>50)
        break;
      else
        used_matrix = tmp_matrix;
      end
    end

    if is_success==1
      [x,y] = resample_pts(x,y,n);  % divide the points to 30 points
      gf = [1 3 3 1];
      x=warp_conv(x,gf);
      y=warp_conv(y,gf);
      break;
    end
end
return
```

```
% Check for crossover
function [out_matrix,is_success]=fill_used_matrix(used_matrix,...
            x1,y1,x2,y2)

  out_matrix=used_matrix;
  is_success=0;

  if (x1==x2) | (y1==y2)
      return
  end

  % Slope
  m = (y2-y1)/(x2-x1);

  % Slope < 1
  if abs(m) < 1
     if x1 > x2
        [x1, x2] = swap(x1, x2);
        [y1, y2] = swap(y1, y2);
     end
     for x=x1+1:x2-1
        y = m*(x-x1) + y1;
        if used_matrix(x, ceil(y)) == 0 & used_matrix(x, floor(y))==0
           used_matrix(x, ceil(y)) = 1;
           used_matrix(x, floor(y)) = 1;
        else
              return;
        end
     end
  else
     if y1 > y2
        [x1, x2] = swap(x1, x2);
        [y1, y2] = swap(y1, y2);
     end
     for y=y1+1:y2-1
        x = (y-y1)/m + x1;
        if used_matrix(ceil(x),y) == 0 & used_matrix(floor(x),y)==0
           used_matrix(ceil(x),y) = 1;
           used_matrix(floor(x),y) = 1;
        else
              return;
        end
     end
  end

  is_success = 1;
  used_matrix(x1,y1) = 1;
  used_matrix(x2,y2) = 1;
  out_matrix = used_matrix;
return

% Swapping
function [a,b] = swap(a,b)
  tmp = a;
  a = b;
  b = tmp;
return
```

C.10 hu7

```
function [m] = hu7(fig)
% Filename: HU7.M
```

```
%
%     Format : m=hu7(fig)
%
%              m   - Hu's 7 moment
%              fig - binary image
%
%      Usage : Calculate the Hu's 7 moment of a binary image fig
%
%    Example : >> hu7(imread('arrow.tif'))
%              >> hu7(imrotate(imread('arrow.tif'),20))
%              >> hu7(imresize(imread('arrow.tif'),[20 20]))
%              >> hu7(imresize(imread('arrow.tif'),[50 50]))
%
%     Author : David Chan
%    Address : CSE Dept, CUHK, Hong Kong
%     E-mail : ymchan@cse.cuhk.edu.hk
%
%       Date : June-1999
%    Version : 1.0

  nm = normalized_moment(fig, [2 0 1 3 1 2 0], [0 2 1 0 2 1 3]);

  n20 = nm(1);
  n02 = nm(2);
  n11 = nm(3);
  n30 = nm(4);
  n12 = nm(5);
  n21 = nm(6);
  n03 = nm(7);

  h1 = n20 + n02;
  h2 = sq(n20-n02) + 4*sq(n11);
  h3 = sq(n30-3*n12) + sq(3*n21-n03);
  h4 = sq(n30+n12) + sq(n21+n03);
  h5 = (n30-3*n12)*(n30+n12)*(sq(n30+n12)-3*sq(n21+n03)) + ...
       (3*n21-n03)*(n21+n03)*(3*sq(n30+n12)-sq(n21+n03));
  h6 = (n20-n02)*(sq(n30+n12)-sq(n21+n03))+4*n11*(n30+n12)*(n21+n03);
  h7 = (3*n21-n03)*(n30+n12)*(sq(n30+n12)-3*sq(n21+n03)) + ...
       (3*n12-n30)*(n21+n03)*(3*sq(n30+n12)-sq(n21+n03));

  m = [h1 h2 h3 h4 h5 h6 h7];
return


% Find the square of b
function [a] = sq(b)
  a = b*b;
return
```

C.11 isclockwise

```
function [yesno] = isclockwise(x,y)
% Filename : ISCLOCKWISE.M
%
%     Format : flag=isclockwise(x,y)
%
%              flag - the return value, 1 or 0
%              x    - x coordinate of the list of points
%              y    - y coordinate of the list of points
%
%      Usage : Assume the list of points form a boundary shape.
%              To test whether the list of points are in the
%              order of clockwise direction or not.
```

```
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0

 % find the index of the array in x and y for the MBR
 id1 = find(y==min(y));
 id2 = find(x==min(x));
 id3 = find(y==max(y));
 id4 = find(x==max(x));
 id = [id1(1) id2(1) id3(1) id4(1)];

 % sort the items
 sid=sort(id);

 % to see whether it is in clockwise or anti-clockwise direction
 if (id==sid)
   yesno = 1;
  elseif (id==[sid(2:4) sid(1)])
   yesno = 1;
  elseif (id==[sid(3:4) sid(1:2)])
   yesno = 1;
  elseif (id==[sid(4) sid(1:3)])
   yesno = 1;
 else
   yesno = 0;
 end

return
```

C.12 moment

```
function [m] = moment(fig,p,q)
% Filename: MOMENT.M
%
%    Format: m=moment(fig,p,q)
%
%            m   - moment
%            fig - binary image
%            p,q - order p,q
%
%     Usage: Calculate the simple moment of a binary image fig
%            with the order of p,q.
%
%   Example: >> m = moment(imread('circle.tif'), [2 0 1], [0 2 1]);
%            >> m20 = m(1)
%            >> m02 = m(2)
%            >> m11 = m(3)
%
%      Note: This function supports vector processing.
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0

 fig = double(fig);
 [ny nx] = size(fig);
 m = zeros(size(p));
```

```
    for i = 1:ny
       for j = 1:nx
         m = m + fig(i,j).*(i.^p).*(j.^q);
       end
    end
return
```

C.13 normalized_moment

```
function [m] = normalized_moment(fig,p,q)
% Filename: NORMALIZED_MOMENT.M
%
%    Format: m=normalized_moment(fig,p,q)
%
%            m   - moment
%            fig - binary image
%            p,q - order p,q
%
%     Usage: Calculate the normalized moment of a binary image fig
%            with the order of p,q.
%
%   Example: >> f = imread('circle.tif');
%            >> m = normalized_moment(f, [2 0 1], [0 2 1]);
%            >> m20 = m(1)
%            >> m02 = m(2)
%            >> m11 = m(3)
%
%      Note: This function supports vector processing.
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0

  m00 = sum(sum( fig ));
  mpq = central_moment(fig,p,q);
  m = mpq ./ ( m00 .^ (( p+q+2)/2));
return
```

C.14 orientation

```
function [o] = orientation(fig)
% Filename: ORIENTATION.M
%
%    Format: o=orientation(fig)
%
%            o    - orienataion in radian (from 0 to pi)
%            fig  - binary image
%
%     Usage: Calculate the orientation of an object, which is
%            defined as the angle between the x-axis and the
%            axis around which the object can be rotated with
%            the minumum inertia.
%
%   Example: >> fig = imread('arrow.tif');
%            >> orientation(fig)
%            >> imshow(fig);
%
%      Note: NaN is returned if the shape is symmetrical
 %            (no orientaion)
 %
 %   Author: David Chan
 %  Address: CSE Dept, CUHK, Hong Kong
 %   E-mail: ymchan@cse.cuhk.edu.hk
 %
 %     Date: June-1999
 %  Version: 1.0

   warning off;
   u = central_moment(fig, [2 0 1 3], [0 2 1 0]);
   u20 = u(1);
   u02 = u(2);
   u11 = u(3);
   u30 = u(4);

   o = 0.5 * atan(2*u11/(u20-u02));
   if (u20 >= u02) | (u30 < 0)
     o = o + pi/2;
   end
   o = mod(o, pi);
   warning on;
 return
```

C.15 resample_pts

```
function [vector_x, vector_y] = resample_pts(x, y, num)
% Filename: RESAMPLE_PTS.M
%
%   Format: [new_x, new_y]=resample_pts(x,y,num)
%
%        vector_x  - the new list of points (x-coordinate)
%        vector_y  - the new list of points (y-coordinate)
%               x  - the old list of points (x-coordinate)
%               y  - the old list of points (y-coordinate)
%             num  - target number of points
%
%    Usage: Resample the list of points [x y]
%           and return a list of num points [vector_x vector_y]
%
%  Example: >> x = [0.2129; 0.2137; 0.3153; 0.3734; 0.4379];
%           >> y = [0.4202; 0.1940; 0.2963; 0.2186; 0.3240];
%           >> plot([x; x(1)],[y; y(1)],'-x');
%           >> [new_x, new_y] = resample_pts(x,y,200);
%           >> figure; plot([new_x;new_x(1)],[new_y;new_y(1)],'-x');
%
%   Author: David Chan
%  Address: CSE Dept, CUHK, Hong Kong
%   E-mail: ymchan@cse.cuhk.edu.hk
%
%     Date: June-1999
%  Version: 1.0

  ori_len=length(x);          % original length
  dis_arr=distance(x,y);      % distance array
  arc_len=sum(dis_arr);       % total arc length
  x=x./arc_len;               % normalize to 1
  y=y./arc_len;
  dis_arr=distance(x,y);      % recalculate the distance array

  vector_x=x(1);
  vector_y=y(1);
  x=[x; x(1)];
  y=[y; y(1)];

  cnt=1;                          % find the points by tracing the boundary
  for i=2:num
    remaining=1/num;
    while (1)
      if remaining > dis_arr(cnt)
        remaining=remaining-dis_arr(cnt);
        cnt=cnt+1;
      else
        next_x=x(cnt+1);
        cur_x=x(cnt);
        next_y=y(cnt+1);
        cur_y=y(cnt);
        d=dis_arr(cnt);

        new_x=cur_x+(next_x-cur_x)/d*remaining;
        new_y=cur_y+(next_y-cur_y)/d*remaining;
        dis_arr(cnt)=d-remaining;
        x(cnt) = new_x;
        y(cnt) = new_y;
        break;
      end
    end

    vector_x=[vector_x; new_x];    % the return values
    vector_y=[vector_y; new_y];
  end

return

% Return a distance vector that store the distances % every 2 points
function [distance_vector] = distance(x,y)

  tmp = x;
  tmp(1) = [];
  x_shift = [tmp; x(1)];

  tmp = y;
  tmp(1) = [];
  y_shift = [tmp; y(1)];

  distance_vector = sqrt((x-x_shift).^2 + (y-y_shift).^2);

return
```

C.16 rectangularity

```
function [r,n_img] = rectangularity(img)
% Filename: RECTANGULARITY.M
%
%   Format: [r, n_img]=rectangularity(img)
%
%           r     - rectangularity
%           n_img - the image with orientation changed to 0
%           img   - binary image
%
%    Usage: Calculate the rectangulrity of an object, which is
%           defined as its area divided by its minumum
%           boundary rectangle.
```

```
%
%
%   Example : >> fig = imread('arrow.tif');
%             >> figure; imshow(fig);
%             >> [r, n_fig] = rectangularity(fig);
%             >> figure; imshow(n_fig);
%             >> r
%
%    Author : David Chan
%   Address : CSE Dept, CUHK, Hong Kong
%    E-mail : ymchan@cse.cuhk.edu.hk
%
%      Date : June-1999
%   Version : 1.0

  % normalize the orientation
  o = orientation(img)*180/pi;
  if (~isnan(o))
    n_img = imrotate(img, -o);
  end

  % area of the image
  a = bwarea(img);

  % area of the MBR
  [i,j] = find(n_img~=0);
  a_rect = (max(i) - min(i) + 1)  * (max(j) - min(j) + 1);

  r = a / a_rect;
return
```

C.17 trace_points

```
function [list_x, list_y] = trace_points(fig)
% Filename : TRACE_POINTS
%
%    Format : [list_x, list_y] = trace_points(fig)
%
%             list_x  - a list of points (x-coordinate)
%             list_y  - a list of points (y-coordinate)
%                fig  - a binary image with a single boundary
%
%     Usage : Given a black-and-white image of a single boundary,
%             the algorithm trace the boundary and output a
%             list of points of the boundary shape.
%
%   Example : >> a = imread('tri.tif');
%             >> figure; imshow(a);
%             >> e = edge(a,'zerocross',0);
%             >> [x,y] = trace_points(e);
%             >> figure; plot(x,y);
%
%    Author : David Chan
%   Address : CSE Dept, CUHK, Hong Kong
%    E-mail : ymchan@cse.cuhk.edu.hk
%
%      Date : June-1999
%   Version : 1.0

  [x,y] = find(fig);                       % find a starting point
  original = [x(10) y(10)];

  set(0,'RecursionLimit',100000)  % increase the recursion limit
```

```
    % find the point list of the image
    [list_x,list_y] = trace_pt(1, original, original, fig);
    list_x = list_x(1:length(list_x)-1);
    list_y = list_y(1:length(list_y)-1);
  return


 % A recursive function for tracing the boundary points of a boundary
 % shape
  function [listx,listy] = trace_pt(is_start, ...
             original, current, pattern)
 warning off;
 sx = current(1);
 sy = current(2);

 tmpx=[];
 tmpy=[];

 if (original==current) & (is_start>10)
    listx=-1;
    listy=-1;
  elseif pattern(sx,sy) ~= 1
    listx=[];
    listy=[];
 else
   pattern(sx,sy)=0;
   for i=1:8

      if i==1
        next=[sx+1, sy];
      elseif i==2
        next=[sx, sy-1];
      elseif i==3
        next=[sx-1, sy];
      elseif i==4
        next=[sx, sy+1];
      elseif i==5
        next=[sx+1, sy-1];
      elseif i==6
        next=[sx-1, sy-1];
      elseif i==7
        next=[sx-1, sy+1];
      elseif i==8
        next=[sx+1, sy+1];
     end

     [tmpx tmpy] = trace_pt(is_start+1, original, next, pattern);
     if ~isempty(tmpx)
        if tmpx(length(tmpx)) == -1
           listx=[sx tmpx];
           listy=[sy tmpy];
           break;
        end
     end
   end
end
```

C.18 warp_conv

```
function [result] = warp_conv(original, filter)
% Filename: WARP_CONV.M
%
%   Format: [C]=warp_conv(A, B)
```

```
%
%     Usage: Warp-around convolution of vector A with vector B
%
%   Example: >> x = [0.2129; 0.2137; 0.3153; 0.3734; 0.4379];
%            >> y = [0.4202; 0.1940; 0.2963; 0.2186; 0.3240];
%            >> [new_x, new_y] = resample_pts(x,y,50);
%            >> figure; plot([new_x;new_x(1)],[new_y;new_y(1)],'-x');
%            >> filter = [1 6 15 20 15 6 1];
%            >> cx = warp_conv(new_x, filter);
%            >> cy = warp_conv(new_y, filter);
%            >> figure; plot([cx; cx(1)],[cy; cy(1)],'-x');
%
%    Author: David Chan
%   Address: CSE Dept, CUHK, Hong Kong
%    E-mail: ymchan@cse.cuhk.edu.hk
%
%      Date: June-1999
%   Version: 1.0

  len_ori=length(original);
  tmp = [conv(original, filter)]';
  len_tmp=length(tmp);

  result = [tmp(1:len_ori) + ...
    [tmp((len_ori+1):len_tmp) zeros(1, 2*len_ori-len_tmp)]]';
return
```

Bibliography

[1] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in QBIC. In *Proceedings of Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 24–35, February 1995.

[2] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331, 1990.

[3] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[5] K.R. Castleman. *Digital Image Processing.* Prentice-Hall, INC., Englewood Cliffs, N.J., 1979.

[6] D.Y.M. Chan. A two-stage approach for shape retrieval using curvature scale space. In *The First ACM Hong Kong Postgraduate Research Day*, pages 150–153, Hong Kong, 1998.

[7] D.Y.M. Chan and I. King. Weight assignment in dissimilarity function for Chinese cursive script character image retrieval using genetic algorithm. To appear in IRAL'99.

[8] D.Y.M. Chan and I. King. Genetic algorithm for weights assignment in dissimilarity function for trademark retrieval. In *Third International Conference on Visual Information and Information Systems (VISUAL'99)*, volume 1614 of *Lecture Notes in Computer Science*, pages 557–565, The Netherlands, 1999. Springer.

[9] K.A. De Jong. Adaptive system design: A genetic approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(9):556–574, 1980.

[10] S.A. Dudani, K.J. Breeding, and R.B. McGhee. Aircraft identification by moment invariants. *IEEE Transactions on Computers*, C-26(1):39–45, January 1977.

[11] J.P. Eakins, J.M. Boardman, and M.E. Graham. Similarity retrieval of trademark images. *IEEE Multimedia*, 5:53–63, apr–jun 1998.

[12] J.P. Eakins, J.M. Boardman, and K. Shields. Retrieval of trade mark images by shape feature - the ARTISAN project. In *IEE Colloquium on Intelligent Image Databases*, 1996.

[13] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligient Information Systems*, 3(3/4):231–262, July 1994.

[14] R.A. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[15] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, September 1995.

[16] H. Freeman. Analysis of the precision of generalized chain codes for the representation of planar curves. *Pattern Analysis and Machine Intelligence*, 3(5):533–539, September 1981.

[17] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[18] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[19] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[20] A. Guttman. R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD*, 14(2):47–57, June 1984.

[21] R.M. Haralick, S.R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987.

[22] K. Hirata and T. Kato. Query by visual example - content based image retrieval. In *Advances in Database Technology EDBT'92, Third International Conference on Extending Database Technology*, pages 56–71, March 1992.

[23] J. Holland. *Adaptation in natural and artifical systems*. The University of Michigan Press, 1975.

[24] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8, 1962.

[25] C.L. Huang and D.H. Huang. A content-based image retrieval system. *Image and Vision Computing*, 16:149–163, 1998.

[26] IBM. IBM - 1997 prototype trademark database, 1997. http://wwwqbic.almaden.ibm.com/tmdemo/.

[27] IPD. Frequently asked questions - trademarks. Trade Marks Registry, Intellectual Property Department, Hong Kong Government, 1998. http://www.houston.com.hk/hkgipd/faq_tm1.html.

[28] B. Jahne. *Digital Image Processing: Concepts, Algorithms and Scientific Applications*. Springer-Verlag, Berlin; New York, 4 edition, 1997.

[29] A.K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, 1996.

[30] A.K. Jain and A. Vailaya. Shape-based retrieval: A case study with trademark image databases. *Pattern Recognition*, 31(9):1369–1390, 1998.

[31] T. Kato. Database architecture for content-based image retrieval. *SPIE Image Storage and Retrieval Systems*, 1662:112–123, 1992.

[32] A. Khotanzad and Y.H. Hong. Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5), May 1990.

[33] W.Y. Kim, , and P. Yuan. A practical pattern recognition system for translation, scale and rotation invariance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 391–396, 1994.

[34] Y.S. Kim and W.Y. Kim. Content-based trademark retrieval system using a visually salient feature. *Image and Vision Computing*, 16(12–13):931–939, 1998.

[35] I. King, A. Fu, L.W. Chan, and L. Xu. Montage: An Image Database for the Hong Kong's Textile, Fashion, and Clothing Industry, 1995. http://www.cse.cuhk.edu.hk/~viplab.

[36] I. King and T.K. Lau. A Feature-Based Image Retrieval Database for the Fashion, Textile, and Clothing Industry in Hong Kong. In *Proceedings of the 1996 International Symposium on Multimedia Information Processing (IS-MIP'96)*, 1996.

[37] S. Kuo and G.R. Cross. A two-step string-matching procedure. *Pattern Recognition*, 24(7):711–716, 1991.

[38] C.P. Lam, J.K. Wu, and B. Mehtre. STAR - a system for trademark archival and retrieval. In *2nd Asian Conference on Computer Vision*, volume 3, pages 214–217, 1995.

[39] T.K. Lau and S.T. Hung. Montage: An Image Database for Effective Digital Image Management. In *Hong Kong International Computer Conference 1997 (HKICC'97)*, volume 1, pages 83–88, 1997.

[40] T.K. Lau and I. King. Montage: An Image Database for the Fashion, Textile, and Clothing Industry in Hong Kong. In *Proceedings of the Third Asian Conference on Computer Vision (ACCV'98)*, volume 1, pages 410–417, 1998.

[41] K.C. Ma. *Shu Pu I Chu / Ma Kuo-Chuan I Chu.* Hong Kong : Shao-Hua Wen Hua Fu Wu She, 1977.

[42] A.K. Mackworth and F. Mokhtarian. The renormalized curvature scale space and the evolution properties of planar curves. Technical Report TR-87-37, Department of Computer Science, University of British Columbia, November 1987.

[43] P. Maragos. Tutorial on advances in morphological image processing and analysis. *Optical Engineering*, 26(7):623–632, 1987.

[44] F. Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):539–544, May 1995.

[45] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *First International Workshop on Image Databases and Multi-Media Search*, pages 35–42, Amsterdam, The Netherlands, August 1996.

[46] F. Mokhtarian and A.K. Mackworth. A theory of multiscale, curvature-based representation of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):789–805, August 1992.

[47] F. Mokhtarian and H. Murase. Silhouette-based recognition through curvature scale space. In *Fourth International Conference on Computer Vision*, pages 269–274, Berlin, Germany, 1993.

[48] D. Mumford. The problem of robust shape descriptions. In *First International Conference on Computer Vision, (London, England, June 8–11, 1987)*, pages 602–606, Washington, DC., 1987. IEEE Computer Society Press.

[49] D. Mumford. Mathematical theories of shape: Do they model perception? In *Geometric Methods in Computer Vision*, volume 1570, pages 2–10. SPIE, 1991.

[50] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture and shape. In *Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187, 1993.

[51] E.A. Patrick. *Fundamentals of Pattern Recognition.* Prentice-Hall, INC., Englewood Cliffs, N.J., 1972.

[52] T. Pavlidis. The use of a syntactic shape analyzer for contour matching. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1:307–310, 1979.

[53] H.L. Peng and S.Y. Chen. Trademark shape recognition using closed contours. *Pattern Recognition Letters*, 18:791–803, 1997.

[54] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Content-Based Manipulation of Image Databases. *International Journal of Computer Vision*, 18(3):223–254, June 1996.

[55] E. Persoon and K.S. Fu. Shape discrimination using Fourier descriptors. *IEEE Transactions on Systems, Man Cybernetics*, 7(2):170–179, 77.

[56] E. Pettit and K.M. Swigger. An analysis of genetic-based pattern tracking. In *National Conference on AI, AAAI'83*, pages 327–332, 1983.

[57] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R^+-tree: a dynamic index for multidimensional objects. In *Proceedings of the 13th VLDB Conference*, pages 507–518, 1987.

[58] J.R. Smith and S.F. Chang. VisualSEEk: a fully automated content-based image query system. In *ACM multimedia - international conference - 1996*, pages 87–98, November 1996.

[59] A. Soffer and H. Samet. Negative shape features for image databases consisting of geographic symbols. In *3rd International Workshop in Visual Form Capri*, May 1997.

[60] A. Soffer and H. Samet. Using negative shape features for logo similarity matching. In *14th International Conference on Pattern Recognition*, volume 1, pages 571–573, 1998.

[61] T.W. Sze and Y.H. Yang. A simple contour matching algorithm. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 3(6):676–678, 1981.

[62] E.G. Talbi and P. Bessière. A parallel genetic algorithm for the graph partitioning problem. In *International Conference on Supercomputing 1991*, pages 312–320, Cologne, Germany, June 17–21, 1991. ACM SIGARCH.

[63] USPTO. Basic facts about registering a trademark. US patent and trademark office, 1995. http://www.uspto.gov/web/offices/tac/doc/basic/basic_facts.html.

[64] A. Vailaya, Y. Zhong, and A.K. Jain. A hierarchical system for efficient image retrieval. In *13th International Conference on Pattern Recognition*, pages 356–360, 1996.

[65] J.K. Wu, B.M. Mehtre, Y.J. Gao, C.P. Lam, and A.D. Narasimhalu. STAR - a multimedia database system for trademark registration. In Witold Litwin and Tore Risch, editors, *First International Conference on Applications of Databases*, volume 819 of *Lecture Notes in Computer Science*, pages 109–122, Vadstena, Sweden, 21–23 June 1994. Springer.

[66] J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.P. Gao. CORE: A content-based retrieval engine for multimedia information systems. *ACM Multimedia Systems*, 3(1):25–41, February 1995.

[67] P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. of the 4th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 311–321, 1993.

[68] C.T. Zahn and R.Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computer*, C-21(1):269–281, 1972.