

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2005

Neural network-based shape retrieval using moment invariants and Zernike moments.

Xiaoliu Chen
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Chen, Xiaoliu, "Neural network-based shape retrieval using moment invariants and Zernike moments." (2005). *Electronic Theses and Dissertations*. 2824.
<https://scholar.uwindsor.ca/etd/2824>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Neural Network-Based Shape Retrieval Using Moment Invariants and Zernike Moments

**by
Xiaoliu Chen**

A Thesis

**Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor**

Windsor, Ontario, Canada

2005

© 2005 Xiaoliu Chen



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-494-09759-0

Our file Notre référence

ISBN: 0-494-09759-0

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

Shape is one of the fundamental image features for use in Content-Based Image Retrieval (CBIR). Compared with other visual features such as color and texture, it is extremely powerful and provides capability for object recognition and similarity-based image retrieval. In this thesis, we propose a Neural Network-Based Shape Retrieval System using Moment Invariants and Zernike Moments. Moment Invariants and Zernike Moments are two region-based shape representation schemes and are derived from the shape in an image and serve as image features. k means clustering is used to group similar images in an image collection into k clusters whereas Neural Network is used to facilitate retrieval against a given query image. Neural Network is trained by the clustering result on all of the images in the collection using back-propagation algorithm. In this scheme, Neural Network serves as a classifier such that moments are inputs to the Neural Network and the output is one of the k classes that have the largest similarities to the query image.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my academic advisor, Dr. Imran Ahmad for giving me the opportunity to conduct research in this area and for his constant guidance throughout its duration. I am very thankful for his instruction, advice, patience and serious attitude that has been an invaluable source of support throughout my graduate program.

I would also like to acknowledge my committee members, Dr. Alioune Ngom, Dr. S. Ejaz Ahmed and Dr. Boubakeur Boufama for spending their precious time in reading my thesis and providing their valuable comments and advice on this work.

My special thanks go to my parents. Without their continuous support, it would not have been possible for me to finish this work.

Table of Contents

Abstract.....	iii
Acknowledgements.....	iv
List of Tables.....	viii
List of Figures	ix
1 Introduction	1
1.1 Low Level Features	2
1.1.1 Color.....	2
1.1.2 Texture	3
1.1.3 Shape	3
1.2 CBIR Systems	4
1.3 Problem Statement.....	5
2 Related Work.....	7
2.1 Shape Representations.....	7
2.1.1 Contour-Based Representations	8
2.1.1.1 <i>Chain Codes</i>	8
2.1.1.2 <i>Signatures</i>	10
2.1.1.3 <i>Fourier Descriptors</i>	11
2.1.2 Region-Based Representations.....	13
2.1.2.1 <i>Grid Descriptors</i>	13
2.1.2.2 <i>Moment Invariants</i>	14
2.1.2.3 <i>Zernike Moments</i>	17
2.2 Multidimensional Indexing.....	18
2.2.1 KDB Tree	18

2.2.2	R/R* Tree	19
2.2.3	SS Tree	20
2.2.4	SR Tree	20
2.2.5	Comparison	20
3	Thesis Approach	22
3.1	Shape Feature Extraction	22
3.1.1	Boundary Sequence	22
3.1.2	Moment Invariants from Extreme Boundary Pixels	23
3.1.3	Complex Zernike Moments and the Magnitudes	30
3.2	Indexing Mechanism	34
3.2.1	Retrieval Strategy	34
3.2.2	Clustering	35
3.2.2.1	<i>k Means Clustering</i>	35
3.2.2.2	<i>Distance Functions</i>	36
3.2.2.3	<i>Clustering Evaluation</i>	39
3.2.3	Neural Network	40
3.2.3.1	<i>Background of Neural Network</i>	41
3.2.3.2	<i>Multilayer Neural Network</i>	42
3.2.3.3	<i>Feed-Forward</i>	45
3.2.3.4	<i>Training by Back-Propagation</i>	50
4	Experiments and Discussions	55
4.1	Image Collection	55
4.2	Comparison on Euclidean and Mahalanobis Distances	56
4.3	Comparison on Activation Functions	58
4.4	Retrieval Performance of Moment Invariants and Zernike Moments	61

5	Conclusion and Future Directions	67
5.1	Conclusion.....	67
5.2	Future Direction	68
	References	70
	Vita Auctoris	77

List of Tables

Table 3.1 Example of Moment Invariants as shape features.....	29
Table 3.2 Example of Zernike Moments as shape features.....	34
Table 4.1 Clustering validity and time with Euclidean and Mahalanobis distances.....	57
Table 4.2 Errors of the Neural Network with three different activation functions	59

List of Figures

Figure 1.1 Image Retrieval System classification.....	1
Figure 1.2 A CBIR image retrieval system architecture	4
Figure 2.1 Example of contour-based and region-based shape similarity	8
Figure 3.1 Turtle procedure in binary object boundary following.....	22
Figure 3.2 Contribution type of boundary pixels	24
Figure 3.3 A Shape consisting of small uniform squares.....	26
Figure 3.4 Example of binary shapes and shapes under geometric transformation.....	29
Figure 3.5 Coordinate system transformation.....	30
Figure 3.6 Six orthogonal radial polynomials plotted for increasing ρ	32
Figure 3.7 Architecture of shape-based image retrieval system using Neural Network...	34
Figure 3.8 The unit spheres under Minkowsky distances	38
Figure 3.9 Euclidean distance and Mahalanobis distance.....	39
Figure 3.10 Basic unit of Neural Network	41
Figure 3.11 Multilayer (3-later) Neural Network.....	42
Figure 3.12 Neural Networks with one or two hidden layers and their corresponding type of decision region	44
Figure 3.13 Binary sigmoid.....	49
Figure 3.14 Bipolar Sigmoid.....	49
Figure 4.1 Sample binary shape images in the image collection	55
Figure 4.2 The values of DB index for k means clustering with Euclidean and Mahalanobis distances.....	57
Figure 4.3 Clustering time with Euclidean and Mahalanobis distances	58
Figure 4.4 The MSE of the Neural Network with three different activation functions	59
Figure 4.5 The average of errors of each node in the output layer of the Neural Network with three different activation functions	60
Figure 4.6 Measuring of retrieval effectiveness.....	61
Figure 4.7 Precision-recall graph of Moment Invariants and Zernike Moments (1)	63

Figure 4.8 Precision-recall graph of Moment Invariants and Zernike Moments (2)	63
Figure 4.9 Precision-recall graph of Moment Invariants and Zernike Moments (3)	64
Figure 4.10 Example of the shape image retrieval results using Moment Invariants (1) .	65
Figure 4.11 Example of the shape image retrieval results using Moment Invariants (2) .	65
Figure 4.12 Example of the shape image retrieval results using Zernike Moments (1) ...	66
Figure 4.13 Example of the shape image retrieval results using Zernike Moments (2) ...	66

1 Introduction

In real world, images play an essential role in a wide selection of fields such as art history, manufacturing, medicine, geologic exploration, astronomy, and even military defense to name a few. With advances in digital technology, image compression techniques, storage capabilities and processing power of computer systems, the volume of digital image collections has experienced a rapid increase in recent years, and diverse image applications based on large digital image collections have been emerging. To utilize the image information effectively, techniques for storage, searching, indexing, and retrieval need to be developed [16, 17].

Since the 1970's, image retrieval has been a very active research area within the disciplines of Database Management and Computer Vision. These two major research communities study image retrieval from two different points of view (Figure 1.1): the former primarily takes a text-based approach, whereas the later relies on visual properties of the image data [35, 3].

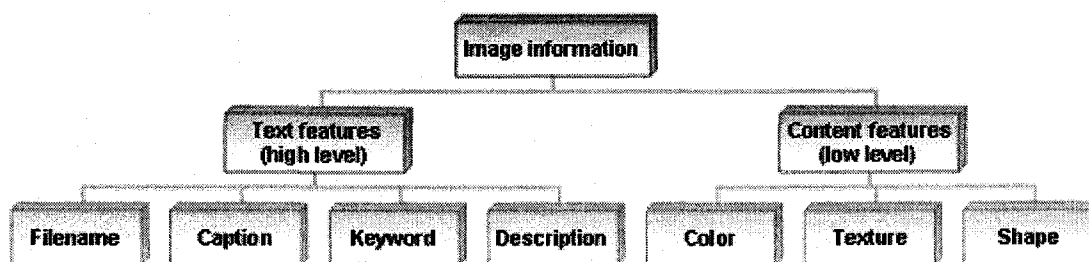


Figure 1.1 Image Retrieval System classification

Text-based image retrieval can be traced back to the late 1970's [45, 5]. A popular image retrieval framework at that time was to first annotate images with text such as filename, caption, keywords and descriptions, and then employ typical database management techniques to perform retrieval from the image database [35]. Text-based image retrieval approaches are simple to implement and perform fast but with two major problems, especially when the size of the image collection becomes very large. First problem is the

prohibitive amount of labor required in manual image annotation, and the second problem, which is probably more critical, results from the contradiction of the rich image contents and subjectivity of the human perception [35].

To overcome these difficulties, Kato [19] introduced the notion of Content-Based Image Retrieval (CBIR) in the early 1990's, and used it to design and implement a project about automatic retrieval of images from a large image database by using image content information. In Content-Based Image Retrieval, images are represented and indexed by visual content, such as color, texture and shape. These features are objective and can be directly derived from the images themselves, without reference to any external knowledge. Thus, allowing low-level numerical features extracted by a computer to be substituted for higher-level text-based manual annotations [35, 3].

1.1 Low Level Features

Unlike high level features which describe the human world properties and understanding of the meaning of whole images or scenes, low level features of an image denote several of its objective perceived characteristics, including color, texture and shape. These features are essential for describing and representing an image, and nowadays, many of them are used by the CBIR systems.

1.1.1 Color

Among all visual features, color is probably the most straightforward one and enables human beings to recognize different images. As a result, color similarity measure has become a very important aspect in the implementation of CBIR systems. Color Histogram [42] is a well known technique used to represent the color of an image. Statistically, it denotes the joint probability of the intensities of the three color channels: red, green and blue. Color Moments [42] is a mathematical way to overcome the quantization effects present in Color Histogram. The first three order moments, i.e., mean, variance and skewness, are extracted as color feature representation. Color Sets [39] is a

selection of bins of colors in the perceptually uniform color space, such as HSV space, which is transformed from the RGB space. Color sets as an approximation to color histograms can facilitate fast retrieval over large image collections because of their binary structures [35].

1.1.2 Texture

Texture is also an important feature in pattern recognition, segmentation and image retrieval. It refers to “the visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity” [35]. Basically, every surface has this property, including clouds, tree bark, bricks, hair, fabric, etc. Haralick *et al.* [14] introduced the co-occurrence matrix representation of texture feature. The two dimensional co-occurrence matrix describes the gray level spatial dependence of the texture. After the psychological studies in human perception of texture, Tamura and Yokoya [45] developed computations from six visual texture properties: contrast, coarseness, directionality, regularity, periodicity, and randomness. The texture of an image can be represented by Wavelet transformation – a technique used in image processing in which a signal is transformed from the time domain to the frequency domain. This transformation based texture analysis approach achieves higher accuracy than typical texture-based approaches [35].

1.1.3 Shape

The shape of objects in an image is another important image feature and helps to represent content of an image. In many situations, people can recognize an object only with its shape. The shape of an object can be obtained by tracking its boundary. Generally, there are two approaches used in shape analysis: boundary-based techniques, e.g., Fourier Descriptors [53, 33], and region-based techniques, e.g., Moment Invariants [15]. More details about shape representation and shape similarity measure are provided in section 2.1.

1.2 CBIR Systems

After the first introduction of Content-Based Image Retrieval, many image retrieval systems have been developed using image content features. In order to make image retrieval more effective, some systems try to combine different visual features, as well as textual annotations [1].

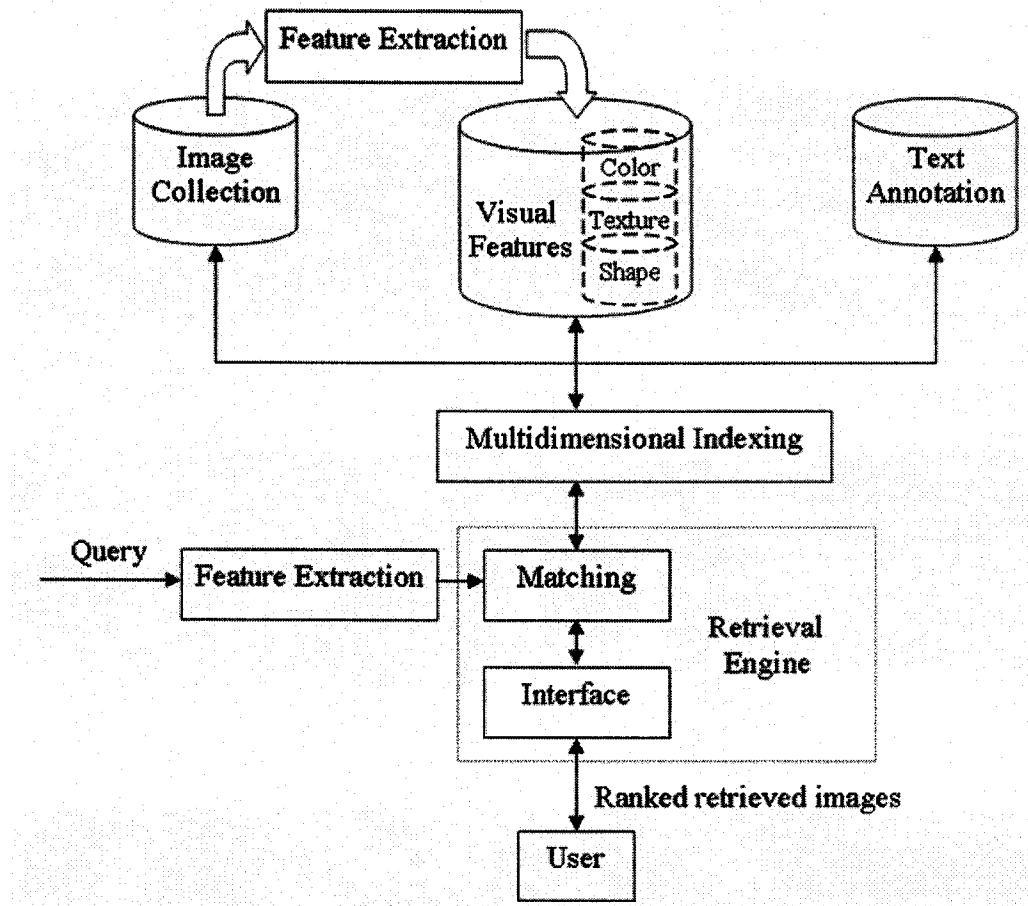


Figure 1.2 A CBIR image retrieval system architecture
(Source: [35], [18])

Figure 1.2 [35, 18] shows one of the possible integrated system architectures with three databases in this system. The image collection database contains raw images for visual display purpose. The visual features database stores color features, texture features and shape features which are extracted from the images using different image representation

techniques. This information is needed to support Content-Based Image Retrieval. Text annotation database contains keywords and free-text descriptions of image contents. In CBIR, text annotations are simply there to assist visual features and indexing techniques are applied to facilitate fast and efficient retrieval.

The query request to the system could be in the form of an image or a rough sketch drawn by the user. The visual features of the query image need to be extracted using the same techniques that are applied to the database images. Features are matched in the retrieval engine and similar images are returned to the user through a graphic interface. The resultant images are normally ranked in terms of similarity distance between the query image and the retrieved database images.

1.3 Problem Statement

Color, texture and shape are the three primary image content features. However, the image retrieval techniques based on color and texture have been developed more thoroughly than shape-based techniques, primarily due to the inherent complexities in representing shape features. On the other hand, shape features are very powerful in object identification and recognition. It is important to note that the human beings can generally recognize objects solely from their shapes. Retrieval of images based on the shapes of objects, generally termed as “Shape-based Retrieval” is an important CBIR technique and has applications in many different fields. Examples of such retrieval can be found but are not limited to recognition and retrieval of trademarks and logos, medical structures, fingerprint, face profiles, hand written signatures, etc.

Primary issues associated with Shape-Based Image Retrieval are: shape representation, similarity measure and retrieval strategy. In shape representation, there are normally two classes of methods: contour-based methods and region-based methods.

The objective of this thesis is to:

- compare two types of moments in the region-based method category: Moment Invariants (MI) and Zernike Moments (ZM). We use these two moment-based representations as feature descriptors to represent shapes and compare their performance in similarity retrieval;
- introduce artificial Neural Network (NN) to Shape-Based Image Retrieval as an intelligent search engine instead of traditional techniques of multidimensional indexing trees. We employ k means clustering method to provide learning samples for the Neural Network to facilitate back propagation training;
- apply the proposed methodology to build an image retrieval system for application to a real-life system.

Remainder of this thesis is organized as follows. Section 2 reviews the fundamental and traditional techniques related to shape-based image retrieval, including contour-based shape representations, region-based representations, and techniques on multidimensional indexing trees. Section 3 describes the methodologies used in this thesis to represent, classify and retrieve shape images. In section 4, we present our experimental results whereas section 5 contains the concluding remarks and directions for future research work.

2 Related Work

2.1 Shape Representations

Shape representations are formalistic identification models of the original shapes so that the important characteristics of the shape are preserved [23]. The goal of shape representation is to derive a numeric shape descriptor (also called a feature vector), which can uniquely characterize the given shape.

The required properties of a shape representation scheme are [32, 23]:

- **Unique.** Each shape must have a unique representation, i.e., two dissimilar shapes should not have a same or very similar representation.
- **Compact.** A good representation should have the ability to provide an excellent compression capability of the shape in the image.
- **Accurate and Reliable.** A good representation must reflect the basic features of a shape correctly and unambiguously. It should be able to handle small changes in an object's shape and be robust to noises.
- **Invariant.** The shape of an object should not change after a series of geometric transformations. Accordingly, the representation of a shape must be invariant under translation, rotation, and scaling.

Two-dimensional shapes can be described in a number of different ways. There are several classifications of shape representation techniques [23], such as classification on the basis of information preservation, classification based on whether the result is numeric or non-numeric, etc. The most popular and widely used classification is the one proposed by Pavlidis [32], which is based on the use of shape boundary points as opposed to the interior features of the shape. The two resulting classes of algorithms are known as contour-based and region-based, respectively.

Contour-based methods emphasize the outer closed curve that surrounds the shape, and region-based methods describe the entire shape region occupied by the shape within the closed boundary on the image plane.

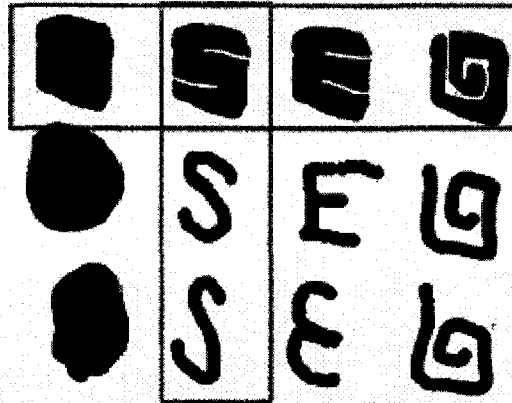


Figure 2.1 Example of contour-based and region-based shape similarity
(After: [2])

Figure 2.1 [2] is an example of the two methods in a 2D case. The shape similarity can be measured based on contour-based representation and region-based representation. Shapes in the first row have similar spatial distribution of pixels and are, therefore, similar according to a region-based criterion. However, they clearly have different outline contours. When contour-based similarity is considered, then shapes shown in each column are similar. If a query is built by the second shape in the first row, the retrieved results will be the shapes from the first row with region-based similarity criteria or the second column with contour-based similarity criteria.

In the following subsections, a brief review of some of the techniques for these two classes is presented.

2.1.1 Contour-Based Representations

2.1.1.1 Chain Codes

A shape can be uniquely represented by its boundary. Let us suppose that all the images we deal with are binary images since we are concerned with only the shape features of objects in the images. Pixels that are part of an object have values of 1, and pixels that are

part of the background have values of 0. "A boundary of an object is represented by a connected path of 1s" [34]. Chain codes describe an object boundary path by a sequence of unit-size line segments with a given orientation. The basic idea was introduced by Freeman [8] in 1961 by establishing a method that permitted the encoding of arbitrary geometric configurations.

In chain codes, the direction vectors between the successive boundary pixels are encoded. The commonly used chain codes employ 4 or 8 elementary directions based on the definition of connectivity [6]. Typically, chain codes contain a starting pixel address followed by a sequence of direction codes. It is obvious that the chain codes representation depends on the starting point of a boundary path and the direction of traversal, clockwise/anticlockwise [34].

If chain codes are used for shape matching and retrieval, there are a number of advantages. First of all, it is compact. Chain codes provide a good compression of boundary description because each chain code element can be encoded with 2 bits (for 4-connected chain codes) or 3 bits (for 8-connected chain codes) only, compared with the 2 bytes required for the storage of the coordinates (x, y) of each boundary pixel [34]. Chain codes can be used to calculate many other shape features, such as perimeter, area, and even Moment Invariants (MI) [6].

Chain codes representation is translation invariant. Scale invariance can be achieved by normalizing the size of the shape. The difference chain codes can be constructed in order to obtain rotation invariance [34].

Lu [25] derived the normalized chain codes, which are invariant to translation, scale and rotation. The normalized chain codes are good for shape representation but quite difficult to use for computing shape similarity. They describe a normalization process to obtain the unique chain code for each shape which is invariant to translation, scale and rotation followed by a method to derive an alternative shape representation to compute shape similarity.

Although chain codes have a compact representation, chain codes could be very long, especially for shapes of large objects. This drawback may cause difficulties in comparison and matching of codes. Also, chain codes are very sensitive to noises.

2.1.1.2 Signatures

In general, a signature is a 1D function representing a 2D boundary. Signatures can be generated in a number of ways but regardless of how a signature is generated, the basic idea is to reduce the boundary representation to a 1D function, which apparently is easier to describe than the original 2D boundary [12].

One of the simplest and the most commonly used signatures is to plot the distance from a shape centroid to its boundary and is called the centroid-to-boundary distance or centroid distance approach.

If (x_i, y_i) are the 2D shape boundary coordinates, where $i=1,2,\dots,N$ and N is the number of boundary pixels, the centroid of the shape (x_c, y_c) is the average of the boundary coordinates [50]:

$$x_c = \frac{1}{N} \sum_{i=1}^N x_i, \quad y_c = \frac{1}{N} \sum_{i=1}^N y_i.$$

The centroid distance signature function is then given by:

$$r(i) = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}.$$

In addition to the centroid distance signature, there are many other types of shape signatures, including complex coordinates, polar coordinates, tangent angle, cumulative angle, curvature, area and chord-length. A detailed study of shape signatures can be found in [31, 30].

Shape signatures are invariant to translation, and are usually normalized to fulfill the scale invariance requirement by standardizing the shape into a specific size. However, in

order to compensate for orientation changes, the shift matching of 1D signature series is needed to find the best match between two shapes. Alternatively, signatures can be quantized into a signature histogram, which is rotation invariant and can be used for shape matching [40].

Shape signatures are normally sensitive to noises, and slight changes in the boundary can cause large errors in the matching. Roh and Kweon [36] devised a contour shape signature descriptor for the recognition of planar curved objects in noisy scenes. A descriptor consisting of five-point invariants was used to index into a hash table. However, it is still impractical or undesirable to directly use shape signatures for shape retrieval. Other descriptors such as Fourier descriptors (FDs) can be further derived from signatures and then utilized for shape matching [50].

2.1.1.3 Fourier Descriptors

Fourier descriptors are obtained by applying Fourier transform on a shape boundary, which is usually represented by a shape signature (section 2.1.1.2). For good shape description, an appropriate shape signature is essential to obtain Fourier descriptors [47]. The method was first introduced by Zahn and Roskies [53]. They represented a shape as a parametric function of tangential direction and cumulative angular bend function of the plane curve, and then transformed the function into the amplitude/phase-angle form. The result of the transformation is a real, continuous, and periodical function. Therefore, it can be described by a Fourier series. The set of Fourier transformed coefficients are called the Fourier Descriptors (FDs) of the shape. They stay constant irrespective of any of the transformations involving: translation or rotation of the shape, change of scale or of origin [53].

Zahn and Roskies's method is restrictive and may have trouble when the modified shape is no longer a closure of the curve. Granlund [11] introduced complex Fourier Descriptors that ensure a closed curve will correspond to any set of descriptors. A shape is described by a set of N vertices $\{r(i): i=1, \dots, N\}$ corresponding to N points of the

boundary. The Fourier Descriptors $\{z(k): k = -N/2 + 1, \dots, N/2\}$ are the coefficients of the Fourier transform of r :

$$r_i = \sum_{k=-N/2+1}^{N/2} z_k \exp(2\pi j \frac{ki}{N}).$$

The inverse relationship exists between $z(k)$ and $r(i)$:

$$z_k = \frac{1}{N} \sum_{i=1}^N r_i \exp(-2\pi j \frac{ik}{N}).$$

and z_k are usually denoted as FD_n , $n=1, \dots, N$.

Zhang and Lu [47] derived Fourier Descriptors from the centroid distance signature, and their experiment results show that the shape signature using centroid distance exhibits better performance than other shape signatures in shape-based image retrieval. The Fourier Descriptors obtained in their approach is translation invariant because of the translation invariance of centroid distance. To achieve rotation invariance, they ignore the phase information of the Fourier Descriptors by using the magnitudes $|FD_n|$ only. Scale invariance is achieved by dividing the magnitudes by $|FD_0|$. Since centroid distance is a real value function, only half of the Fourier Descriptors are needed for indexing the shapes. Thus, the shape can be represented by feature vector [47]:

$$\mathbf{x} = \left[\frac{|FD_1|}{|FD_0|}, \frac{|FD_2|}{|FD_0|}, \dots, \frac{|FD_{N/2}|}{|FD_0|} \right].$$

Two shapes can be compared by comparing the subsets of the Fourier Descriptors, beginning with the lower order coefficients and then using higher order coefficients. To make the description simpler, one can get rid of the Fourier descriptors of the higher frequencies. However, the high-frequency components account for fine detail, and low-frequency components determine global shape features. Thus, the fewer the FD terms used, more are the details that are lost on the shape boundary [12].

The excellent properties of Fourier Descriptors are their robustness, ability to capture some perceptual characteristics of the shape and ease to derive. Although there is a trade-off between using low order coefficients and high order coefficients, usually only a small

number of low order coefficients are enough to capture the overall shape features. Therefore, the representation is also compact. In addition, since noise can only appear in very high frequency [12], with Fourier Descriptors, noise can be easily truncated out.

The disadvantage of Fourier Descriptors is that local features cannot be located, because in “Fourier Transform only the magnitudes of the frequencies, not the location, are known” [50]. To overcome the drawbacks of existing shape representation techniques, a Generic Fourier Descriptor (GFD) has been proposed [52]. GFD is derived by applying 2D Fourier transformation on a polar raster sampled shape image, and it is said to be application independent and robust [52].

An Enhanced Generic Fourier Descriptor (EGFD) is obtained based on GFD [51]. It is acquired by deriving GFD from the shape that has been rotation and scale normalized. The EGFD outperforms GFD significantly. It solves GFD’s low retrieval performance on severely skewed and stretched shapes. It also improves GFD’s robustness to general shape distortions.

2.1.2 Region-Based Representations

2.1.2.1 Grid Descriptors

In grid shape representation, a shape is projected onto a grid of fixed size square cells, say 16x16 pixels for example [24, 4, 43]. The grid is just big enough to completely cover the shape such that some grid cells are fully or partially covered by the shape and some are not. The grid cells are assigned the value of ‘1’ if they are covered by the shape region (or covered beyond a threshold, e.g., 15% of pixels) and ‘0’ if they are outside the shape. A shape number consisting of a binary sequence is created by scanning the grid in left-right and top-bottom order, and this binary sequence is used as a shape descriptor to index the shape [24].

For two shapes to be comparable using Grid Descriptors (GDs), several normalization processes have to be done to achieve scale, rotation and translation invariance [24, 49].

The process begins with finding out the major axis, i.e., the line joining the two furthest points on the boundary. Rotation normalization is achieved by turning the shape so that the major axis is parallel with the x -axis. To avoid multi-normalization result for mirrored shapes and flipped shapes, the centroid of the rotated shape may be restricted to the lower-left part, or a mirror and a flip operation on the shape number are applied in the matching stage. Scale normalization can be done by resizing the shape so that the length of the major axis is equal to the grid width, and by shifting the shape to the upper-left of the grid, leaving the representation to be translation invariant [24, 49]. The distance between two set of grid descriptors is simply the number of elements having different values.

Grid representation is a straightforward shape representation which may be suitable for shape coding as is adopted in MPEG-4 [49]. However, it is questionable for retrieval purposes, because a slight shape distortion such as affine transform can cause very big difference in the similarity measure. Furthermore, since “the normalizations are mainly based on major axis (which is unreliable in essence) and eccentricity (which is only reliable for convex shapes or compact shapes), shapes otherwise similar may be treated as different due to this normalization” [49]. Furthermore, it is clear that the smaller the cell size, the more accurate the shape representation but more the storage and computation requirements.

2.1.2.2 Moment Invariants

Moments are extensively used in the area of pattern recognition and for shape representation and similarity measure. Moment Invariants (MI) are derived from moments of shapes and are unchanged under 2D geometric transformations such as translation, scale and rotation [15, 1].

The theory of moments provides an interesting and useful alternative to a series of expansions for representing a real bounded function [9]. Suppose $f(x,y) \geq 0$ is such a 2D function on a region R , the geometric moment of order $p + q$ is defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy .$$

for $p, q = 0, 1, 2, \dots$. A uniqueness theorem [27] states that if $f(x, y)$ is piecewise continuous and has nonzero values only in a finite part of the xy -plane, moments of all orders exist, and the moment sequence m_{pq} is uniquely determined by $f(x, y)$. Conversely, m_{pq} uniquely determines $f(x, y)$.

Many shape features can be conveniently represented in terms of geometric moments such as the total mass (area), the centroid, the angle of the principal axis, the bounding box, the best-fit ellipse and the eccentricity [6].

The central moments are defined as

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy ,$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} , \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

is the centre of mass.

The normalized moments are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$$

with

$$\gamma = \frac{p+q}{2} + 1 .$$

Hu [15] first introduced seven Moment Invariants which are given as follow:

$$\Phi_1 = \eta_{20} + \eta_{02}$$

$$\Phi_2 = (\eta_{20} + \eta_{02})^2 + 4(\eta_{11})^2$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$

$$\begin{aligned}
& + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\Phi_6 = & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
& + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\Phi_7 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
& + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{Equation 2.1}$$

$\Phi_1 - \Phi_6$ are invariant with respect to rotation and reflection, and Φ_7 remains unchanged under rotation but changes sign under reflection (i.e., invariant in absolute value). If we replace μ_{pq} by η_{pq} in the above equations or normalize them directly with m_{00} (μ_{00}), we have:

$$\begin{aligned}
\Psi_1 &= \Phi_1 / m_{00}^2 \\
\Psi_2 &= \Phi_2 / m_{00}^4 \\
\Psi_3 &= \Phi_3 / m_{00}^5 \\
\Psi_4 &= \Phi_4 / m_{00}^5 \\
\Psi_5 &= \Phi_5 / m_{00}^{10} \\
\Psi_6 &= \Phi_6 / m_{00}^7 \\
\Psi_7 &= |\Phi_7| / m_{00}^{10}
\end{aligned} \tag{Equation 2.2}$$

$\Psi_1 \sim \Psi_7$ are dependent on unique shape regardless of their location, size and orientation.

A Feature vector consists of the seven components: $\mathbf{x} = [\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Psi_6, \Psi_7]$ and is used to index shapes in the image database.

The values of the computed geometric moments are usually small, values of higher order moment invariants, in some cases are close to zero. Therefore, all of the Moment Invariants can be further normalized into [0,1] by the limit values of each dimension [49].

The advantage of using Moment Invariants is that it is a very compact shape representation with low computational overhead. However, it is difficult to obtain higher order moment invariants.

2.1.2.3 Zernike Moments

Based on the idea of replacing the conventional kernel of moments with a general transform, orthogonal moments have been proposed to recover the image from moments [44]. Zernike Moments (ZMs), which allow independent moment invariants to be constructed to an arbitrarily high order, are orthogonal moments. The complex Zernike moments are derived from the Zernike polynomials:

$$V_{nm}(x, y) = V_{nm}(\rho \cos \theta, \rho \sin \theta) = R_{nm}(\rho) \exp(jm\theta), \text{ and}$$

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s}.$$

where n and m are subject to conditions:

$$n - |m| = \text{even}, \text{ and } |m| \leq n.$$

Zernike polynomials are a complete set of complex-valued function orthogonal over the unit circle, i.e., $x^2 + y^2 = 1$. Then the complex Zernike moments of order n with repetition m are defined as:

$$A_{nm} = \frac{n+1}{\pi} \iint_{x,y} f(x, y) V_{nm}^*(x, y) dx dy.$$

The theory of Zernike Moments is similar to that of the Fourier transform, to expand a signal into a series of orthogonal basis. The precision of shape representation depends on the number of moments truncated from the expansion. Since Zernike basis functions take the unit disk as their domain, this disk must be specified before moments can be calculated [49]. The unit disk is then centered on the shape centroid, thus, making the obtained moments scale and translation invariant. Rotation invariance is achieved by using only the magnitudes of the moments [49]. The magnitudes are then normalized into $[0,1]$ by dividing them by the mass of the shape.

Zernike Moments do not need to know boundary information, thus, making them suitable for more complex shape representation. Like Fourier Descriptors, Zernike Moments can also be constructed to some arbitrary order, and this overcomes the drawback of the Moment Invariants in which higher order moments are difficult to construct [48]. However, Zernike moments lose the important perceptual meaning as reflected in Fourier Descriptors and Moment Invariants [48]. In other words, we don't know what shape feature each Zernike Moment represents.

2.2 Multidimensional Indexing

Normally, image descriptors or shape descriptors are represented by multidimensional vectors, which are often used to measure the similarity of two images by calculating a descriptor distance in the feature space. When the number of images in the database is small, a sequential linear search can provide a reasonable performance. However, with large scale image databases, indexing support for similarity-based queries becomes necessary. In traditional databases, data is indexed by key attributes, which are selected to support the most common types of searches. Similarly, the indexing of an image database should support an efficient search based on the image contents or extracted features. In relational database management systems, the most popular class of indexing techniques is the B-tree family, most commonly the B⁺-tree [45, 5]. B-trees, in general, allow extremely efficient searches when the key is a scalar. However, they are not suitable to index the content of images represented by high-dimensional features. Popular multidimensional indexing techniques include KDB Tree, R/R* Tree, SS Tree and SR Tree.

2.2.1 KDB Tree

The KDB tree is an index structure for multidimensional point data. It is a height-balanced tree similar to the B⁺-tree and its structure is constructed by dividing the search space into subregions with coordinate planes recursively [37]. Nodes and leaves correspond to subregions and a disk block is allocated for each of them. The distinctive

characteristic of the KDB tree is the disjointness among subregions on the same tree level. This makes the search path of a point query to be single branch from the root to a leaf. Therefore, the search time of a point query is logarithmic to the size of a data set [10]. However, the KDB tree cannot ensure the minimum storage utilization, and this reduces the performance of the KDB tree on range and nearest neighbor queries.

2.2.2 R/R* Tree

R tree was initially proposed to index data objects of non-zero size in high-dimensional spaces [10]. “This index structure can be simply adapted to indexing multidimensional points with some small modifications to its insertion and search algorithms” [10]. An R tree is built based on the clustering of all the points in a data space. Each nonterminal node in the tree structure corresponds to a cluster in the space with its minimum bounding rectangle representing the extent of the cluster [10, 20, 3]. When a node is overfilled, the cluster represented by the node is partitioned and the node is split into two new nodes.

An R tree is completely dynamic since its insertions and deletions can be intermixed with queries and no periodic global reorganization is required. However, since the structure allows the bounding rectangles of different entries to overlap on each other, the search algorithm must traverse more than one path to search for the desired data. Hence, generally it is not possible to achieve a worst-case performance [10]. With R tree, the optimization criterion used by the data insertion and node split algorithms is to minimize the area of enclosing rectangles in the resulting nodes. This criterion has a tendency to generate strip-like bounding rectangles in leaf nodes, thereby resulting in a large overlap among covering rectangles in non-leaf nodes [10].

R* tree is one of the most successful variants of the R tree. It shares the same tree structure with R tree, but improves the performance of R tree by modifying the insertion and node split algorithms and by introducing the forced reinsertion mechanism [20, 10].

2.2.3 SS Tree

The SS tree has a similar configuration as that of R and R* trees, but it improves the performance and enhances the nearest neighbor queries. The major difference between the tree structures is that the SS tree makes use of minimum bounding spheres instead of minimum bounding rectangles in its non-leaf nodes [10, 20]. Because a bounding sphere is determined by only the center and the radius, it requires much less (nearly half) storage compared to bounding rectangles [20]. The algorithms for data search, insertion, deletion, and node split from both the R and R* trees can be applied to the SS tree with some minor modifications.

Using bounding spheres has certain advantages to partition data space. However, “experimental evaluations have discovered that minimum bounding spheres in an SS tree tend to have larger volumes than minimum bounding rectangles in the equivalent R tree or R* tree” [10]. In addition, larger sphere volumes result larger overlaps between different nodes, which deteriorates the data retrieval performance.

2.2.4 SR Tree

To Combine the benefits and overcome the disadvantages of R/R* trees and SS tree, SR tree was proposed by Katayama and Satoh [20]. The SR tree uses the intersection area of the minimum bounding spheres and the minimum bounding rectangles in its internal nodes and leaves. The distinguishing feature of the SR tree is that it specifies a region by the intersection of the bounding sphere and the bounding rectangle, and this permits to divide points into regions with both small volumes and short diameters.

2.2.5 Comparison

A comparison study of multidimensional indexing trees was made by Katayama and Satoh [20]. In their evaluation, both uniform data set and real data set were used for performance test. Test results revealed that the four tree structures can be divided into two groups: KDB tree and R/R* tree, SS tree and SR tree. Furthermore, SS tree and SR

tree group outperforms the KDB tree and R/R* tree groups in terms of both tree building cost and the retrieval performance. SR and SS trees require less CPU time than R/R* tree, because the centroid-based insertion algorithm of the SS tree requires significantly less CPU time than the algorithm of the R/R* tree [20]. SR tree contains not only bounding spheres but also bounding rectangles, therefore it requires higher creation cost and more disk accesses than the SS tree. However, SR tree enhances the query performance remarkably. It allows to divide data point into regions with smaller volume and shorter diameter than any other index structures. SR tree is especially effective for high dimensional and non-uniform data sets, which can be practical in actual image similarity indexing.

3 Thesis Approach

3.1 Shape Feature Extraction

In this thesis, we study and compare two shape representations: Moment Invariants (MI) and Zernike Moments (ZM). Moment Invariants and Zernike Moments are both region-based shape representations but need boundary sequence of the shape object for computation.

3.1.1 Boundary Sequence

We assume that all the images we deal with are binary images since we are concerned only with the shape features of the objects in the images. We also assume that the pixels in the object have value of '1', and pixels on the background have value of '0'. Therefore, a boundary sequence is a list of connected pixels on the edge of the object, separating the shape region (1) and the background (0).

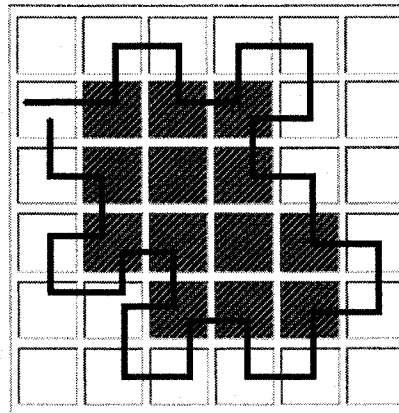


Figure 3.1 Turtle procedure in binary object boundary following
(Source: [34])

The boundary of a binary object can be easily followed by employing a Turtle procedure [34]. For a clockwise boundary sequence, the Turtle begins from a boundary pixel (the start point), and then if the current pixel value is 1, it turns left and advances one pixel; if

the current pixel value is 0, it turns right and advances on pixel. The algorithm terminates when the Turtle returns to the starting boundary point as shown in Figure 3.1 [34].

However, the Turtle algorithm has its limitation. It may not always be able to identify a part of the shape which is only one pixel diagonally connected to the other part of the shape. The following procedure elegantly avoids this problem, and is employed in our proposed approach:

Algorithm: Finding the Boundary Sequence

Input: a digital binary shape image

Output: a boundary sequence $C[i]$

Method:

- (1) Find the start point $S(s_x, s_y)$ in the image whose value is 1;
 $i = 0$;
 Let the current pixel be $C[i](c_x, c_y)$, i.e., set $C = S$;
- (2) Let the 4th neighbor (neighbor pixel on the left) of C be $B(b_x, b_y)$;
- (3) $i++$;
- (4) Let $D[1], D[2], \dots, D[8]$ be the 8 neighbors (anticlockwise, starting with B) of C ;
 Find the smallest $k, k = 1, 2, \dots, 8$, that $D[k]$ has value of 1;
- (5) Let $C = D[k]$ and $B = D[k - 1]$;
- (6) If C is the start point S then terminate;
 Otherwise goto step (3).

Then, $C[i]$ s are the boundary pixels, starting with S , in the anticlockwise order.

3.1.2 Moment Invariants from Extreme Boundary Pixels

Moment Invariants are derived from geometric moments. Dai *et al.* [6] suggest that geometric moments can be directly calculated from the chain codes and the method is called integral method. However, we need to know only the extreme boundary pixels of an object's shape to compute the geometric moments and Moment Invariants.

We assume that each boundary pixel has a contribution type: negative, zero, or positive (Figure 3.2 [6]). The left extreme pixels of every scan line are of the negative contribution type, the right extreme pixels are of the positive contribution type, and others are of type zero. For pixels which are both left extreme and right extreme, we consider them as the hybrid case which contains all characteristics of negative case and positive case.

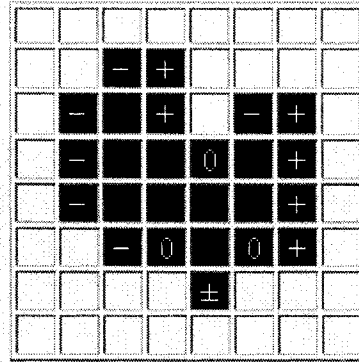


Figure 3.2 Contribution type of boundary pixels

(Source: [6])

From the definition of geometric moments given in section 2.1.2.2, $f(x,y) \geq 0$ is a 2D real bounded function, the $(p + q)$ th order moment of the shape enclosed by $f(x,y)$ is:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \quad (\text{Equation 3.1})$$

for $p, q = 0, 1, 2, \dots$

In computer vision applications, for the reasons of simplicity and speed, in many cases binary shape representation are used. The binary shape of an object may be directly represented by the region it occupies. The binary function

$$u(x,y) = \begin{cases} 1, & \text{if } (x,y) \in R \\ 0, & \text{otherwise} \end{cases}$$

is a simple representation of the region R . If we replace $f(x,y)$ by $u(x,y)$ in Equation 3.1, it will give us the moments of the region R representing a binary shape. In this case, Equation 3.1 will become:

$$m_{pq} = \iint_R x^p y^q dx dy .$$

To compute geometric moments of a digital image, the double integrals could be approximated by the double summations in the moment definitions

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

For a binary shape $u(x, y)$:

$$m_{pq} = \sum_x \sum_y x^p y^q$$

where $(x, y) \in R$.

A large number of multiplications and additions are involved in above equations. In real-time applications, the speed of computation is of vital importance. For a binary shape, an algorithm called the “Delta method” was proposed by Zakaria *et al.* in 1987 [54]. This method used the contribution of each row rather than individual pixels for the improvement of computation speed. Only the coordinates of the first pixel and the length of the chained pixels of the shape R in each row are needed in the “Delta method”. The Integral method proposed by Dai *et al.* [6] uses the contribution of each extreme point of the shape in each row instead of the contribution of the total row, and this computation is directly based on the integral operation. The geometric moments and Moments Invariants in this method are derived from the chain codes.

Based on the same idea of computing from the integral and extreme pixels, we propose the following methods to calculate the moments:

Assuming a given shape as a pixel matrix shown in Figure 3.3 [6], where $x_{L,i}$, $x_{R,i}$ are the abscissas of the first pixel (extreme left) and the last pixel (extreme right) of the shape in row i , δ_i is the number of connected pixels in row i , i.e., $\delta_i = x_{L,i} - x_{R,i} + 1$. y_i is the ordinate of row i , so the geometric moments m_{pq} could be written as:

$$m_{pq} = \sum_i m_{pq,i} .$$

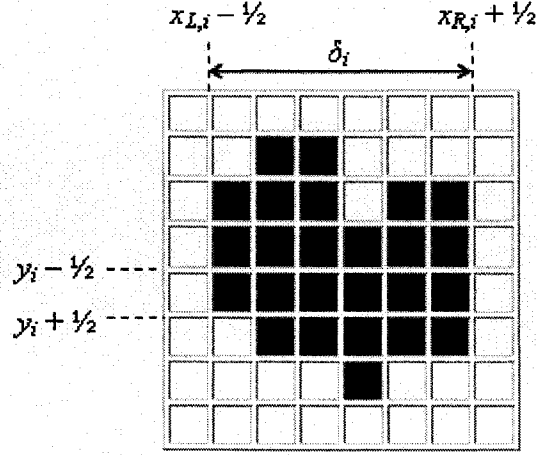


Figure 3.3 A Shape consisting of small uniform squares
(Source: [6])

The contribution of row i in terms of $x_{L,i}$, $x_{R,i}$ and y_i for a horizontal convex shape is considered as a region consisting of small uniform squares of size 1×1 . Therefore, we adjust the coordinates by $\pm 1/2$, and the contribution of row i is derived using the Newton-Leibniz Formula:

$$\begin{aligned}
 m_{pq,i} &= \int_{y_i - 1/2}^{y_i + 1/2} \int_{x_{L,i} - 1/2}^{x_{R,i} + 1/2} x^p y^q dx dy \\
 &= \int_{y_i - 1/2}^{y_i + 1/2} \int_0^{x_{R,i} + 1/2} x^p y^q dx dy - \int_{y_i - 1/2}^{y_i + 1/2} \int_0^{x_{L,i} - 1/2} x^p y^q dx dy \\
 &= \left[\frac{x^{p+1}}{p+1} \right]_0^{x_{R,i} + 1/2} \cdot \left[\frac{y^{q+1}}{q+1} \right]_{y_i - 1/2}^{y_i + 1/2} - \left[\frac{x^{p+1}}{p+1} \right]_0^{x_{L,i} - 1/2} \cdot \left[\frac{y^{q+1}}{q+1} \right]_{y_i - 1/2}^{y_i + 1/2} \\
 &= \left(\left[\frac{x^{p+1}}{p+1} \right]_0^{x_{R,i} + 1/2} - \left[\frac{x^{p+1}}{p+1} \right]_0^{x_{L,i} - 1/2} \right) \cdot \left[\frac{y^{q+1}}{q+1} \right]_{y_i - 1/2}^{y_i + 1/2}
 \end{aligned}$$

For $p + q \leq 3$,

$$\begin{aligned}
 m_{00,i} &= \left([x]_0^{x_{R,i} + 1/2} - [x]_0^{x_{L,i} - 1/2} \right) \cdot [y]_{y_i - 1/2}^{y_i + 1/2} \\
 &= [(x_{R,i} + 1/2) - 0] - [(x_{L,i} - 1/2) - 0] \cdot [(y_i + 1/2) - (y_i - 1/2)] \\
 &= (x_{R,i} + 1/2) - (x_{L,i} - 1/2)
 \end{aligned}$$

$$m_{10,i} = \left(\left[\frac{x^2}{2} \right]_0^{x_{R,i} + \frac{1}{2}} - \left[\frac{x^2}{2} \right]_0^{x_{L,i} - \frac{1}{2}} \right) \cdot [y]_{y_i - \frac{1}{2}}^{y_i + \frac{1}{2}}$$

$$= \frac{(x_{R,i} + \frac{1}{2})^2}{2} - \frac{(x_{L,i} - \frac{1}{2})^2}{2}$$

$$m_{20,i} = \frac{(x_{R,i} + \frac{1}{2})^3}{3} - \frac{(x_{L,i} - \frac{1}{2})^3}{3}$$

$$m_{30,i} = \frac{(x_{R,i} + \frac{1}{2})^4}{4} - \frac{(x_{L,i} - \frac{1}{2})^4}{4}$$

$$m_{01,i} = \left([x]_0^{x_{R,i} + \frac{1}{2}} - [x]_0^{x_{L,i} - \frac{1}{2}} \right) \cdot \left[\frac{y^2}{2} \right]_{y_i - \frac{1}{2}}^{y_i + \frac{1}{2}}$$

$$= m_{00,i} \cdot \frac{1}{2} \left((y_i + \frac{1}{2})^2 - (y_i - \frac{1}{2})^2 \right)$$

$$= m_{00,i} \cdot y_i$$

$$m_{02,i} = m_{00,i} \cdot \frac{1}{3} \left((y_i + \frac{1}{2})^3 - (y_i - \frac{1}{2})^3 \right)$$

$$= m_{00,i} \cdot \frac{1}{3} \left(y_i^2 + \frac{1}{4} \right)$$

$$= m_{00,i} \cdot y_i^2 + \frac{1}{12} m_{00,i}$$

$$m_{03,i} = m_{00,i} \cdot \frac{1}{4} \left((y_i + \frac{1}{2})^4 - (y_i - \frac{1}{2})^4 \right)$$

$$= m_{00,i} \cdot \frac{1}{4} \left(4y_i^3 + y_i \right)$$

$$= m_{00,i} \cdot y_i^3 + \frac{1}{4} m_{01,i}$$

$$m_{11,i} = m_{10,i} \cdot y_i$$

$$m_{12,i} = m_{10,i} \cdot y_i^2 + \frac{1}{12} m_{10,i}$$

$$m_{21,i} = m_{20,i} \cdot y_i$$

The $(p + q)$ th moment of the whole binary shape is the sum of the contributions of every row:

$$m_{pq} = \sum_i m_{pq,i}.$$

For a binary shape, the central moments can be given as:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q.$$

After m_{pq} have been calculated by the extreme pixels, the central moments μ_{pq} could be obtained by using the following formulas:

$$\mu_{00} = \sum_x \sum_y 1 = m_{00}$$

$$\mu_{01} = \sum_x \sum_y (y - \bar{y}) = m_{01} - \frac{m_{01}}{m_{00}} \cdot m_{00} = 0$$

$$\mu_{02} = \sum_x \sum_y (y - \bar{y})^2 = m_{02} - \frac{m_{01}^2}{m_{00}} = m_{02} - \bar{y}m_{01}$$

$$\mu_{03} = \sum_x \sum_y (y - \bar{y})^3 = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2m_{01}$$

$$\mu_{10} = \sum_x \sum_y (x - \bar{x}) = m_{10} - \frac{m_{10}}{m_{00}} \cdot m_{00} = 0$$

$$\mu_{11} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) = m_{11} - \frac{m_{10} \cdot m_{01}}{m_{00}} = m_{11} - \bar{y}m_{10}$$

$$\mu_{12} = \sum_x \sum_y (x - \bar{x})(y - \bar{y})^2 = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2m_{10}$$

$$\mu_{20} = \sum_x \sum_y (x - \bar{x})^2 = m_{20} - \frac{m_{10}^2}{m_{00}} = m_{20} - \bar{x}m_{10}$$

$$\mu_{21} = \sum_x \sum_y (x - \bar{x})^2 (y - \bar{y}) = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01}$$

$$\mu_{30} = \sum_x \sum_y (x - \bar{x})^3 = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2m_{10}$$

Then, by using Equations 2.1 and 2.2, we can easily get the seven Moment Invariants, as mentioned earlier.

Figure 3.4 shows some binary shape images: (a) and (b) are two kinds of butterflies; (c) is a pier of pincers. If we flip the pincers, rotate it by 12° and 45° in clockwise, and resize it to 50% of its original width and height, then we obtain the transformed shapes (d), (e), (f), and (g), respectively.

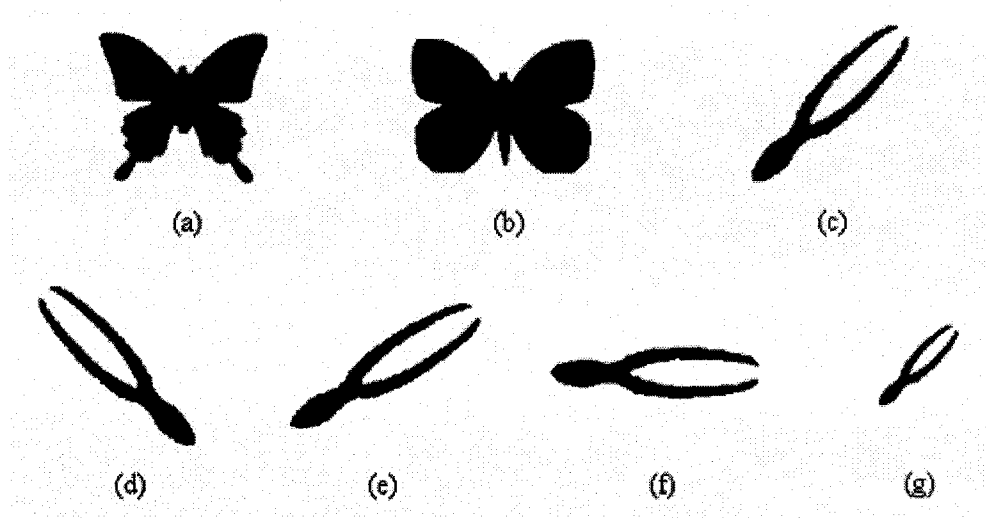


Figure 3.4 Example of binary shapes and shapes under geometric transformation

Table 3.1 shows the corresponding Moment Invariants of the shapes in Figure 3.4. From this table, we can see that: (a) and (b), two butterflies, are close in the Moment Invariants feature space, but are both far away from (c), the pincers. The transformed shapes (d) (e) (f) and (g) are very close to the original shape (c).

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Ψ_1	2.3579E-01	2.1699E-01	7.4873E-01	7.5060E-01	7.6499E-01	7.1600E-01	6.7028E-01
Ψ_2	2.1204E-03	7.8817E-03	4.4297E-01	4.4536E-01	4.6700E-01	3.9698E-01	3.5543E-01
Ψ_3	1.5002E-03	1.8027E-04	8.8653E-04	8.6760E-04	5.3087E-04	6.4979E-04	2.7781E-06
Ψ_4	1.6026E-05	2.2426E-05	1.0625E-02	1.0589E-02	8.9542E-03	9.6474E-03	2.8673E-03
Ψ_5	-2.4427E-09	-1.4237E-09	3.2580E-05	3.2092E-05	1.8712E-05	2.4048E-05	-6.8134E-08
Ψ_6	-7.3511E-07	-1.9900E-06	7.0714E-03	7.0667E-03	6.1122E-03	6.0777E-03	1.7053E-03
Ψ_7	4.5564E-10	7.8733E-11	1.3684E-06	5.2998E-07	5.5653E-06	2.2624E-06	2.4583E-07

Table 3.1 Example of Moment Invariants as shape features

3.1.3 Complex Zernike Moments and the Magnitudes

Essentially, Zernike Moments do not need to know the boundary information. Zernike Moments are defined over a unit disk which is a circle of unit radius on a Cartesian plane. However, the unit disk is centered on the shape centroid, and the centroid of a shape (x_c, y_c) is the average of its boundary coordinates (x_i, y_i) , $i = 1, 2, \dots, N$

$$x_c = \frac{1}{N} \sum_{i=1}^N x_i, \quad y_c = \frac{1}{N} \sum_{i=1}^N y_i.$$

Before computing the complex Zernike Moments, we need to transform the shape from its digital image coordinate system into the Zernike unit coordinate system. As shown in Figure 3.5, the origin of the digital image coordinate system is the top-left corner of the image whereas positive u points to horizontal right, and positive v points to vertical down. The coordinate system transformation consists of three steps: translation, rotation and scaling.

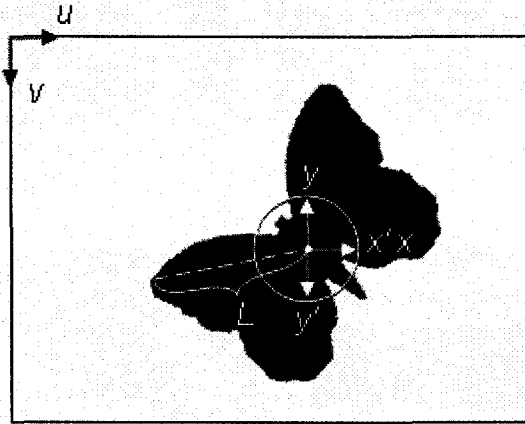


Figure 3.5 Coordinate system transformation

We first translate the origin from the left-top corner to the shape centroid (x_c, y_c) . x_c and y_c can be considered as positive quantities in uv system. The relationship between uv system and $x'y'$ system can be expressed as:

$$\begin{cases} x' = u - x_c \\ y' = v - y_c \end{cases}$$

Then, we flip the y' axis so that it fits the typical xy system in geometry:

$$\begin{cases} x = x' \\ y = -y' \end{cases}$$

Finally, we normalize the shape into a unit disk of radius R . Suppose the distance from the shape centroid (circle center) to its far most boundary pixel is L , the scaling factor for this normalization would be R/L . From the above transformations, we obtain:

$$\begin{cases} u = \frac{L}{R}x + x_c \\ v = y_c - \frac{L}{R}y \end{cases}$$

The complex Zernike moments are derived from the Zernike polynomials as:

$$V_{nm}(x, y) = V_{nm}(\rho \cos \theta, \rho \sin \theta) = R_{nm} e^{jm\theta}$$

and

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s}$$

where ρ is the length of the vector from the shape centroid to pixel (x, y) , θ is the angle between vector ρ and the x axis in the anticlockwise direction and $j = \sqrt{-1}$.

Zernike moments allow independent moment invariants to be constructed to an arbitrarily high order n with repetition m [49] where n and m are integers and subject to the condition:

$$n - |m| = \text{even}, |m| \leq n.$$

Possible values of n and m are:

$$n = 0, |m| = 0$$

$$n = 1, |m| = 1$$

$$n = 2, |m| = 0, 2$$

$$n = 3, |m| = 1, 3$$

$$n = 4, |m| = 0, 2, 4$$

$$n = 5, |m| = 1, 3, 5$$

...

The first six orthogonal radial polynomials are:

$$R_{00}(\rho) = (-1)^0 \frac{0!}{0! \cdot 0! \cdot 0!} \rho^0 = 1$$

$$R_{11}(\rho) = (-1)^0 \frac{1!}{0! \cdot 1! \cdot 0!} \rho^{1-0} = \rho$$

$$R_{20}(\rho) = (-1)^0 \frac{(2-0)!}{0! \cdot (1-0)! \cdot (1-0)!} \rho^{2-0} + (-1)^1 \frac{(2-1)!}{1! \cdot (1-1)! \cdot (1-1)!} \rho^{2-2} = 2\rho^2 - 1$$

$$R_{22}(\rho) = (-1)^0 \frac{(2-0)!}{0! \cdot (2-0)! \cdot (0-0)!} \rho^{2-0} = \rho^2$$

$$R_{31}(\rho) = (-1)^0 \frac{(3-0)!}{0! \cdot (2-0)! \cdot (1-0)!} \rho^{3-0} + (-1)^1 \frac{(3-1)!}{1! \cdot (2-1)! \cdot (1-1)!} \rho^{3-2} = 3\rho^3 - 2\rho$$

$$R_{33}(\rho) = (-1)^0 \frac{(3-0)!}{0! \cdot (3-0)! \cdot (0-0)!} \rho^{3-0} = \rho^3$$

Figure 3.6 shows these six radial responses. From the figure it can be seen that the polynomials are separated between 0 and 1. They become more grouped as they approach 1, and then scatter again.

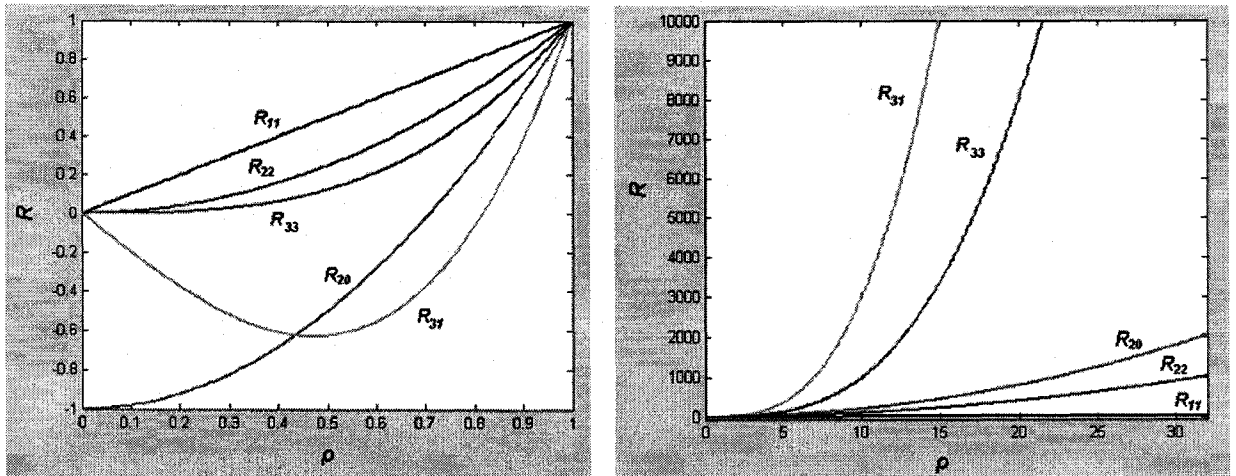


Figure 3.6 Six orthogonal radial polynomials plotted for increasing ρ

Since Zernike basis functions take the unit disk as their domain, this disk must be specified before moments can be calculated. In our implementation, all the shapes are normalized into a unit disk of fixed radius of 32 ($=2^5$) pixels, i.e., $R = 32$. We can also

choose 8 ($=2^3$), 16 ($=2^4$), or 64 ($=2^6$) as the value of the disk radius. However, the larger the disk is, the more computation is required and the more details the feature reflects. The unit disk is then centered on the shape centroid by the transformations we discussed before.

For digital images, we substitute the integrals with summations and the complex Zernike moments of order n with repetition m are defined as:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x, y) V_{nm}^*(x, y) \text{ where } x^2 + y^2 \leq R^2.$$

In the equation above, $f(x, y)$ can be replaced by $u(x, y)$ for binary images, therefore,

$$\begin{aligned} A_{nm} &= \frac{n+1}{\pi} \sum_x \sum_y u(x, y) V_{nm}^*(x, y) \\ &= \frac{n+1}{\pi} \sum_x \sum_y u(x, y) R_{nm} (e^{jm\theta})^* \\ &= \frac{n+1}{\pi} \sum_x \sum_y u(x, y) R_{nm} [e^0 (\cos(m\theta) + j \sin(m\theta))]^* \\ &= \frac{n+1}{\pi} \sum_x \sum_y u(x, y) (R_{nm} \cos(m\theta) + j R_{nm} \sin(m\theta))^* \end{aligned}$$

Furthermore,

$$\begin{aligned} V_{nm}^*(x, y) &= R_{nm} \cos(m\theta) - j R_{nm} \sin(m\theta) \\ &= R_{nm} \cos(-m\theta) + j R_{nm} \sin(-m\theta) \\ \therefore V_{nm}^*(x, y) &= V_{n, -m}(x, y) \end{aligned}$$

Zernike transformation makes the obtained moments scale and translation invariant. Rotation invariance is achieved by only using magnitudes (absolute values) of the moments.

Table 3.2 shows the Zernike Moments features of shapes in Figure 3.4.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)
A_{00}	3.9088E+02	5.0516E+02	9.7721E+01	9.6767E+01	9.8676E+01	1.0791E+02	1.1268E+02
A_{11}	1.0576E+04	1.5920E+04	2.4752E+03	2.3695E+03	2.6036E+03	3.0097E+03	2.7164E+03
A_{20}	6.8396E+05	1.0420E+06	1.5806E+05	1.4688E+05	1.5142E+05	1.7404E+05	1.7090E+05
A_{22}	2.0218E+05	4.1820E+05	5.2699E+04	4.7746E+04	6.7507E+04	8.3522E+04	5.6793E+04
A_{31}	2.5204E+07	4.3413E+07	6.2397E+06	5.6012E+06	6.2481E+06	7.3978E+06	6.6449E+06
A_{33}	3.0881E+06	1.1208E+07	9.0766E+05	7.3056E+05	1.9273E+06	2.4398E+06	1.0317E+06

Table 3.2 Example of Zernike Moments as shape features

3.2 Indexing Mechanism

3.2.1 Retrieval Strategy

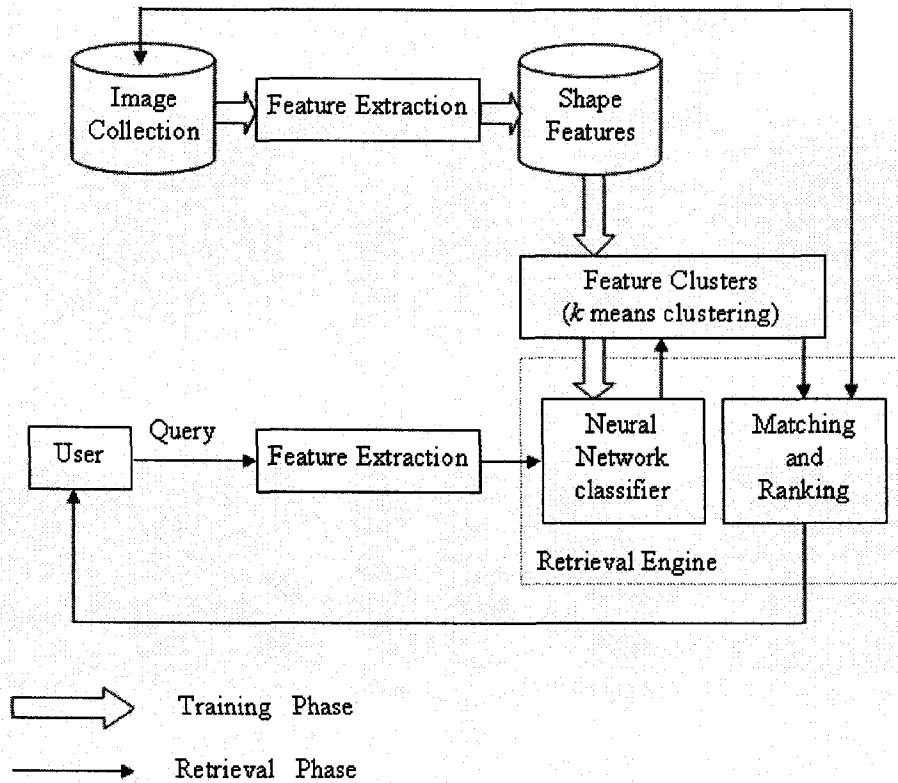


Figure 3.7 Architecture of shape-based image retrieval system using Neural Network

Traditional method for Content-Based Image Retrieval is to index the extracted visual feature vectors with multidimensional indexing trees as discussed in Section 2.2. In this thesis, we introduce clustering techniques and Neural Network for shape-based image retrieval and build an automatic intelligent retrieval engine.

Our strategy consists of two stages: training and retrieving (testing), as shown in Figure 3.7. In the training stage, we use all of the images in our image database as training samples. We first group the training samples into clusters with their feature vectors. Then, we use an Artificial Neural Network and train it with the result of clustering. In the testing stage, for a query image q , we extract its feature vector and feed it into the trained Neural Network and the network assigns it to one or more similar clusters. Then, we compare all of the images in the selected cluster(s) against the query image q . Finally, similar images are ranked by their similarities using distance functions and returned as the retrieval results.

3.2.2 Clustering

3.2.2.1 k Means Clustering

k means clustering [26] is one of the best known clustering algorithm in Pattern Recognition. It is a non-hierarchical approach to forming good clusters – specify a desired number of clusters, say, k , then assign each sample to one of the k clusters so as to minimize a measure of dispersion within the clusters. The procedure of determining the membership for every sample is through an unsupervised learning procedure.

Of the various techniques that can be used to simplify the computation and accelerate convergence, we consider one elementary but very popular method. The algorithm of k means clustering is very simple. It first randomly initializes k cluster means, and then assigns each pattern vector to the cluster of the nearest mean. Re-compute cluster means and re-assign pattern vectors, until no changes occurred. The algorithm shows as following:

Algorithm: k Means Clustering**Input:** Number of clusters, k ;A training dataset $D = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_N\}$ **Output:** k mean vectors $\mu_1, \mu_2, \mu_3, \dots, \mu_k$ **Method:**

- (1) Initialize $\mu_1, \mu_2, \mu_3, \dots, \mu_k$;
- (2) **repeat** {
- (3) Assign $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_N$ to the cluster of the nearest μ_i
- (4) Recompute $\mu_1, \mu_2, \mu_3, \dots, \mu_k$
- (5) } **until** no change in μ_i

Although it is unsupervised, k Means Clustering algorithm needs to provide the number of clusters, k , to start with. It also needs some knowledge about the problem to determine the value of k . The nearest mean μ_i is found by using an arbitrary distance function, and will be discussed more in Section 3.2.2.2. Initial values of μ_i affect the convergence since different initialization values may lead to different membership results. Therefore, we either guess initial values based on the knowledge of the problem, or choose k random samples from $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$.

3.2.2.2 Distance Functions

Distance functions are used to measure the similarity or dissimilarity of two feature vectors. In a d -dimensional space, for any two elements \mathbf{x} and \mathbf{y} , there exists a real number $D(\mathbf{x}, \mathbf{y})$, called the distance function (or metric), and it satisfies the following properties [12, 3]:

- | | |
|--|-----------------------|
| (1) $D(\mathbf{x}, \mathbf{y}) \geq 0$ | (non-negativity) |
| (2) $D(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$ | (identity) |
| (3) $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$ | (symmetry) |
| (4) $D(\mathbf{x}, \mathbf{z}) \leq D(\mathbf{x}, \mathbf{y}) + D(\mathbf{y}, \mathbf{z})$ | (triangle inequality) |

There are many distance functions available, for example, Manhattan distance, Euclidean distance, Minkowski distance and Mahalanobis distance. We have tested and compared Euclidean distance and Mahalanobis distance in our thesis work.

Euclidean Distance

Euclidean distance is a very simple, well known and widely used distance function. It is defined as:

$$D(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=0}^{d-1} (x_i - y_i)^2 \right)^{\frac{1}{2}} = \sqrt{(\mathbf{x} - \mathbf{y})' (\mathbf{x} - \mathbf{y})},$$

where \mathbf{x} and \mathbf{y} are two feature vectors of the same dimension d .

Euclidean distance is a member of the Minkowski-form distance family. The general definition of Minkowski-form distance is:

$$D_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=0}^{d-1} (x_i - y_i)^p \right)^{\frac{1}{p}}.$$

This family of distance functions is parameterized by p . When $p = 1$, it is called Manhattan distance, and is also known as the city block distance:

$$D_1(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{d-1} |x_i - y_i|;$$

When $p = 2$, it is our Euclidean distance; And when $p = \infty$ ($p \rightarrow \infty$), it is called Chebychev distance:

$$D_{\infty}(\mathbf{x}, \mathbf{y}) = \max_{0 \leq i < d} \{|x_i - y_i|\}.$$

To intuitively convey the difference between these distance functions, we can construct a metric space where a set of points centered at a point O with the same distance r make up a ball (sphere). Figure 3.8 [3] shows the unit spheres under Manhattan (D_1), Euclidean (D_2), Minkowsky with $p = 14$ (D_{14}), and Chebychev (D_{∞}) distance. Balls in Euclidean spaces, as we all know, are the spherical surfaces. A ball in D_{∞} is a hypersquare aligned with the coordinate axes. A ball in D_1 is a hypersquare, having vertices on the coordinate

axes. A ball in D_p , for $1 < p < 2$, looks like a “slender sphere” that lies between the D_1 and D_2 balls, whereas for $p > 2$, lies between the D_2 and D_∞ balls and looks like a “fat sphere”.

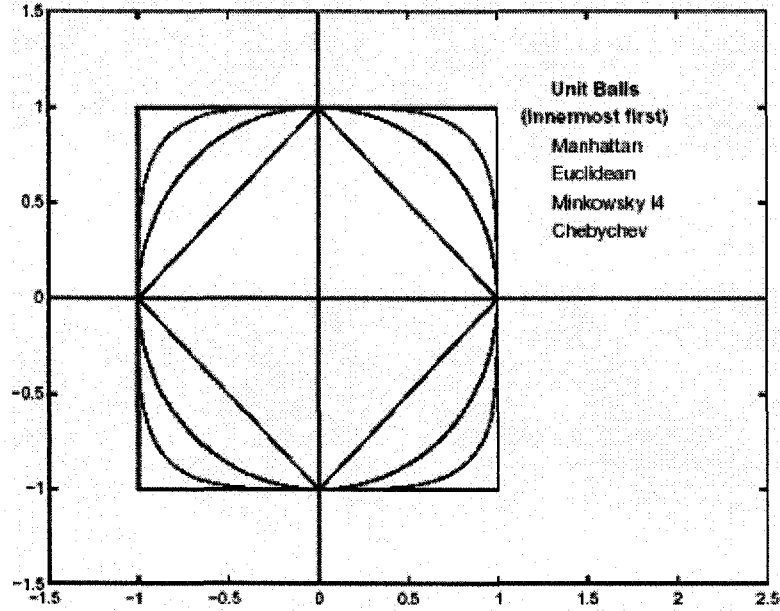


Figure 3.8 The unit spheres under Minkowsky distances
(After: [3])

Mahalanobis Distance

The Mahalanobis distance is a generalization of the Euclidean distance. It is defined as:

$$D(\mathbf{x}, \mathbf{y}) = [(\mathbf{x} - \mathbf{y})' \Sigma^{-1} (\mathbf{x} - \mathbf{y})]^{\frac{1}{2}}$$

where Σ is the covariance matrix of \mathbf{x} :

$$\Sigma = \frac{1}{N} \sum_{i=0}^{N-1} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})',$$

$$\boldsymbol{\mu} = \sum_{i=0}^{N-1} \mathbf{x}_i.$$

Σ is a positive, semidefinite and symmetric matrix. When $\Sigma = \mathbf{I}$, the distance is reduced to Euclidean distance and when $\Sigma \neq \mathbf{I}$, it is called Mahalanobis distance. Both Euclidean distance and Mahalanobis distance are commonly used in clustering algorithms. We

know that the center of a cluster is determined by its mean vector, μ , and the shape of the cluster is determined by its covariance matrix, Σ .

The use of the Mahalanobis distance removes several of the limitations of the Euclidean distance:

- it automatically accounts for scaling of the coordinate axes.
- it corrects for correlation between different features.
- it can provide curved as well as linear decision boundaries.

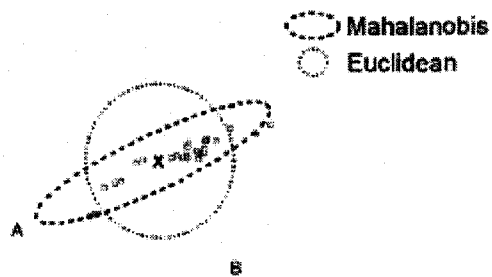


Figure 3.9 Euclidean distance and Mahalanobis distance
(Source: [3])

Figure 3.9 [3] graphically illuminates the relationship between the Euclidean distance and the Mahalanobis distance. For most datasets, Mahalanobis distance can present a precise shape of the cluster but with more time complexity than the Euclidean distance which though corresponds to less time complexity but at the cost of reduced precision.

3.2.2.3 Clustering Evaluation

Clustering techniques generally aim to partition the dataset into a number of clusters. The two fundamental questions that need to be addressed in any typical clustering system are: (i) how many clusters are actually present in the dataset and (ii) how real or good is the clustering itself [28].

In our approach, our aim is to use a clustering algorithm that could provide us a minimum within-cluster distance and maximum inter-cluster distance. Based on this idea, there are several cluster validity indices to evaluate the partitioning obtained by the clustering algorithm. Davies-Bouldin (DB) Index is one of them. This index is a function of the ratio of the sum of within-cluster scatter to the inter-cluster separation. The scatter within the i^{th} cluster, S_i , is computed as:

$$S_i = \frac{1}{N_i} \sum_{\mathbf{x} \in C_i} |\mathbf{x} - \boldsymbol{\mu}_i|,$$

and the distance between two clusters C_i and C_j , denoted by D_{ij} , is defined as:

$$D_{ij} = |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|.$$

Here, $\boldsymbol{\mu}_i$ represents the i^{th} cluster center. The Davies-Bouldin (DB) index is then defined as:

$$DB = \frac{1}{k} \sum_{i=1}^k R_{i,ql},$$

where

$$R_{i,ql} = \max_{j, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{D_{ij,l}} \right\},$$

with the objective to minimize the DB index for achieving proper clustering.

3.2.3 Neural Network

In our approach, after clustering, we use the Neural Network as part of the retrieval engine. Neural Network is a computer model to imitate human brain. It has great potential in image retrieval and has been successfully used in pattern recognition, data mining, industrial process control, data validation, etc. [38]. Neural Network is often used as a classifier. The best advantage of Neural Network classifier is its ability to recognize unfamiliar patterns.

3.2.3.1 Background of Neural Network

Neural Network (also called Artificial Neural Network) is a mathematically modeled information processing simulation of Biological Neural Network, which has the following characteristics [7]:

- information processing takes place at neurons where many signals are transmitted.
- neurons communicate with each other via axons, each of which has a synapse to multiply signal transmitted over it.
- each neuron applies certain activation to its input signal to determine its output signal. The output from a particular neuron may go to many other neurons.
- a synapse's strength may be modified by experience.

In Neural Network, neurons (or nodes) are mathematically modeled by an activation function f , such as the identity function, binary step function, binary sigmoid function and bipolar sigmoid function, etc. Axons (or links) are modeled by the combination function u . The most common combination function is the sum of weighted input. Nodes and links compose basic unit of Neural Network as shown in Figure 3.10 [22].

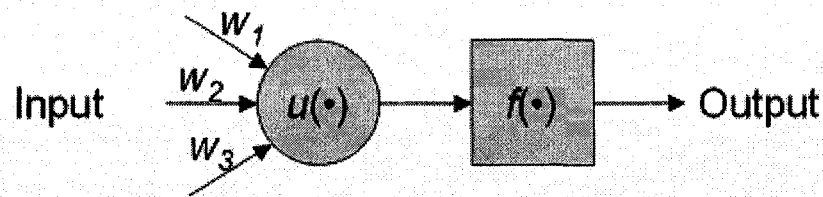


Figure 3.10 Basic unit of Neural Network

(Source: [22])

In its applications in the field of Pattern Recognition, Neural Network can predict an input pattern to a familiar class or recognize unfamiliar patterns. Neural Network has great potential of achieving human-like performance in the fields of image recognition [22] and is a promising technique for indexing image features [35].

3.2.3.2 Multilayer Neural Network

A multilayer Neural Network consists of an input layer, an output layer, and one or more hidden layers as shown in Figure 3.11 [22]. Computing nodes in each layer are arranged so that the output of every node in one layer becomes an input of every node in the next layer [12].

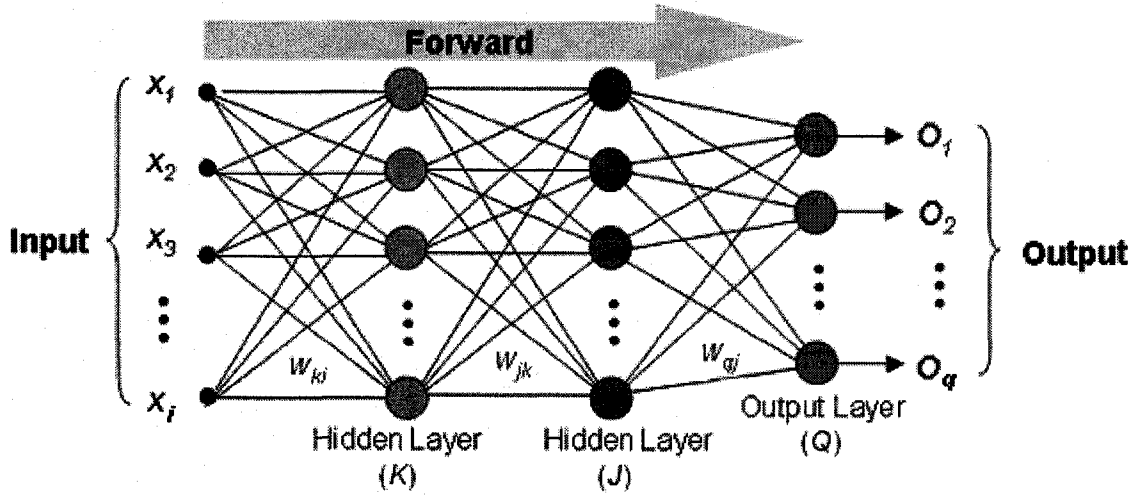


Figure 3.11 Multilayer (3-layer) Neural Network

(Source: [22])

Input Layer

Feature vectors $\mathbf{x} = [x_1, x_2, x_3, \dots, x_i]$ serves as an input to the Neural Network. The number of nodes in the input layer is the number of elements of the feature vector, i.e., i . In this thesis, Moment Invariants and Zernike Moments serve as feature vectors, so that every element of the moment vector is fed in to the Neural Network. The number of nodes in the input layer in our case are 7 for Moment Invariants, and 6 for Zernike Moments.

Normally, the inputs of the Neural Network do not need to be normalized. However, there is a significant difference of magnitude between values of Moment Invariants and those of Zernike Moments. Therefore, we normalized the feature vector as follows:

$$\mathbf{x} = [x_1, x_2, x_3, \dots, x_i],$$

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_i^2},$$

$$\mathbf{x}_u = \left[\frac{x_1}{\|\mathbf{x}\|}, \frac{x_2}{\|\mathbf{x}\|}, \frac{x_3}{\|\mathbf{x}\|}, \dots, \frac{x_i}{\|\mathbf{x}\|} \right].$$

where \mathbf{x}_u is the unit vector after normalization.

Hidden Layer

The hidden layer is used to represent the interaction between the input layer and the output layer. The number of hidden layers and nodes depends on the particular problem being attacked using a Neural Network, and in many cases, such issues can be solved in practice by trial and error [29].

For the number of hidden layers and the number of nodes in each layer, there is always a tradeoff: with too few nodes and links, the Neural Network may not be powerful enough to form a decision region that is as complex as required by a given learning task, and may perform poorly on new testing samples; on the other hand, with a large size network, it may not be possible to reliably estimate weights from the available training data and computation would be too expensive.

Assuming that all members in a cluster belong to the same class and samples of each class may be spread over different clusters, Lippmann [22] has studied the types of decision regions that can be formed by a single- and multi-layer Neural Network with one and two layers of hidden units and two inputs, and are illustrated in Figure 3.12 [22].

The smooth closed contours labeled ω_1 and ω_2 in Figure 3.12 are the input distribution for the two classes. Distributions for the two classes for the exclusive OR problem are disjoint and can not be separated by a single straight line. Input distributions for the second problem shown in this figure are meshed and also can not be separated by a single straight line.

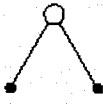
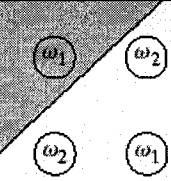
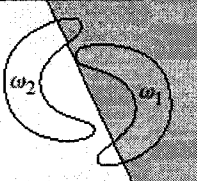
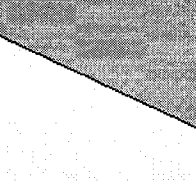
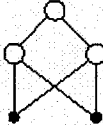
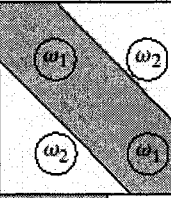
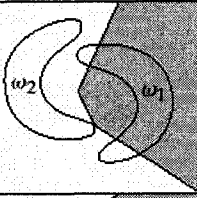
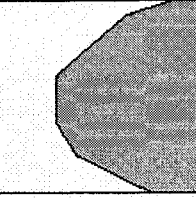
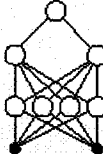
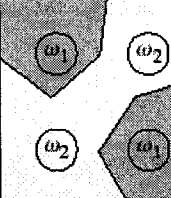
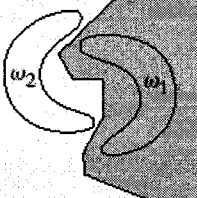
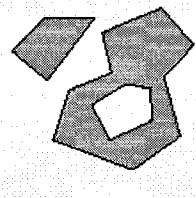
Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)			

Figure 3.12 Neural Networks with one or two hidden layers and their corresponding type of decision region

(After: [22])

With only two layers (one layer of hidden units), a Neural Network can generate a convex open or closed decision region as shown in the second row of Figure 3.12. It is effective enough for the exclusive OR problem, but still has difficulties with meshed classes. A three-layer network can form arbitrarily complex decision regions and can separate the meshed classes as shown in the figure. It can form regions as complex as those formed using mixture distributions and nearest-neighbor classifiers.

The above analysis demonstrates that a three layer network may be sufficient for a feed-forward Neural Network because it can outline arbitrarily complex decision regions. It can also provide some insight into the problem of selecting the number of nodes to use in the three-layer networks. The number of nodes in the second layer must be greater than 1 when decision regions are disconnected or meshed and cannot be formed from one convex area. The number of second layer nodes required in the worst case is equal to the number of disconnected regions in input distributions. The number of nodes in the first

layer must typically be sufficient to provide three or more edges for each convex area generated by every second-layer node. There should typically be more than three times as many nodes in the second layer as in the first layer [22].

Output Layer

The output layer in our system is used to predict the cluster to which the testing sample may belong. There are two kinds of representation schemes that can be used on the output layer [13]: (i) direct representation, and (ii) binary coding representation. Direct representation uses the number of clusters of dataset as the number of nodes on the output layer. In this representation, only one node gets the “high” value and the node that has the “high” value is the cluster of the testing sample. For example, there are eight nodes on the output layer if there are eight clusters of shape images. If a test shape belongs to the third cluster (index number of cluster is 3), then the output of the output layer will be 00001000. The other representation scheme, binary coding representation, uses $\lceil \log_2 k \rceil$ nodes on the output layer for k clusters of shape images. For example, if a shape belongs to the third cluster, then the output of the layer will be 011.

In this thesis, we use the direct representation because it is straightforward, easy to use and has the capability of extension (see section 5.2).

3.2.3.3 Feed-Forward

Feed-forward is a process in which a sample vector feeds the input layer, goes through the computation of functions of every node in every layer in an order, and finally reaches the output layer.

As illustrated in Figure 3.10, each neuron (node) has the same structure as the basic unit in the Neural Network. Since the combination function u is the weighted sum, the input to a node in any layer is the weighted sum of the outputs from the previous layer. In our 3-layer Neural Network model (Figure 3.11), K denotes the first hidden layer, J denotes

the second hidden layer, and Q denotes the output layer. If the inputs to the activation function f of each node in each layer are denoted as I , then we have:

$$I_k = \sum_i w_{ki} x_i ,$$

$$I_j = \sum_k w_{jk} O_k , \text{ and}$$

$$I_q = \sum_j w_{qj} O_j .$$

The outputs of each node in each layer are:

$$O = f(I) ,$$

where f is the activation function.

Activation Functions

An ideal activation function for a Neural Network should have several important characteristics [7, 29]:

- The same activation function should be used for all nodes in any particular layer of a Neural Network, even though it is not a requirement.
- In order to achieve the advantages of multilayer network, in most cases, a nonlinear activation function should be used.
- The activation function should be continuous, differentiable, and monotonically non-decreasing.
- Aiming at the back-propagation, for computational efficiency, it is desirable that the derivative of the activation function be easy to compute. For the most commonly used activation functions, the values of the derivative can be expressed in terms of the value of the function.

Therefore, the soft-limiting sigmoid functions (S-shape curves) are useful activation functions whereas the binary/bipolar sigmoid function and the hyperbolic tangent function are the most common ones. It is especially advantageous to use them in Neural Networks trained by back-propagation, because the simple relationship between the value of the function at a point and the value of the derivative at the point reduces the computational burden during training.

1. Binary sigmoid:

The Binary sigmoid function is expressed as:

$$f(x) = \frac{1}{1 + e^{-x}},$$

and as shown in Figure 3.13, it is a soft-limiting function. It is always positive, and can reach its limiting values of 0 and 1 only if the input to the activation element is infinitely negative or positive, respectively. For this reason, values near 0 and 1 (say, 0.05 and 0.95) are considered as “low” and “high” values at the output of the nodes in the network.

The derivative procedure of binary sigmoid function is given as follows:

$$\begin{aligned} f(x) &= \frac{1}{1 + e^{-x}} \rightarrow \\ f'(x) &= -\frac{1}{(1 + e^{-x})^2} (1 + e^{-x})' \\ &= -\frac{1}{(1 + e^{-x})^2} (e^{-x})' \\ &= -\frac{1}{(1 + e^{-x})^2} e^{-x} (-x)' \\ &= \frac{e^{-x}}{(1 + e^{-x})^2} \\ &= \frac{1}{1 + e^{-x}} \frac{e^{-x}}{1 + e^{-x}} \\ &= \frac{1}{1 + e^{-x}} \left(\frac{1 + e^{-x}}{1 + e^{-x}} - \frac{1}{1 + e^{-x}} \right) \end{aligned}$$

$$\therefore f'(x) = f(x)[1 - f(x)]$$

2. Bipolar sigmoid:

Sigmoid function can be scaled to have any range of values that may be appropriate for a given problem. The most common range is from -1 to 1. We call this sigmoid a bipolar sigmoid and is illustrated in Figure 3.14, whereas its expression is given as:

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}.$$

The bipolar sigmoid is closely related to the hyperbolic tangent function, which is also often used as the activation function when the desired range of output values is between -1 and 1. The hyperbolic tangent is:

$$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}.$$

The derivatives of bipolar sigmoid function and hyperbolic tangent function are derived as follows:

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}} = \frac{2 - 1 - e^{-x}}{1 + e^{-x}} = \frac{2}{1 + e^{-x}} - 1 = 2f(x) - 1 \rightarrow$$

$$\begin{aligned} g'(x) &= 2f'(x) \\ &= 2f(x)[1 - f(x)] \\ &= f(x)[2 - 2f(x)] \\ &= \frac{1}{2}[1 + 2f(x) - 1][1 - 2f(x) + 1] \end{aligned}$$

$$\therefore g'(x) = \frac{1}{2}[1 + g(x)][1 - g(x)]$$

$$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} = \frac{2}{1 + e^{-2x}} - 1 \rightarrow$$

$$\begin{aligned} h'(x) &= \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2} \\ &= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ &= 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right)^2 \end{aligned}$$

$$\therefore h'(x) = 1 - h^2(x)$$

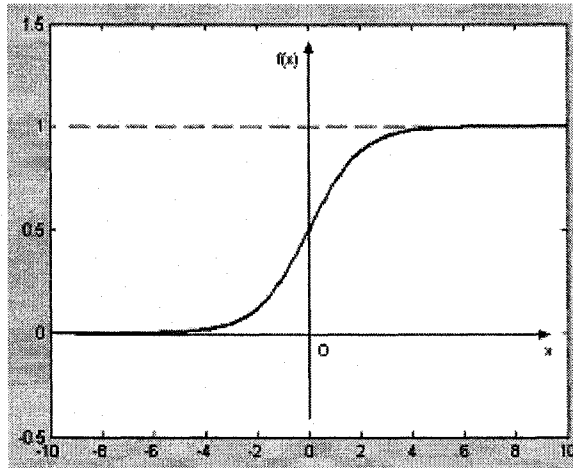


Figure 3.13 Binary sigmoid

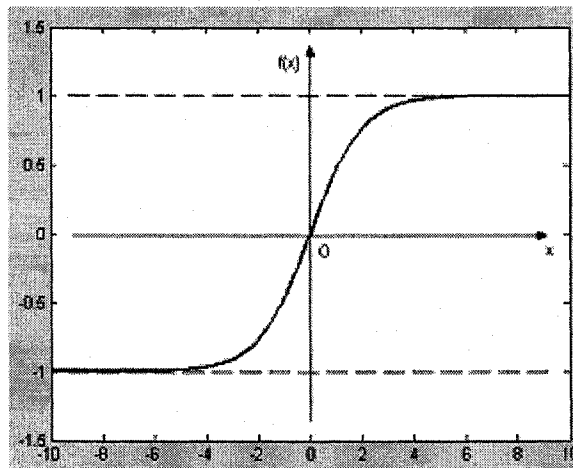


Figure 3.14 Bipolar Sigmoid

Initialization of Weights

Computation in a Neural Network generally starts by choosing the initial weights. Typically, the weights chosen are small random values (between -1.0 and 1.0 or -0.5 to +0.5), since larger weight magnitudes may drive the output of layer 1 nodes to saturation, requiring much larger training time to emerge from the saturated states [29]. This phenomenon results from the behavior of the sigmoid functions.

If the magnitudes of some of the inputs are much larger than the others, as is the case with Moment Invariants and Zernike Moments in this thesis, random initialization may

bias the network to give much greater importance to the inputs whose magnitudes happen to be large. In such a situation, weights can be initialized as stated below so that the network input to each hidden node is roughly of the same magnitude.

For the weights associated with inputs:

$$w_{ki} = \pm \frac{1}{2N} \sum_{p=0}^{N-1} \frac{1}{|x_i|},$$

where N is the size of the dataset. For the weights associated with other layers in the network:

$$w_{jk} = \pm \frac{1}{2N} \sum_{p=0}^{N-1} \frac{1}{f(\sum w_{ki} x_i)},$$

$$w_{qj} = \pm \frac{1}{2N} \sum_{p=0}^{N-1} \frac{1}{f(\sum w_{jk} O_k)}.$$

3.2.3.4 Training by Back-Propagation

Training a Neural Network by back-propagation involves three stages [7]: feed-forward the input training sample, back-propagate the associated error of each layer to its previous layer, and adjust the weights connecting these two layers accordingly.

Starting from the output layer, the total squared error between the actual output O_q and the target result T_q is:

$$Err = \frac{1}{2} \sum_{q=1}^{N_q} (T_q - O_q)^2,$$

where N_q is the number of nodes in the output layer Q .

The objective of training is to adjust the weights in each of the layers in a way to minimize the total squared error Err . To achieve this result, we can adjust the weights in proportion to the partial derivative of the error with respect to the weights:

$$\Delta w_{qj} = -\alpha \frac{\partial Err}{\partial w_{qj}}. \quad (\text{Equation 3.2})$$

The error Err is a function of the output O_q , which in turn is a function of the input I_q . Using the chain rule, we evaluate the partial derivative of Err as follows:

$$\frac{\partial Err}{\partial w_{qj}} = \frac{\partial Err}{\partial I_q} \cdot \frac{\partial I_q}{\partial w_{qj}}. \quad (\text{Equation 3.3})$$

From the combination function,

$$\frac{\partial I_q}{\partial w_{qj}} = \frac{\partial}{\partial w_{qj}} \sum_j w_{qj} O_j = O_j. \quad (\text{Equation 3.4})$$

Substituting Equation 3.3 and 3.4 into 3.2 yields:

$$\Delta w_{qj} = -\alpha \frac{\partial Err}{\partial I_q} O_j = \alpha \delta_q O_j,$$

where

$$\delta_q = -\frac{\partial Err}{\partial I_q}.$$

Then, we use the chain rule again to express the partial derivative in terms of the rate of change of Err with respect to O_q and the rate of change of O_q with respect to I_q . That is,

$$\delta_q = -\frac{\partial Err}{\partial I_q} = -\frac{\partial Err}{\partial O_q} \cdot \frac{\partial O_q}{\partial I_q}. \quad (\text{Equation 3.5})$$

From the definition of the total squared error,

$$\frac{\partial Err}{\partial O_q} = -(T_q - O_q). \quad (\text{Equation 3.6})$$

From the activation function,

$$\frac{\partial O_q}{\partial I_q} = \frac{\partial}{\partial I_q} f(I_q) = f'(I_q). \quad (\text{Equation 3.7})$$

Substituting Equation 3.6 and 3.7 into 3.5 yields

$$\delta_q = (T_q - O_q) f'(I_q),$$

which is proportional to the error quantity $(T_q - O_q)$. Therefore, after the activation function f has been specified, all the terms in the above questions are known or can be observed in the network.

Continuing to work our way back from the output layer, let us now analyze what happens at the hidden layer J . Proceeding in the same manner as above yields

$$\Delta w_{jk} = \alpha \delta_j O_k,$$

where the error term is

$$\delta_j = (T_j - O_j) f'(I_j).$$

With the exception of T_j , all the terms in the above two equations are either known or can be observed in the network.

Going back to the error term for layer J , we can write it as

$$\delta_j = -\frac{\partial \text{Err}}{\partial I_j} = -\frac{\partial \text{Err}}{\partial O_j} \cdot \frac{\partial O_j}{\partial I_j},$$

The term $\partial O_j / \partial I_j$ is easy to explain. As before, it is given as:

$$\frac{\partial O_j}{\partial I_j} = \frac{\partial}{\partial I_j} f(I_j) = f'(I_j).$$

The term that produced T_j is the derivative $\partial \text{Err} / \partial O_j$, so it must be expressed in a way that does not contain T_j . Using the chain rule, we write the derivative as:

$$\begin{aligned} -\frac{\partial \text{Err}}{\partial O_j} &= -\sum_q \frac{\partial \text{Err}}{\partial I_q} \cdot \frac{\partial I_q}{\partial O_j} = \sum_q \left(-\frac{\partial \text{Err}}{\partial I_q} \right) \cdot \frac{\partial}{\partial O_j} \sum_j w_{qj} O_j \\ &= \sum_q \left(-\frac{\partial \text{Err}}{\partial I_q} \right) \cdot w_{qj} \\ &= \sum_q \delta_q w_{qj} \end{aligned}$$

where the last step follows from

$$\delta_q = -\frac{\partial \text{Err}}{\partial I_q}.$$

Finally, we can express δ_j as

$$\delta_j = f'(I_j) \sum_q \delta_q w_{qj},$$

and now that all of its terms are known, it can be computed very easily.

Similarly, for the hidden K , we can write:

$$\Delta w_{ki} = \alpha \delta_k O_i = \alpha \delta_k x_i, \text{ and}$$

$$\delta_k = f'(I_k) \sum_j \delta_j w_{jk}.$$

Training termination criterion

Training is complete when the weights no longer change much or until the network has gone through the training set for a specified number of iterations. More specifically, the termination condition in this thesis is:

- 1) Minimum Squared Error (MSE) is less than a threshold ϵ , say $\epsilon = 0.05$
- 2) OR
 - a) $| \text{MSE}(t) - \text{MSE}(t-1) | < \epsilon$
 - b) AND the network has gone through the training samples a maximum number of times.

In summary, the back-propagation algorithm is illustrated as follows:

Algorithm: Back-Propagation. Neural Network learning for classification, using the back-propagation algorithm.

Input: The training samples, *Samples*; a multilayer feed-forward neural network, *NN*.

Output: A Neural Network trained to classify the samples.

Method:

- (6) Initialize all weights in *NN*;
- (7) **while** terminating condition is not satisfied {
- (8) **for each** training sample X in *Samples* {
- (9) // Propagate the input forward;
- (10) **for each** hidden or output layer unit j {
- (11) $I_j = \sum_i w_{ij} O_i$;


```

// Compute the net input of unit  $j$  with respect to the previous layer  $i$ 
(12)  $O_j = f(I_j)$ ; // Compute the output of each unit  $j$ 
(13) // Back-propagate the errors:
(14) for each unit  $j$  in the output layer
(15)  $\delta_j = (T_j - O_j)f'(I_j)$ ; // Compute the error
(16) for each unit  $j$  in the hidden layers, from the last to the first hidden layer
(17)  $\delta_j = f'(I_j)\sum_k Err_k w_{jk}$ ;
// Compute the error with respect to the next higher layer  $k$ 
(18) for each weight  $w_{ij}$  in  $NN$  {
(19)  $\Delta w_{ij}(t) = (\eta)\delta_j O_i + (\alpha)\Delta w_{ij}(t-1)$ ;
// Weight increment where  $0 < \alpha, \eta < 1$  with typical values of  $\eta = 0.5$  to  $0.6$ ,
//  $\alpha = 0.8$  to  $0.9$ 
(20)  $w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$ ; } // Weight update
(21) } }

```

4 Experiments and Discussions

4.1 Image Collection

Since our main objective is to classify and retrieve specific shapes and to validate our proposed theoretical steps, we chose to deal with only binary shape images. Use of binary images allowed us to focus more on training and retrieval strategies rather than the image processing steps such as edge detection, etc. However, it must be noted that a system can be developed to deal with any type of images, including color images, and that system needs to employ specific image processing steps. It is also important to note that in order to deal with the requirements of robustness and invariance under various geometrical transformations, data set needs to contain images with noise and geometric transformations such as rotation and scaling.

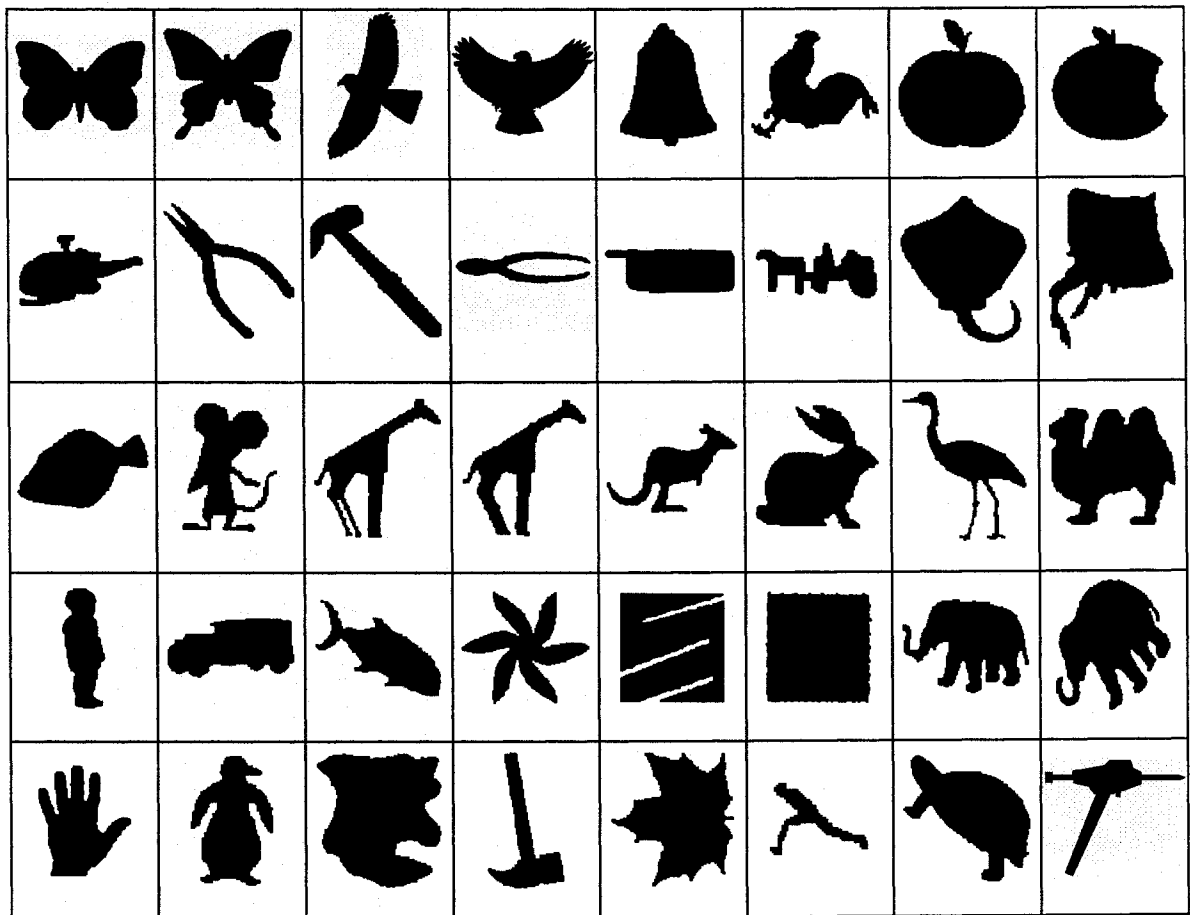


Figure 4.1 Sample binary shape images in the image collection

Our image data set consists of images from “A Large Binary Image Database” [55] which is a large database of closed binary shapes collected by the LEMS Vision Group at Brown University. This collection contains more than 5000 binary shape images and has many variations of same shape and object. This data set consists of some groups of similar shapes that are scaled, rotated and slightly distorted, as well as some images that are unique. Figure 4.1 is an example of some of the images in this collection.

Programs to extract Moment Invariants and Zernike Moments from binary shape images, k means clustering and back-propagation Neural Network for indexing and retrieval are implemented using C# on Microsoft .NET Framework 1.1. Algorithms have been preliminarily tested using Matlab 6.1 with a small set of data (150 images). Experiments are done and results are collected on an Intel Pentium III PC, running Microsoft Windows 2000.

4.2 Comparison on Euclidean and Mahalanobis Distances

From our analysis in section 3.2.2.2, we know that compared with Euclidean distance, Mahalanobis distance presents a more precise shape of a cluster because it takes into account not only the average value but also the variance and the covariance of the cluster members. Table 4.1 shows the experiments on k means clustering with these two distances. We use the Davies-Bouldin (DB) Index mentioned in section 3.2.2.3 to evaluate the result of clustering, and plot the values of DB index in Figure 4.2 for different value of k (from 3 to 10). From this figure, we can see that for the small values of k , for example $k = 3$, $k = 4$, clustering results with both the Euclidean distance and the Mahalanobis distance are very similar, but when k is greater than 5, Mahalanobis distance performs much better than Euclidean distance. It is primarily due to the fact that the Mahalanobis distance presents a precise shape of clusters whereas the Euclidean distance always forms a circle irrespective of the shape of the data set.

k	Euclidean Distance		Mahalanobis Distance	
	DB Index	Time (s)	DB Index	Time (s)
3	0.061037	0.170	0.061037	0.140
4	0.077329	0.250	0.071343	0.761
5	0.173958	0.330	0.086476	7.360
6	0.189725	0.451	0.106956	28.680
7	0.187746	0.741	0.114014	56.600
8	0.189074	0.912	0.108673	68.847
9	0.175987	1.392	0.123772	99.871
10	0.179971	1.663	0.105527	122.222

Table 4.1 Clustering validity and time with Euclidean and Mahalanobis distances

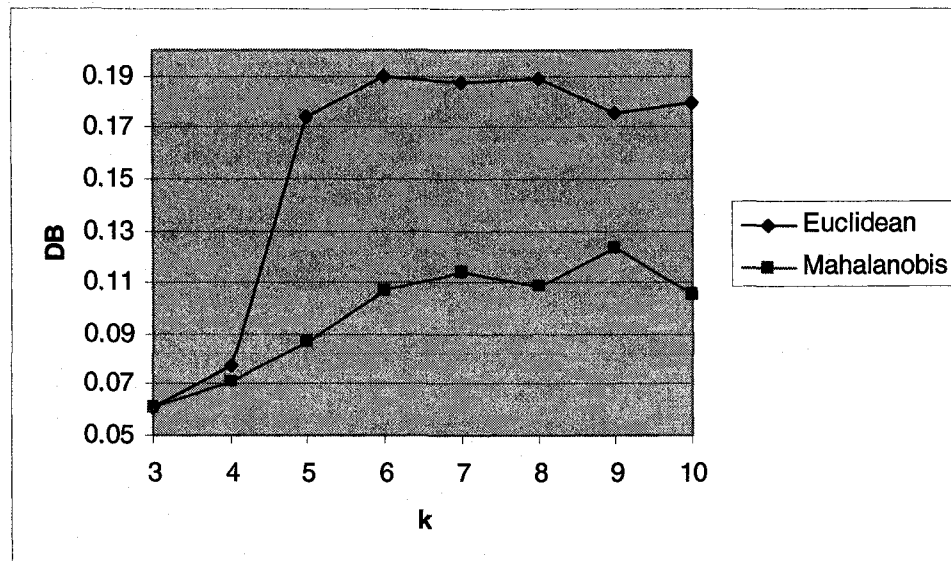


Figure 4.2 The values of DB index for k means clustering with Euclidean and Mahalanobis distances

However, Mahalanobis distance function has a higher time complexity than Euclidean distance function. In the k means clustering algorithm with Mahalanobis distance, after

assigning samples to the cluster of the nearest mean, we have to recompute not only the new cluster mean, but also the new cluster covariance in each loop of our computation.

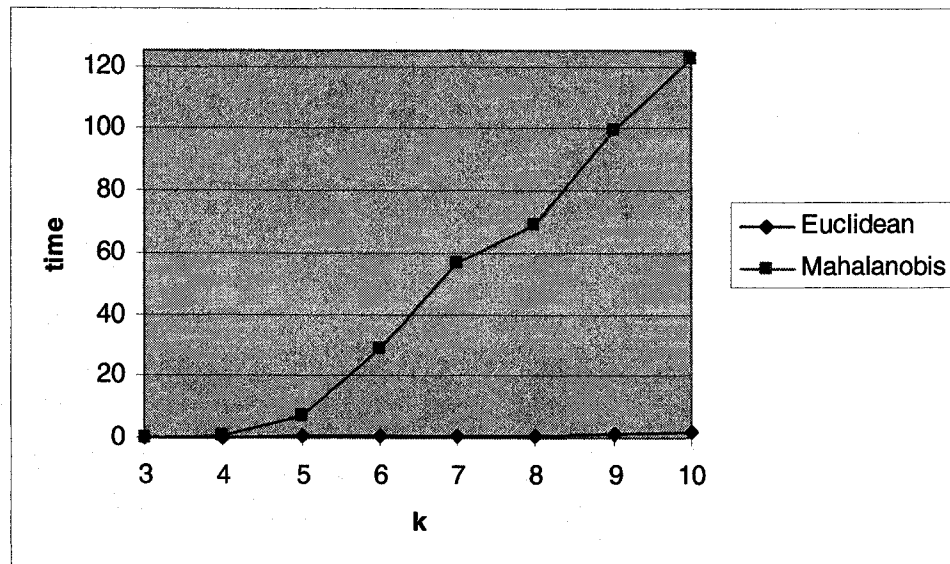


Figure 4.3 Clustering time with Euclidean and Mahalanobis distances

For Euclidean distance, even after few iterations, it is easy to have the cluster means stable, but for Mahalanobis distance, it always takes much longer despite the fact that we use a cutoff or threshold as a termination condition (new cluster means doesn't move very much, say less than 0.1%). Figure 4.3 shows the clustering time (in seconds) with these two distances. Obviously, the larger the value of k , more is the time needed for k means clustering; with end result that the Mahalanobis distance is more time consuming than the Euclidean distance.

4.3 Comparison on Activation Functions

As mentioned before, in our Neural Network system, we have used and collected results with three activation functions: binary sigmoid, bipolar sigmoid and hyperbolic tangent. Results in terms of errors of Neural Network after training and corresponding training time are given in Table 4.2. During the training phase, we can observe that the minimum squared error (MSE) fluctuates in the first few training loops, and then keeps on

decreasing at a very slow rate. Since the error reduction is very slow, the training termination in our case mostly occurs due to the second condition as mentioned in section 3.2.3.4. This allows the Neural Network to go through all the training samples a maximum number of times, i.e., 10000 times in our experiments.

k	Binary Sigmoid			Bipolar Sigmoid			Hyperbolic Tangent		
	MSE	AV Err	Time (h)	MSE	AV Err	Time (h)	MSE	AV Err	Time (h)
5	0.1373	0.2343	1.1142	0.1213	0.2203	1.1536	0.1345	0.2319	1.2782
6	0.1550	0.2273	1.3451	0.1464	0.2209	1.3607	0.1361	0.2130	1.4401
7	0.1734	0.2226	1.5935	0.1548	0.2103	1.6546	0.1496	0.2067	1.6932
8	0.1918	0.2189	1.8410	0.1632	0.2019	1.9484	0.1630	0.2018	1.9462
9	0.2137	0.2179	2.0627	0.1724	0.1957	2.2414	0.1725	0.1958	2.2261
10	0.2345	0.2165	2.1988	0.1793	0.1893	2.3379	0.1793	0.1893	2.3149

Table 4.2 Errors of the Neural Network with three different activation functions

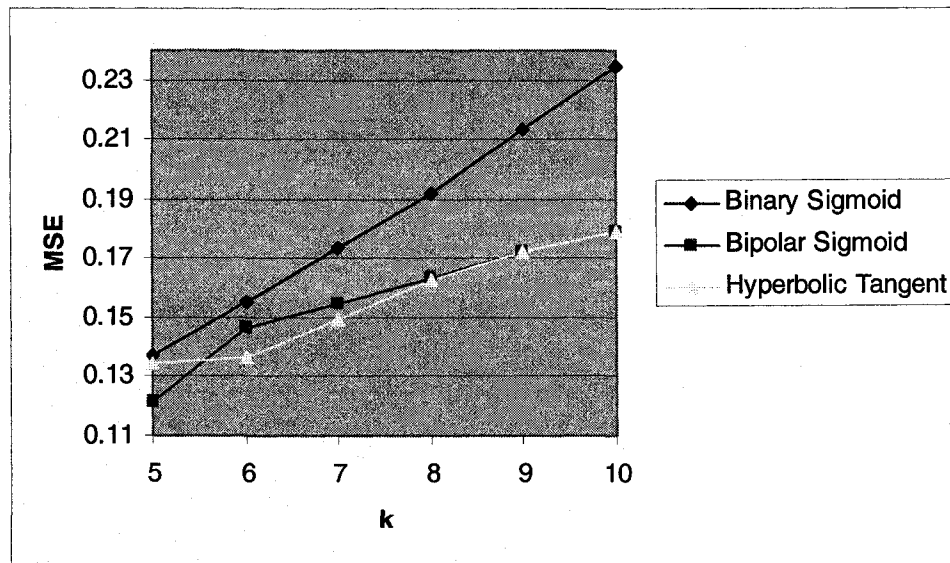


Figure 4.4 The MSE of the Neural Network with three different activation functions

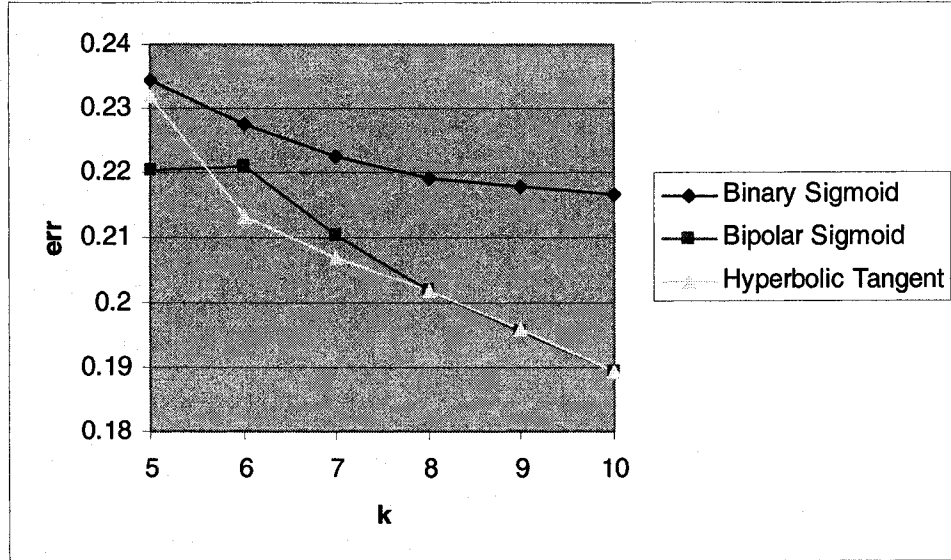


Figure 4.5 The average of errors of each node in the output layer of the Neural Network with three different activation functions

We know that the total squared error on the output layer of the Neural Network is:

$$MSE = \frac{1}{2} \sum_{q=1}^{N_q} (T_q - O_q)^2$$

where the number of nodes in the output layer is the number of clusters k in our case. Thus the average of errors of each node in the output layer of the Neural Network can be estimated by:

$$err = \sqrt{\frac{2 \cdot MSE}{k}}.$$

We plot the MSE and the average of errors of the output nodes with the three mentioned activation functions in Figure 4.4 and Figure 4.5 respectively. Both figures illustrate that the bipolar sigmoid and hyperbolic tangent functions with values in the range from -1 to 1 perform better than the binary sigmoid whose values ranges from 0 to 1.

4.4 Retrieval Performance of Moment Invariants and Zernike Moments

After finishing the training of Neural Network – our retrieval engine, we can run the test program to see how the system performs on a given query image. Ideally, the shape-based image retrieval system must return all the similar shape images and must not return any dissimilar shape images. Such a system is considered to have high precision. However, real image retrieval systems are able to return only some of the similar images in the image collection, and may retrieve some irrelevant images as well.

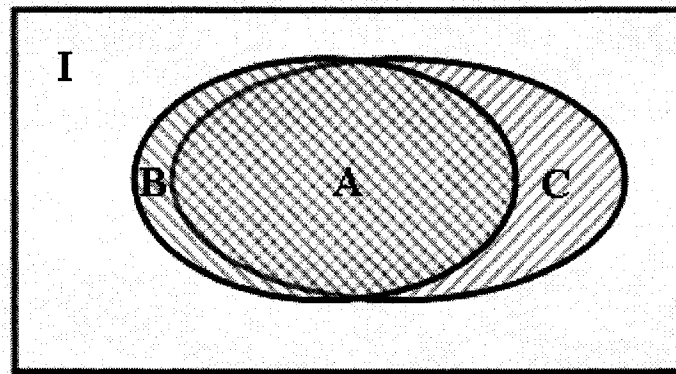


Figure 4.6 Measuring of retrieval effectiveness

(Source: [56])

In the CBIR, we often use precision and recall as a measure of evaluating the effectiveness of the retrieval engine. As shown is Figure 4.6, if rectangular I is the whole image collection, the left ellipse represents the set of the relevant images in the image collection against a given query image, and the right ellipse represents the set of images retrieved by the retrieval engine, then, area A represents the relevant images retrieved; area B is the relevant images not retrieved whereas area C is the irrelevant images retrieved.

The relevant images not retrieved are also known as “false negative”, and the irrelevant images retrieved are called “false positive”. We want to minimize “false negative” (B) as well as “false positive” (C). However, “false positive” is still acceptable in real image

retrieval system because they can be visually discarded by the user, whereas absence of “false negative” is vital to a retrieval system because the user may never know if there are images in the data set which have not been retrieved.

Precision and recall are defined as [56]:

$$\text{Precision} = \frac{\text{Number of Relevant Images Retrieved}}{\text{Number of Images Retrieved}} = \frac{|A|}{|A| + |C|}$$

$$\text{Recall} = \frac{\text{Number of Relevant Images Retrieved}}{\text{Total Number of Relevant Images in the Image Collection}} = \frac{|A|}{|A| + |B|}$$

from where we can see that “false negative” effects recall, and “false positive” effects precision.

Recall is a measure of how well the retrieval engine performs in returning relevant images, whereas precision is a measure of how well the retrieval engine performs by not returning the irrelevant images. Recall and precision are inversely related: Recall is 1.0 or 100% when every relevant image is retrieved, and precision is 1.0 or 100% when every image returned to the user is relevant to the query. In theory, it is easy to achieve good recall by just simply returning every image in the image collection for every query. Therefore, recall by itself is not a good measure of the quality of a retrieval engine. However, there is no easy way to achieve 100% precision other than in the trivial cases where no irrelevant image is ever returned for any query [56].

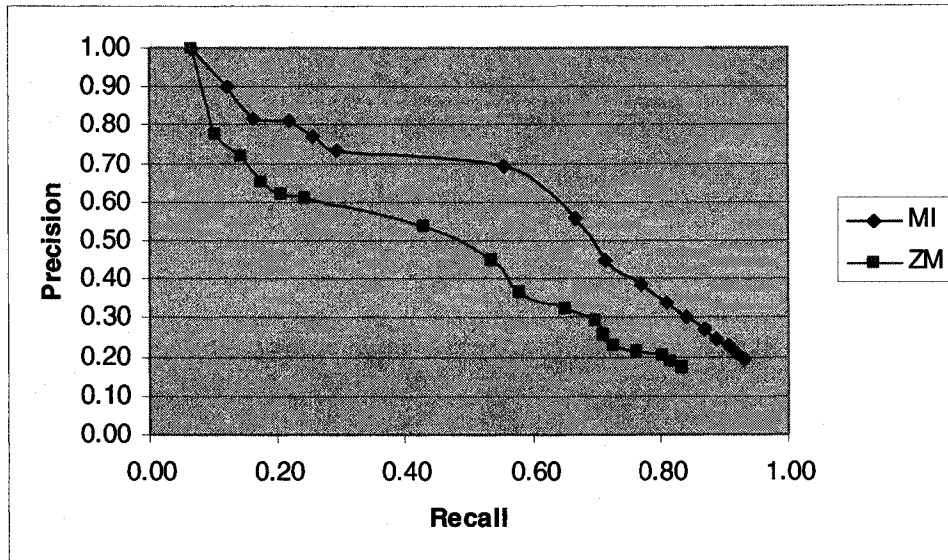


Figure 4.7 Precision-recall graph of Moment Invariants and Zernike Moments (1)

Figure 4.7 shows the precision-recall graph for Moment Invariants and Zernike Moments. The query images used in this experiment are part of training samples, and the retrieval precision achieved is as high as 100%.

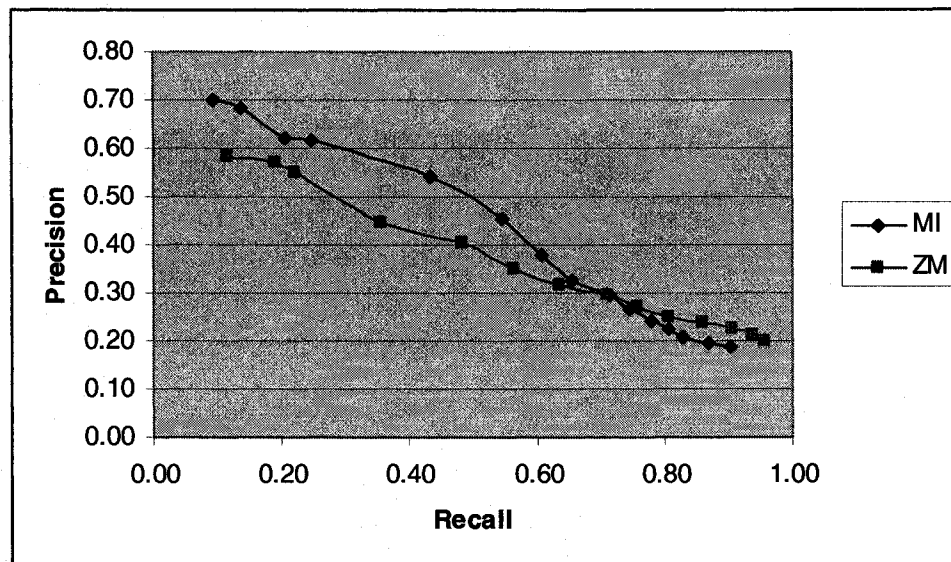


Figure 4.8 Precision-recall graph of Moment Invariants and Zernike Moments (2)

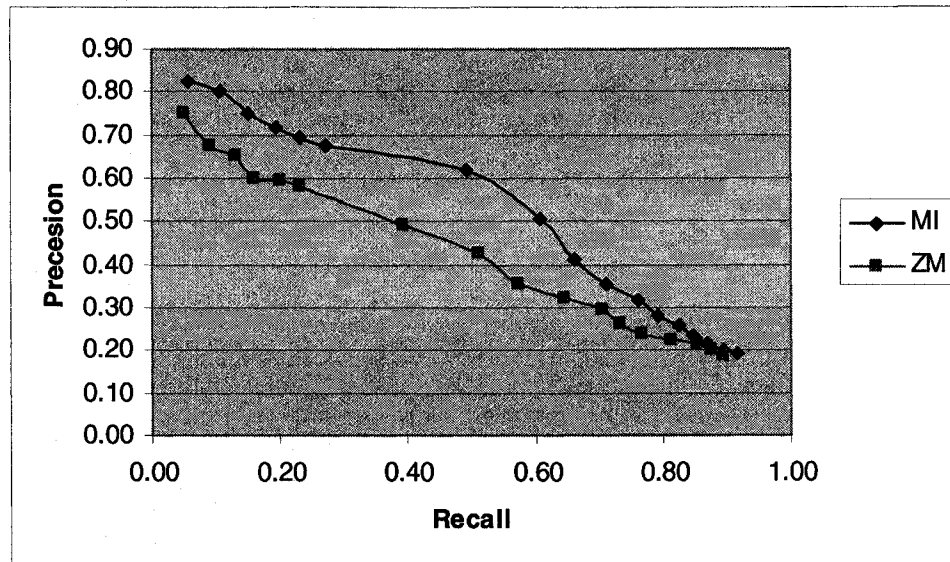


Figure 4.9 Precision-recall graph of Moment Invariants and Zernike Moments (3)

Figure 4.8 shows the precision-recall graph of the query images that the shape-based image retrieval system has never seen before and have not been part of training, and the retrieval precision reaches to about 70% for Moment Invariants, and about 58% for Zernike Moments.

The average result for precision and recall is shown in Figure 4.9 such that the shape retrieval using Moment Invariants reaches a precision of about 83%, and shape retrieval using Zernike Moments reaches a precision of about 75% in precision. More than 90% of relevant images can be retrieved by the system.

From above experimental results we can see that the Moments Invariants perform better than the Zernike Moments in terms of shape similarity retrieval for both precision and recall. Figures 4.10 and 4.11 illustrate some examples of shape retrieval results using Moment Invariants. The query image of example shown in Figure 4.10 is a new shape pattern to the retrieval system, whereas the query image of example shown in Figure 4.11 is a mirror image of one of the images in the database. Figure 4.12 and 4.13 are the retrieval results using Zernike Moments, for same exact query images.

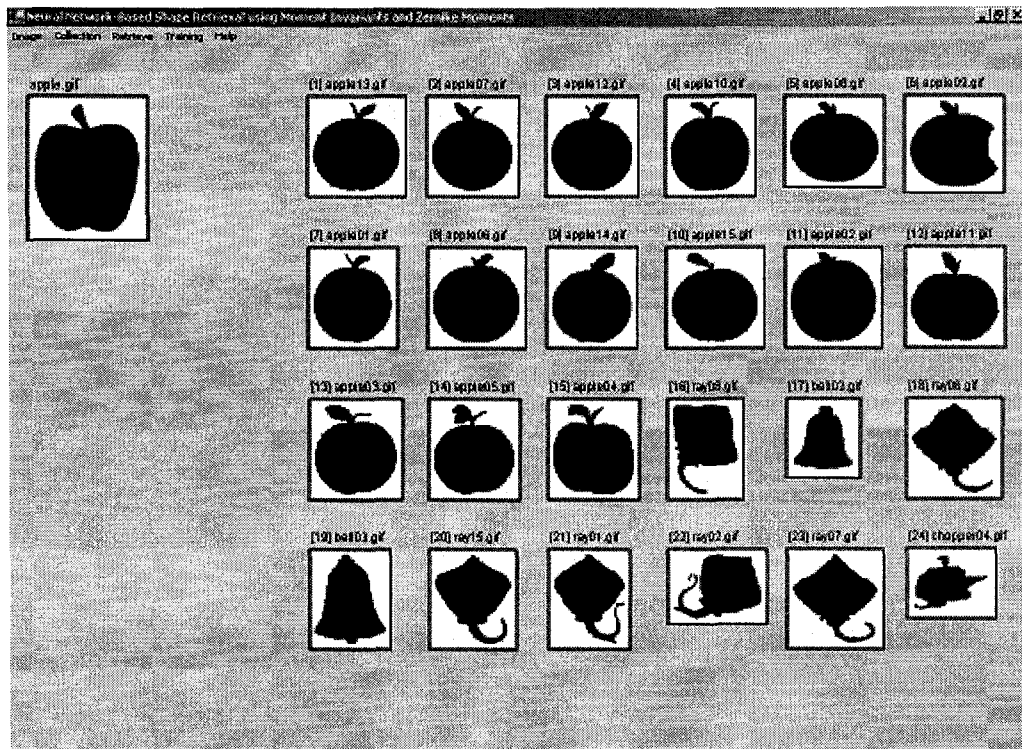


Figure 4.10 Example of the shape image retrieval results using Moment Invariants (1)

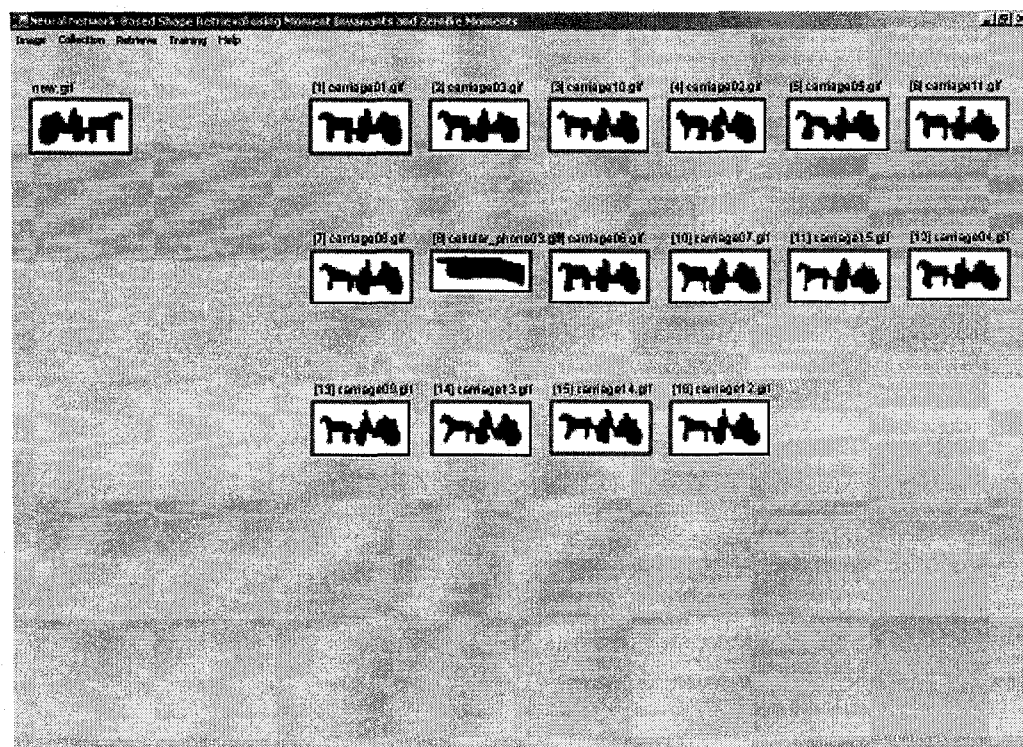


Figure 4.11 Example of the shape image retrieval results using Moment Invariants (2)

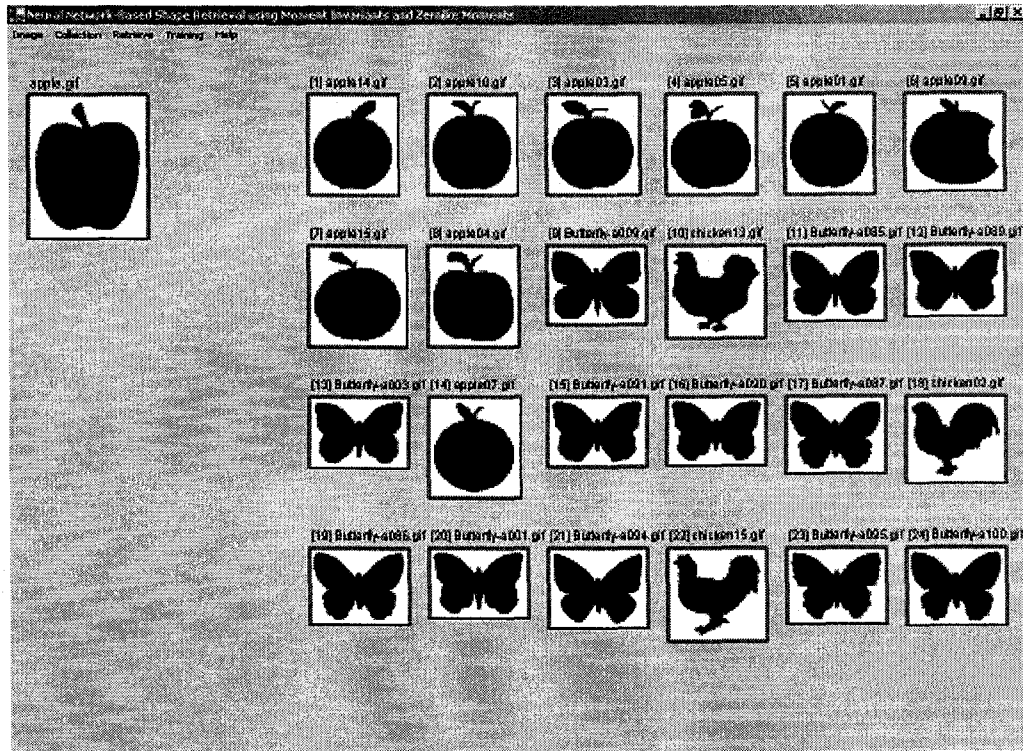


Figure 4.12 Example of the shape image retrieval results using Zernike Moments (1)

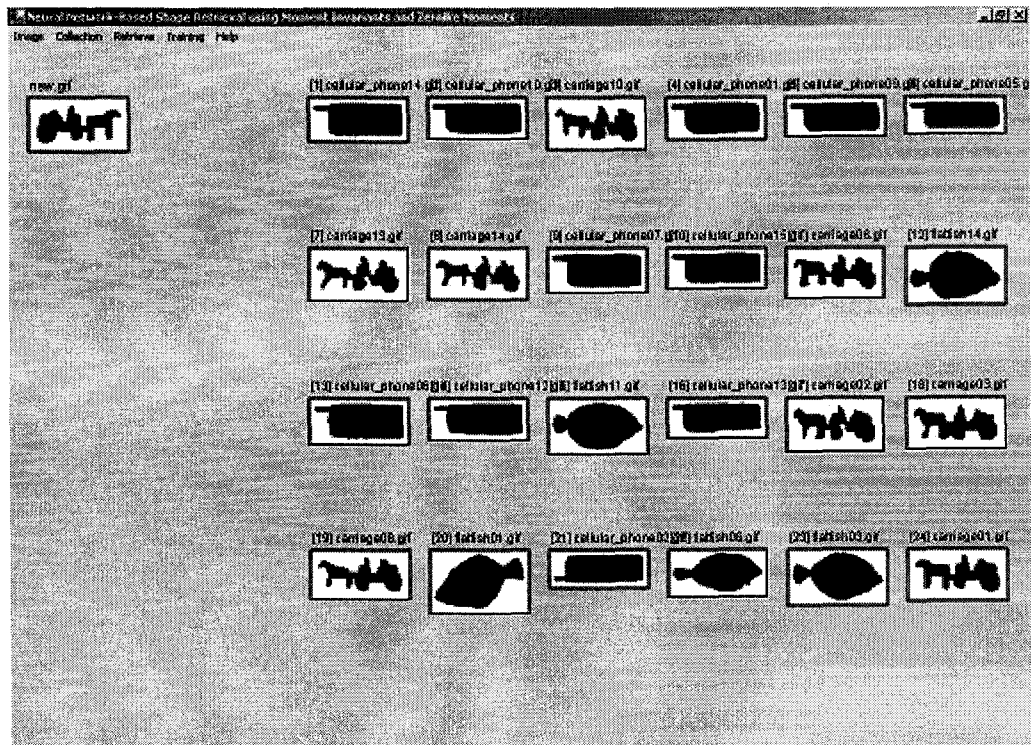


Figure 4.13 Example of the shape image retrieval results using Zernike Moments (2)

5 Conclusion and Future Directions

5.1 Conclusion

Shape is an important image feature and provides an ability to recognize individual objects in an image as well as for similarity based search and retrieval in Content-Based Image Retrieval. Because of inherent complexity associated with the shape description, this image feature is fairly less used than color and texture in image retrieval techniques but is still an active research area. Generally, in shape-based image retrieval, shape features of an image are first extracted as shape descriptors, and are represented by multidimensional vectors. These feature vectors are then used to measure the similarity of two images by calculating a descriptor distance in the feature space. Since the feature vectors are usually of high dimension, multidimensional indexing techniques are often employed for efficient and effective retrieval.

In this thesis, we have studied and compared two different types of moments: Moment Invariants and Zernike Moments. Generally speaking, there are two types of shape representation schemes: contour-based representations and region-based representations. In contour-based representations, the main emphasis is on the shape boundary whereas region-based representations emphasize on the interior region of a closed shape boundary. Moment Invariants and Zernike moments are both region-based representations schemes. Moment Invariants are robust under geometric transformations. The first six orders of Moment Invariants are translation and rotation independent for shape images and the sign of the seventh order of Moment Invariants can be used to detect mirror images. However, higher orders of Moments Invariants are very intricate to compute. Zernike Moments, on the other hand, are complex numbers and are orthogonal. We can calculate any order of Zernike Moments, but may not necessarily know the meaning of each of the value. Both Moment Invariants and Zernike Moments can uniquely represent a shape and, thus, provide visual features to facilitate shape-based image retrieval.

In this thesis, we have proposed use of k means clustering to organize the image collection. Shape images in the image database are grouped with the similarity of their

feature vectors. Neural Network serves as part of our retrieval engine. The clustering result of k means clustering on all the images in the image collection serves as the training samples of the Neural Network. After training, the Neural Network has the intelligence to assign a query image to the nearest clusters which contain the most similar images to it or recognize it as an unfamiliar shape. Although the training of the Neural Network is a time consuming process, we realize that training and retrieving are not symmetric and once the training is done, it can achieve higher retrieval efficiency and lower computational cost.

5.2 Future Direction

The shape-based image retrieval approach and application presented in this thesis can be further improved in the following ways:

- **Normalize the shape before feature computation:** Although Moment Invariants and Zernike Moments and other shape representations have some properties of remaining invariant under basic forms of planar shape distortions, there is a normalization algorithm called shape compacting [21] which can normalize a shape and its distorted versions so that they all become similar to each other. Therefore, after shape compacting, the moment-based shape descriptors may become more effective to geometric transformations.
- **Apply fuzzy k means clustering or Simulated Annealing for better training samples:** We used k means clustering algorithm in this thesis. It is a very well known clustering technique but may not necessarily be the most effective clustering method. Fuzzy k means clustering algorithm groups the data points into k clusters represented by a fuzzy membership matrix, which indicates a sample's degree of similarity to each cluster. Both k means and fuzzy k means clustering methods need preliminary knowledge on the number of clusters, k , which may not be feasible in many practical applications. Simulated Annealing [46], which has succeeded in grouping gene data, can be extended and applied to determine the optimal number of clusters k .

- **Accelerate the training procedure of the Neural Network:** Back-propagation training of Neural Network is a time consuming procedure and often takes hours or even days. Since MSE decreases very slowly in every training epoch, faster solutions are needed to improve the training algorithm while maintaining the same or higher prediction accuracy of the Neural Network.
- **Add user's feedback to amend the search engine:** new features such as relevant feedback [41] can be added to the shape-based image retrieval application to improve retrieval effectiveness.

References

- [1] I. Ahmad. "Image Indexing and Retrieval using Moment Invariants". In *Proceedings of the Fourth International Workshop on Information Integration and Web-based Applications and Services*, pp.93-104. Bandung, Indonesia. September 10-12, 2002.
- [2] M. Bober. "MPEG-7 Visual Shape Descriptors". In *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):716-719. June 2001.
- [3] V. Castelli, L. D. Bergman. *Image Databases – Search and Retrieval of Digital Imagery*. John Wiley & Sons, Inc. 2002. ISBN: 0471321168.
- [4] K. Chakrabarti, M. O. Binderberger, K. Porkaew, P. Zuo, S. Mehrotra. "Similar Shape Retrieval in MARS". In *Proceeding of IEEE International Conference on Multimedia and Expo*. New York City, NY, USA. July 30-August 2, 2000.
- [5] S. K. Chang, A. Hsu. "Image Information Systems: Where Do We Go From Here?". In *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431-442. October 1992.
- [6] M. Dai, P. Baylou, M. Najim. "An Efficient Algorithm for Computation of Shape Moments from Run-Length Codes or Chain Codes". In *Pattern Recognition*, 25(10):1119-1128. 1992.
- [7] L. Fausett. *Fundamentals of Neural Networks – Architectures, Algorithms, and Applications*. Prentice-Hall. 1994. ISBN: 0-13-334186-0.
- [8] H. Freeman. "On the Encoding of Arbitrary Geometric Configurations". In *IRE Transactions on Electronic Computers*, 10:260-268. 1961.

- [9] J. Flusser, T. Suk. "Pattern Recognition by Affine Moment Invariants". In *Pattern Recognition*, 26(1): 167-174. 1993.
- [10] Y. Gong. *Intelligent Image Databases: Towards Advanced Image Retrieval*. Kluwer Academic. 1998. ISBN: 0792380150.
- [11] G. H. Granlund. "Fourier Preprocessing for Hand Print Character Recognition". In *IEEE Transactions on Computers*, 21(2):195-201. February 1972.
- [12] R. C. Gonzalez, R. E. Woods. *Digital Image Processing*. Prentice Hall. 2002. ISBN: 0201180758.
- [13] T. H. Hou, M. D. Pern. "A Shape Classifier by using Image Projection and a Neural Network". In *International Journal of Pattern Recognition and Artificial Intelligence*, 14(2):225-242. 2000.
- [14] R. M. Haralick, K. Shanmugan, I. Dinstein. "Texture Features for Image Classification". In *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610-621. 1973
- [15] M. K. Hu. "Visual Pattern Recognition by Moment Invariants". In *IEEE Transactions on Information Theory*, 50:179-187. 1962.
- [16] R. Jain. "Workshop Report: NSF Workshop on Visual Information Management Systems". In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*. San Jose, CA, USA. February 1993.
- [17] R. Jain, A. Pentland, D. Petkovic. "NSF-ARPA Workshop Report: NSF-ARPA Workshop on Visual Information Management Systems". Boston, MA, USA. June 1995.

- [18] A. K. Jain, A. Vailaya. "Image Retrieval using Color and Shape". In *Pattern Recognition*, 29:1233-1244. August 1996.
- [19] T. Kato. "Database Architecture for Content-Based Image Retrieval". In *Proceedings of SPIE: Image Storage and Retrieval Systems*, 1662:112-123. 1992.
- [20] N. Katayama, S. Satoh. "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries". In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp.13-15. Tucson, Arizona. May 1997.
- [21] J. Leu. "Shape Normalization through Compacting". In *Pattern Recognition Letters*, 10:243-250. 1989.
- [22] R. P. Lippmann. "An Introduction to Computing with Neural Nets". In *IEEE Acoustics, Speech and Signal Processing Magazine*, 4(2):4-22. 1987.
- [23] S. Loncaric. "A Survey of Shape Analysis Techniques". In *Pattern Recognition*, 31(8):983-1001. 1998.
- [24] G. Lu, A. Sajjanhar. "Region-Based Shape Representation and Similarity Measure Suitable for Content-Based Image Retrieval". In *Multimedia Systems*, 7:165-174. 1999.
- [25] G. Lu. "Chain Code-Based Shape Representation and Similarity Measure". In *Proceedings of Conference on Visual Information and Information Systems*, pp.135-150. Melbourne, Australia. 1996.
- [26] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1:281-297. 1967.

- [27] S. Maitra. "Moment Invariants". In *Proceedings of the IEEE*, 67(4):697-699. April 1979.
- [28] U. Maulik, S. Bandyopadhyay. "Performance Evaluation of Some Clustering Algorithms and Validity Indices". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650-1654. 2002.
- [29] K. Mehrotra, C. K. Mohan, S. Ranka. "Elements of Artificial Neural Networks". The MIT Press. 1996. ISBN: 0-262-13328-8.
- [30] A. Y. Mustafa. "Boundary Signature Matching for Object Recognition". In *Proceedings of Vision Interface Annual Conference*. Ottawa, ON, Canada. June 7-9, 2001.
- [31] P. J. V. Otterloo. "A Contour-Oriented Approach to Shape Analysis". Prentice Hall. pp.90-108, 1991.
- [32] T. Pavlidis. "A Review of Algorithms for Shape Analysis". In *Computer Graphics and Image Processing*, 7:243-258. 1978.
- [33] E. Persoon, K. S. Fu. "Shape Discrimination using Fourier Descriptors". In *IEEE Transactions on Systems, Man, and Cybernetics*, 7(3):170-179. March 1977.
- [34] I. Pitas. *Digital Image Processing Algorithms*. Prentice Hall. 1993. ISBN: 0131458140.
- [35] Y. Rui, T. S. Huang, S. F. Chang. "Image Retrieval: Current Techniques, Promising Directions and Open Issues". In *Journal of Visual Communication and Image Representation*, 10(4):39-62. April 1999.

- [36] K. Roh, I. Kweon. "2D Object Recognition using Invariant Contour Descriptor and Projective Refinement". In *Pattern Recognition*, 31(4):441-455. 1998.
- [37] J. T. Robinson. "The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp.10-18. 1981.
- [38] R. Rickman, J. Stonham. "Similarity Retrieval from Image Databases – Neural Networks can Deliver". In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, 1908: 85-94. 1993.
- [39] J. R. Smith, S. F. Chang. "Tools and Techniques for Color Image Retrieval". In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, 2670:426-437. 1996.
- [40] D. M. Squire, T. M. Caelli. "Invariance Signature: Characterizing Contours by Their Departures from Invariance". In *Computer Vision and Image Understanding*, 77:284-316. 2000.
- [41] E. Di Sciascio, G. Mingolla, M. Mongiello. "Content-Based Image Retrieval over the Web Using Query by Sketch and Relevance Feedback". In *Proceedings of the Third International Conference on Visual Information and Information Systems*, pp.123-130. Amsterdam, the Netherlands. June 2-4, 1999.
- [42] M. Stricker, M. Orengo. "Similarity of Color Images". In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases III*, 2420:381-392. San Jose, CA, USA. February 1995.
- [43] M. Safar, C. Shahabi, X. Sun. "Image Retrieval by Shape: A Comparative Study". In *IEEE International Conference on Multimedia and Expo*, 1:141-144. 2000.

- [44] C. H. Teh, R. T. Chin. "On Image Analysis by the Methods of Moments". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4): 496-513. July 1988.
- [45] H. Tamura, N. Yokoya. "Image Database Systems: A Survey". In *Pattern Recognition*, 17(1):29-43. 1984.
- [46] W. Yang, L. Rueda, A. Ngom. "Determining the Number of Clusters on Time-Series Microarray Data by a Simulated Annealing". To be published.
- [47] D. Zhang, G. Lu. "Shape Retrieval using Fourier Descriptors". In *Proceedings of International Conference on Multimedia and Distance Education*. Fargo, ND, USA. June 2001.
- [48] D. Zhang, G. Lu. "Content-Based Shape Retrieval using Different Shape Descriptors: A Comparative Study". In *Proceedings of IEEE Conference of Multimedia and Expo*, pp.317-320. Tokyo, Japan. August 2001.
- [49] D. Zhang, G. Lu. "A Comparative Study of Three Region Shape Descriptors". In *Digital Image Computing Techniques and Applications*. Melbourne, Australia. January 21-22, 2002.
- [50] D. Zhang, G. Lu. "A Comparative Study of Fourier Descriptors for Shape Representation and Retrieval". In *Proceedings of the Fifth Asian Conference on Computer Vision*. Melbourne, Australia. January 23-25, 2002.
- [51] D. Zhang, G. Lu. "Enhanced Generic Fourier Descriptors for Object-Based Image Retrieval". In *Proceedings of IEEE the 2002 International Conference on Acoustics, Speech, and Signal Processing*, 4:3668-3671. Orlando, FL, USA. May 13-17, 2002.

- [52] D. Zhang, G. Lu. "Generic Fourier Descriptor for Shape-Based Image Retrieval". In *Proceedings of IEEE International Conference on Multimedia and Expo*, 1:425-428, 2002.
- [53] C. T. Zahn, R. Z. Roskies. "Fourier Descriptors for Plane Closed Curves". In *IEEE Transactions on Computers*, 21(3):269-281. March 1972.
- [54] M. F. Zakaria, L. J. Vroomen, P. J. A. Zsombor-Murray, J. M. H. M. Vankessel. "Fast Algorithm for the Computation of Moment Invariants". In *Pattern Recognition*, 20: 639-643. 1987.
- [55] "A Large Binary Image Database". The LEMS Vision Group, Brown University. URL: <http://www.lems.brown.edu/~dmc/> (2005)
- [56] K. Mahesh. "Text Retrieval Quality: A Primer". Oracle Corporation. URL: http://www.oracle.com/technology/products/text/htdocs/imt_quality.htm (2005)

Vita Auctoris

Xiaoliu Chen was born in 1980 in Beijing, P.R.China. She graduated from Beijing No. 4 High School in 1998. From there she went on to the Beijing Polytechnic University where she obtained a B.Eng. in Computer Science in 2002. She is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in Winter 2005.