

**SIMILARITY MEASURES FOR RETRIEVAL  
TEXTURAL IMAGES**

**By**

**LATHA THIYAGARAJAN**

**Bachelor of Engineering**

**Madurai Kamaraj University**

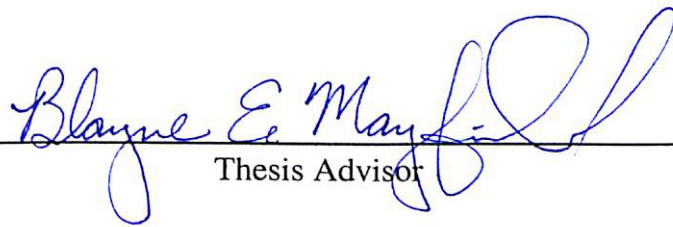
**Madurai, India.**

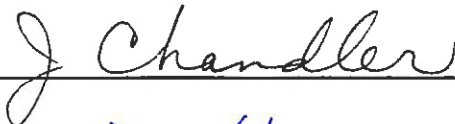
**1999**

**Submitted to the Faculty of  
the Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
May, 2004.**

SIMILARITY MEASURES FOR RETRIEVAL OF  
TEXTURAL IMAGES

Thesis Approved:

  
Thesis Advisor







Dean of the Graduate College

## ACKNOWLEDGEMENT

It gives me great pleasure to present my gratitude to Dr. B. Mayfield, my major advisor for the M.S. program. His guidance and encouragement was instrumental in completing my program successfully. I would like to thank my advisory committee members, Dr. G. Hedrick and Dr. J. Chandler for their critique on my research work. Appreciation is extended to OSU Computer Science Department for providing me teaching assistantship.

Without the motivation from my husband, Jeyamkondan Subbiah, it would have been tough to complete this study on time. During the course of this program, I was blessed with a baby boy, Vishal, who made my life wonderful. I am grateful to my mother, Valliammai Thiyagarajan who came from India to support me, while writing my thesis with a newborn baby.

## TABLE OF CONTENTS

1.	Introduction.....	1
1.1.	Content-based Image Retrieval.....	2
1.2.	Objectives .....	3
2.	Review of Literature .....	4
2.1.	Spatial Domain.....	4
2.2.	Frequency Domain.....	5
2.3.	Linear Space Invariant filter .....	8
2.4.	Gabor Filter.....	12
3.	Materials & Methods .....	13
3.1.	Gabor filter.....	13
3.2.	Similarity Measures .....	22
3.2.1.	Minkowsky Distance .....	22
3.2.2.	Mahalanobis Distance.....	23
3.3.	Normalization .....	25
3.3.1.	Linear scaling to unit range.....	25
3.3.2.	Linear scaling to unit variance.....	25
3.3.3.	Rank normalization.....	25
3.3.4.	Transformation to a Uniform [0, 1] random variable .....	26
3.3.5.	Fitting a Normal ( $\mu, \sigma^2$ ) density.....	26
3.3.6.	Fitting a Lognormal ( $\mu, \sigma^2$ ) density .....	27
3.3.7.	Fitting an Exponential ( $\mu$ ) density .....	27
3.3.8.	Fitting an Gamma ( $\alpha, \beta$ ) density .....	28
3.4.	Principal component analysis .....	29
3.4.1.	Principal component scores .....	33
3.4.2.	Determining the Number of Principal Components .....	33
3.4.3.	Steps in implementation of PCA.....	34
3.5.	Evaluation Procedure .....	35
4.	Results & discussion.....	39
4.1.	Minkowsky Distance Measure.....	39

4.1.1.	Without normalization .....	39
4.1.2.	Normalization to unit range .....	40
4.1.3.	Linear scaling to unit variance .....	40
4.1.4.	Rank normalization.....	42
4.1.5.	Transformation to a Uniform [0, 1] random variable .....	43
4.1.6.	Fitting a Normal Density .....	43
4.1.7.	Fitting a LogNormal Density .....	44
4.1.8.	Fitting an Exponential Density .....	45
4.1.9.	Fitting a Gamma Density .....	47
4.1.10.	Principal Component Analysis .....	47
4.2.	Mahalanobis Distance .....	52
4.3.	Comparison of Similarity Measures .....	53
4.4.	Graphical User Interface .....	57
4.4.1.	Testing with Brodatz image database .....	57
4.4.2.	Testing with new textural images .....	65
4.4.3.	Testing with new non-textural images.....	69
5.	Conclusions.....	74
6.	References.....	756
	Appendix A.....	79

## LIST OF FIGURES

Figure 1. A representation of a digital image (Adapted from Smith, 1997).....	4
Figure 2. Demonstration of Fourier idea. The function at the bottom is the sum of four sine functions above it (Adapted from Gonzales and Woods, 2002).....	6
Figure 3. Image representation in spatial and frequency domains.....	7
Figure 4. Impulse response of a LSI filter. ....	9
Figure 5. Convolution operation. ....	10
Figure 6. Frequency response of a low-pass filter (Adapted from Gonzalez and Woods, 2002). ....	10
Figure 7. Basic steps of implementing a filter .....	11
Figure 8. Impulse and frequency response of Gabor filter ( $K=3; S=3; n=0; m=0,1,2$ ). .	15
Figure 9. Frequency response of the Gabor filter ( $K=3; S=3; n=0; m=0$ ) .....	17
Figure 10. Frequency response of the Gabor filter ( $K=3; S=3; n=0; m=1$ ) .....	17
Figure 11. Frequency response of the Gabor filter ( $K=3; S=3; n=0; m=2$ ) .....	18
Figure 12. Impulse and frequency response of Gabor filter ( $K=3; S=3; n=1; m=0,1,2$ ). 19	
Figure 13. Impulse and frequency response of Gabor filter ( $K=3; S=3; n=2; m=0,1,2$ ). 19	
Figure 14. Frequency coverage by Gabor filters ( $K=3, S=3$ ) .....	20
Figure 15. Gabor filters ( $K=6; S=4$ ). Adapted from Manjunath and Ma (1996). ....	20
Figure 16. Basic steps of filtering using a Gabor filter.....	21
Figure 17. Illustration of differences between Euclidean and Mahalanobis distances ....	24

Figure 18. Image of tree bark from Brodatz album. ....	36
Figure 19. Image of a straw from Brodatz album. ....	37
Figure 20. An image from Brodatz album. ....	37
Figure 21. An image of a cork from Brodatz album. ....	38
Figure 22. Performance of Minkowsky distance measure without normalization. ....	39
Figure 23. Performance of Minkowsky distance measure with 'minmax' normalization. .....	41
Figure 24. Performance of Minkowsky distance measure with 'meanstd' normalization. .....	41
Figure 25. Performance of Minkowsky distance measure with 'rank' normalization. ....	42
Figure 26. Performance of Minkowsky distance measure with 'uniform' transformation. .....	43
Figure 27. Performance of Minkowsky distance with normalization using normal distribution. ....	44
Figure 28. Performance of Minkowsky distance with normalization using lognormal distribution. ....	45
Figure 29. Performance of Minkowsky distance with normalization using exponential distribution. ....	46
Figure 30. Performance of Minkowsky distance with normalization using gamma distribution. ....	46
Figure 31. Scree plot for evaluating number of principal components. ....	48
Figure 32. Cumulative percent variation explained by PCs. ....	48
Figure 33. Eigenvectors for PC1 and PC2. ....	49

Figure 34. Loading vectors for PC1 and PC2.....	50
Figure 35. Minkowsky measure with PCA preprocessing with 4 PCs.....	51
Figure 36. Minkowsky measure with PCA preprocessing with 17 PCs.....	51
Figure 36. Performance of Mahalanobis distance measure .....	52
Figure 37. Image retrieval for test image of wood grain .....	57
Figure 38. Image retrieval for test image of fieldstone.....	58
Figure 39. Image retrieval for test image of reptile skin.....	58
Figure 40. Image retrieval for test image of crocodile skin.....	59
Figure 41. Image retrieval for test image of bark of a tree. ....	59
Figure 42. Image retrieval for test image of ice crystals on an automobile.....	60
Figure 43. Original image of ice crystals on an automobile.....	61
Figure 44. Original image of beach sand.....	61
Figure 45. Image retrieval for test image of varied swinging of light bulb.....	62
Figure 46. Original image of varied swinging of light bulb.....	62
Figure 47. Original image of swinging lights in the darkened room.....	63
Figure 48. Original image of abstract effect of swinging lights.....	63
Figure 49. Image retrieval failure for the test image of Japanese rice paper.....	64
Figure 49. Image retrieval failure for the test image of handmade paper.....	64
Figure 50. Image retrieval for a real test image of square textural pattern.....	65
Figure 51. Image retrieval for a real test image of horizontal line textural pattern.....	66



Figure 52. Image retrieval for a real test image of pebble textural pattern.....	67
Figure 53. Image retrieval for a real test image of brick textural pattern. ....	67
Figure 54. Image retrieval for a real test image of capsule textural pattern. ....	68
Figure 55. Image retrieval for a test image of a tree.....	68
Figure 56. Image retrieval for a real test image of a sword. ....	69
Figure 57. Image retrieval for a real test image of an aeroplane. ....	70
Figure 58. Image retrieval for a real test image of a vertical sword. ....	71
Figure 59. Image retrieval for a real test image of the American flag.....	71
Figure 60. Image retrieval for a real test image of the Statue of Liberty.....	72
Figure 61. Image retrieval for a real test image of a cat. ....	72
Figure A1. Illustration of convolution (Adapted from Gonzalez and Woods, 2002). ....	80

## LIST OF TABLES

Table 1. Effect of normalization of features for Minkowsky distance on performance of image retrieval.....	55
Table 2. Comparison of performance of image retrieval with literature.....	56

## 1. INTRODUCTION

Text-based queries are commonly used for Internet searches. Searching for images on the Internet is gaining interest. Currently, many images in the Internet have been indexed with a few keywords either manually or automatically. For example, the URL <http://www.cs.okstate.edu/~bem/graphics/teapot.jpg> would have keywords graphics, teapot, and okstate. In Internet search engines such as Google, when a keyword 'teapot' is entered to search images in the database, this URL could be retrieved.

A picture cannot be explained completely by keywords. Retrieval of images based on pictorial queries is gaining interest as multimedia image libraries are increasing day-by-day. This is a challenging task because two pictures can be considered similar even if their resolutions and lighting conditions are different. Even if all the images are resized to the same resolution, comparing millions of images pixel-by-pixel is a daunting task and is computationally prohibitive. Typically, features based on color, shape, and texture are identified to describe the content of an image. It is envisioned that such features will be extracted from most of the images available in the web, and stored in a database. When an image is queried, the same features are extracted from the queried image and these features then are compared with the features of the images present in the database. Images with features closest to the features of the queried image are retrieved. This is called content-based image retrieval (CBIR).

Another application of CBIR is for quick retrieval of similar images in a large image database based on a sample image. For example, NASA has a large image database of satellite remote-sensing images. A researcher might want to buy a set of

images similar to a sample image with desired texture characteristics.

### **1.1. Content-based Image Retrieval**

Image color, shape of an object in the image, and image texture are all features that can describe the content of an image (Smeulders et al., 2000). To describe image color, histogram features are extracted. The histogram describes the distribution of pixel values (gray-levels) in an image (Gonzales and Woods, 1993). However for many applications, users want to retrieve similar images irrespective of color and lighting conditions. The shape of a predominant object in an image can be a valuable feature for retrieving images. This is complicated because the prominent object first must be identified in each and every image. A segmentation algorithm can identify objects in an image. Although there are several segmentation algorithms available, the input parameters for the algorithms must be fine-tuned for each type of image to achieve good segmentation. In an image retrieval application, there are several type of images encountered. Also object in each image differ from image to image. Typically, segmentation algorithms are also computationally intensive. Due to these factors, identifying a predominant object in an image is complex. Image texture has been used successfully to retrieve images (Manjunath and Ma, 1996; Li and Castelli, 1997).

Texture is primarily related to the frequency content of an image. An image from the spatial domain can be converted to the frequency domain using the Fourier transform (FT). The spatial domain (original image) contains complete spatial information, but the frequency information is not obvious. The FT identifies all spectral components of an image; however it does not contain spatial information. The FT analyzes the image

globally rather than locally. Gabor filters (Manjunath and Ma, 1996) can analyze the image in both the spatial and the frequency domains. Principal component analysis (Johnson, 1996) can be used to preprocess the correlated set of textural features into a smaller set of new uncorrelated features called principal components. Various similarity measures such as Euclidean distance, Manhattan distance, Mahalanobis distance, can be used to retrieve the similar images in the database. These similarity measures are defined in the “Materials & Methods” chapter (Chapter 3).

## **1.2. Objectives**

A Gabor filter was implemented to extract textural features. The objective of this study was to compare similarity measures for image retrieval based on textural features extracted from a Gabor filter. Principal component analysis was used as a preprocessing step for reducing the dimension of textural features. This preprocessing step was evaluated based on accuracy of image retrieval and computation time. Various normalization techniques were applied before implementing Minkowsky similarity measures. The effects of normalization of textural features before applying similarity measures were evaluated.

## 2. REVIEW OF LITERATURE

### 2.1. Spatial Domain

A gray-scale digital image can be represented as a two-dimensional matrix,  $f(i, j)$ , where  $i$  and  $j$  indicate the rows and columns of the image. Figure 1 shows a digital image of the earth. The resolution of this image is  $200 \times 200$ . Each element in this matrix is called picture element, which is shortened as a pixel. If a part of this image is zoomed, the matrix representation can be seen clearly (Figure 1). The pixel value represents the gray-scale (intensity). In a photographic (analog) image, the gray tones are continuous. When this image is digitized using a scanner, the gray tones are quantized into discrete gray-levels. An 8-bit quantization would produce  $2^8=256$  gray-levels. An 8-bit image matrix,  $f$ , will have values ranging from 0-255, where 0 represents 'black' and 255 represents 'white'.

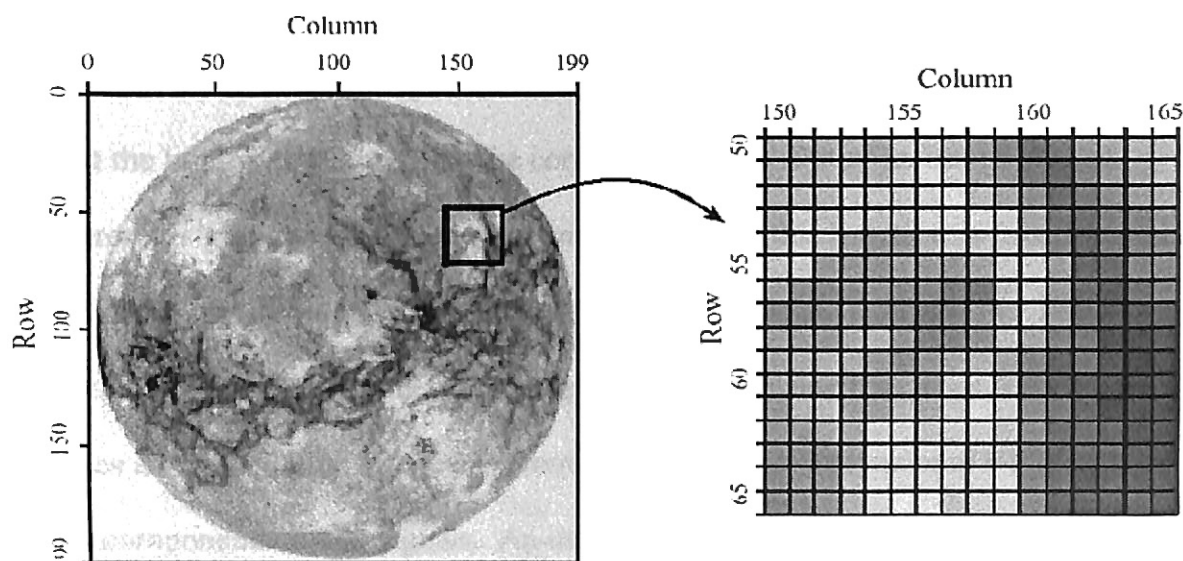


Figure 1. A representation of a digital image (Adapted from Smith, 1997)

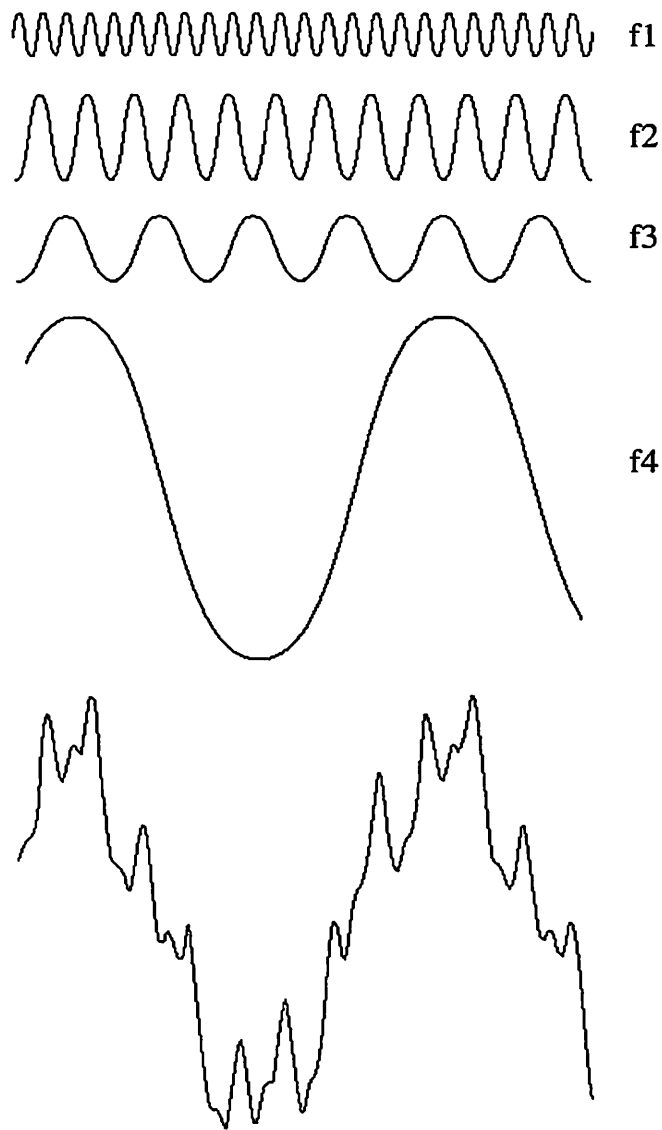
A color image can be represented by a 3-D matrix, which is an extension of a 2-D gray-

scale image. The third dimension consists of three colors such as 'red', 'green', and 'blue' (RGB). The three primary light colors, RGB, can be mixed to produce millions of colors. In this study, only gray-scale images are considered.

## 2.2. Frequency Domain

This matrix type of image representation using gray-level values is called spatial domain representation. An image also can be represented by the Fourier or frequency domain that is useful in many image processing applications. In December 1807, a French mathematician, Jean B. Joseph Fourier published a revolutionary concept that periodic functions could be represented as a weighted sum of sines and cosines. Figure 2 shows a function that is the sum of 4 sine functions with 4 distinct frequencies  $f_1, f_2, f_3, f_4$ . Note that  $f_1 > f_2 > f_3 > f_4$ . A frequency represents rate of change or number of cycles per unit distance. The top function has a higher frequency than other functions.

Similar to a spatial domain having spatial axes  $x$  and  $y$ , the Fourier domain has frequency axes  $u$  and  $v$ . For simplicity, let us consider a one-dimensional function. The function at the bottom of Figure 2 can be considered as one row of an image whose values represents gray-levels. If Fourier transform (FT) of this function is taken, nonzero values will be obtained only at 4 points in the frequency axis at  $f_1, f_2, f_3, f_4$ . The value of those points would be the same as the amplitude of those sine functions. Therefore FT decomposes an image into its frequency components. A one-dimensional FT identifies frequency components in 1-D signals. An image can be considered as a 2-D signal. A 2-D FT converts an image from the 2-D spatial domain to 2-D Fourier domain, which is given by:



**Figure 2. Demonstration of Fourier idea. The function at the bottom is the sum of four sine functions above it (Adapted from Gonzales and Woods, 2002).**

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp\left(-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right) \quad (2.1)$$

where:  $f(x, y)$  is the pixel value of the image,  $f$  at coordinates  $(x, y)$ ,

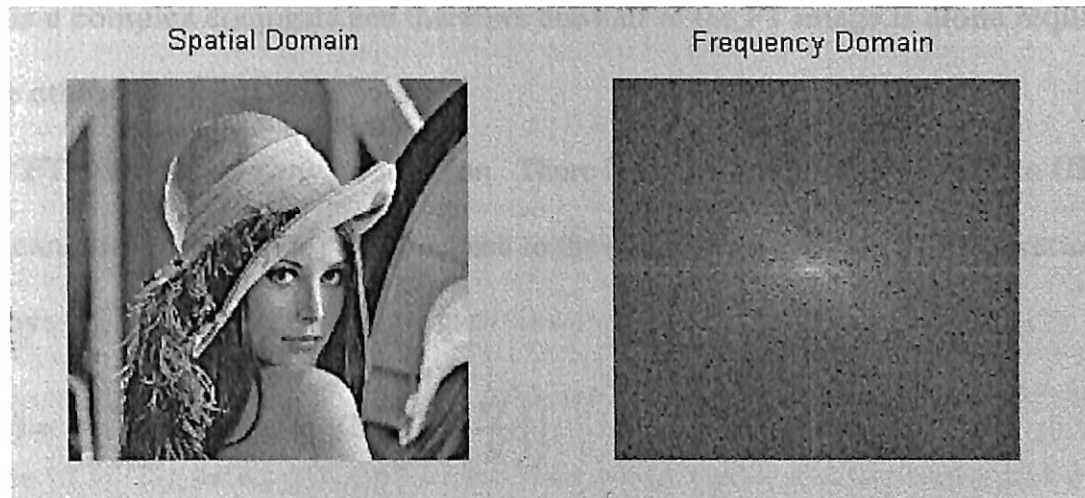
$F$  is the FT of the image with new coordinates  $(u, v)$ ,



$j$  is the imaginary number  $= \sqrt{-1}$ , and

$M \times N$  is the resolution of the image  $f$ .

A 2-D FT consists of  $u, v$  axes, which are also defined as horizontal and vertical frequency axes. Figure 3 shows an image and its FT.



**Figure 3. Image representation in spatial and frequency domains.**

The central part of the Fourier-transformed image represents lowest frequency content, whereas the corners of the image represent the highest frequency content image. Let 'F' be the FT of the image. The central value, represent zero frequency or a constant, is the mean value of all gray-level values of the image in the spatial domain,  $B$ . A high frequency of the image represents fast-changing details, which is fine texture, whereas low frequency content represents coarse texture.

The basis or kernel functions of FT are complex exponentials (sinusoids). As sines and cosines are infinite in length, FT analyzes the signal globally. Another way of understanding the FT is that the FT coefficients give the correlation of the image with cosines and sines of various frequencies. A higher coefficient indicates the higher

presence of that frequency in the image, and vice versa.

FT coefficients are complex; they have both real and imaginary parts.

Commonly, the magnitude and phase are calculated. In FT literature, the magnitude and phase matrix is called magnitude and phase spectrum, respectively. The FT of a real image is a complex conjugate and therefore one half of the FT image is alone required and the other half is redundant.

FT does not lose any information. There is an inverse Fourier transform (IFT), which can convert frequency domain back to the spatial domain. The IFT operation is given by:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp\left(j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right) \quad (2.2)$$

### 2.3. Linear Space Invariant filter

A digital filter can be used to filter certain frequency content in an image. For instance, a high-pass filter attenuates low frequency content and passes high frequency content of an image. The converse is true for a low-pass filter. A band-pass filter passes only a certain band of frequencies. A digital filter is called a linear filter when it satisfies the following conditions:

$$H(f+g) = H(f)+H(g) \quad (2.3)$$

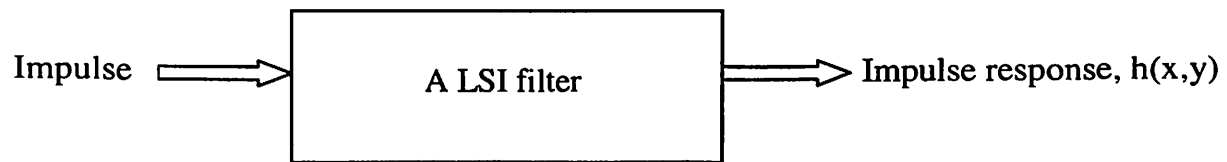
$$H(cf) = cH(f) \quad (2.4)$$

where:  $H(f)$  and  $H(g)$  are the outputs of a linear filter on images  $f$  and  $g$ , respectively, and  $c$  is a constant.

Equation 2.3 defines the *additivity* property; if two images were added and passed through the filter, the output image is equal to the sum of the filtered first image and the filtered second image. This implies that the property of the filter is independent of the input image. Equation 2.4 defines the *homogeneity* property; if the input image is scaled, the output image is also scaled by the same factor. Equations 2.3 and 2.4 can be combined into one equation, as follows:

$$H(cf+dg) = cH(f)+dH(g) \quad (2.5)$$

A space-invariant filter means that the filter characteristics do not vary with the location within the image. A linear space-invariant (LSI) filter can be completely characterized by the *impulse response*. The impulse response is the response of the filter to an *impulse* (Figure 4). An impulse is a matrix with a value 1 at the center and 0 at all other places.

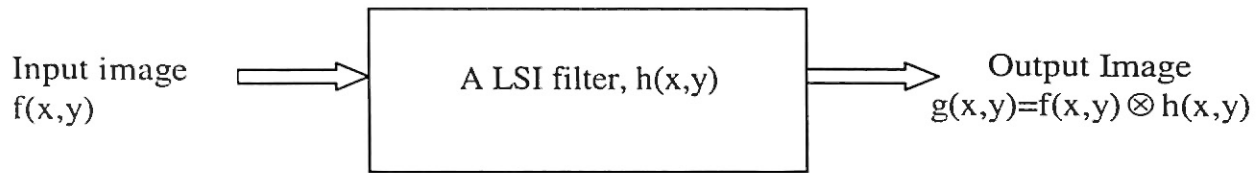


**Figure 4. Impulse response of a LSI filter.**

The output image of the LSI filter is given by the convolution of the impulse response of the filter with the input image (Figure 5).

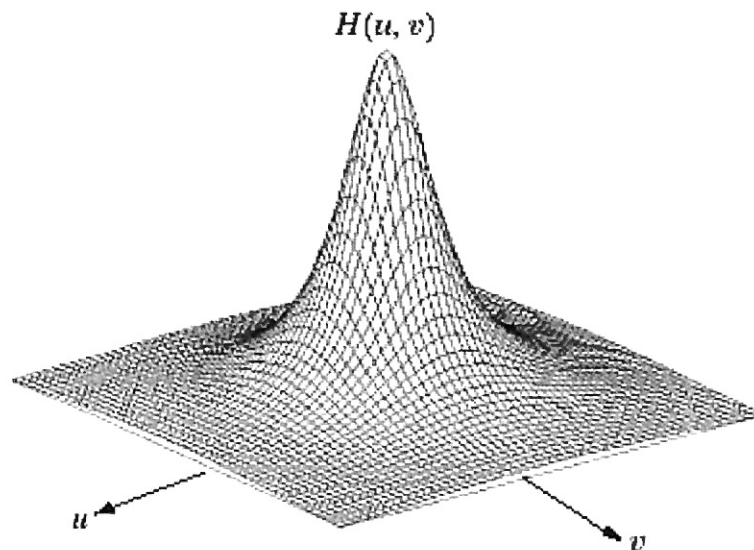
$$g = h ** f \quad (2.6)$$

where:  $g$  is the output image from the filter,  $f$  is the input image to the filter,  $h$  is the impulse response of the filter, and  $**$  is the two-dimensional convolution operation.



**Figure 5. Convolution operation.**

The concept of the convolution can be seen in image or signal processing textbooks such as Oppenheim and Schaffer (1991). The convolution operation is conducted in the spatial domain. Convolution is computationally intensive and is explained in Appendix A. Convolution in a spatial domain is equivalent to multiplication in the frequency domain. Therefore, convolution is usually implemented in the frequency domain. If we take the FT of the impulse response of a filter, we get frequency response of the filter. The frequency response indicates how the frequency content in the input image is attenuated or magnified.



**Figure 6. Frequency response of a low-pass filter (Adapted from Gonzalez and Woods, 2002).**

Figure 6 shows the frequency response of a Gaussian-shaped, low-pass filter. When this filter response is multiplied with frequency content of an image, the low frequency (central portion) is passed whereas the high frequency is attenuated. Convolution in the spatial domain (Eqn. 2.5) is equivalent to the following equation in the frequency domain.

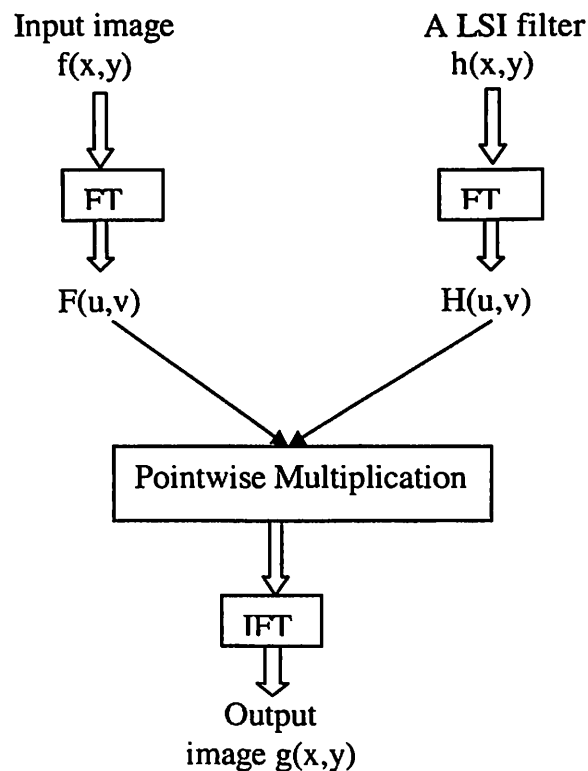
$$G(u,v) = H(u,v).F(u,v) \tag{2.7}$$

Where  $G$  = FT of the output image,  $g$

$H$  = frequency response of the LSI filter, which is FT of impulse response,  $h$

$F$  = FT of the input image,  $f$ .

If we take the IFT of the resultant, we would get the filtered image (Figure 7). The basic steps of implementing a filter are shown in Figure 7.



**Figure 7. Basic steps of implementing a filter**

## 2.4. Gabor Filter

The spatial domain (original image) contains complete spatial information, but the frequency information is not obvious. The FT identifies all spectral components of an image; however, it does not contain spatial information. As previously indicated, FT analyzes the image globally rather than locally. In order to obtain spatial localization of frequency components, the image must be analyzed locally. To meet this requirement, short-time Fourier transform (STFT) was introduced. An image was localized spatially by multiplying the image by a *window*. A window is a two-dimensional function that is multiplied pointwise (element-by-element) to the image. By multiplying an image with a window, the specific region in the image can be given more weight than other regions. The resultant image then is Fourier-transformed to get the frequency component of that spatial-localized image. Various windows including rectangular, triangular, or Gaussian can be used. When the window is a Gaussian, the STFT is also known as the Gabor Transform, from its inventor's name, Dennis Gabor (Misiti et al., 1996). Gabor transform is implemented by a bank of band-pass filters; the name "Gabor filter" is commonly used in the literature.

Manjunath and Ma (1996) designed Gabor filters in the spatial domain. By specifying the number of orientations and scales, their algorithm divides the FT space by the filter frequency responses of a bank of band-pass filters. Mean and standard deviation of the filtered images were used as features. Manjunath and Ma (1996) demonstrated the application of a Gabor filter for image retrieval. Details and implementation of a Gabor filter are explained in the next section, 3.1.

### 3. MATERIALS & METHODS

#### 3.1. Gabor filter

The algorithm for the Gabor filter described in Manjunath and Ma (1996) was implemented as part of this research in Matlab 6.1. (Mathworks, Natick, MA). The band-pass filters are designed in the spatial domain, as given by:

$$h(x, y) = a^{-m} \left( \frac{1}{2\pi\sigma_x\sigma_y} \right) \exp \left( -\frac{1}{2} \left( \frac{(x')^2}{\sigma_x^2} + \frac{(y')^2}{\sigma_y^2} \right) + j2\pi x' U_h \right) \quad (3.1)$$

where:  $h(x,y)$  is the impulse response of the band-pass filter

$x'$  and  $y'$  are oriented/ rotated coordinates

$$x' = a^{-m} \left( x \cos \left( \frac{n\pi}{K} \right) + y \sin \left( \frac{n\pi}{K} \right) \right) \quad (3.2)$$

$$y' = a^{-m} \left( -x \sin \left( \frac{n\pi}{K} \right) + y \cos \left( \frac{n\pi}{K} \right) \right) \quad (3.3)$$

where:  $K$  = number of orientations

$S$  = number of scales

$m = 0, 1, 2, \dots, S-1$

$n = 0, 1, 2, \dots, K-1$

$$a = \left( \frac{U_h}{U_l} \right)^{\frac{1}{S-1}} \quad (3.4)$$

where:  $U_h$  – center frequency of Gaussian of the largest scale ( $m=K-1$ ) along u-axis

$U_l$  – center frequency of Gaussian of the smallest scale ( $m=0$ ) along u-axis

The variance of Gaussian in the frequency domain was selected such that all bank

filters covers most of the frequency spectrum.

$$\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}} \quad (3.5)$$

$$\sigma_v = \frac{\tan\left(\frac{\pi}{2K}\right)\left(U_h - 2\ln\left(\frac{2\sigma_u^2}{U_h}\right)\right)}{\sqrt{2\ln 2 - \frac{(2\ln 2)^2 \sigma_u^2}{U_h^2}}} \quad (3.6)$$

Then, the variance of Gaussians in the spatial domain is given by:

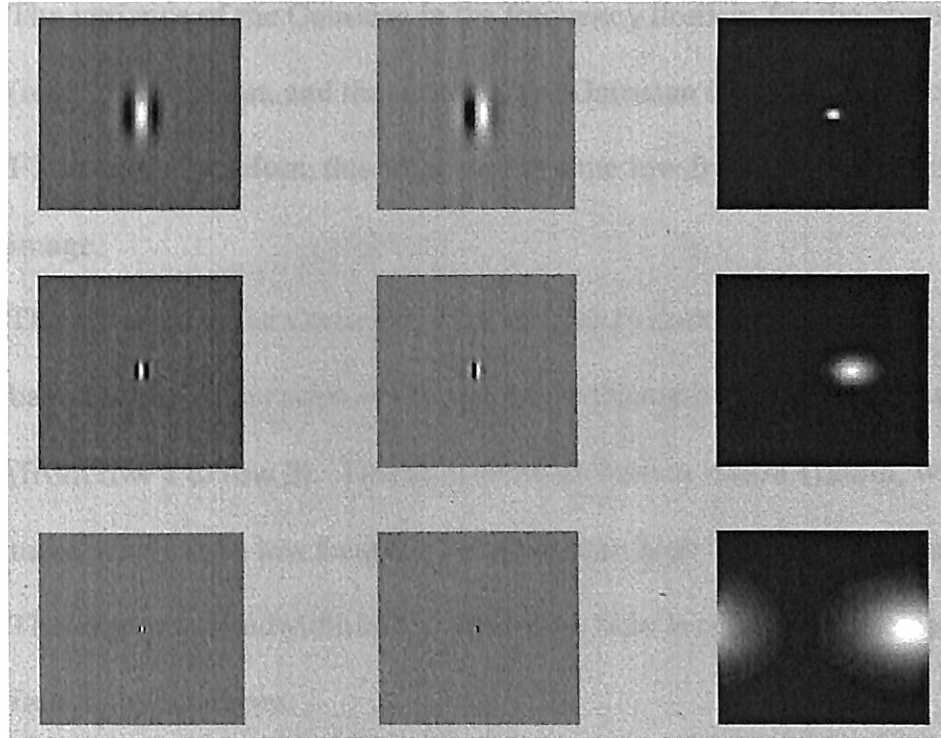
$$\sigma_x = \frac{1}{2\pi\sigma_u}, \sigma_y = \frac{1}{2\pi\sigma_v} \quad (3.7)$$

The kernel function of FT is a complex exponential (Eqn. 2.1), whereas the kernel function of a Gabor filter is a Gaussian function modulated by a complex exponential (Eqn. 3.1). An interesting property of the Gaussian is that the FT of the Gaussian is also a Gaussian (Oppenheim and Schaffer, 1991). Therefore, the Gaussian window is continuous in both spatial and frequency domain. When the Gaussian filter is rotated in the spatial domain (Eqn. 3.2, Eqn. 3.3), the Gaussian in the frequency domain is also rotated. The Variances of Gaussians in spatial and frequency domains are inversely related (Eqn. 3.7). When a Gaussian is modulated (multiplied) by a complex exponential in the spatial domain, the Gaussian in the frequency domain is shifted (added to) by the phase of the exponential. To illustrate Gabor filter design, a Gabor filter of 3 scales and 4 orientations is designed.

Figure 8 shows the Gabor filters for orientation 1 and 3 scales. The first row in Figure 8 represents the coarse scale, whereas the last row represents the fine scale, and



the middle row represents the medium scale. The first and second columns represent real and imaginary parts of the impulse response of the Gabor filter; whereas, the third column represents the frequency response of the Gabor filter.



**Figure 8. Impulse and frequency response of Gabor filter ( $K=3$ ;  $S=3$ ;  $n=0$ ;  $m=0,1,2$ ).**

The following observations can be made from Figure 8.

1. Both impulse and frequency responses are Gaussian shaped.
2. The impulse response is a Gaussian modulated by an exponential. The real part is the Gaussian modulated by a cosine, whereas the imaginary part is the Gaussian modulated by a sine.
3. The frequency response is also a Gaussian, but shifted by the amount equal to the frequency of the exponential which modulated the impulse response.
4. The variance of Gaussian in the frequency domain is inversely related to the

variance of the Gaussian in the spatial domain.

5. The variance of the Gaussian in the spatial domain for coarse scale (row 1) is largest, whereas that for fine scale (row 3) is smallest.
6. The variance of the Gaussian in the frequency domain for the coarse scale (row 1) is smallest, and the center of the Gaussian is closer to the center of the FT image. Therefore, this filter captures the low frequency content of the image.
7. The variance of the Gaussian in the frequency domain increases in log to the base 2 scale, as the scale or the central frequency of the Gaussian increases (from row 1 to row 3). This is to simulate human vision system, which is more sensitive to low frequency content than high frequency content. Therefore the bandwidth of the band-pass filter increases, as the central frequency increases.

The third column in Figure 8 is the image display of the frequency response of the Gabor filters. The gray-level corresponds to the amplitude of the frequency response. Another way of visualizing is to look at 3-D mesh plot. Figures 9-11 show the frequency response for coarse, medium, and fine scale. These frequency responses are band-pass in nature. The response shape is Gaussian and the surface is smooth.

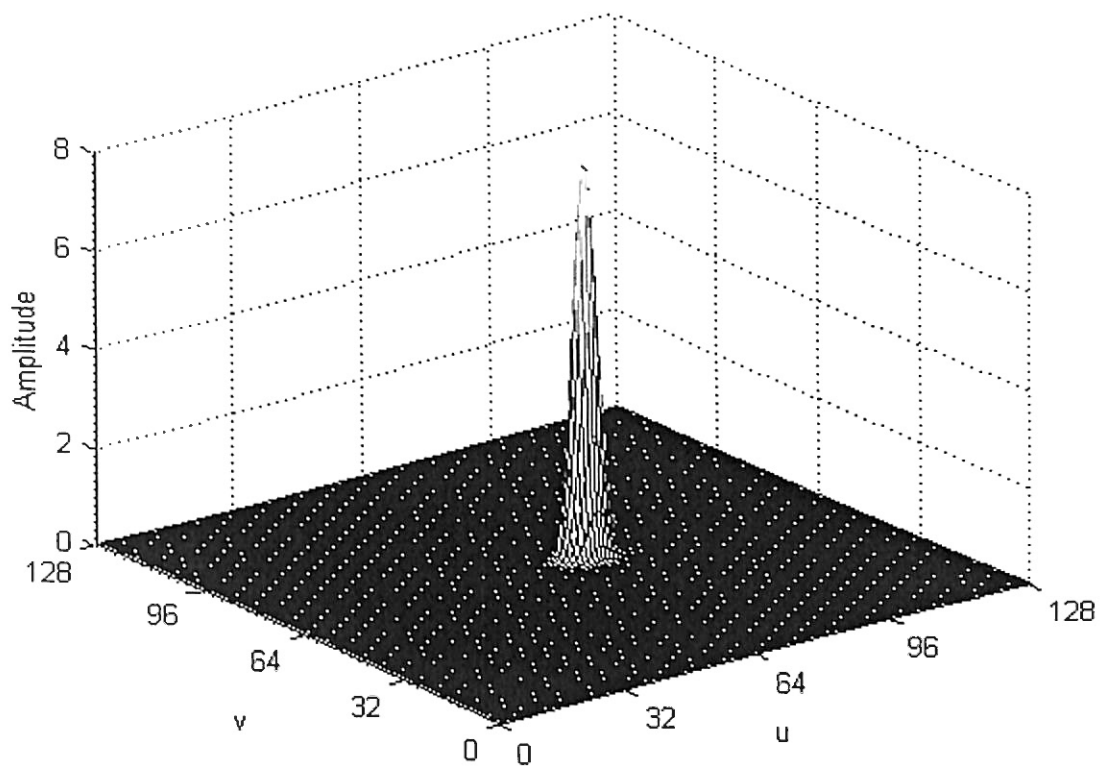


Figure 9. Frequency response of the Gabor filter ( $K=3$ ;  $S=3$ ;  $n=0$ ;  $m=0$ )

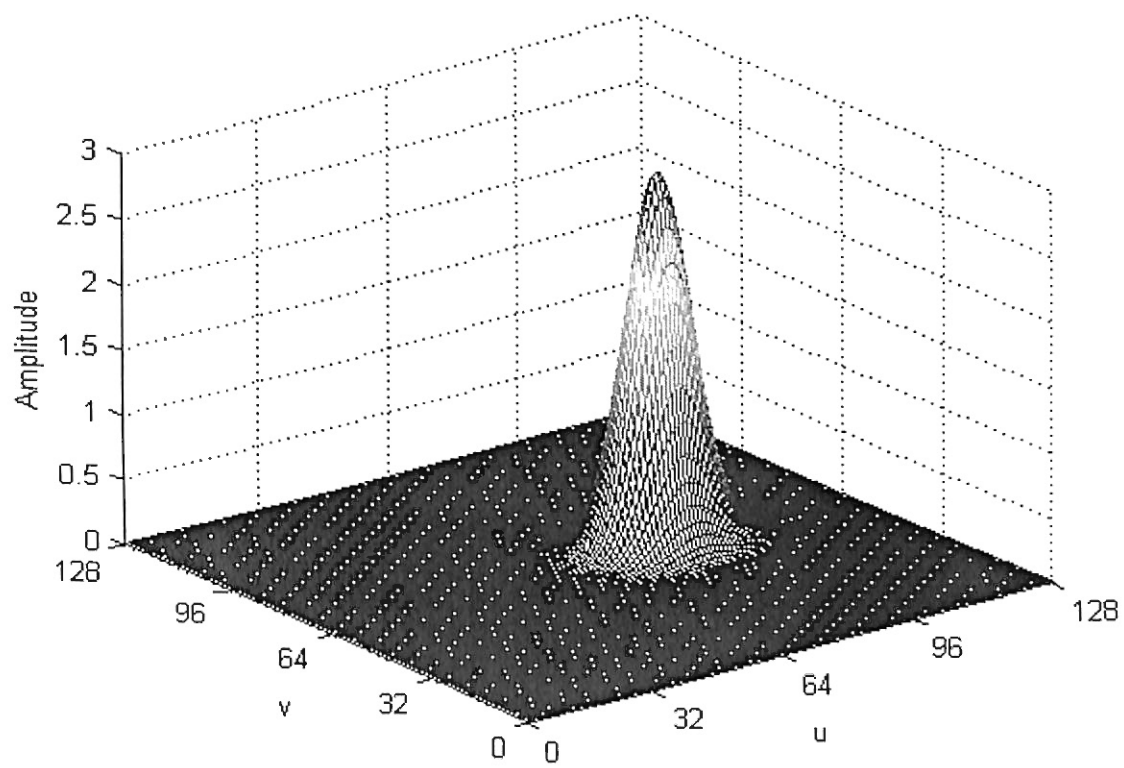
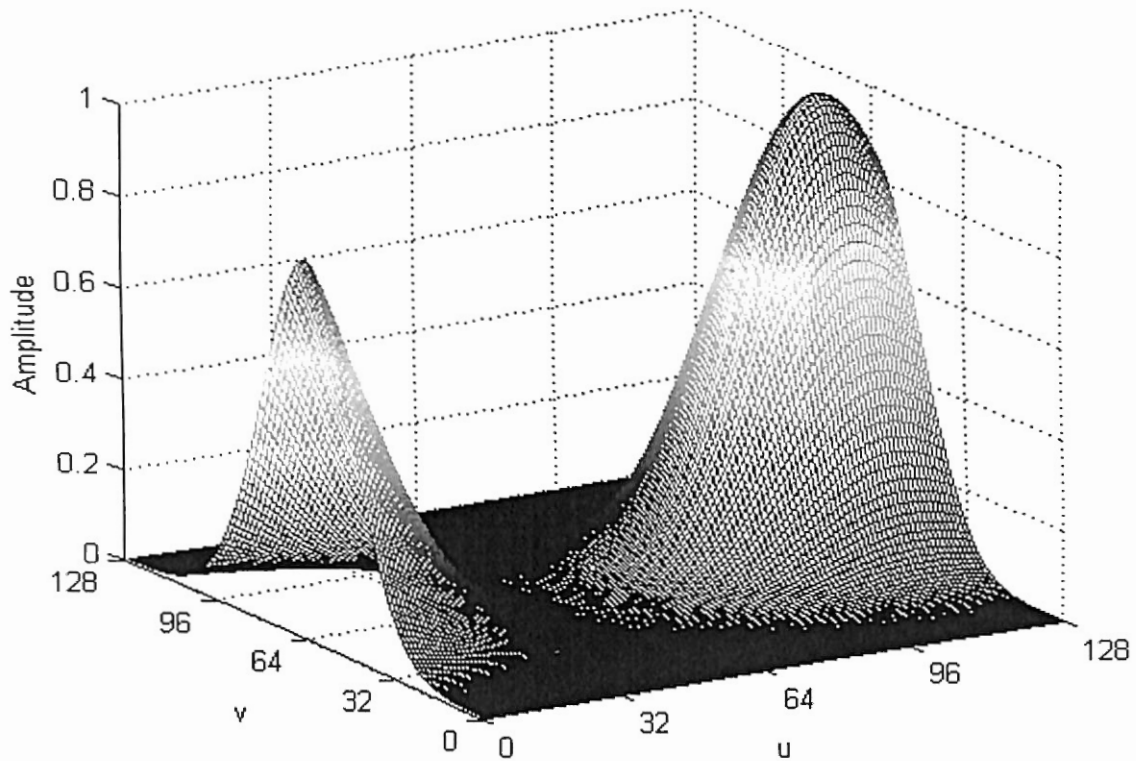


Figure 10. Frequency response of the Gabor filter ( $K=3$ ;  $S=3$ ;  $n=0$ ;  $m=1$ )



**Figure 11. Frequency response of the Gabor filter ( $K=3; S=3; n=0; m=2$ )**

The peak amplitude for coarse scale is higher than that for fine scale. The filters are designed such that the volume under the Gaussian for any scale and orientation is same, so that equal weight is given for all textural scale and orientation. The scale ( $s=1$ ) would capture coarse texture, whereas the scale ( $s=3$ ) would capture fine texture features.

Similar to Figure 8, the Gabor filters for 3 other orientations are given in Figures 12-14. All these filters would constitute a bank of band-pass filters, which would cover half of the frequency space. Figure 14 shows the frequency responses of all filters in the same image. Note that the coarse scale filters are bright, however the bandwidth is small.

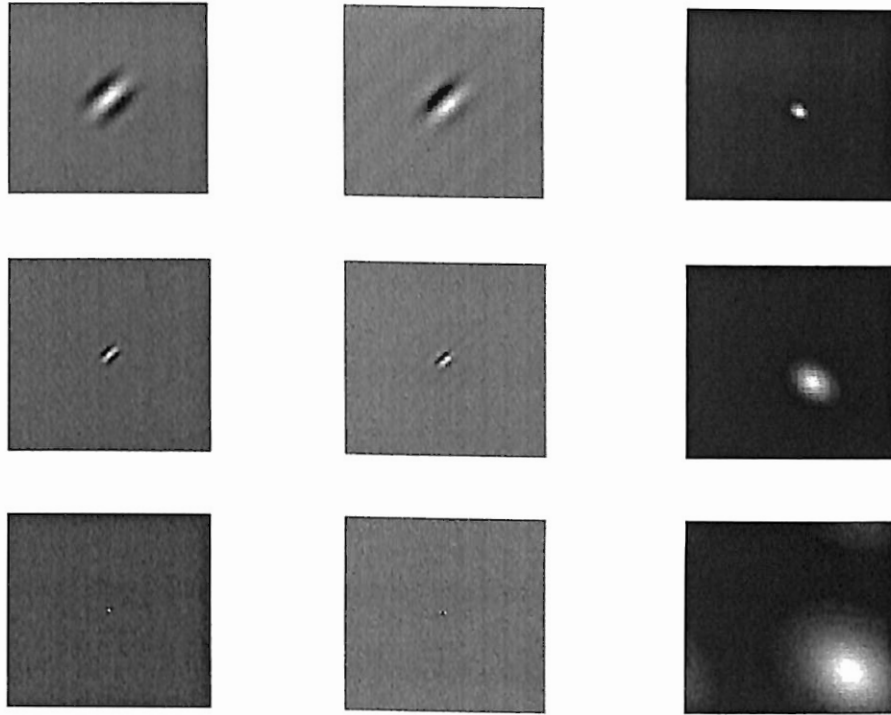


Figure 12. Impulse and frequency response of Gabor filter ( $K=3$ ;  $S=3$ ;  $n=1$ ;  $m=0,1,2$ ).

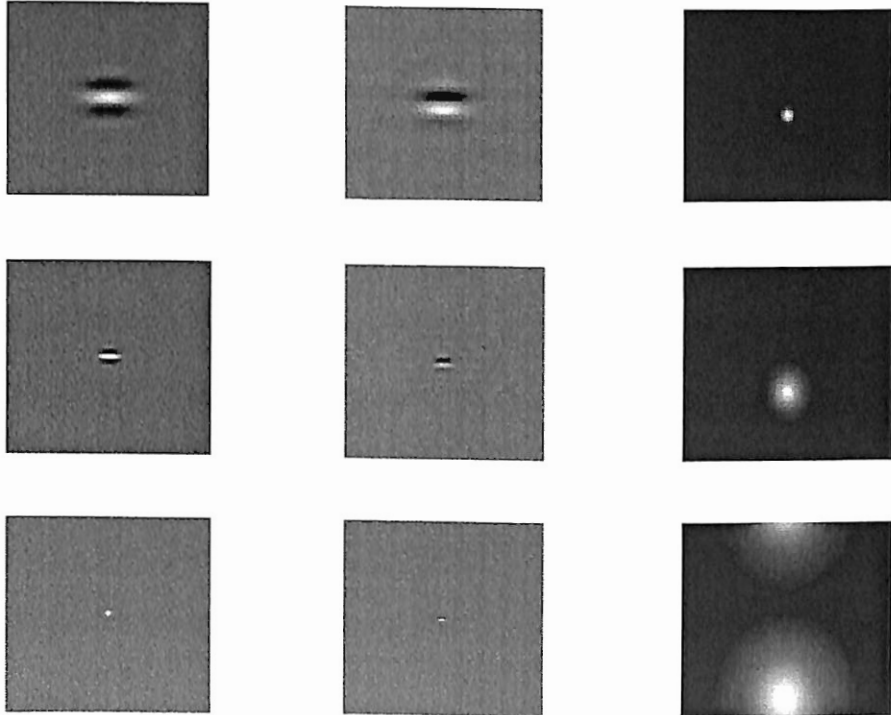


Figure 13. Impulse and frequency response of Gabor filter ( $K=3$ ;  $S=3$ ;  $n=2$ ;  $m=0,1,2$ ).

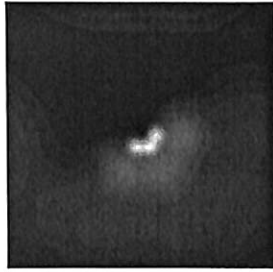


Figure 14. Frequency coverage by Gabor filters ( $K=3, S=3$ ).

Whereas, the fine scale filters are not bright and their bandwidth is large. All filters have the same volume under each Gaussian.

Manjunath and Ma (1996) designed bank of band-pass filters ( $K=6; S=4$ ) such that the half-peak of each Gaussian touches the neighboring Gaussian (Figure 15). Note that the filters cover only half of the frequency space, as the other half is redundant. This is because FT of a real image is conjugate symmetric.

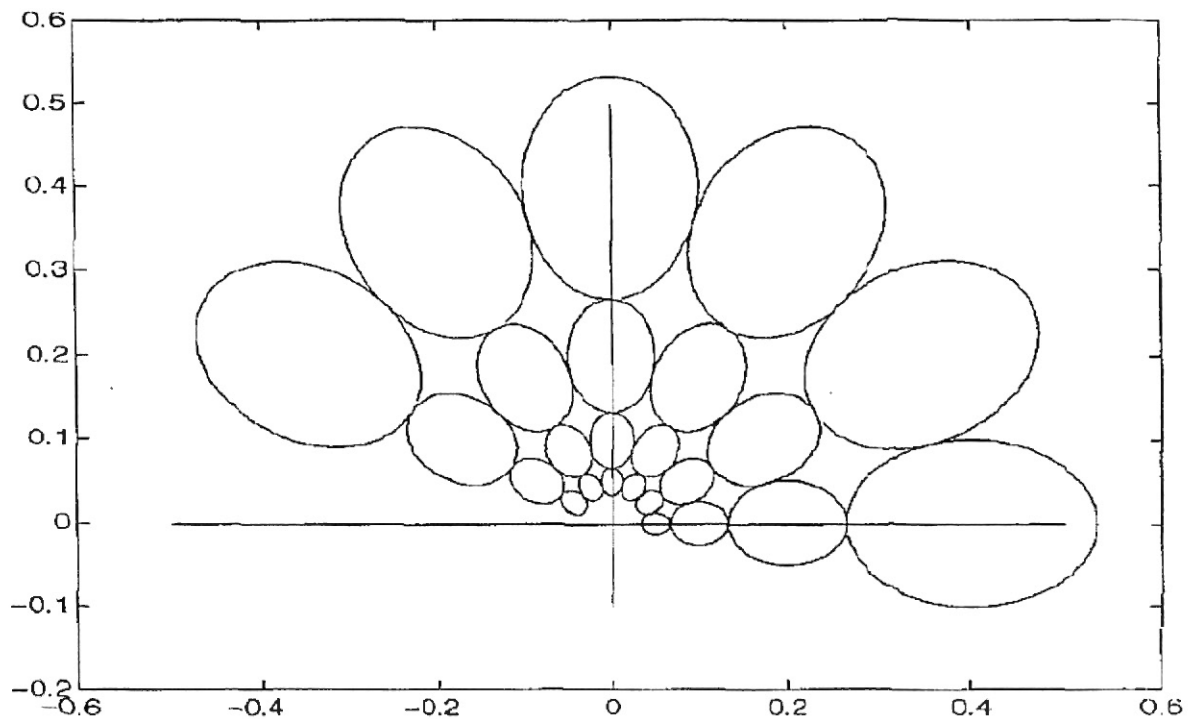


Figure 15. Gabor filters ( $K=6; S=4$ ). Adapted from Manjunath and Ma (1996).

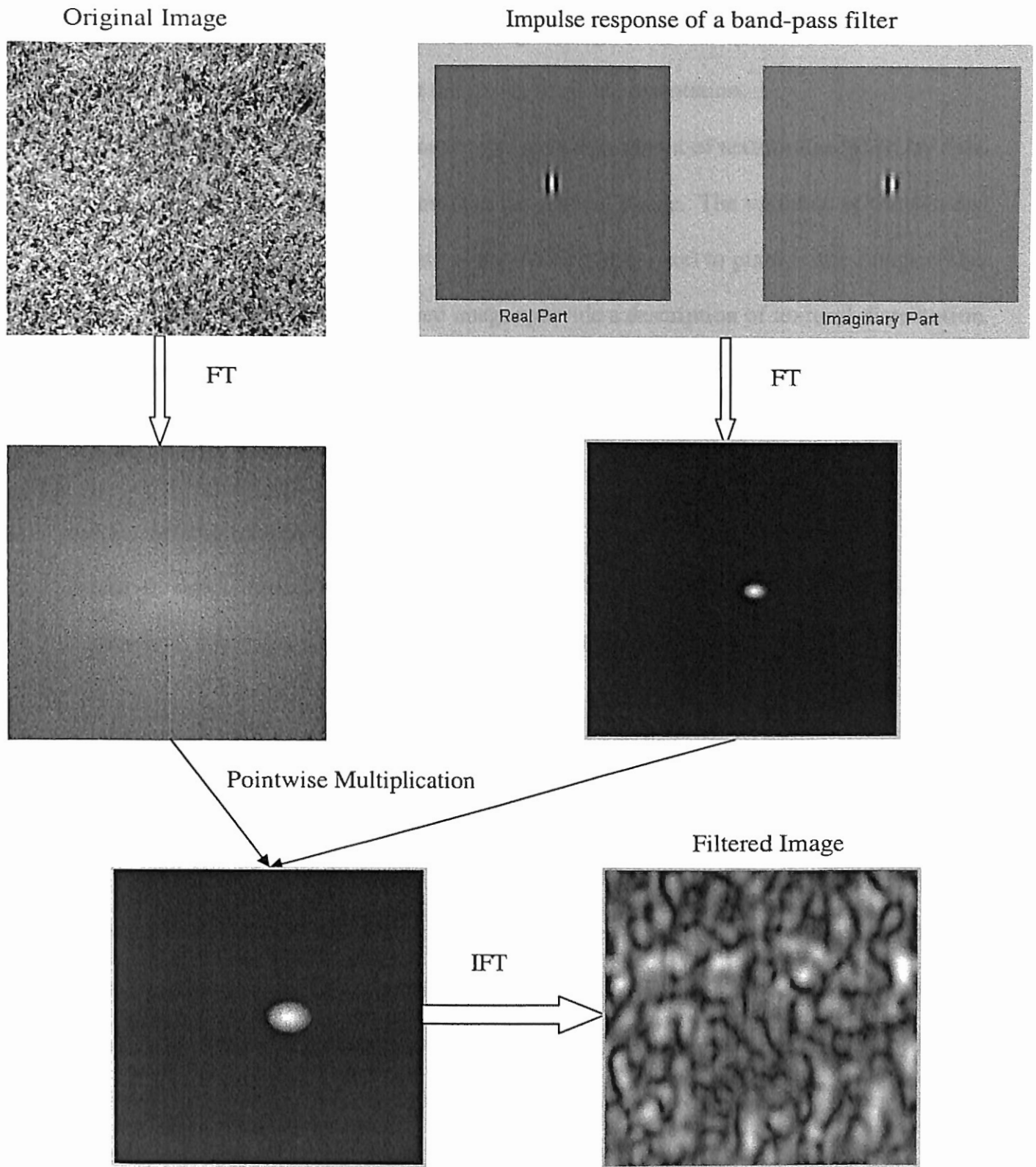


Figure 16. Basic steps of filtering using a Gabor filter.

Figure 16 shows the basic steps of filtering using Gabor filters, similar to Figure 7. If a particular location in the filtered image is bright, then that particular location has more texture corresponding to that frequency band and orientation.

The mean of the filtered image represents the amount of texture described by that frequency band and orientation present in the original image. The variance of the filtered image would describe the variability of that texture from pixel to pixel in the image. The mean and the variance of the filtered images provide a description of textural distribution at various scales and orientations.

## 3.2. Similarity Measures

### 3.2.1. Minkowsky Distance

Basic distance measures commonly are used for similarity searches. Minkowsky distance ( $L_p$ ) is defined as (Saastamoinen et al., 2002)

$$L_p(x, y) = \left( \sum_{i=1}^N |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (3.8)$$

where  $x$  and  $y$  are vectors. This is also commonly known as  $L_p$  norm.

When  $p=1$ , this distance is called “Manhattan” distance. When  $p=2$ , this distance is called as “Euclidean distance”. Manhattan distance gives equal importance to all features. As the value of ‘ $p$ ’ increases, the weight of a large difference between the values of a feature increases. Therefore, Euclidean distance gives more weight to the features whose differences are large. When  $p$  goes to infinity, the Minkowsky norm will identify the largest (maximum) difference between all features.



### 3.2.2. Mahalanobis Distance

Minkowsky distance does not take into account the magnitude differences and covariance (relationship) among features. To overcome this limitation, Mahalanobis distance is introduced (Johnson, 1998). The Mahalanobis distance between the two vectors  $\mathbf{x}$  and  $\mathbf{y}$  is given by:

$$d^2 = (\mathbf{x} - \mathbf{y})\Sigma^{-1}(\mathbf{x} - \mathbf{y})' \quad (3.9)$$

where:  $d$  = Mahalanobis distance

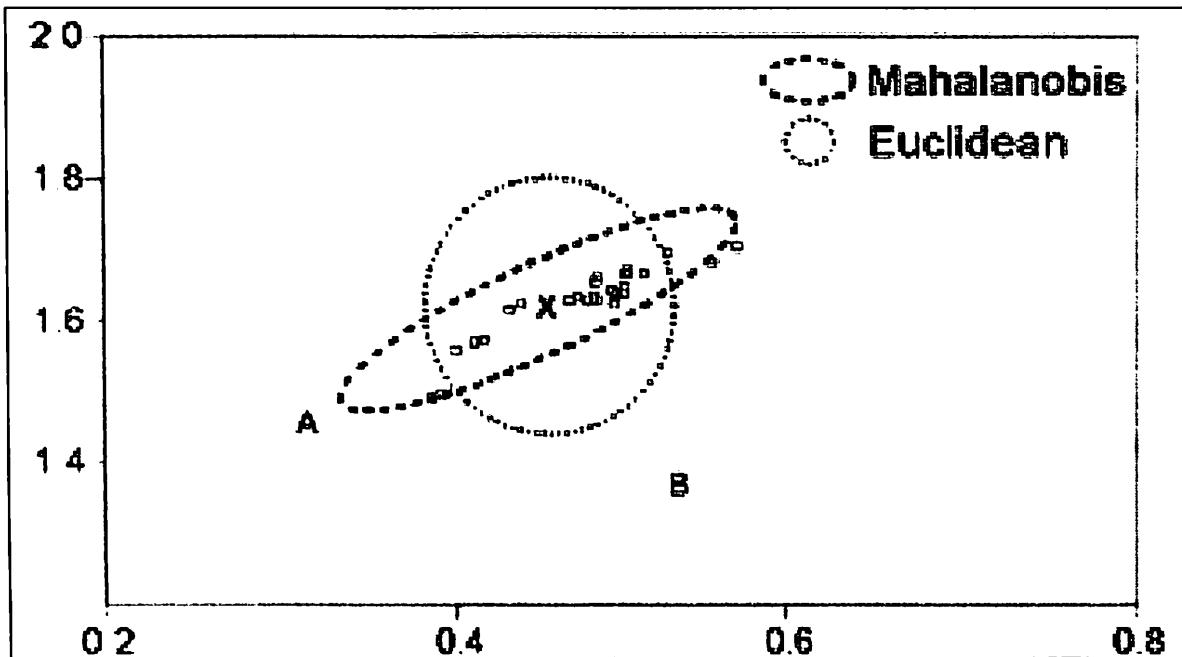
$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ \dots \ x_N],$$

$$\mathbf{y} = [y_1 \ y_2 \ y_3 \ \dots \ y_N], \text{ and}$$

$$\Sigma = \text{covariance matrix}$$

Note that  $\mathbf{x}$  and  $\mathbf{y}$  are  $(1 \times N)$  vectors representing  $N$  textural features extracted from two different images. The size of the covariance matrix is  $(N \times N)$ .

Mahalanobis distance can be explained graphically (Fig. 17). For simplicity, let us consider that there are 2 features and therefore the size of  $\mathbf{x}$  and  $\mathbf{y}$  vectors are  $(1 \times 2)$ . There are 'n' observations or images and the objective is to determine the Euclidean and Mahalanobis distances from the mean or cluster center. Therefore,  $\mathbf{y}$  is represented by the cluster center, which is marked by a 'cross' in the Figure 17. Observe that there is some relationship between the two features. In general feature 1 increases, when feature 2 increases. Therefore, the 2 features are correlated to some degree.



**Figure 17. Illustration of differences between Euclidean and Mahalanobis distances**

The Euclidean distance contours are circular, as this distance measure does not take into account of these relationships among the features, which are described by the covariance matrix. Therefore, the Euclidean distance would indicate that observations A and B in Figure 17 are equidistant from the cluster center. Mahalanobis distance takes into account the correlation between the features, and the contours of distances for this example are elliptical. All observations on this elliptic contour are equidistant from the cluster center. Mahalanobis distances are calculated based on the unit of standard deviation, and therefore this distance measure removes the effect of magnitude differences between features and correlation among features. Mahalanobis distance would clearly identify that observation 'B' is far different from 'A', with respect to the cluster center.

### 3.3. Normalization

Some features may be of a larger magnitude/range and these get more weight than features of lower magnitude/range. Aksoy and Haralick (2001) conducted various feature normalizations to improve Minkowsky measures.

#### 3.3.1. *Linear scaling to unit range (minmax)*

One simple way is to linearly normalize the range of each feature to (0,1), as given by:

$$\tilde{x} = \frac{x - l}{u - l} \quad (3.10)$$

where  $\tilde{x}$  is the transformed variable, and  $u$  and  $l$  are the upper and lower bounds of the original variable  $x$ . This is also known as ‘minmax’ normalization. This normalization is not robust.

#### 3.3.2. *Linear scaling to unit variance (meanstd)*

The other more common method is normalizing each feature to zero mean and unit standard deviation.

$$\tilde{x} = \frac{x - \mu}{\sigma} \quad (3.11)$$

where  $\mu$  is the mean and  $\sigma$  is the standard deviation of  $x$ . This is also known as ‘meanstd’ normalization. This normalization is not very robust.

#### 3.3.3. *Rank normalization*

Given a feature for all images  $x_1, x_2, \dots, x_n$ , the order statistics or rank  $x_{(1)}, x_{(2)}, \dots, x_{(n)}$  is determined. Then, each image’s feature value is replaced by its corresponding rank.

This is repeated for all the features. This procedure uniformly maps all feature values to the [0, n] range. If more than one image has the same feature value, the average rank then is assigned to those images. Aksoy and Haralick (2000) further scaled the ranks to [0, 1] range. Because all features were normalized already to [0, n] range, further linear scaling to [0, 1] will not affect the performance of image retrieval. Therefore this step was not performed in this study.

### 3.3.4. Transformation to a Uniform [0, 1] random variable

A feature 'x' can be considered a random variable. The empirical cumulative distribution function  $F(x)$  can be defined as

$$F(x) = (\text{number of images having value } \leq x) / (\text{total number of images}) \quad (3.12)$$

By setting the transformed variable,  $\tilde{x} = F(x)$ , the new variable will be distributed uniformly in the [0, 1] range (Papoulis, 1991).

### 3.3.5. Fitting a Normal ( $\mu, \sigma^2$ ) density

The features can be assumed to be distributed normally with probability density function (PDF) given by:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.13)$$

The two parameters for this distributions are  $\mu$  ( mean) and  $\sigma$  (standard deviation). They can be estimated by sample average  $\bar{x}$  and s.

$$\begin{aligned}\bar{x} &= \frac{1}{N} \sum_{i=1}^N x_i \\ s^2 &= \frac{1}{N-1} \sum_{i=1}^{N-1} (x_i - \bar{x})^2\end{aligned}\tag{3.14}$$

By using the set of training images, mean and standard deviation for each feature are calculated. Then, the cumulative probability density function (CDF), which is the probability that a single observation falls in the normal distribution defined by  $N(\mu, \sigma^2)$  in the region  $[-\infty, x]$ , is calculated by

$$F(x | \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt\tag{3.15}$$

These CDF values then are assigned to the corresponding values of the features. Note that probability ranges from  $[0, 1]$  and therefore the values are transformed to range  $[0, 1]$ , assuming a normal distribution function.

### 3.3.6. *Fitting a Lognormal $(\mu, \sigma^2)$ density*

A lognormal distribution becomes a normal distribution when natural logarithm of the distribution is taken. Therefore, all features are transformed simply by natural logarithm, and then a normal distribution is fitted, as explained in section 3.3.5. The CDF values then are used as transformed variables.

### 3.3.7. *Fitting an Exponential $(\mu)$ density*

The exponential PDF is given by:

$$f(x) = \frac{1}{\mu} e^{-\frac{x}{\mu}}\tag{3.16}$$

The mean  $\mu$  is estimated by sample average. Some statistics books refer  $\mu$  as  $\lambda$  in the

context of exponential distribution. The CDF is calculated by:

$$F(x | \mu) = 1 - e^{-\frac{x}{\mu}} \quad (3.17)$$

Therefore, each value then is transformed by the above equation. The transformed values will be in the range of [0, 1].

### 3.3.8. Fitting an Gamma ( $\alpha$ , $\beta$ ) density

A gamma PDF is given by:

$$f(x) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}} \quad (3.18)$$

where:  $\alpha$ ,  $\beta$  are parameters of the gamma distribution and  $\Gamma$  is a gamma function, defined by:

$$\Gamma(\alpha) = \int_0^{\infty} e^{-t} t^{\alpha-1} dt \quad (3.19)$$

The parameters  $\alpha$ ,  $\beta$  are first estimated using training image database by (Aksoy and Haralick, 2000):

$$\begin{aligned} \bar{\alpha} &= \frac{\sum_{s=1}^S x_s^{-2}}{S} \\ \bar{\beta} &= \frac{\sum_{s=1}^S x_s^{-2}}{S} \end{aligned} \quad (3.20)$$

The CDF then can be determined by:

$$F(x | \alpha, \beta) = \frac{1}{\beta^\alpha \Gamma(\alpha)} \int_0^x t^{\alpha-1} e^{-\frac{t}{\beta}} dt \quad (3.21)$$

The CDF values then are used as transformed variables, which is in the range of [0, 1].

### 3.4. Principal component analysis

All the above similarity measures do not work well when the features are highly correlated. The features can be preprocessed to reduce the dimensionality by principal component analysis (PCA).

Catalan and Gedeon (1999) have used principal component analysis (PCA) to reduce the dimensions of the feature space. PCA is based on Eigenvector decomposition. PCA involves a mathematical procedure that transforms a set of correlated features into a new smaller set of uncorrelated features called *principal components* (Johnson, 1998).

Let 'x' be the feature vector of an image, given by:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}$$

where: p = number of textural features extracted. We assume that this vector has a multivariate normal distribution.

Let 'XA' be the feature matrix of size (N x p),

$$\mathbf{XA} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \quad (3.22)$$

where: p = number of textural features extracted,

N = number of images,

$x_{rk}$  = value of the  $k^{\text{th}}$  texture feature on the  $r^{\text{th}}$  image for  $r=1, \dots, N$  and  $k=1, \dots, p$

The mean of all features can be calculated and stored as a mean vector:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_p \end{bmatrix} \quad (3.23)$$

where  $\mu_k$  is the mean of the  $k^{\text{th}}$  feature of  $N$  images, which is the mean of the  $k^{\text{th}}$  column of  $\mathbf{XA}$ .

The relationships between the features can be measured by the covariances and/or the correlations. Covariance of  $x_i$  and  $x_k$ :  $\sigma_{ij} = \text{Cov}(x_i, x_k) = E[(x_i - \mu_i)(x_k - \mu_k)]$ ; where  $E(\cdot)$  stands for estimate of a variable, which is the mean.

Covariance of  $x_i$  and  $x_i$  (variance of  $x_i$ ):  $E[(x_i - \mu_i)^2] = \sigma_{ii}$

The covariance matrix can be arranged as:

$$\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})'] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix} \quad (3.24)$$

Note that the covariance matrix is symmetric, i.e.,  $\sigma_{ik} = \sigma_{ki}$

Correlation coefficient of  $x_i$  and  $x_k$  is given by:

$$\rho_{ik} = \frac{\sigma_{ik}}{\sqrt{\sigma_{ii}\sigma_{kk}}}$$

The correlation matrix is given by:

$$\mathbf{P} = \text{Corr}(\mathbf{x}) = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \cdots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \cdots & 1 \end{bmatrix} \quad (3.25)$$



Note that correlation values always range from  $-1$  to  $+1$ . The value of  $+1$  indicates a perfect positive linear relationship between 2 variables, whereas  $-1$  indicates a perfect negative linear relationship. The value of  $0$  indicates no relationship between those 2 variables.

Eigenvalues are also known as characteristic roots. Eigenvalues of  $\Sigma$  are the roots of the polynomial equation defined by

$$|\Sigma - \lambda \mathbf{I}| = 0 \quad (3.26)$$

where  $\mathbf{I}$  is an identity matrix.

If  $p=2$  features, then the determinants are the roots of

$$\begin{aligned} \left| \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| &= \left| \begin{bmatrix} \sigma_{11} - \lambda & \sigma_{12} \\ \sigma_{21} & \sigma_{22} - \lambda \end{bmatrix} \right| \\ &= (\sigma_{11} - \lambda)(\sigma_{22} - \lambda) - \sigma_{21}\sigma_{12} \\ &= \lambda^2 - \lambda(\sigma_{11} + \sigma_{22}) - (\sigma_{21}\sigma_{12} - \sigma_{11}\sigma_{22}) = 0 \end{aligned}$$

$$\lambda = \frac{\sigma_{11} + \sigma_{22} \pm \sqrt{(\sigma_{11} + \sigma_{22})^2 + 4(\sigma_{21}\sigma_{12} - \sigma_{11}\sigma_{22})}}{2} \quad (3.27)$$

In general, the eigenvalues are the  $p$  roots of

$$c_1\lambda^p + c_2\lambda^{p-1} + c_3\lambda^{p-2} + \dots + c_p\lambda + c_{p+1} = 0 \text{ where } c_k \text{ denote constants.}$$

As  $\Sigma$  is a symmetric matrix, the eigenvalues are real numbers and can be ordered from largest to smallest as  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ . The sum of all eigenvalues represents the total variance explained by all original variables.

Each eigenvalue of  $\Sigma$  has a corresponding nonzero vector  $\mathbf{a}$ , called an eigenvector, that satisfies

$$\Sigma \mathbf{a} = \lambda \mathbf{a}. \quad (3.28)$$

PCA involves a mathematical procedure that transforms a set of correlated features into a new smaller set of uncorrelated variables called *principal components* (Johnson, 1996). When variables are highly correlated, they can often be represented just as well with a smaller set of variables. These new variables are linear combinations of the original variables.

Characteristics of the principal components:

- Uncorrelated
- The first principal component accounts for as much variability in the data as possible
- Each successive principal component accounts for as much variability in the data as possible

The first principal component can be calculated by:

$$\mathbf{y}_1 = \mathbf{a}'_1(\mathbf{x} - \boldsymbol{\mu}) \quad (3.29)$$

where:  $\mathbf{a}_1$  is chosen so that  $\text{Var}[\mathbf{a}'_1(\mathbf{x} - \boldsymbol{\mu})]$  is maximized over all vectors  $\mathbf{a}_1$  satisfying  $\mathbf{a}'_1\mathbf{a}_1=1$ . Note that  $\mathbf{a}'_1\mathbf{a}_1=1$  so  $\mathbf{a}_1$  is the first eigenvector normalized to have a length of 1. The maximum value of the variance is  $\lambda_1$  (largest eigenvalue) and it occurs when  $\mathbf{a}_1$  is the corresponding eigenvector of  $\Sigma$ . Since the variance of  $\mathbf{y}_1 = \mathbf{a}'_1(\mathbf{x} - \boldsymbol{\mu})$  is being maximized, the new variable  $y_1$  will explain as much variability of  $\mathbf{x}$  as possible.

The second principal component is given by:

$$\mathbf{y}_2 = \mathbf{a}'_2(\mathbf{x} - \boldsymbol{\mu}) \quad (3.30)$$

where  $\mathbf{a}_2$  is chosen so that  $\text{Var}[\mathbf{a}'_2(\mathbf{x} - \boldsymbol{\mu})]$  is maximized over all vectors  $\mathbf{a}_2$  that

are uncorrelated with  $\mathbf{a}_1$  and satisfying  $\mathbf{a}_2' \mathbf{a}_2 = 1$ . Uncorrelated means that  $\mathbf{a}_1$  and  $\mathbf{a}_2$  are orthogonal (i.e.,  $\mathbf{a}_1' \mathbf{a}_2 = 0$ ). The maximum value is  $\lambda_2$  (2<sup>nd</sup> largest eigenvalue) and it occurs when  $\mathbf{a}_2$  is the corresponding eigenvector of  $\Sigma$ .

Third, fourth, and further principal components can be found similarly to the first and second principal components. It can be shown that since the eigenvectors are orthogonal, the principal components are orthogonal.

### 3.4.1. *Principal component scores*

For each image, we need to calculate the  $k^{\text{th}}$  *principal component value* or *score* as:

$$y_{rk} = \mathbf{a}_k' (\mathbf{x}_r - \boldsymbol{\mu}) \quad (3.31)$$

These principal component scores will be used instead of the original feature set.

PCA can be done on the correlation matrix, instead of the covariance matrix. Note that the correlation coefficient is a unitless measure. Using  $\mathbf{P}$  is equivalent to performing PCA using features that are normalized to zero mean and unit standard deviation. As the magnitude of each texture feature is different from that of other features, PCA will be done on the covariance matrix.

### 3.4.2. *Determining the Number of Principal Components*

It is desirable to find the smallest number of principal components that explains most of the variability explained by original features. The smallest number of principal components ( $d$ ) is selected so that new PCs account for  $\gamma\%$  of the variability such that

$$\sum_{q=1}^d \lambda_q / \sum_{i=1}^p \lambda_i > \gamma \quad (3.32)$$

Note that the numerator in the above term represents the variances explained by first 'd' PCs, whereas the denominator represents the total variance explained by all original variables or all PCs.

### ***3.4.3. Steps in implementation of PCA***

1. For every image in the database, textural features were extracted
2. A feature matrix (Eqn. 3.22) is arranged such that each row represents an image and each column represents a feature.
3. A correlation matrix (Eqn. 3.24) describing the relationship between features is calculated from the feature matrix.
4. An eigenvalue decomposition of the correlation matrix is done using Equation 3.26.
5. For each eigenvalue, a corresponding eigenvector is calculated using Equation 3.28.
6. Based on eigenvalues, only the first few eigenvectors are selected such that they explain 99% of total variance of all original variables (Eqn. 3.32).
7. Then, principal component (PC) scores are calculated using Equation 3.31.

The above steps are done offline, and the following data are stored:

1. Mean vector of all features
2. Selected first few Eigenvectors
3. PC scores for all images in the database.

When a test image is presented for retrieving similar images in the database, the following steps are performed.

1. Textural features are extracted from the test image.
2. PC scores are calculated using Equation 3.31, eigenvector, and mean vector.
3. Similarity measures are calculated between the PC scores of test images and every image in the image database.
4. The top 15 images with smallest distances are retrieved.

### **3.5. Evaluation Procedure**

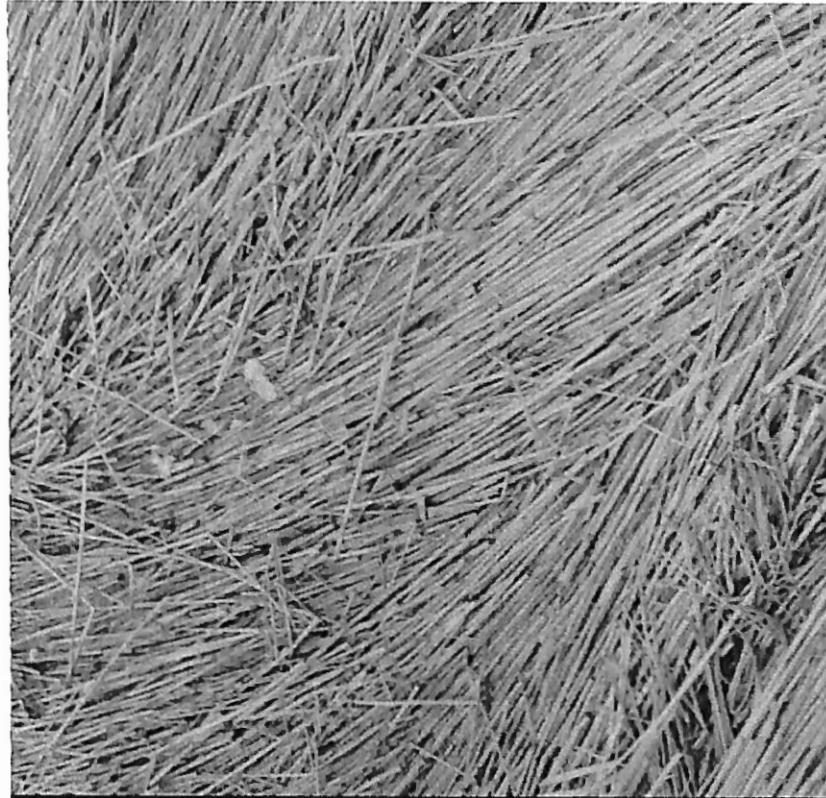
To evaluate the similarity measure, an image database was required. Initial studies on textural analysis have used computer-generated texture patterns. It is better to evaluate naturally occurring textures instead of computer-generated patterns. Brodatz (1966) published a photographic album consisting of 112 images of naturally occurring textures such as grass, stones, pebbles, cork, clouds, water, mesh, wood, etc. Some images from the Brodatz album are illustrated in Figures 18-21. The Brodatz album is a well-established standard or benchmark for testing texture analysis algorithms (Payne et al., 1999). These photographic images were digitized to a resolution of 1024x1024 using a scanner.

As seen from Figures 18-21, these are naturally occurring textures and therefore most images are not uniform. These images (1024x1024) were divided into 16 sub-images of size (256x256). Randomly one sub-image was chosen for testing purposes. The other fifteen sub-images for each image were stored in the training image database. This resulted in 1620 sub-images in the training database and 112 sub-images in the testing database. The 112 images in the testing database were presented one-by-one, and retrieval performance was evaluated. The top 15 images were retrieved. If the retrieval

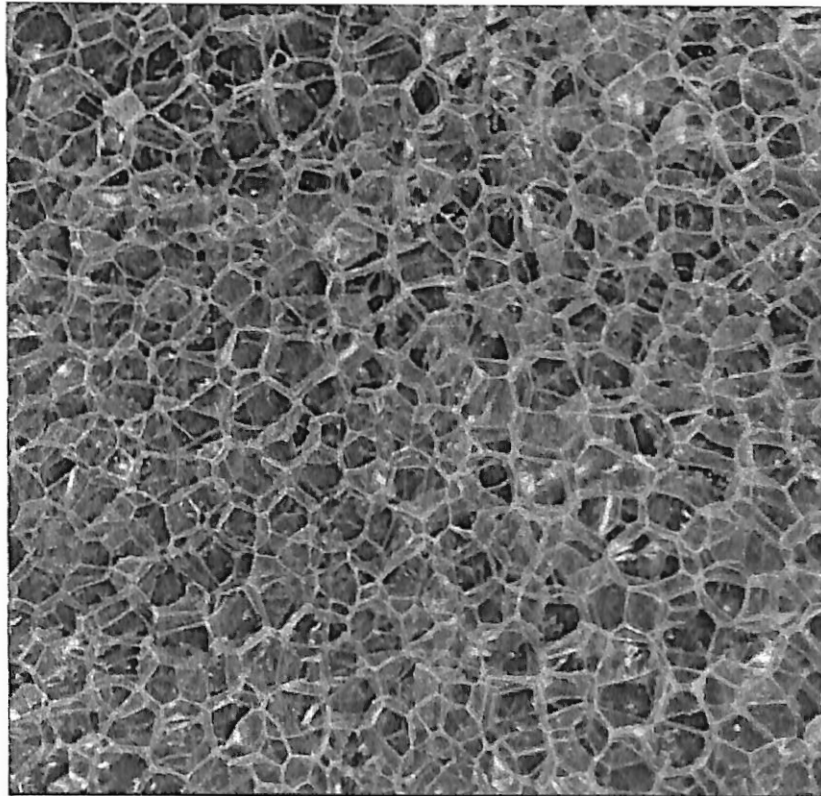
was perfect, the remaining 15 sub-images should have been retrieved for the test image. The percentage of correct retrievals for the top 1 retrieval to top 15 retrievals was calculated. The average percentage of correct retrieval for all 112 test images then was calculated and plotted. Based on this evaluation, similarity measures were compared.



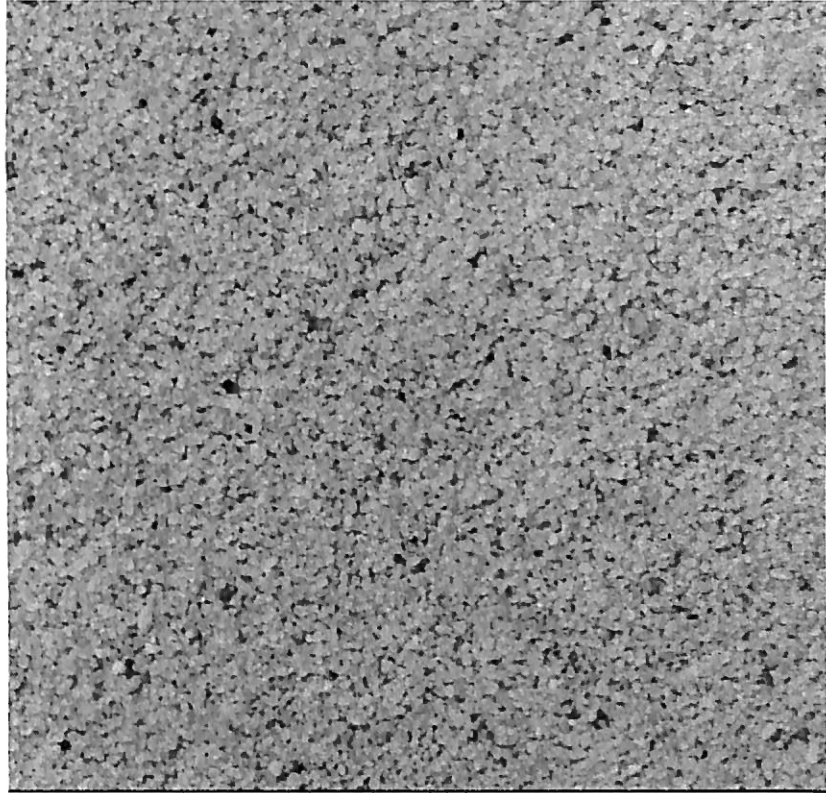
**Figure 18. Image of tree bark from Brodatz album.**



**Figure 19. Image of a straw from Brodatz album.**



**Figure 20. An image from Brodatz album.**



**Figure 21. An image of a cork from Brodatz album.**



## 4. RESULTS & DISCUSSION

### 4.1. Minkowsky Distance Measure

#### 4.1.1. Without normalization

Average percent correct retrievals were plotted against number of retrievals for various  $p$  values for Minkowsky distance measure (Eqn. 3.8). This measure is called Manhattan and Euclidean distance, when  $p$  value is equal to 1 and 2, respectively.

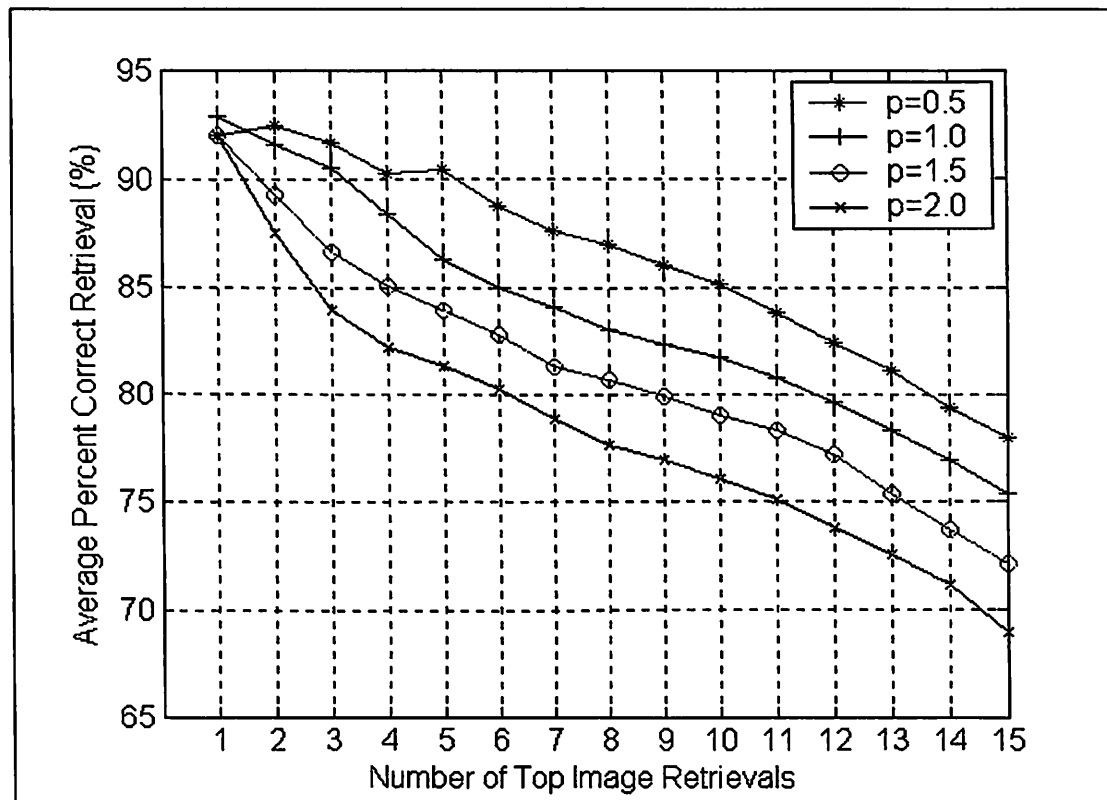


Figure 22. Performance of Minkowsky distance measure without normalization.

For each number of image retrieval, say 10, top 10 retrievals for every test image were considered. For each test image, percentage of images correctly retrieved was calculated. Then, the average of percent correct retrieval for all test images ( $n=112$ ) was calculated.

Similarly, average percent correct retrievals were calculated for other number of retrievals from 1 to 15. For users, performance of an image retrieval system for top few correct matches is more important than further matches. Minkowsky index,  $p$ , affected the performance of image retrieval slightly (Fig. 22). When  $p=0.5$ , the average percent correct retrievals were the highest for all top retrievals except for top 1<sup>st</sup> retrieval. Therefore, a  $p$ -value of 0.5 gave the best result. Average percent correct retrieval was greater than 90% up to top 5 retrievals for  $p$ -value of 0.5.

#### ***4.1.2. Normalization to unit range***

Textural features first were normalized by minmax operation (Eqn. 3.10), and Minkowsky distance measure then was used for image retrieval. As before, 4 different  $p$ -values were used. Performance was greater than 90% for up to top 8 retrievals. Minmax normalization has improved the performance compared to 'without' normalization. This is to be expected since this minmax operation normalizes all features to unit range and gives equal weight to all textural features. Performance was similar for all  $p$ -values (Fig. 23). The least computation time was found when  $p$  value is equal to 1. For other  $p$ -values, power operation in Equation 3.8. increased the computation time. Therefore, the  $p$ -value of 1 can be used for this distance measure.

#### ***4.1.3. Linear scaling to unit variance***

Textural features were normalized to zero mean and unit variance and Minkowsky distance measure then was used for image retrieval (Fig. 24). For this normalization also,  $p$ -values did not considerably affect the image retrieval performance.

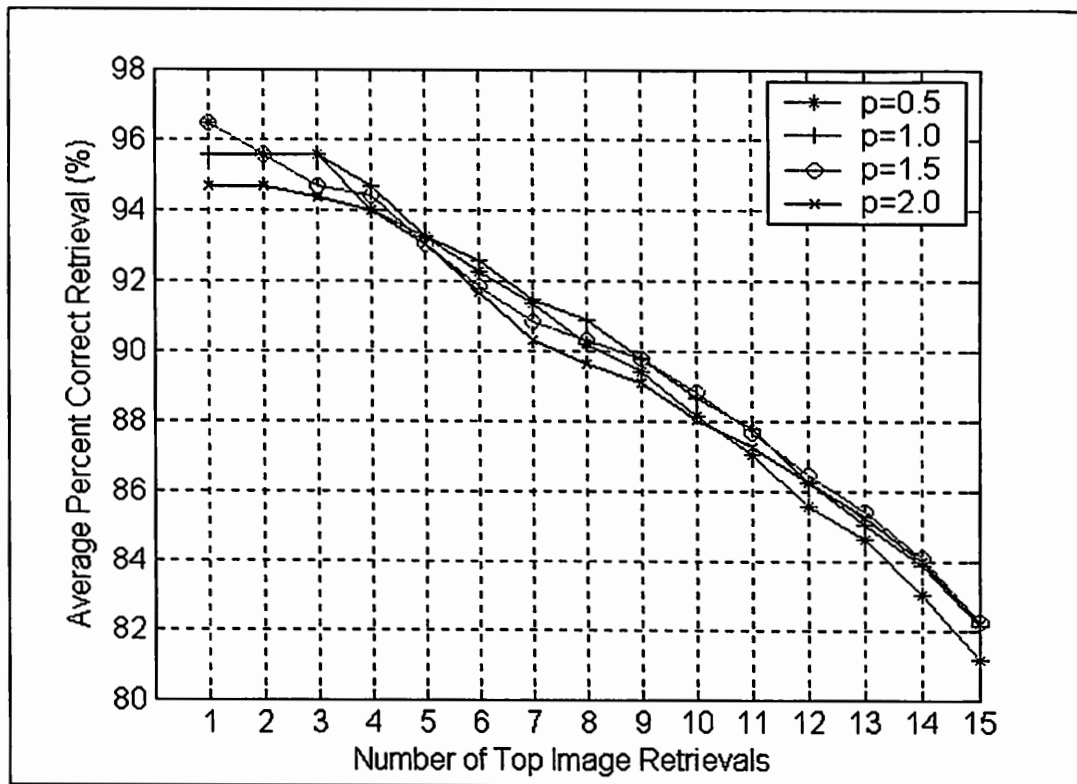


Figure 23. Performance of Minkowsky distance measure with 'minmax' normalization.

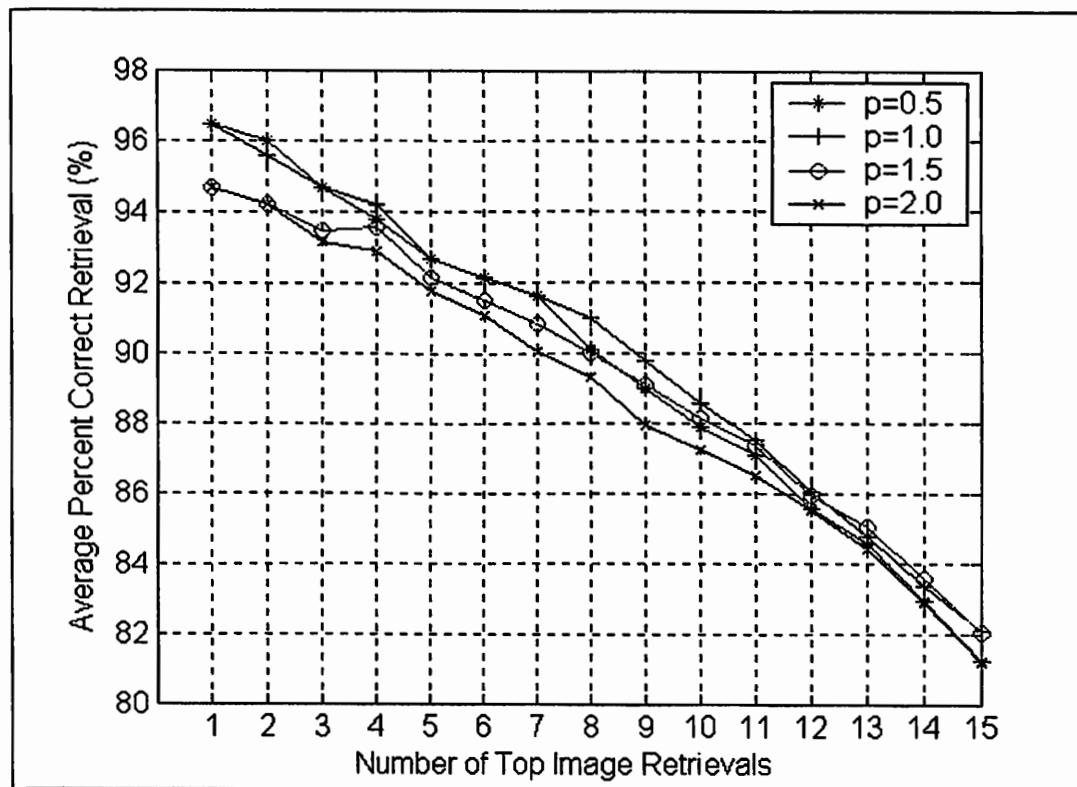


Figure 24. Performance of Minkowsky distance measure with 'meanstd' normalization.

Up to top 3 retrievals, p-values of 0.5 and 1.0 produced higher average percent retrievals than other p-values. Therefore, p-value of 1 can be used for this normalization, as the computation time was the least. Performance of 'meanstd' normalization was better than 'without' normalization, but was similar to 'minmax' normalization.

#### 4.1.4. Rank normalization

Ranks were used in Minkowsky distance measure instead of raw textural features.

Performance was similar to 'minmax' and 'meanstd' normalizations. The p-values of 0.5 and 1 performed better than other p-values (Fig. 25). Therefore, p-value of 1 was better for this normalization as computation time was the least for this p-value. Average percent correct retrieval was greater than 90% for up to top 8 retrievals.

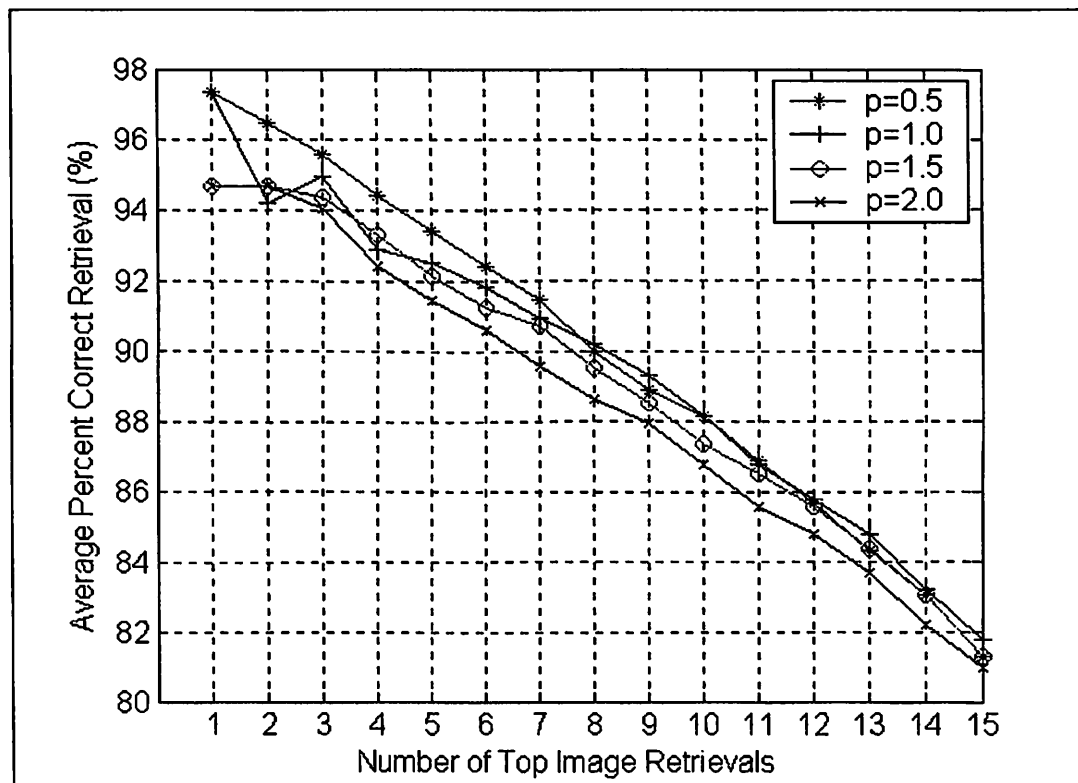


Figure 25. Performance of Minkowsky distance measure with 'rank' normalization.

#### 4.1.5. Transformation to a Uniform [0, 1] random variable

Textural features were normalized to a uniform [0, 1] random variable using empirical cumulative distribution as transformation function (Eqn. 3.12). The p-value of 0.5 performed slightly better than other p-values (Fig. 26). The performance of this transformation was similar to that of other three normalizations (Sections 4.1.2-4.1.5). The average percent correct retrieval was greater than 90% for up to top 8 retrievals. The computation time for this transformation was higher than other simple normalizations like 'minmax' or 'meanstd'. Therefore, simple normalizations were preferred rather than this transformation.

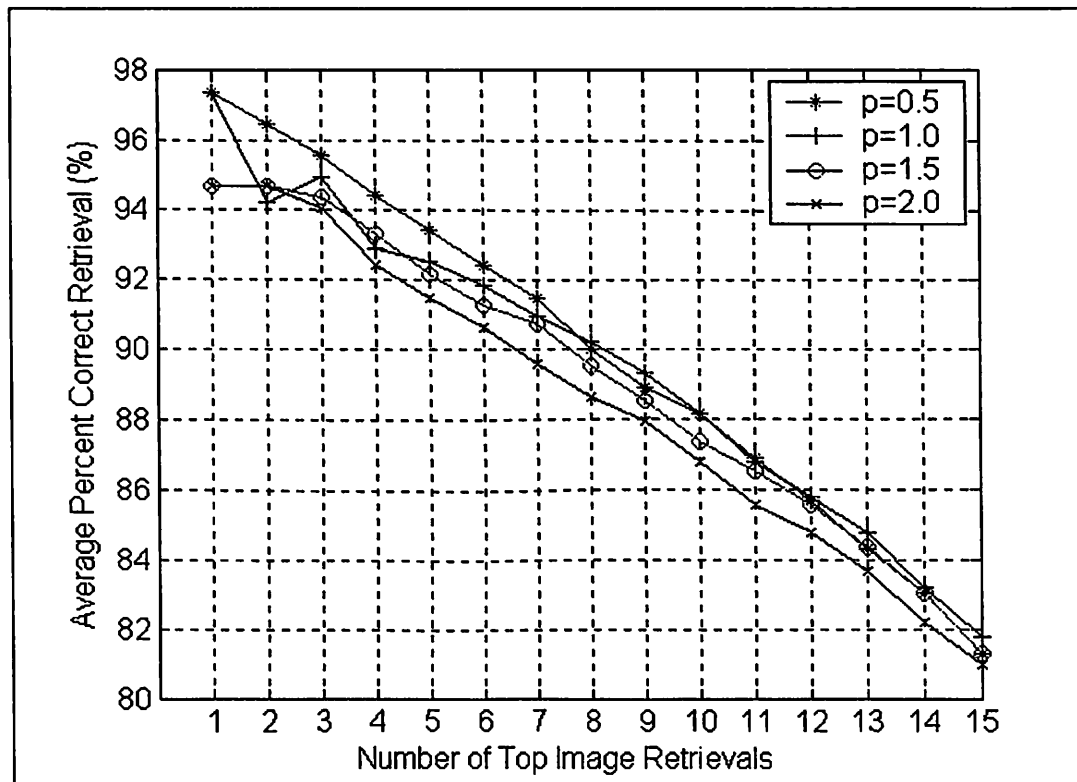


Figure 26. Performance of Minkowsky distance measure with 'uniform' transformation.

#### 4.1.6. Fitting a Normal Density

Each feature was fitted to a normal distribution and the probability of the distribution less

than or equal to that value was used as a transformed variable. After transformation, Minkowsky distance with different p-values was used to retrieve top 15 images for all test images. Average percent correct retrieval is shown in Figure 27. Among all p-values, the p-value of 1 performed the best. Average percent correct retrievals were greater than 90% for up to top 8 retrievals.

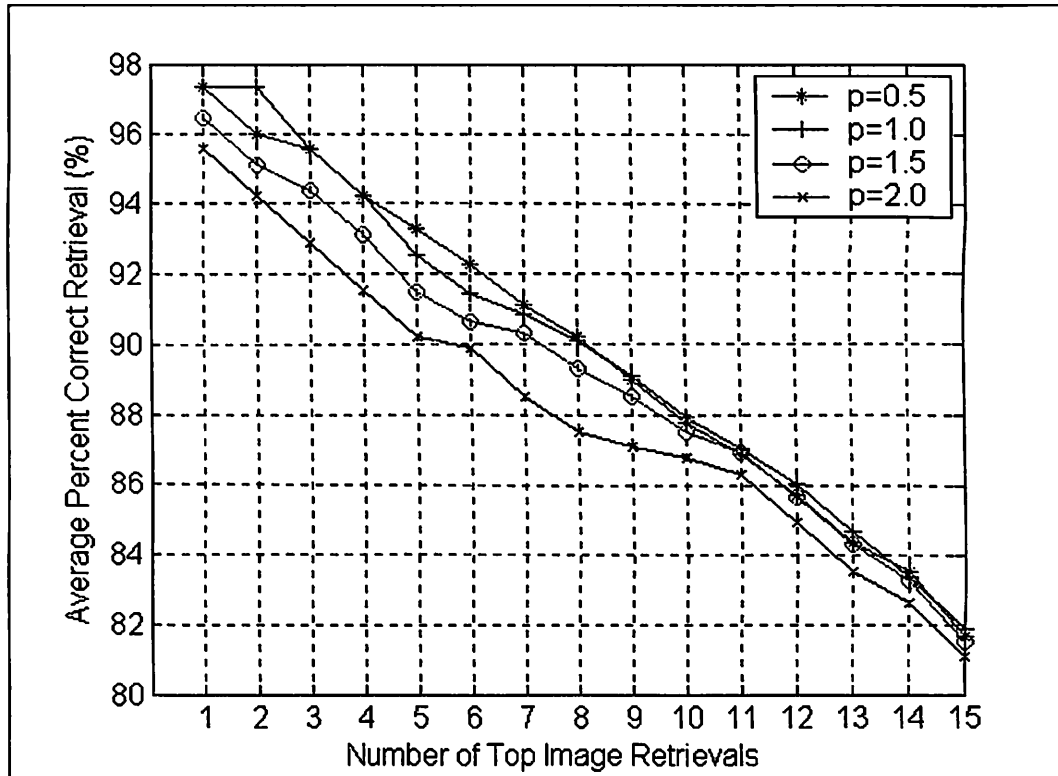
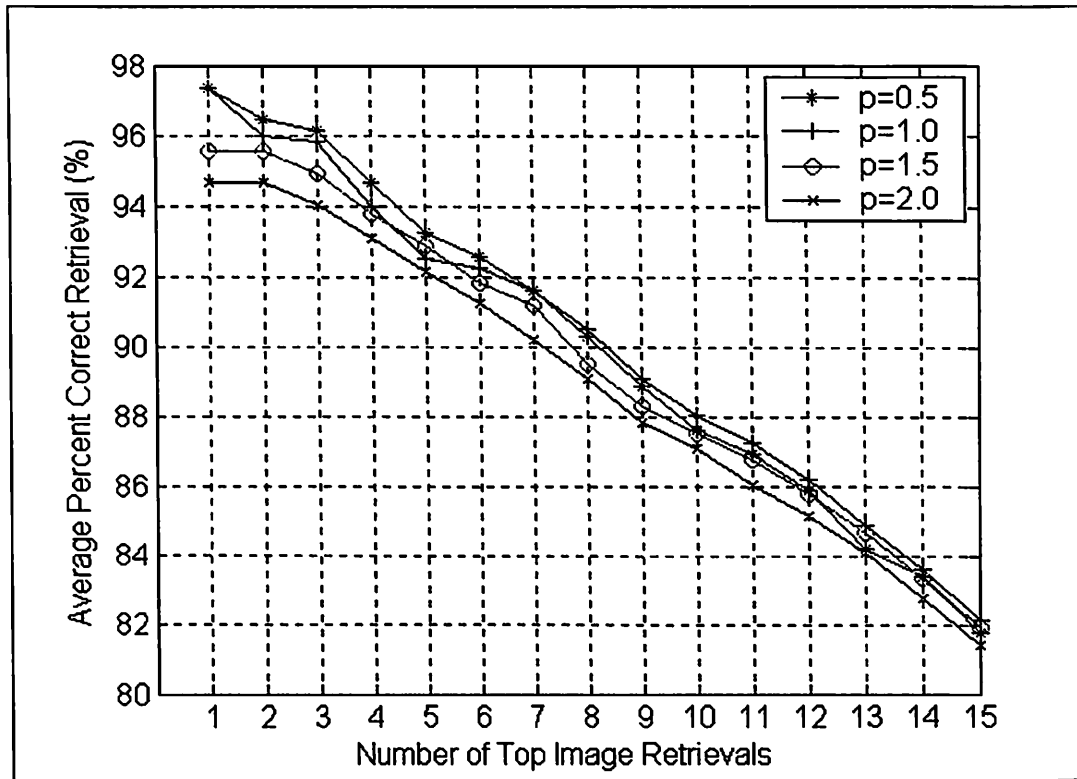


Figure 27. Performance of Minkowsky distance with normalization using normal distribution.

#### 4.1.7. Fitting a LogNormal Density

After taking natural logarithm of textural features, a normal density was fitted. The performance was similar to other normalizations (Fig. 28). Even though, p-value of 0.5 performed slightly better than p-value of 1, the later value might be preferred due to its low computation time. Average percent correct retrievals were greater than 90% for up to top 8 retrievals.



**Figure 28. Performance of Minkowsky distance with normalization using lognormal distribution.**

#### **4.1.8. Fitting an Exponential Density**

Each feature was fitted to an exponential distribution and the probability of the distribution less than or equal to that value was used as a transformed variable. After transformation, Minkowsky distance with different p-values was used to retrieve top 15 images for all test images. Average percent correct retrieval is shown in Figure 29. Even though, p-value of 0.5 performed slightly better than p-value of 1, the later value might be preferred due to its low computation time. Average percent correct retrievals were greater than 90% for up to top 8 retrievals.

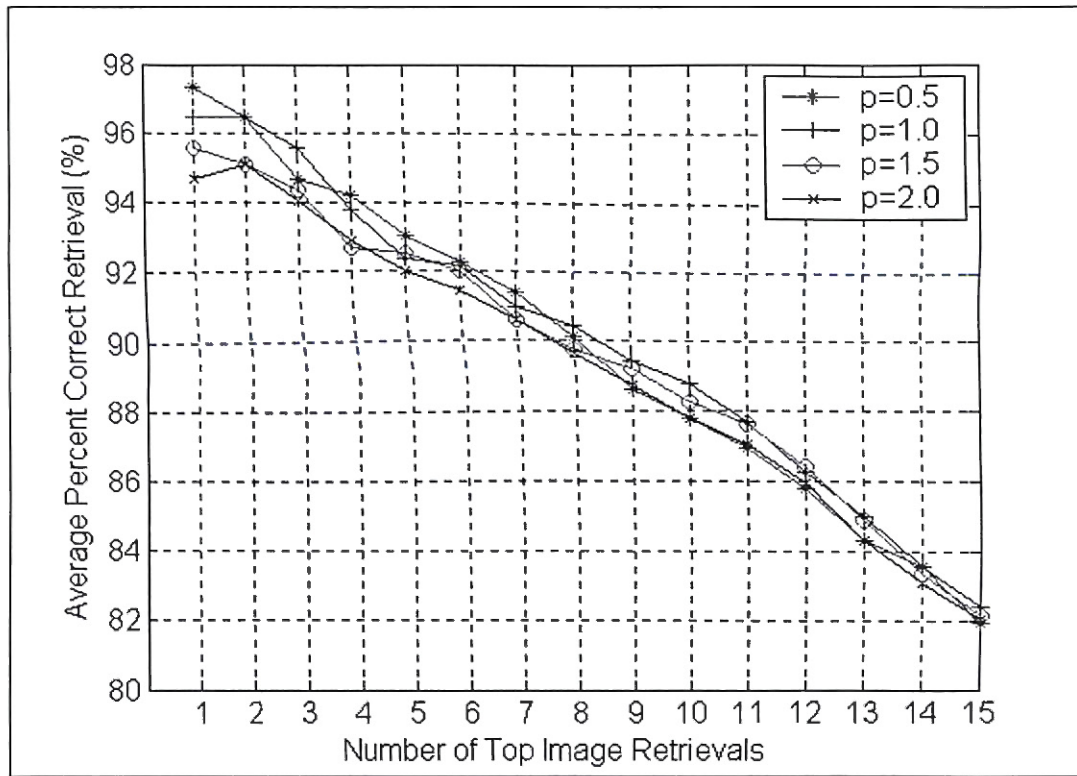


Figure 29. Performance of Minkowsky distance with normalization using exponential distribution.

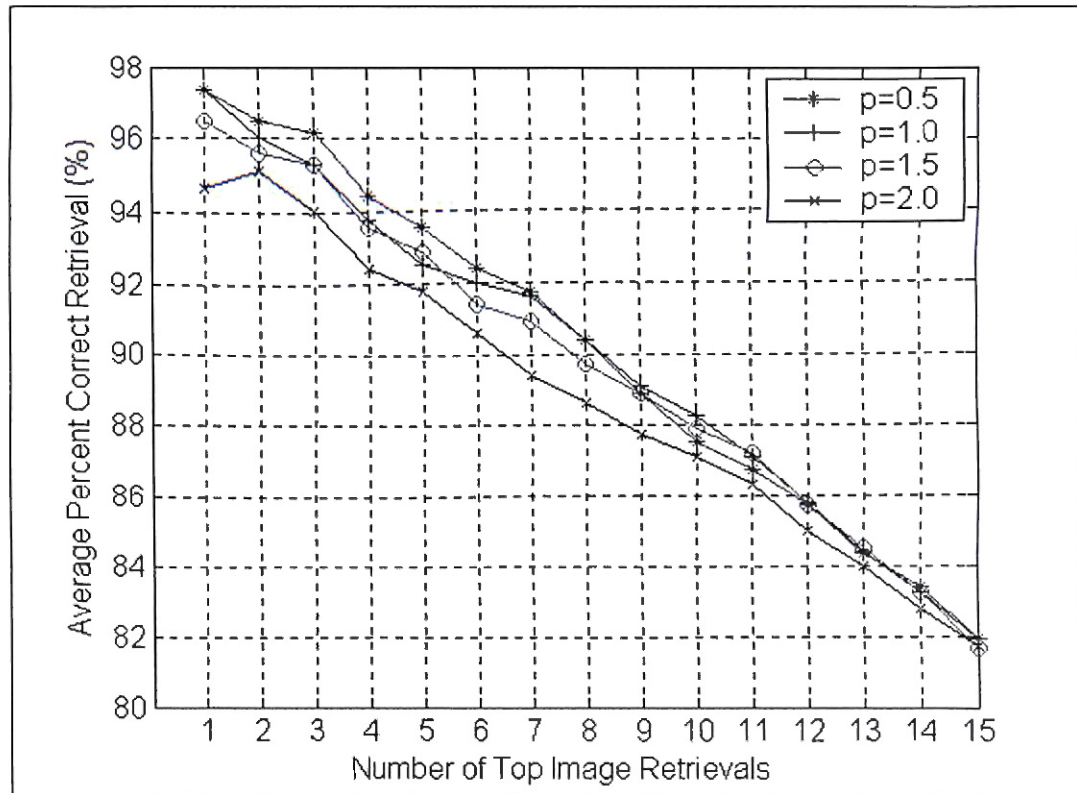


Figure 30. Performance of Minkowsky distance with normalization using gamma distribution.



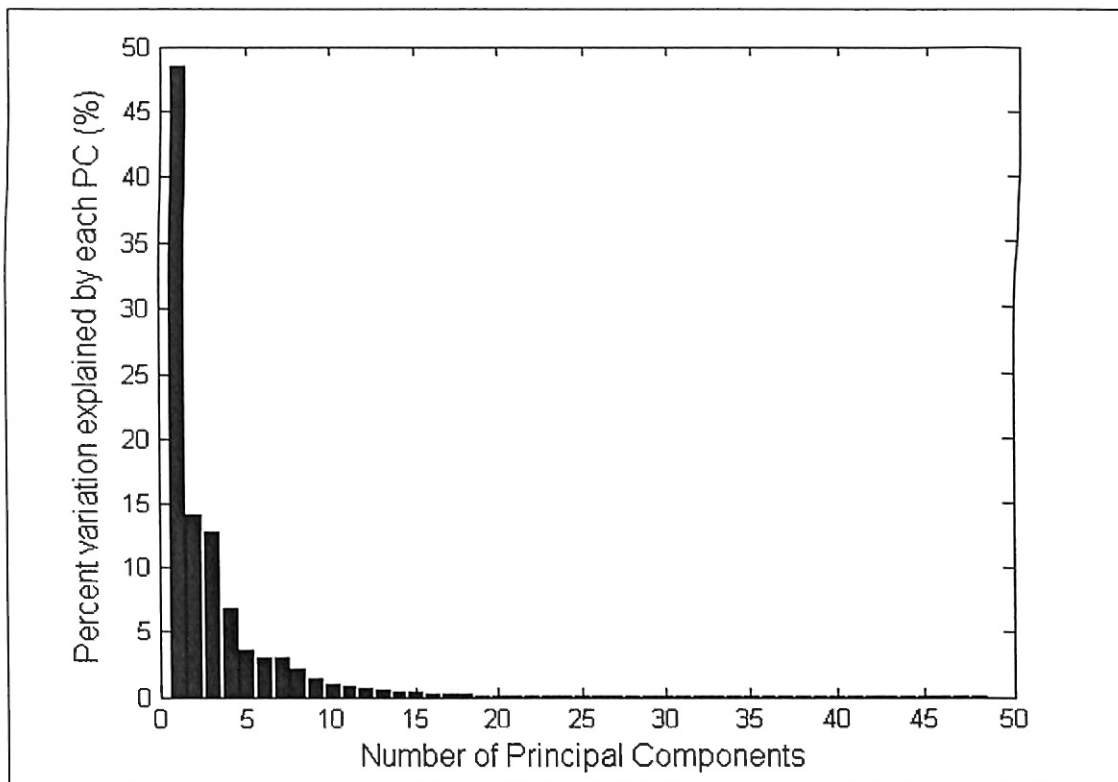
#### ***4.1.9. Fitting a Gamma Density***

Each feature was fitted to a gamma distribution and the probability of the distribution less than or equal to that value was used as a transformed variable. After transformation, Minkowsky distance with different p-values was used to retrieve top 15 images for all test images. Average percent correct retrieval is shown in Figure 30. Even though, p-value of 0.5 performed slightly better than p-value of 1, the later value might be preferred due to its low computation time. Average percent correct retrievals were greater than 90% for up to top 8 retrievals.

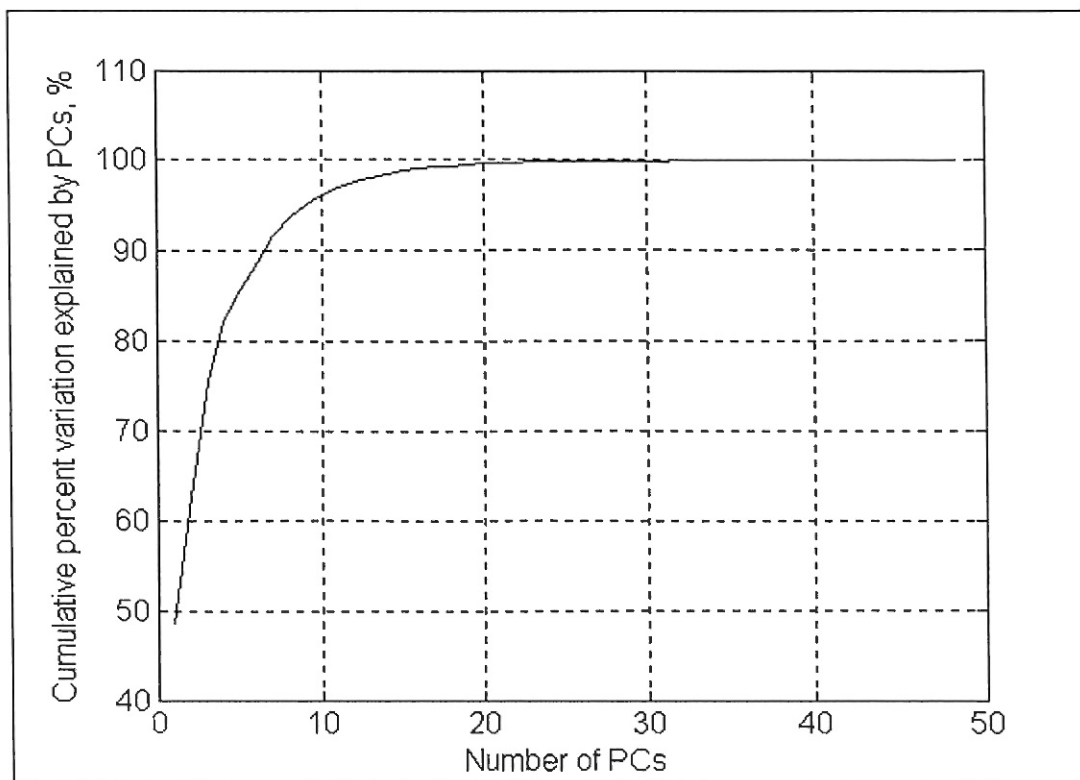
#### ***4.1.10. Principal Component Analysis***

The covariance of the feature matrix was calculated (Eqn. 3.24) and eigenvalue decomposition was performed (Eqns. 3.26 and 3.27). The scree plot, which is a graph of percent variation explained by each PC, is shown in Figure 31. The first principal component explained 48.39% of all variations described by 48 textural features. The second, third, and fourth PCs explained 14.16, 12.86, 6.80, and 3.57 of all variations, respectively. Further PCs explained lesser and lesser variance. Figure 32 shows the cumulative percent variation explained by PCs. For instance, if 4 PCS were retained, we would keep 82.21% of information explained by 48 features. PCA is a dimensionality reduction technique. In some cases, the information lost could be the noise present in original features.

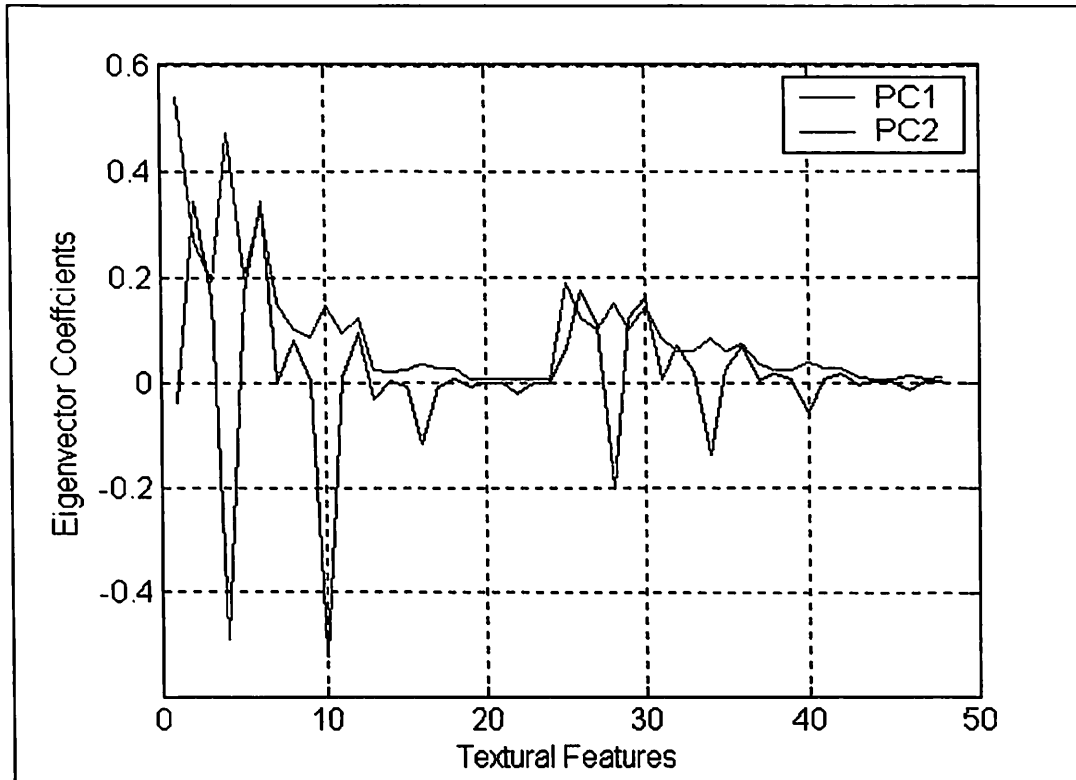
Principal components are linear combinations of original features. Weights are called eigenvectors. Figure 33 shows the eigenvectors for PC1 and PC2. Note that the length (norm) of eigenvectors is 1. Eigenvector 1 is orthogonal to Eigenvector 2.



**Figure 31. Scree plot for evaluating number of principal components.**



**Figure 32. Cumulative percent variation explained by PCs.**

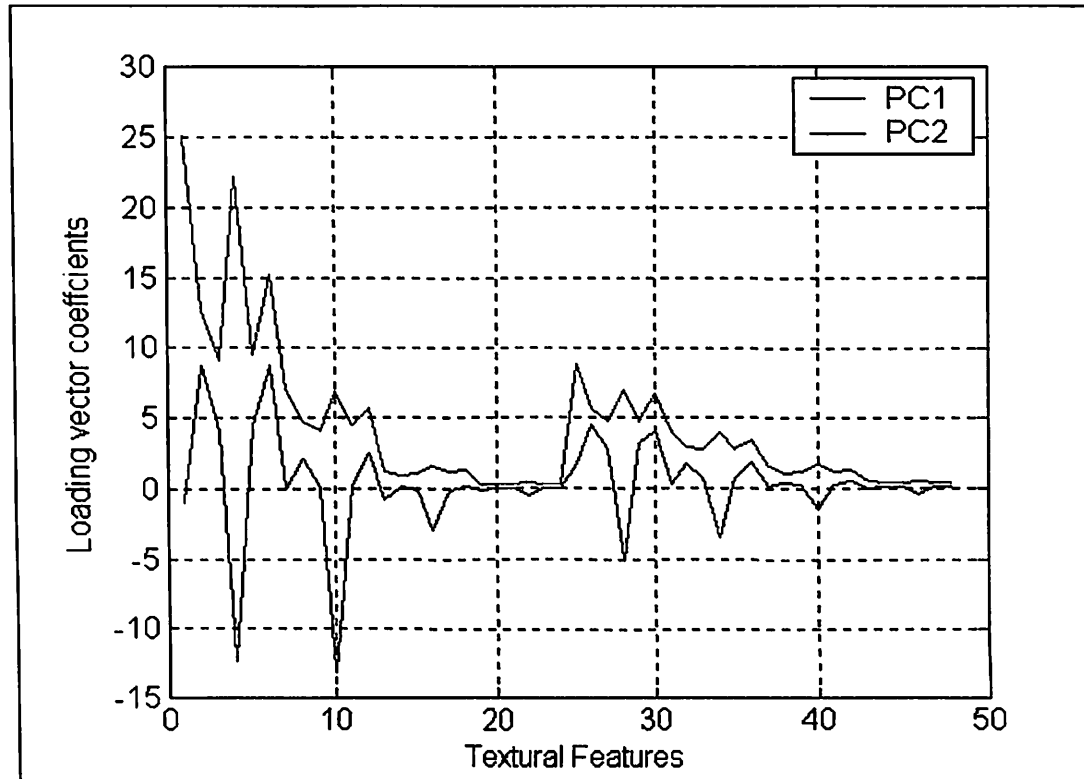


**Figure 33. Eigenvectors for PC1 and PC2.**

Sometimes, eigenvectors give a misleading picture because all eigenvectors are normalized to unit length. In reality, eigenvector 1 is more useful than eigenvector 2. The usefulness of each eigenvector is given by its eigenvalue. Therefore, eigenvector can be scaled by the square root of its eigenvalue to produce a loading vector. The length of the loading vector is the eigenvalue, which is the variance explained by that PC. Figure 34 shows the loading vectors for PC1 and PC2. Note that the magnitude of loading vector 1 is larger than loading vector 2. Still, loading vectors 1 and 2 are orthogonal.

To determine the number of principal components, scree plot or Eqn. 3.32 can be used. The scree plot shows the graph levels off after 4 PCs and therefore 4 PCs can be used for image retrieval. Four PCs explain 82.21% of all variations. If 99% of variation

is desired, then 17 PCs are required (Eqn. 3.32). Image retrieval results were reported separately, when number of PCs used was 4 and 17.



**Figure 34. Loading vectors for PC1 and PC2.**

When 4 PCs were used, the average percent correct retrievals were less than 80% (Fig. 35). This shows that the information lost was useful for image retrievals. When 17 PCs were used, the average percent correct retrievals were improved (Fig. 36). However, the performance was poorer than other normalization techniques. Preprocessing of textural features by principal component analysis was not suitable for Gabor features for image retrieval application.

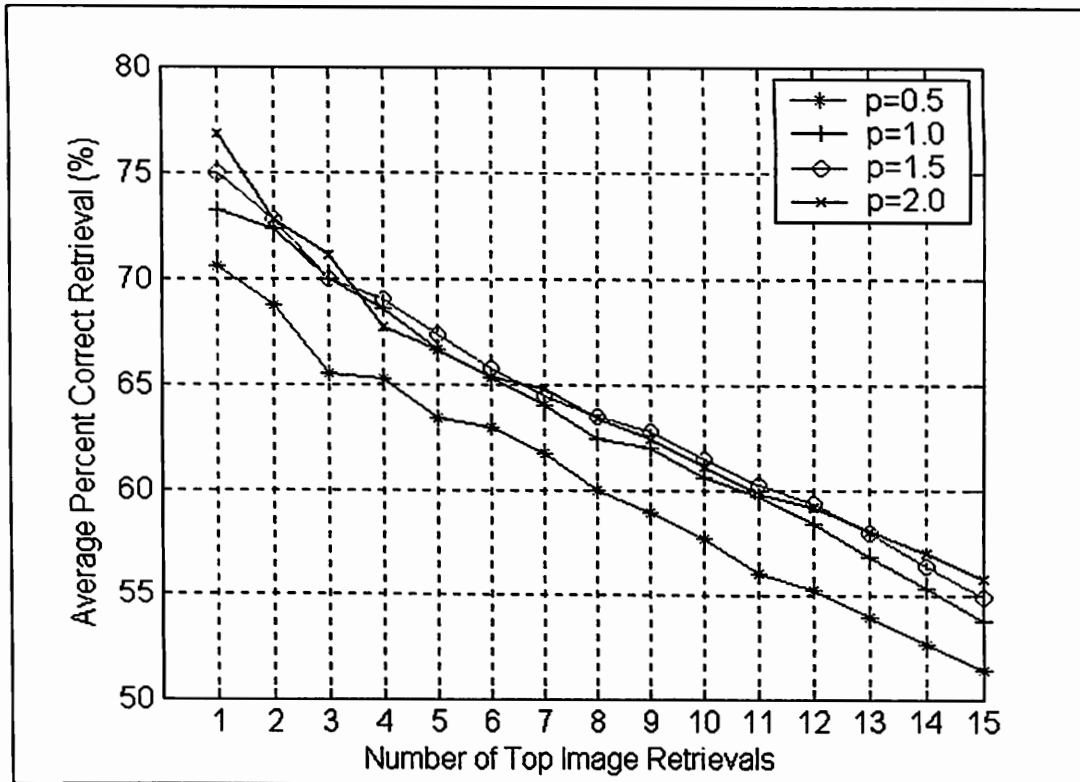


Figure 35. Minkowsky measure with PCA preprocessing with 4 PCs.

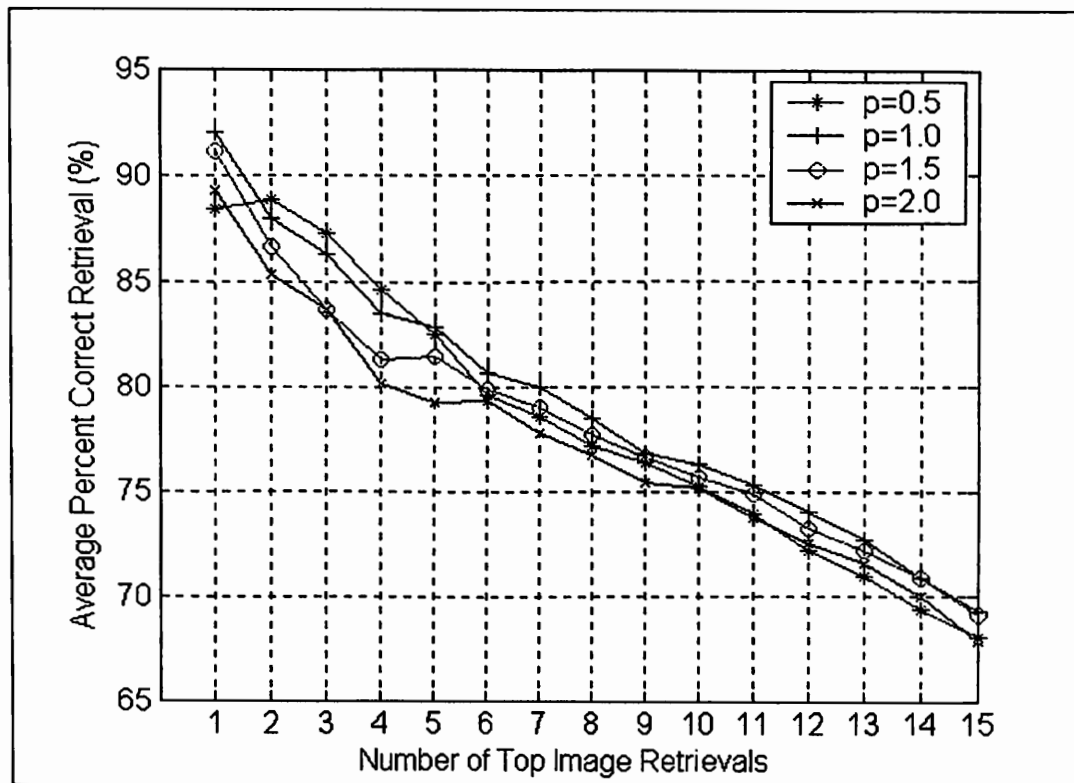
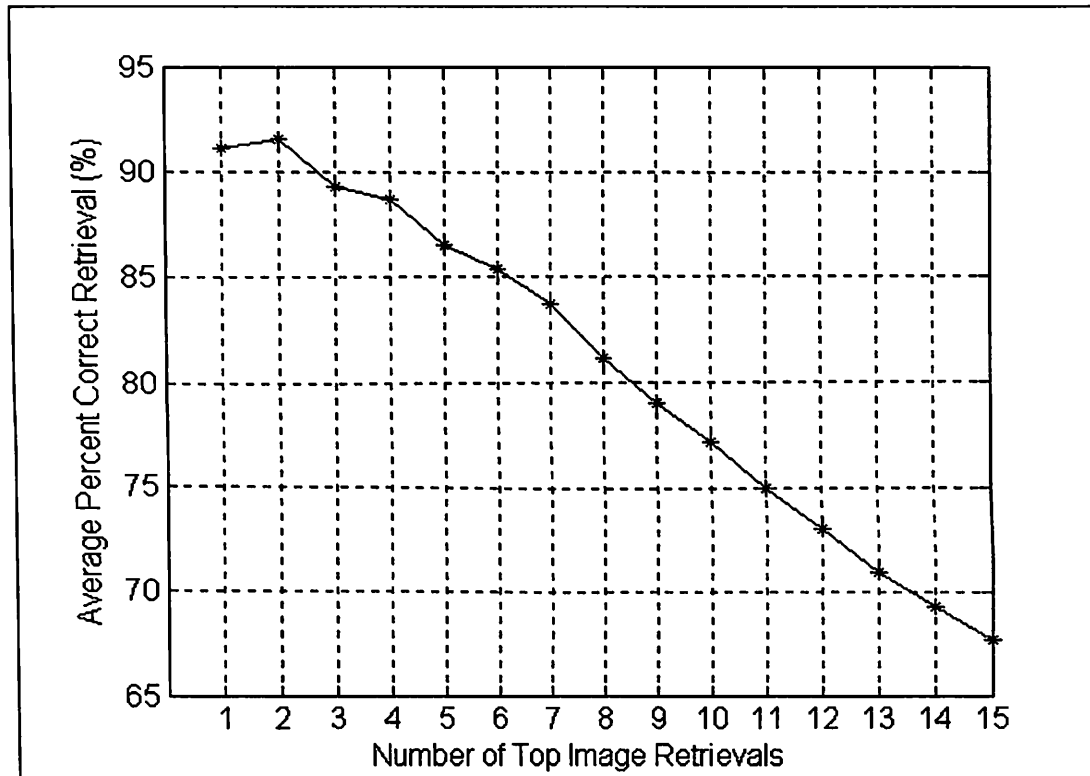


Figure 36. Minkowsky measure with PCA preprocessing with 17 PCs.

## 4.2. Mahalanobis Distance

Mahalanobis distance does not require any normalization, as this measure incorporates covariance matrix in the calculation (Eqn. 3.9). Figure 21 shows the performance of Mahalanobis distance measure without any normalization of Gabor textural features.



**Figure 36. Performance of Mahalanobis distance measure**

Average percent correct retrieval was plotted against the number of image retrievals. The average correct percent retrievals were greater than 85% for up to top 6 retrievals (Fig. 36). Average percent correct retrieval was greater than 90% for only up to top 2 retrievals. Performance of Mahalanobis distance measure was poorer than that of Minkowsky distance measure with various normalizations.

### 4.3. Comparison of Similarity Measures

Average percent correct retrieval was greater than 90% for only up to top 2 retrievals for Mahalanobis distance and the computation time was 0.24 s. Minkowsky distance performed better than Mahalanobis distance. Average percent correct retrievals with greater than 90% were obtained for up to top 8 retrievals for many normalization methods (Table 1). As expected, the computation time was the least for Minkowsky distance without any normalization as there was no preprocessing of features involved in the calculation. Manhattan distance ( $p=1$ ) with 'minmax' normalization had the least computation times, only next to no normalization. At the same time, performance of the Manhattan distance with 'minmax' normalization was one of the best with average percent correct retrieval greater than 90% for up to top 8 retrievals. Fitting various density functions such as normal, exponential, lognormal, gamma, or uniform did not improve the accuracy of image retrieval. Preprocessing of features by PCA reduced the accuracy of image retrieval. Therefore, Manhattan distance with 'minmax' normalization performed the best for image retrieval in Brodatz image database.

Manhattan distance with 'minmax' normalization achieved an average percent correct retrieval of 82.16% for top 15 retrievals. Manjunath and Ma (1996) included all test images in the training image database and reported an average percent correct retrieval of 74.37%. Hatipoglu et al. (2000) also included all test images in training image database and achieved an average percent correct retrieval of 81.3% for top 16 retrievals using textural features extracted by complex wavelet transform. Huang and Chang (1999) used orthogonal wavelet transform and discrete cosine transform to extract textural features from images in Brodatz album and reported an average percent correct

retrieval of 71% for top 15 retrievals. Zhou et al. (2001) extracted textural features using local Fourier transform and reported an average percent correct retrieval of 72% for top 15 retrievals. Xu et al. (2000) divided each image in the Brodatz album into 9 non-overlapping sub-images and reported an average percent correct retrieval of 83% for top 9 retrievals using textural features extracted by multi resolution simultaneous autoregressive model and Bhattacharya similarity measure. This study (Manhattan distance with 'minmax' normalization) achieved an average percent correct retrieval of 89.78% for top 9 retrievals. Guo et al. (2001) divided each image in the Brodatz album into 49 non-overlapping sub-images and used first 33 sub-images in the training image database and the last 16 sub-images in the test image database. They reported an average percent correct retrieval of 87.61% for top 15 retrievals, which is higher than what is reported in this study. However, Guo et al. (2001) algorithm retrieved top 15 images out of possible 33 correct images available in the training image database. In contrast, this study retrieved top 15 images out of possible 15 correct images available in the training image database. Studies by Guo et al. (2001) and Xu et al. (2000) are not directly comparable to this study, as those studied had not divided the images in to 16 sub-images. The image retrieval performance achieved in this study is higher than the image retrieval performances reported in the literature (Table. 2).



Table 1. Effect of normalization of features for Minkowsky distance on performance of image retrieval.

Normalization	Time (s)				No. of top retrievals with > 90% accuracy			
	p =0.5	p =1.0	p=1.5	p=2.0	p = 0.5	p = 1.0	p = 1.5	p = 2.0
None	0.19	0.16	0.24	0.15	5	3	1	1
Minmax	0.22	0.18	0.27	0.19	8	8	8	7
Meanstd	0.23	0.20	0.29	0.20	8	8	8	7
Rank	0.26	0.23	0.32	0.25	7	8	7	6
Normal	0.66	0.65	0.74	0.64	8	8	7	5
Lognormal	0.69	0.69	0.77	0.68	8	8	7	7
Exponential	0.54	0.54	0.62	0.53	8	8	7	7
Gamma	2.56	2.44	2.50	2.42	8	8	7	6
Unifom	13.39	13.30	13.25	13.04	7	8	7	6
PCA - 4 PCs	4.16	4.01	4.01	3.99	0	0	0	0
PCA - 17 PCs	4.60	4.32	4.12	4.14	0	1	1	0

Table 2. Comparison of performance of image retrieval with literature.

<b>Reference</b>	<b>Textural Feature Extraction Method</b>	<b>Similarity Measure</b>	<b>Preprocessing</b>	<b>Number of Top Retrievals</b>	<b>Average % correct retrieval</b>
This study	Gabor filter	Manhattan	minmax	15	82.16%
Manjunath and Ma (1996)	Adaptive Gabor filter	Manhattan	meanstd	15	74.37%
Hatipoglu et al. (2000)	Complex wavelet transform	Euclidean	none	16	81.30%
Huang and Chang (1999)	Discrete cosine transform	Manhattan	meanstd	15	71%
Zhou et al. (2001)	Local Fourier transform	Manhattan	none	15	72%
Xu et al. (2000)	Autoregressive model	Bhattacharya	none	9	83%
Guo et al. (2001)	Support vector machines	Boundary distance	rank	15	87.61%

#### 4.4. Graphical User Interface

A graphical user interface (GUI) was built so that the user can open a test image and visualize the retrieved images. The user can select Mahalanobis or Minkowsky distance measure. If Minkowsky distance measure was selected, the user had an option to enter any p-value and any normalization techniques. Distance versus number of images retrieved was plotted and shown in the GUI. Under each image, ID of image retrieved was displayed.

##### 4.4.1. Testing with Brodatz image database

Some examples of successful image retrievals were shown in Figures 37-41. These figures demonstrate that the Gabor filter can retrieve similar textural images, even though there were small differences existed between test image and retrieved images.

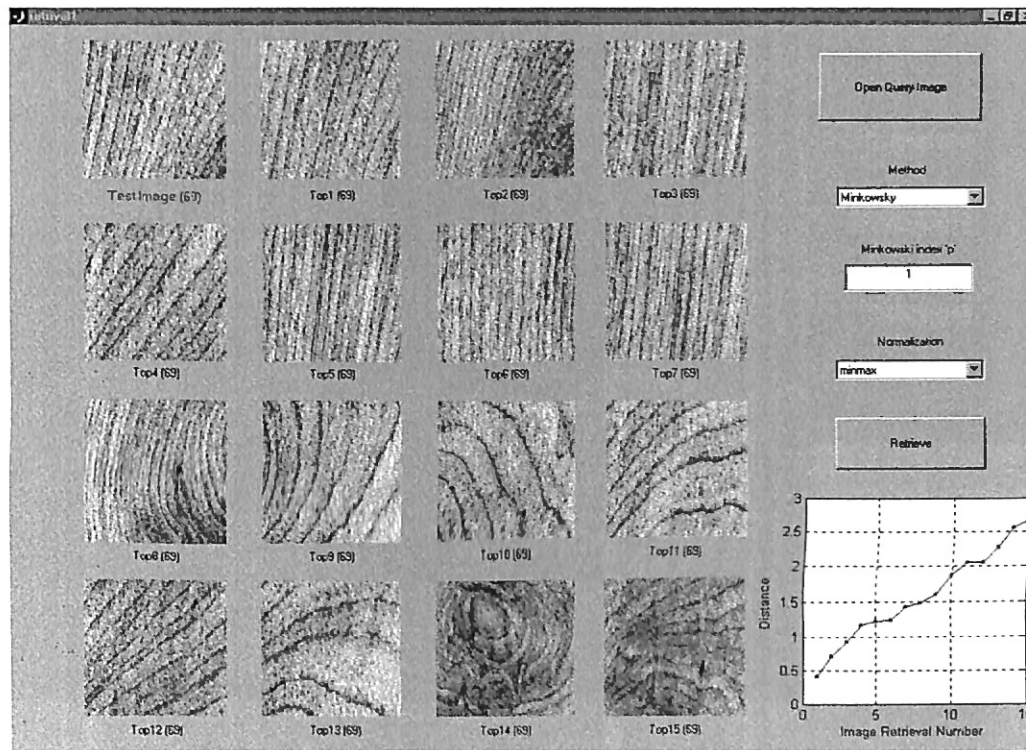


Figure 37. Image retrieval for test image of wood grain

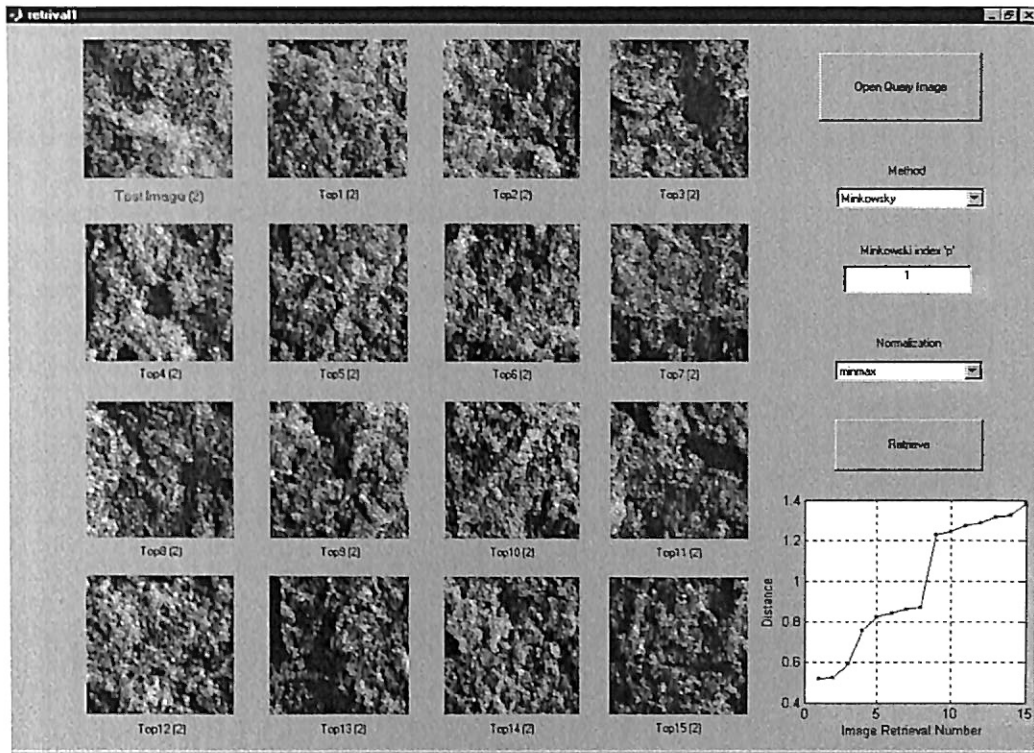


Figure 38. Image retrieval for test image of fieldstone.

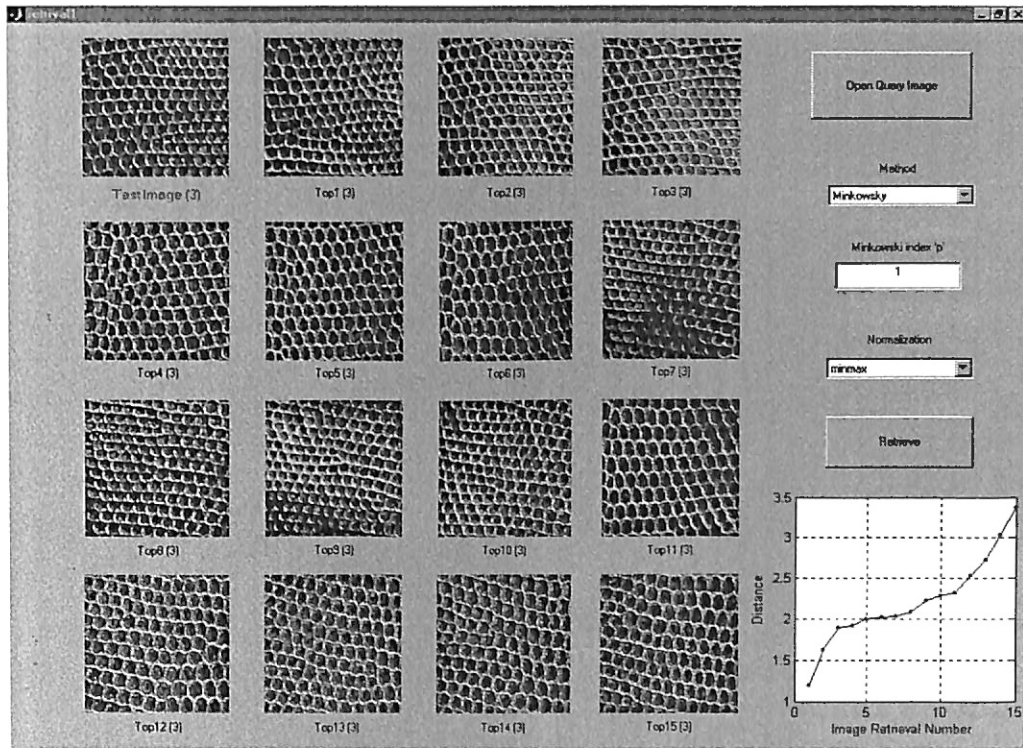


Figure 39. Image retrieval for test image of reptile skin.

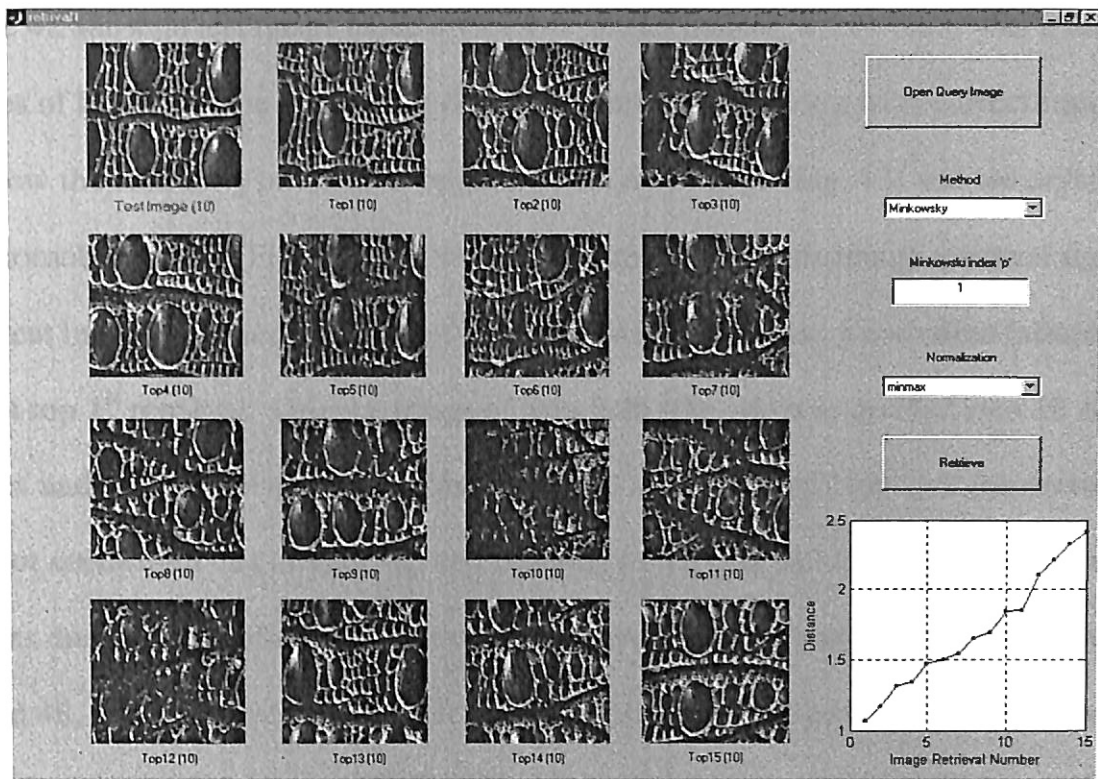


Figure 40. Image retrieval for test image of crocodile skin.

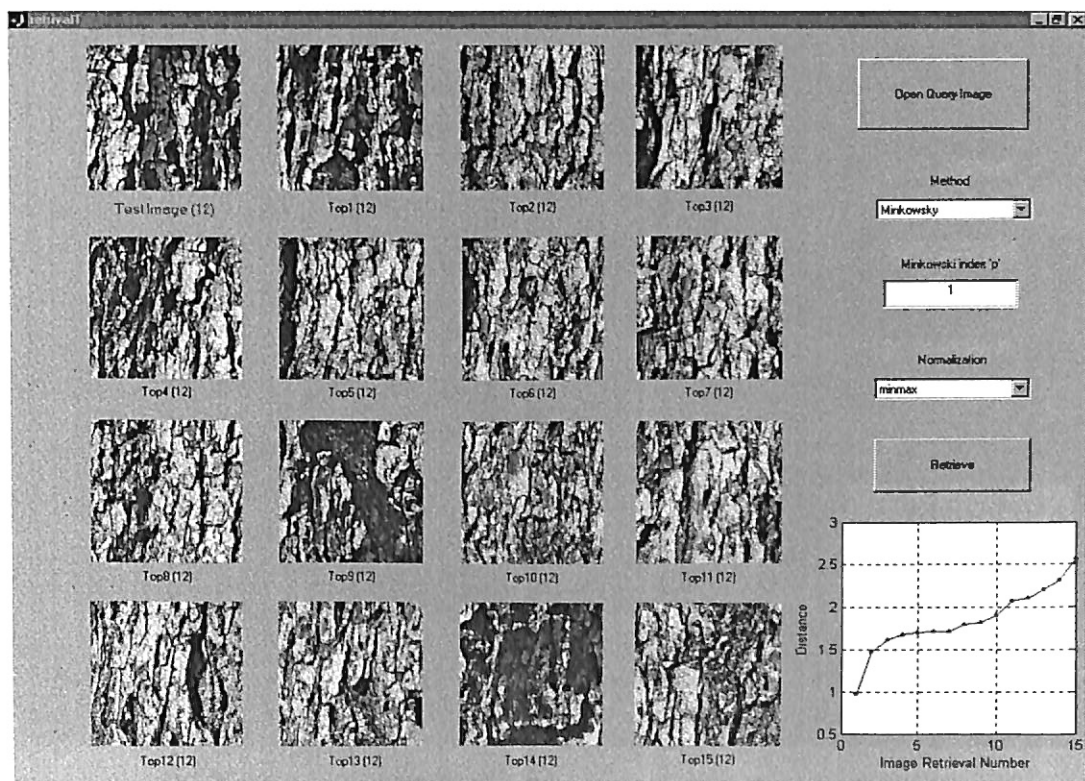


Figure 41. Image retrieval for test image of bark of a tree.

Figure 42 shows a test image of ice crystals on an automobile, where there was a failure. Images of beach sand in the training database were retrieved along with correct images. To show the similarity of textures, original beach sand image (Fig. 43) and ice crystals on an automobile image (Fig. 44) were shown. Figure 45 shows the image retrieval results for a test image of varied swinging of light bulb, where there was a complete failure, except top 1<sup>st</sup> retrieval. Original image of light bulb (Fig. 46) was divided into 16 sub-images and one sub-image was randomly selected as test image. Note that the texture was not uniform in this image (Fig. 46), resulting in retrieval failure. Image retrieval failures due to non-uniformities in texture also were found for images shown in Figures 47 and 48. Poor results were found for test images shown in Figures 49 and 50. The wrongly retrieved images had similar texture, but the negative of the test image.

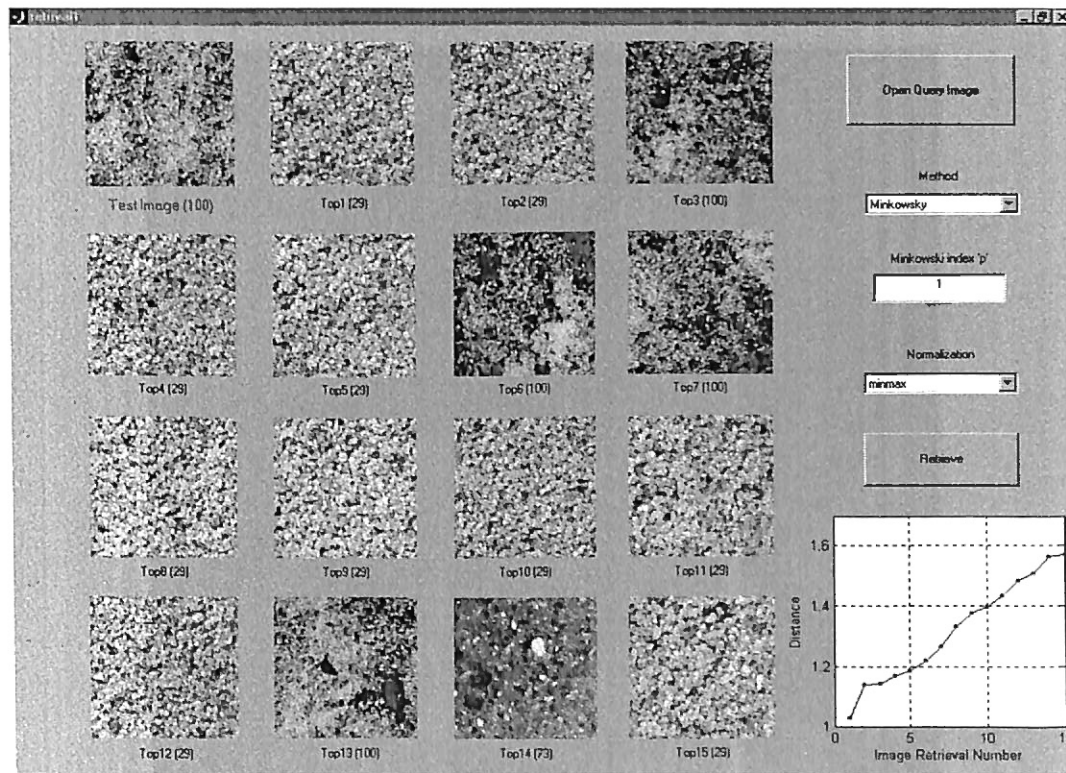
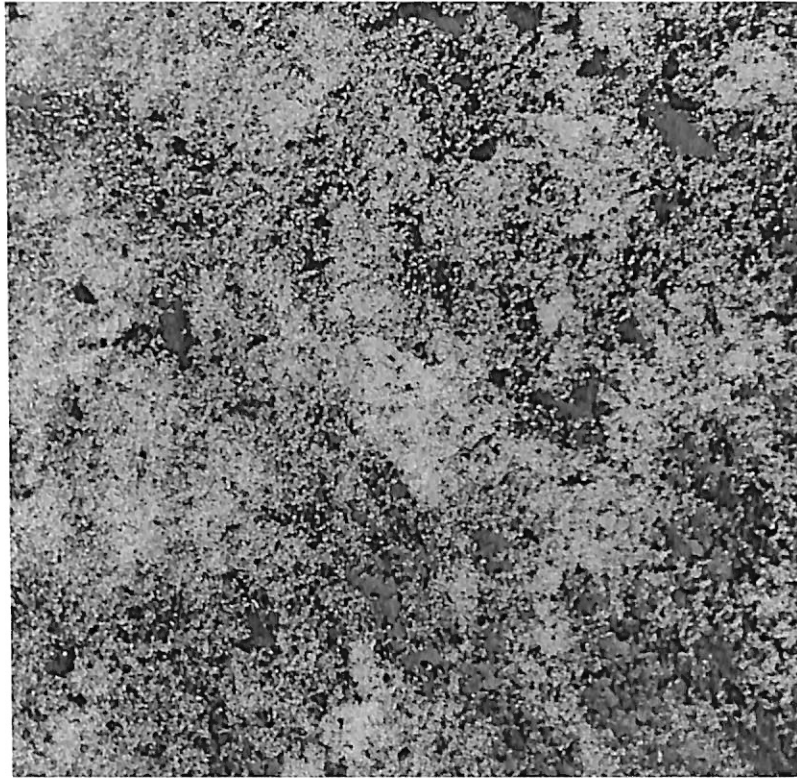
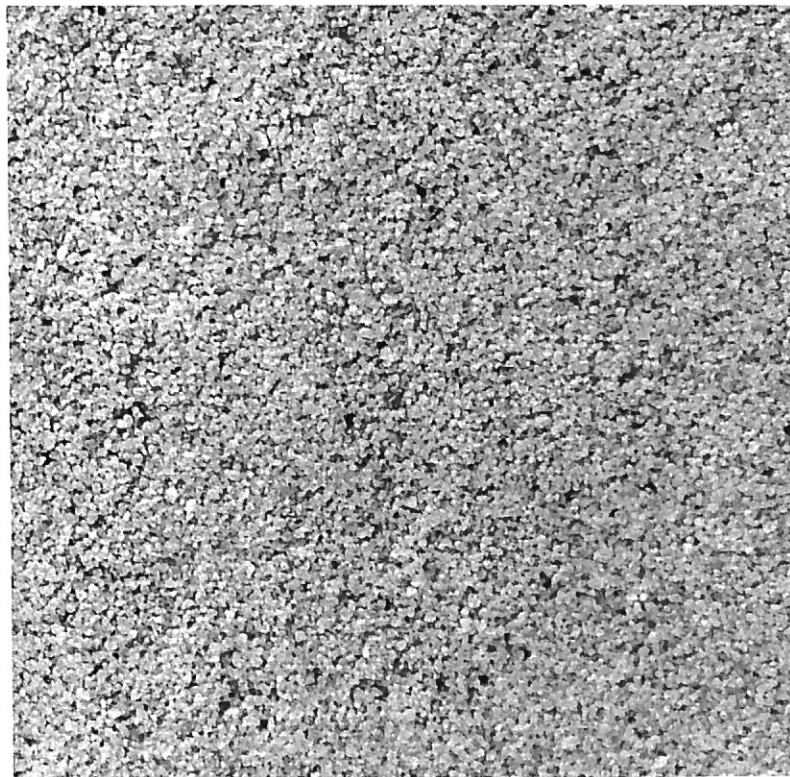


Figure 42. Image retrieval for test image of ice crystals on an automobile.



**Figure 43. Original image of ice crystals on an automobile.**



**Figure 44. Original image of beach sand.**

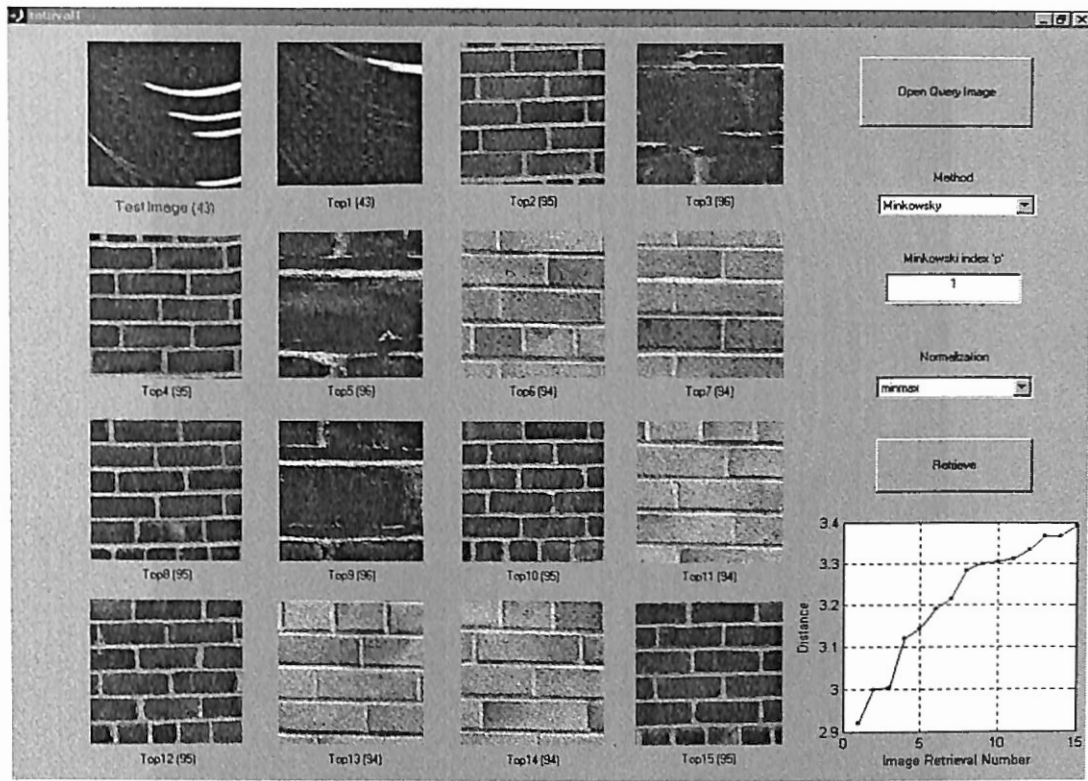


Figure 45. Image retrieval for test image of varied swinging of light bulb.

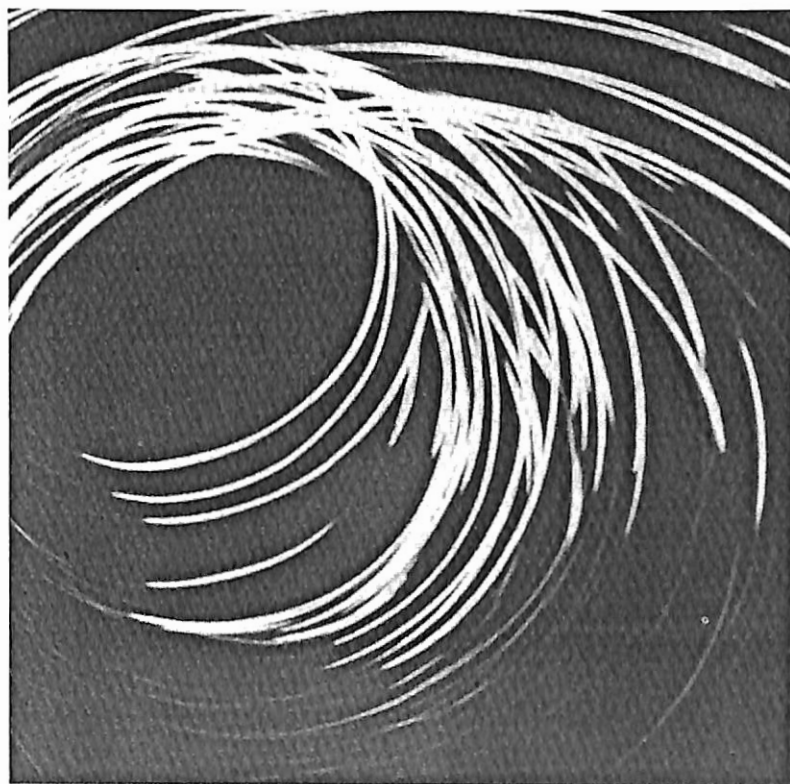
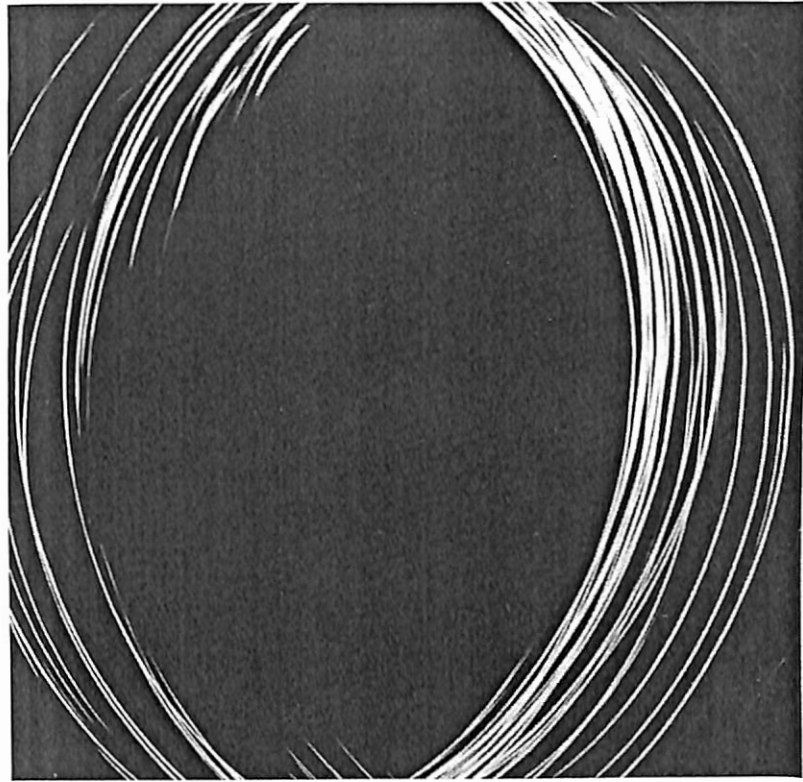


Figure 46. Original image of varied swinging of light bulb.





**Figure 47. Original image of swinging lights in the darkened room.**



**Figure 48. Original image of abstract effect of swinging lights.**

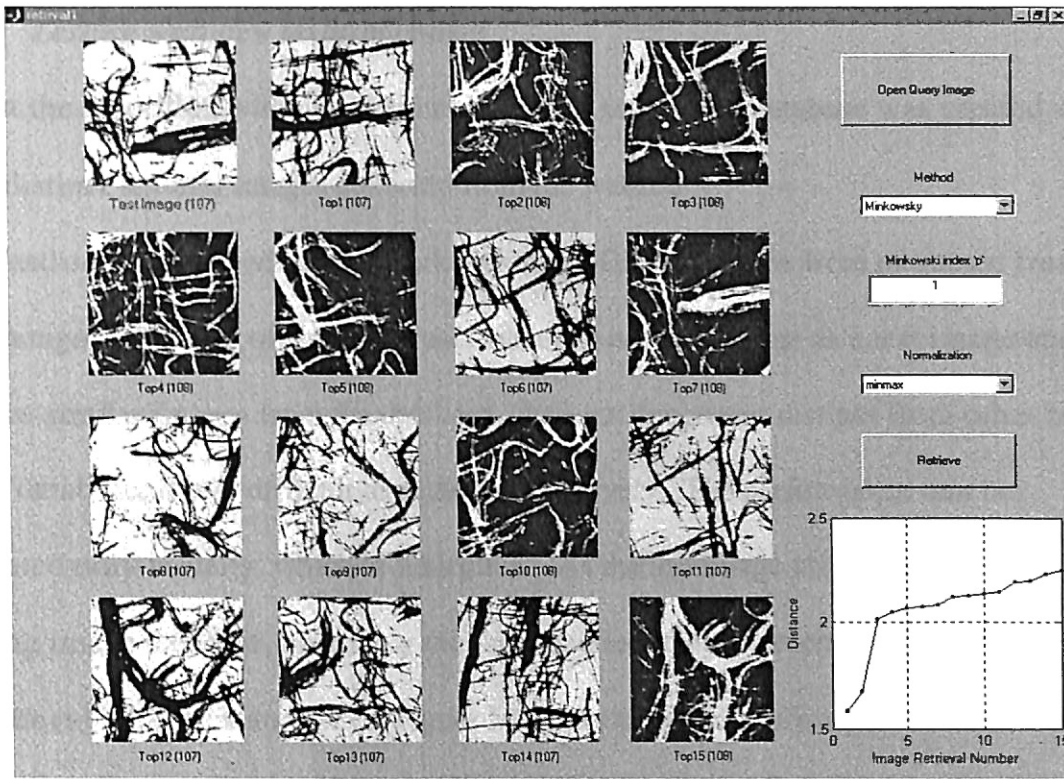


Figure 49. Image retrieval failure for the test image of Japanese rice paper.

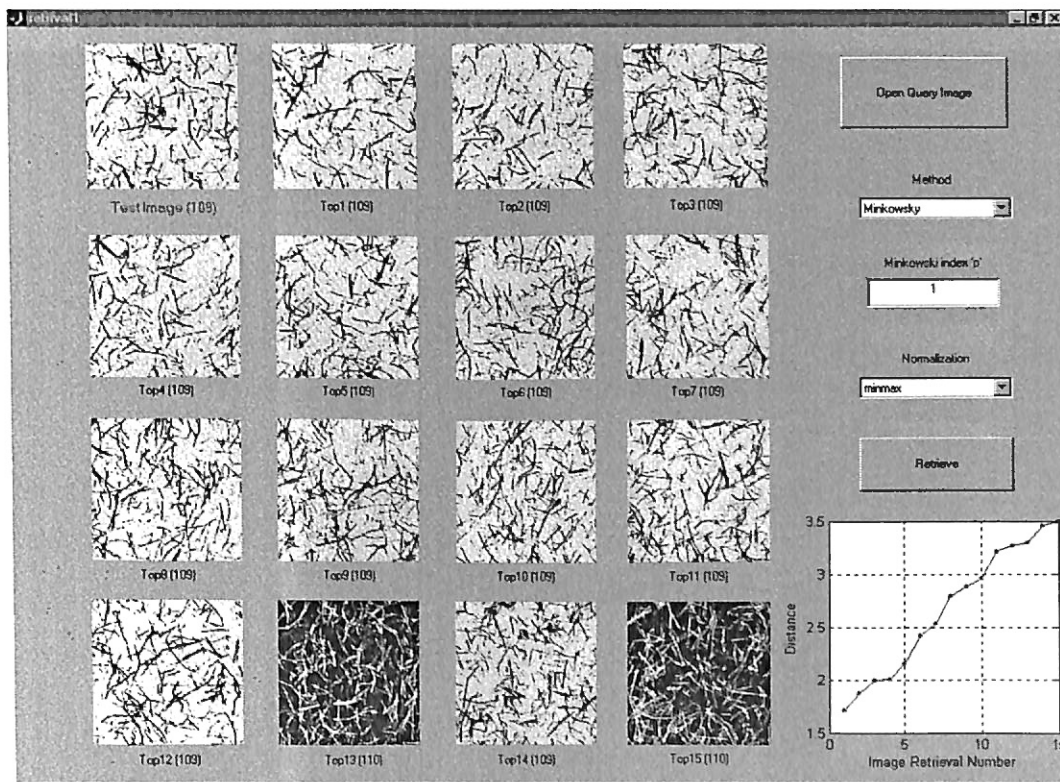


Figure 49. Image retrieval failure for the test image of handmade paper.

#### 4.4.2. Testing with new textural images

To test the algorithm with new textural images, a new image database was created with 1480 distinct textural images collected from the website <http://astronomy.swin.edu.au/~pbourke/texture/>. Gabor features were extracted from each image and stored offline. The user can open any one image as a test image and retrieve similar images from the database. As each image was distinct from other images in the database, it was difficult to evaluate objectively. The performance can be evaluated only visually, which is subjective. As the test image also was present in the training image database, invariably the same queried image was correctly retrieved as top 1<sup>st</sup> retrieval with a distance of 0. Figure 50 shows the results of image retrieval, when a square textural pattern image was given as the test image. The retrieved images also had square or rectangular or line patterns.

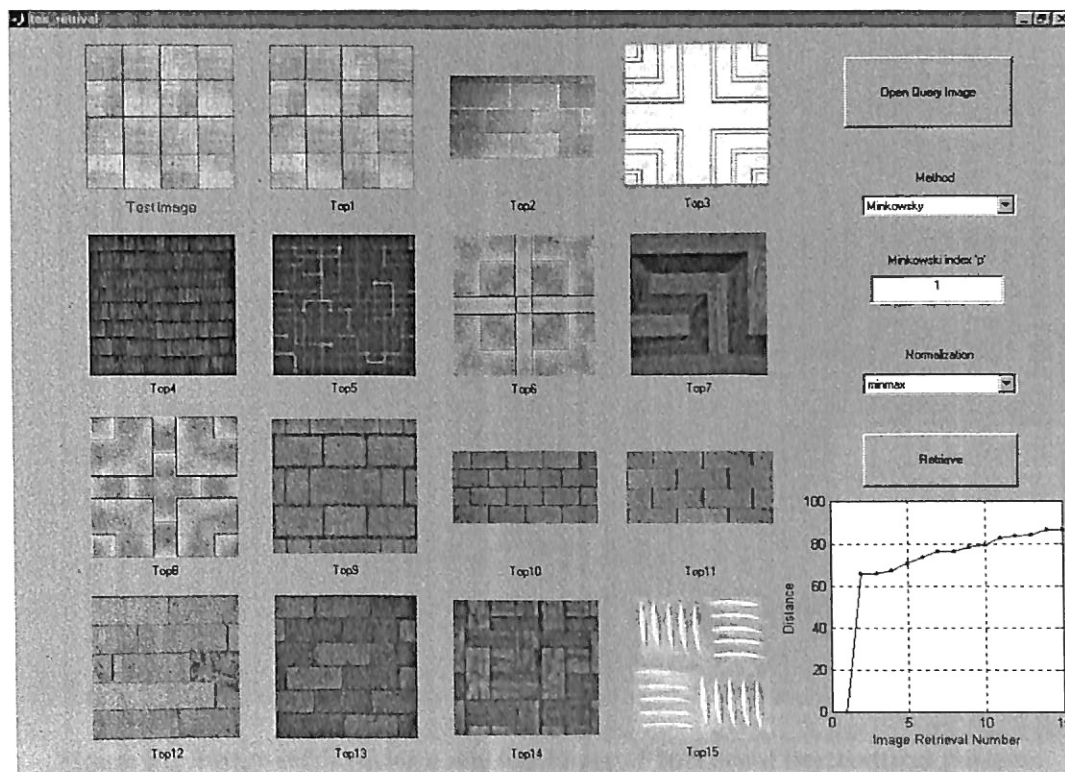


Figure 50. Image retrieval for a real test image of square textural pattern.

When an image with a horizontal line pattern was presented as the test image, the retrieved images also had horizontal line patterns (Figure 51). Figure 52 shows the retrieval results, when an image of pebbles was presented. Most of the retrieved images had similar pebble patterns. Similar results were obtained for two other test images (Figures 53 and 54). Figure 55 shows the image retrieval results, when an image of a tree was presented. It was interesting to see that the top 7 retrieved images also contained a tree. There were no more tree images in the training image database and therefore other irrelevant images were retrieved. Obviously, a large training image database would improve the results and users were also typically more interested in top few retrievals. Note that color, lighting, and size of the image did not influence the retrieved results. Images were retrieved based on textural content alone.

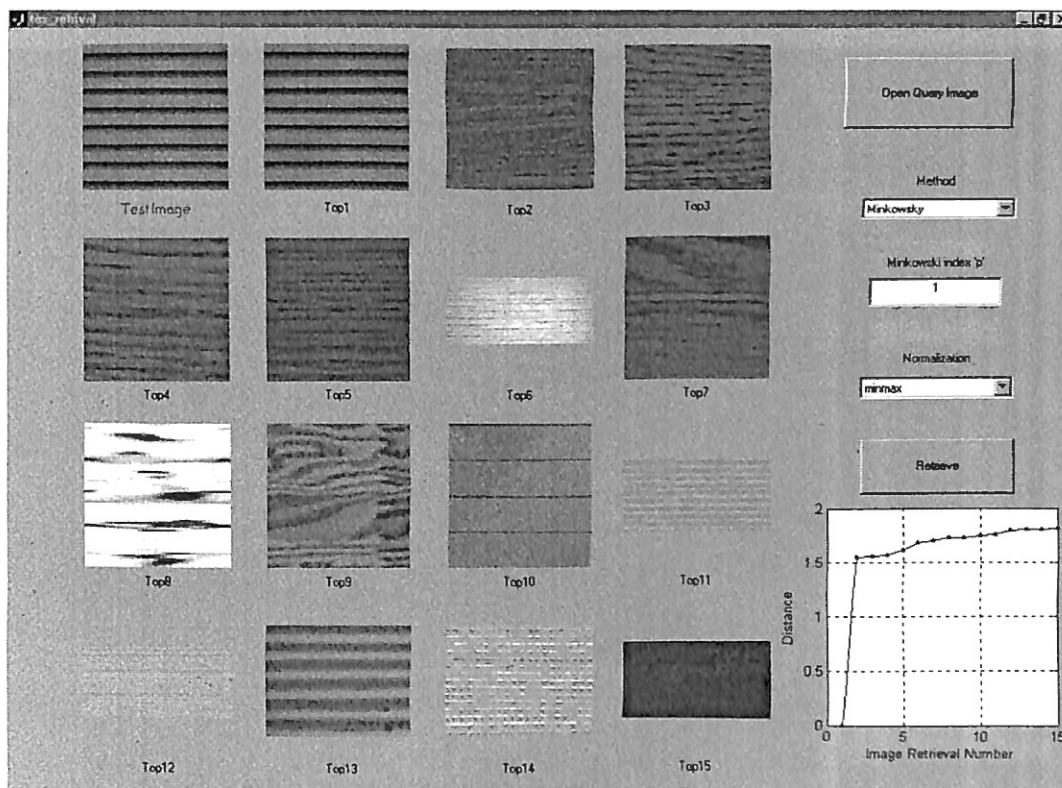


Figure 51. Image retrieval for a real test image of horizontal line textural pattern.

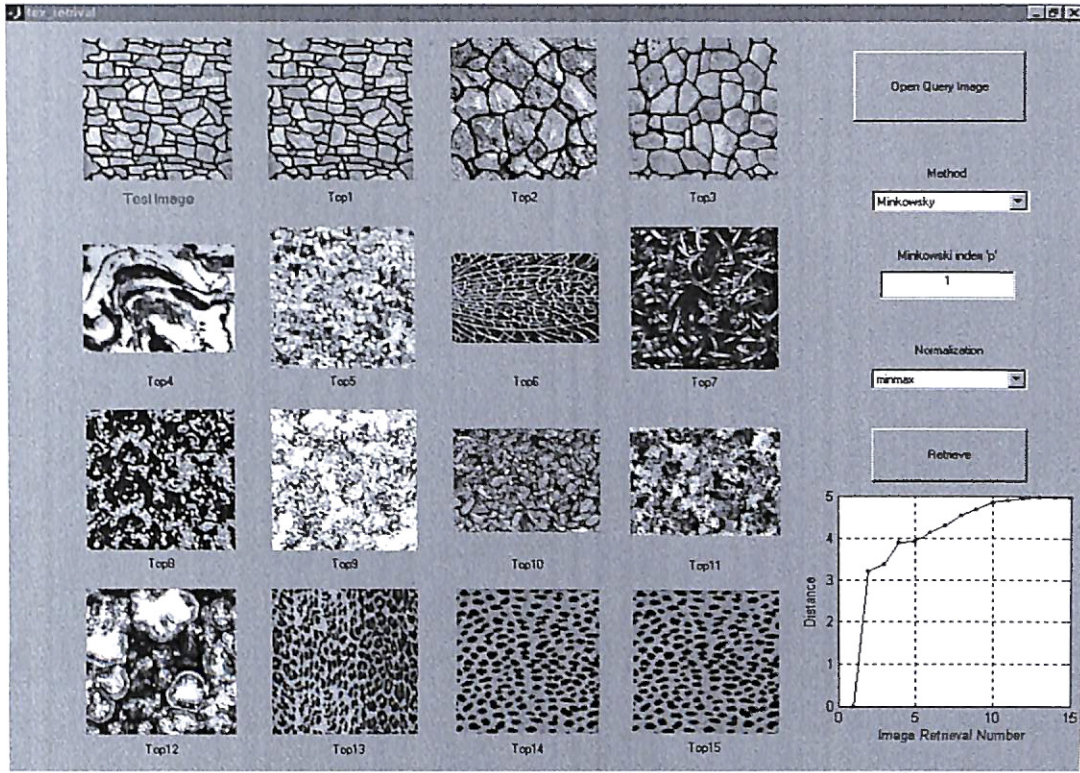


Figure 52. Image retrieval for a real test image of pebble textural pattern.

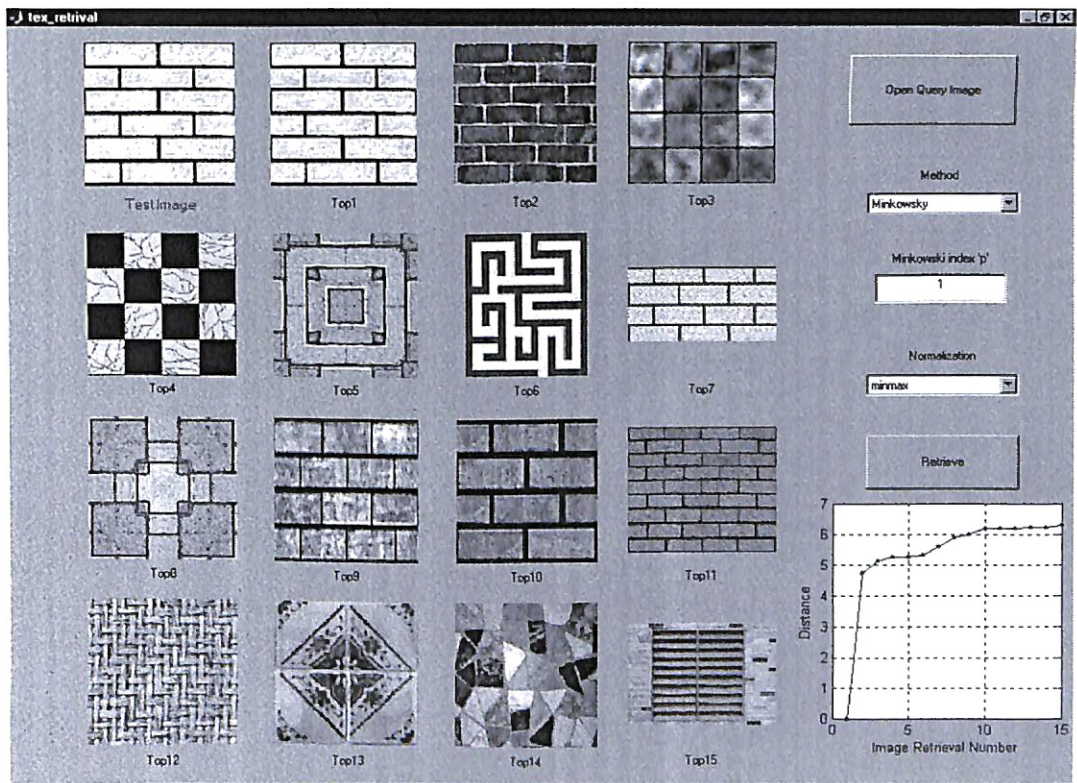


Figure 53. Image retrieval for a real test image of brick textural pattern.

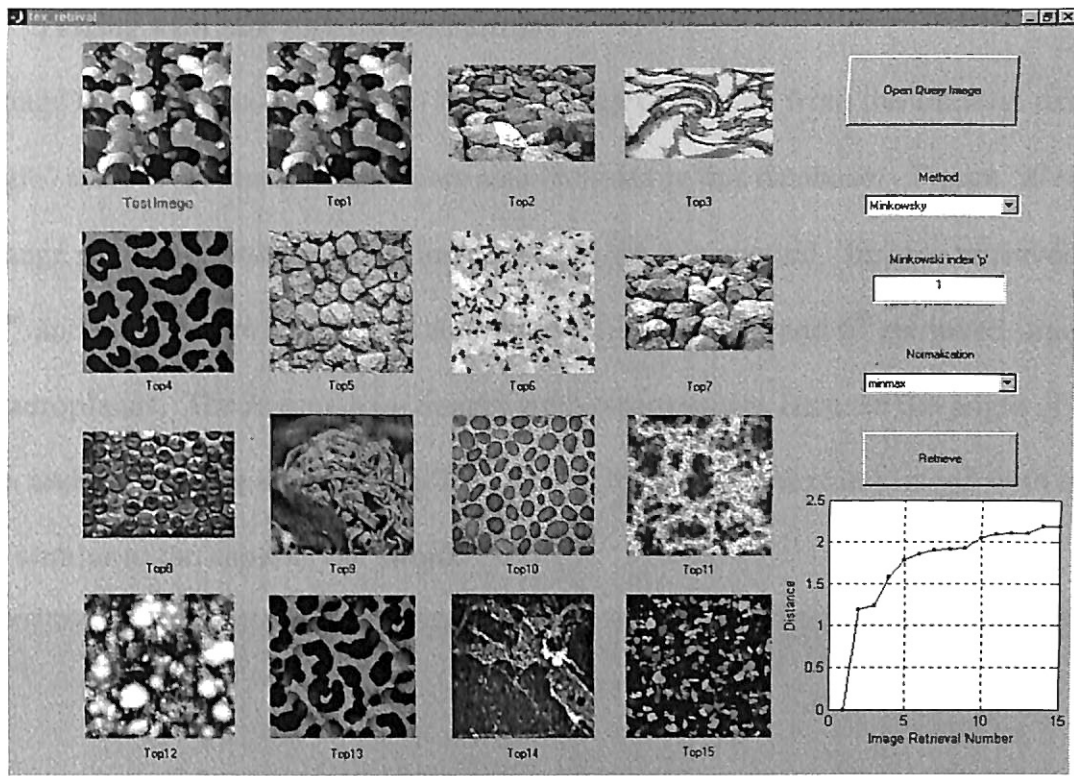


Figure 54. Image retrieval for a real test image of capsule textural pattern.

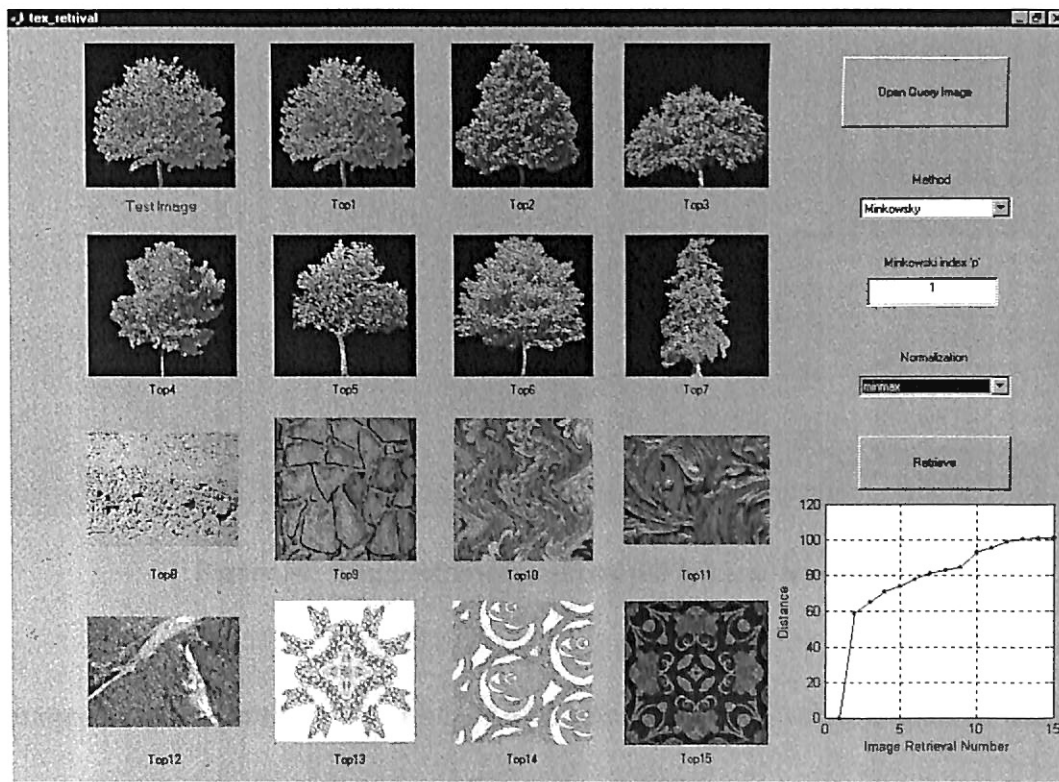


Figure 55. Image retrieval for a test image of a tree.

#### 4.4.3. Testing with new non-textural images

An image database was created with few real images collected from the Internet using a 'Google' search. Textural images were also included in this database. Figure 56 shows the image retrieval results when an image of a sword was queried. Images retrieved as top 2<sup>nd</sup> and top 3<sup>rd</sup> were similar, but slightly different. Top 5<sup>th</sup> and 6<sup>th</sup> retrieved images were aeroplanes. The reason these images were retrieved was because the angle of the planes and swords was similar. Top 7<sup>th</sup> retrieved image was a textural image with an angle similar to the angle of the sword.

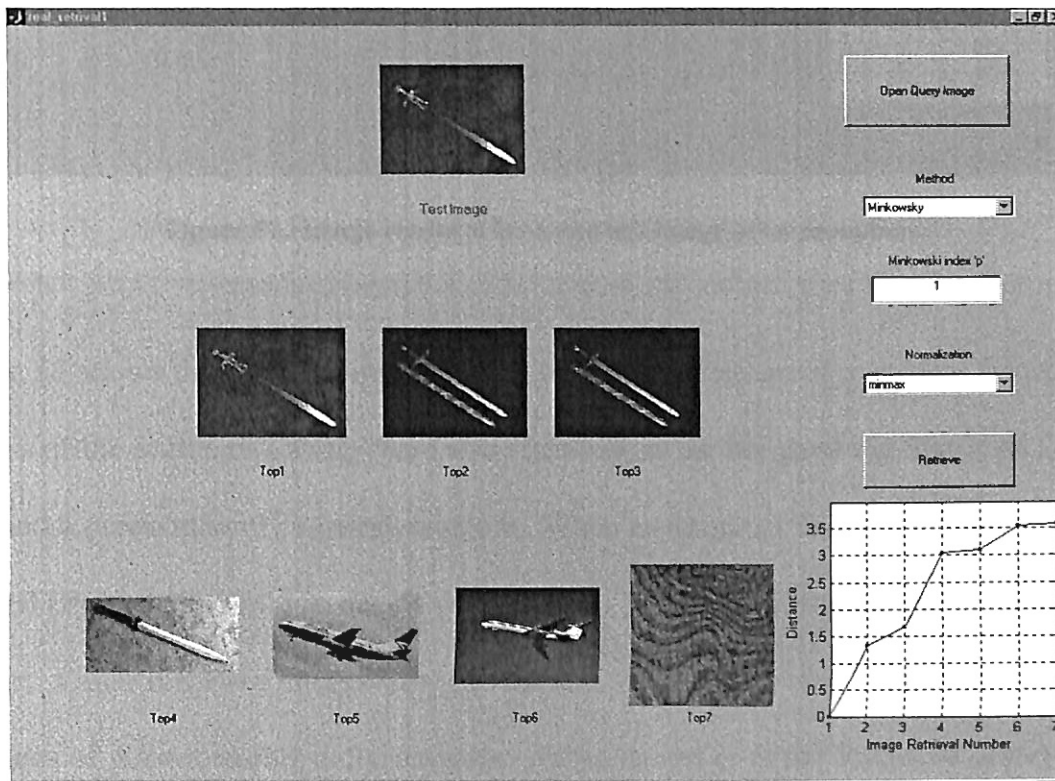
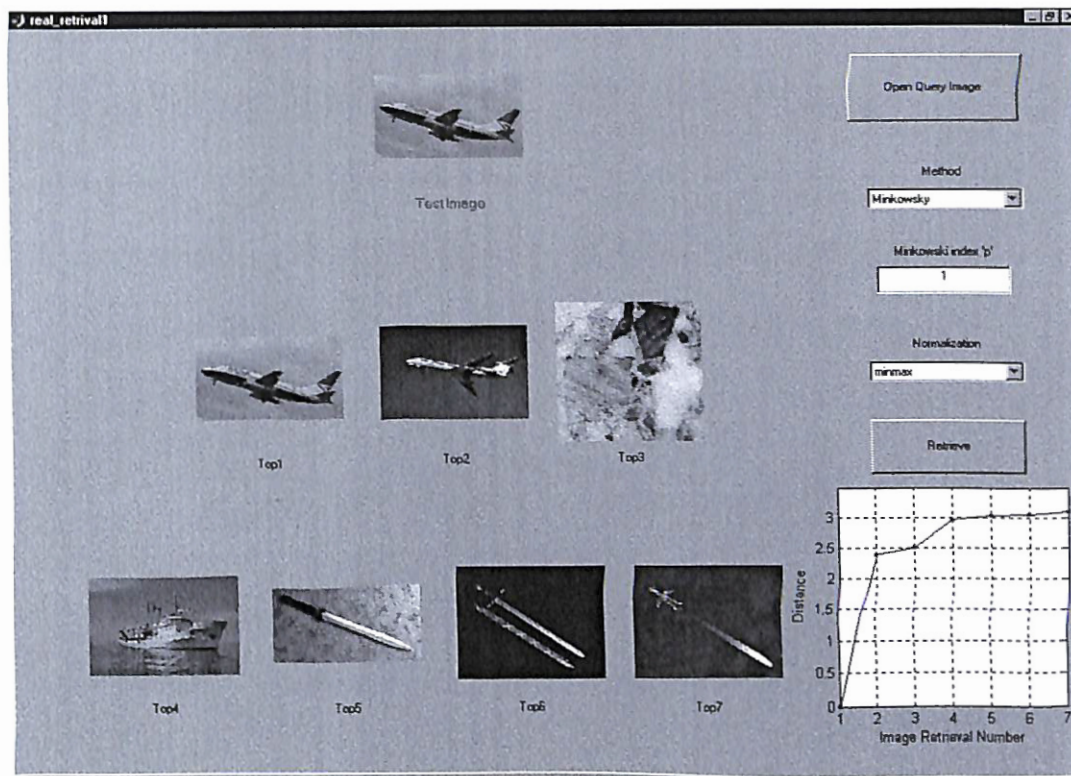


Figure 56. Image retrieval for a real test image of a sword.

The image retrieval result for a test image of a plane (Figure 57) was similar to those for the sword (Figure 56). Top 3 retrieved image was a failure.



**Figure 57. Image retrieval for a real test image of an aeroplane.**

Figure 58 shows the retrieved image results for a queried image of a vertical sword. Images of the Statue of Liberty were also retrieved, as the image of the Statue of Liberty also had a predominantly vertical structure. When an image of the American flag was queried (Figure 59), top 2 retrievals were correct. Top 5<sup>th</sup> retrieved image was also correct. Other retrieved images were not similar. It should be noted that there were only 3 images of American flag in the database. When an image of the Statue of Liberty was queried (Figure 60), top 3 retrievals were correct and top 4<sup>th</sup> and 5<sup>th</sup> retrievals had some vertical components. When an image of a cat was queried (Figure 61), top 3 retrievals were correct and others were failures.



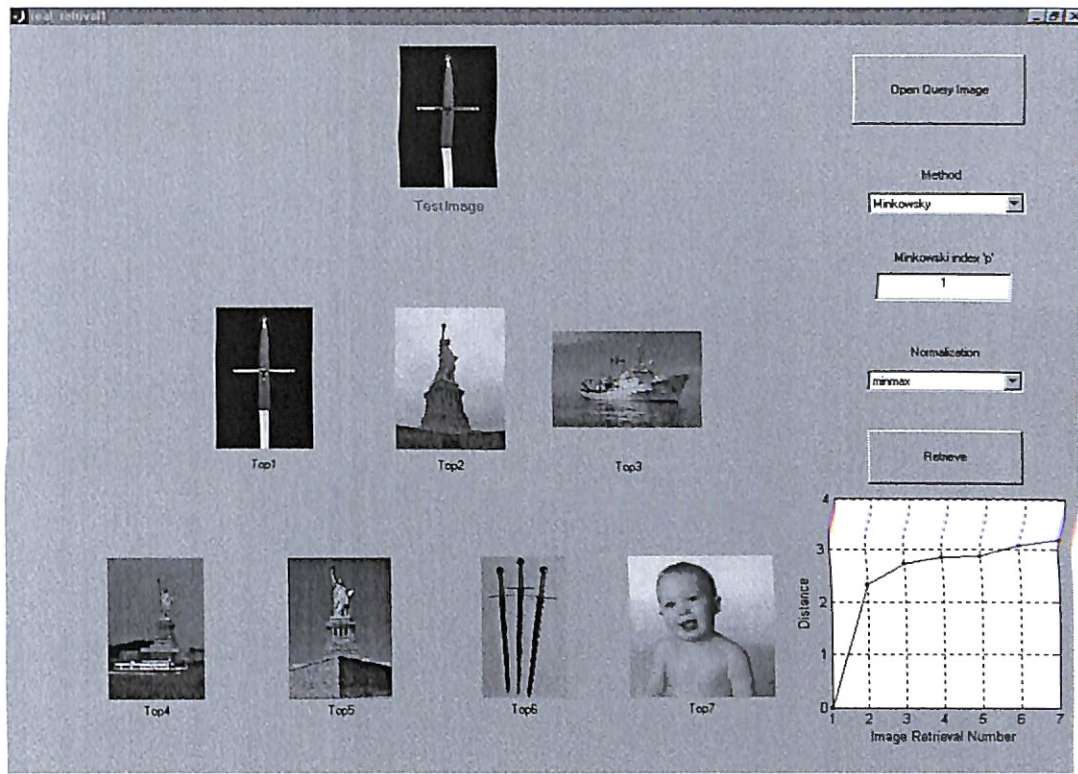


Figure 58. Image retrieval for a real test image of a vertical sword.

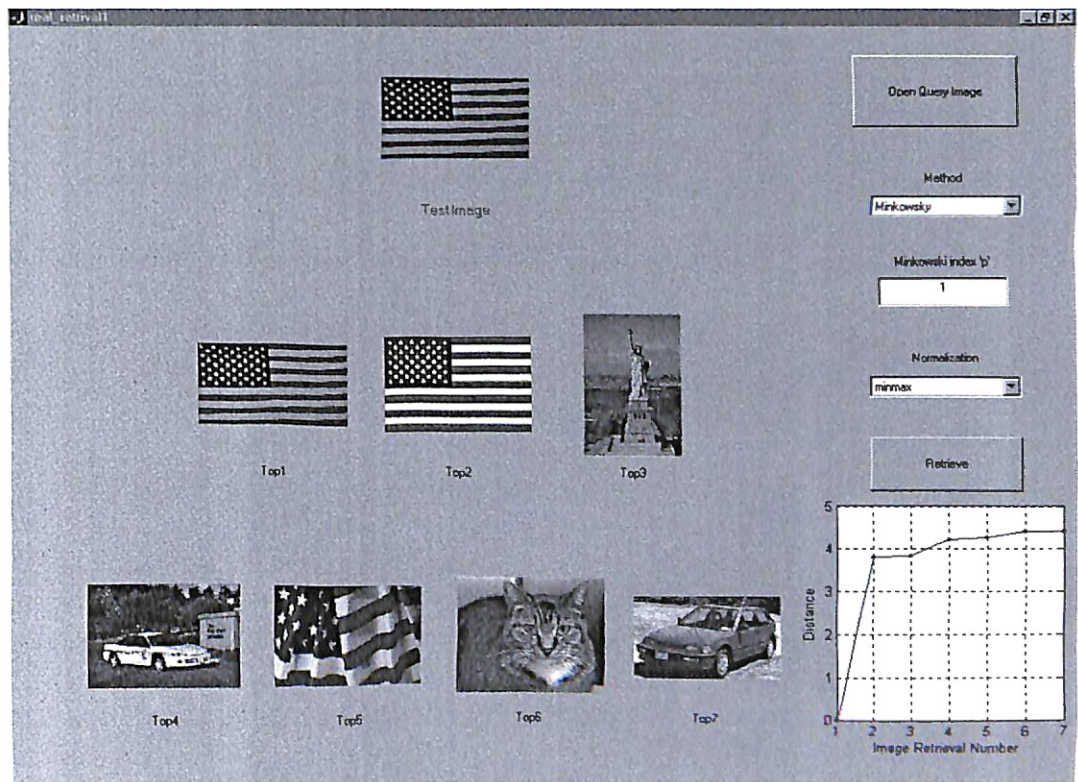


Figure 59. Image retrieval for a real test image of the American flag.

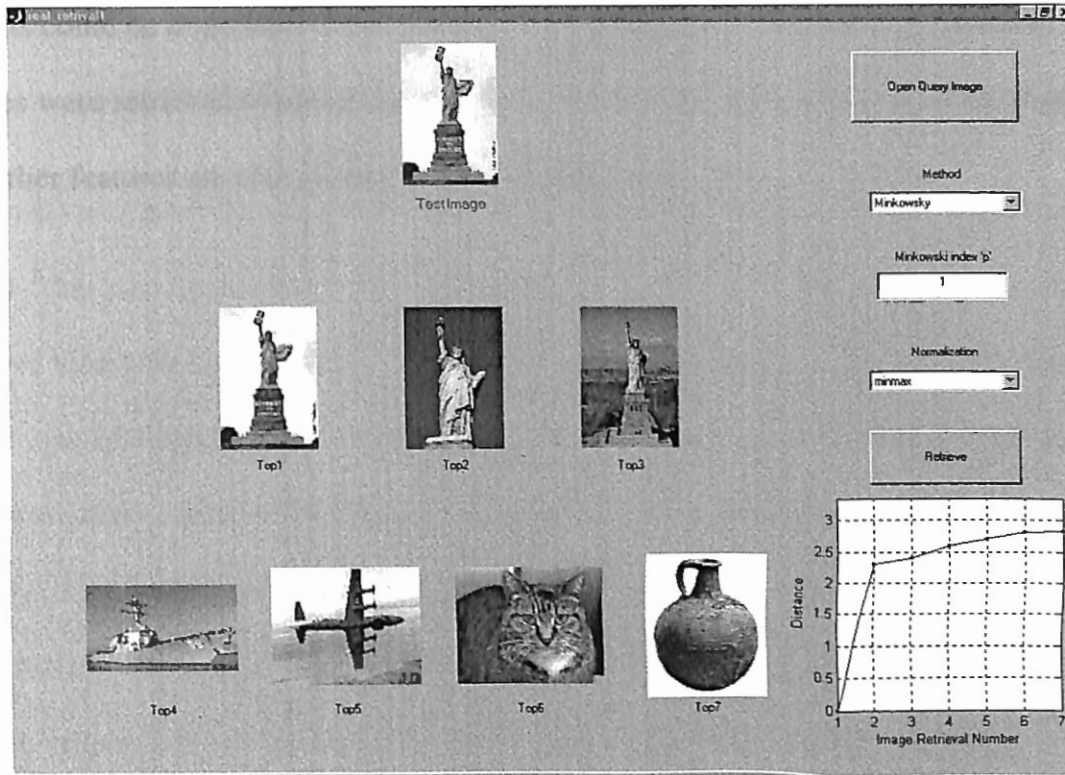


Figure 60. Image retrieval for a real test image of the Statue of Liberty.

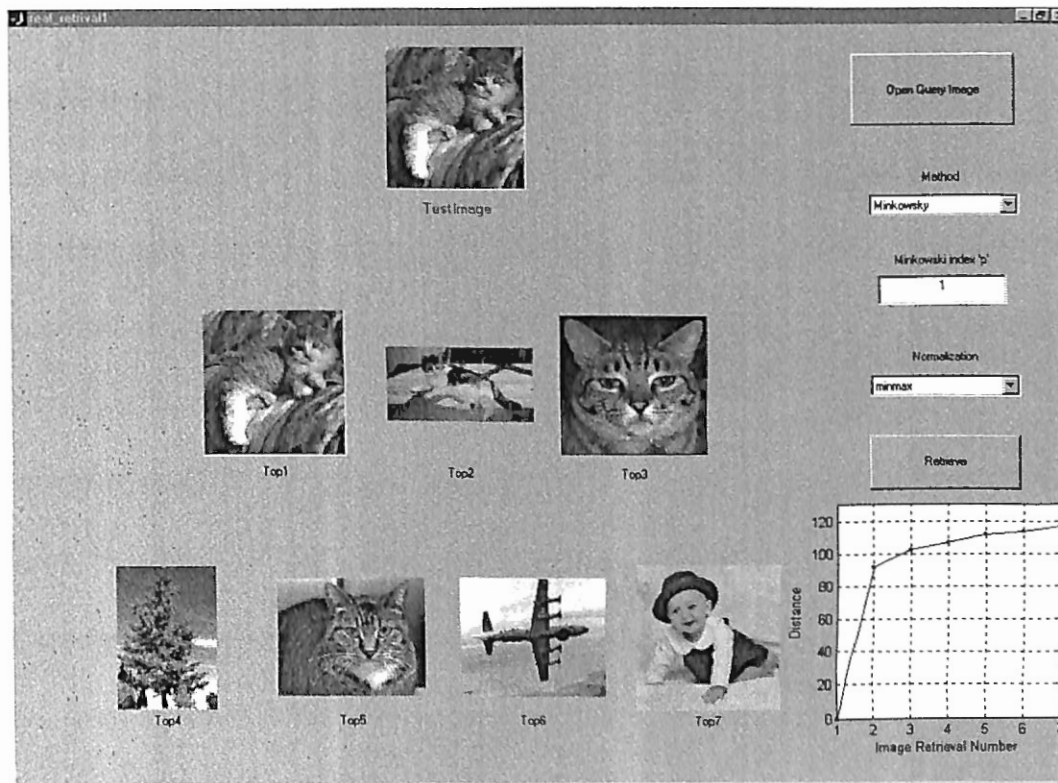


Figure 61. Image retrieval for a real test image of a cat.

Results could be improved, if there had been more images in the training database. Also, images were retrieved based on texture alone. Performances can be improved, if color and other features are also incorporated in addition to texture.

## 5. CONCLUSIONS

Similarity measures for content-based image retrieval using texture were studied. A Gabor filter was implemented to extract textural features from images. These textural features then were used to retrieve images. Minkowsky distance *measure performed* better than Mahalanobis distance measure. Preprocessing by PCA did not work well for this application. Normalization of textural features for Minkowsky distance measure improved the performance. However, all normalizations performed similarly. Therefore, the simplest normalization technique 'minmax,' which linearly scales all textural features to unit range [0, 1] can be used. Minkowsky index (p-value) did not affect the performance of image retrieval considerably. Manhattan distance (p=1) had the least computation time and therefore can be selected. The best similarity measure for image retrieval based on textural content using Gabor features was the Manhattan distance with 'minmax' normalization. Average percent correct retrieval was greater than 90% for up to top 8 retrievals. The image retrieval performance achieved in this study is higher than the image retrieval performances reported in the literature. A graphical user interface was developed to visualize the results.

For retrieving real images, performance can be improved by incorporating other features *like color, in addition* to texture. The 'minmax' normalization is not robust. A more robust normalization must be *identified to improve* the accuracy in retrieving real images. Scaling and rotation of images affect Gabor textural *features. If the training* image database contains zoomed or rotated images of the queried image, similarity

measures based on Gabor textural features will not retrieve those images. Rotational-invariant and scale-invariant textural features must be extracted to improve the robustness of image retrieval.

## 6. REFERENCES

- Aksoy, S. and R. M. Haralick. 2001. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5): 563-582 .
- Brodatz, P. 1966. Textures: A photographic album for artists & designers. Dover Publications, Inc., New York.
- Catalan, J.A., J.S. Jin, and T. Gedeon. 1999. "Reducing the dimensions of texture features for image retrieval using multilayer neural networks", *Pattern Analysis and Applications*, 2 (2):196-203.
- Gonzalez, R.C. and R.E.Woods. 2002. Digital Image Processing. Second Edition. Addison-Wesley Publishing Company, Reading, MA.
- Guo, G., H. Zhang, and S.Z. Li. 2001. Distance-from-boundary as a metric for texture image retrieval. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3: 1629 –1632.
- Huang, Y and R. Chang. 1999. Texture features for DCT-coded image retrieval and classification. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6: 3013-3016.

Hatipoglu, S., S.K. Mitra, and N. Kingsbury. 2000. Image texture description using complex wavelet transform. *International Conference on Image Processing*. 2: 530 -533

Johnson, D.E. 1998. *Applied multivariate methods for data analysts*. Duxbury Press, Pacific Grove, CA.

Li, C. and V. Castelli. 1997. Deriving texture feature set for content-based retrieval of satellite image database. *Proceedings of the International Conference on Image Processing*, 1: 576 –579.

Manjunath, B.S. and W.Y. Ma. 1996. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (Special Issue on Digital Libraries), 18 (8): 837-842.

Misiti, M., Y. Misiti, G. Oppenheim, J. Poggi. 1996. *Wavelet Toolbox for use with Matlab - User's Guide*. Version One. MA.: Natick, The Mathworks, Inc.

Oppenheim, A.V. and R.W.Schafer. 1991. *Digital signal processing*. Prentice Hall.

Papoulis, A. 1991. *Probability, random variables, and stochastic processes*. 3<sup>rd</sup> ed. McGraw-Hill, NY.

Payne, J.S., L. Hepplewhite, and T.J. Stonham, 1999. *Perceptually based metrics for the evaluation of textural image retrieval methods*. IEEE International Conference on Multimedia Computing, 2: 793-797.

Saastamoinen, K., V. Kononen, and P. Luukka. 2002. A classifier based on the fuzzy similarity in the Lukasiewicz structure with different metrics. Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, 1: 363 –367.

Smeulders, A.W.M., M. Worring, S. Santini, A. Gupta, and R. Jain. 2000. Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(12):1349-1380.

Smith, S. W. 1997. The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, San Diego, CA.

Xu, K., B. Georgescu, D. Comaniciu, P. Meer. 2000. Performance analysis in content-based retrieval with textures. 15th International Conference on Pattern Recognition, 4: 275-278.

Zhou, F., J.F. Feng, and Q.Y. Shi. 2001. Texture feature based on local Fourier transform. International Conference on Image Processing, 2: 610-613.



## Appendix A

### Convolution

The output or filtered image can be obtained by convolving the impulse response of the filter with the input image. Impulse responses of filters used in various image processing operations are usually symmetric. The convolution of image  $f$  of size  $M \times N$  with a filter impulse response of  $m \times n$  is given by:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t) \quad (\text{A1})$$

where:  $g(x, y)$  is the value of the filtered image at  $(x, y)$ ,

$$a = (m-1)/2,$$

$$b = (n-1)/2, \text{ and}$$

$s$  and  $t$  are temporary variables.

To generate a complete filtered image, this equation must be applied to  $x=0, 1, 2, \dots, M-1$  and  $y=0, 1, 2, \dots, N-1$ .

This convolution operation can be explained graphically (Figure A1). In this example, the size of the impulse response is  $3 \times 3$  (i.e.,  $m=n=3$ ;  $a=b=1$ ). This impulse response is also commonly called a filter mask or convolution kernel. The convolution process consists of simply moving the mask over the image pixel-by-pixel. At each point  $(x, y)$ , the response  $g(x, y)$  is given by a sum of products of the filter coefficients and the corresponding image pixels in the area covered by the mask (Gonzalez and Woods, 2002). The output is given by:

$$g(x, y) = w(-1, -1)f(x-1, y-1) + w(-1, 0)f(x-1, y) + \dots$$

$$+w(0,0)f(x,y) + \dots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \quad (A2)$$

Note that  $w(0,0)$  coincides with  $f(x,y)$ .

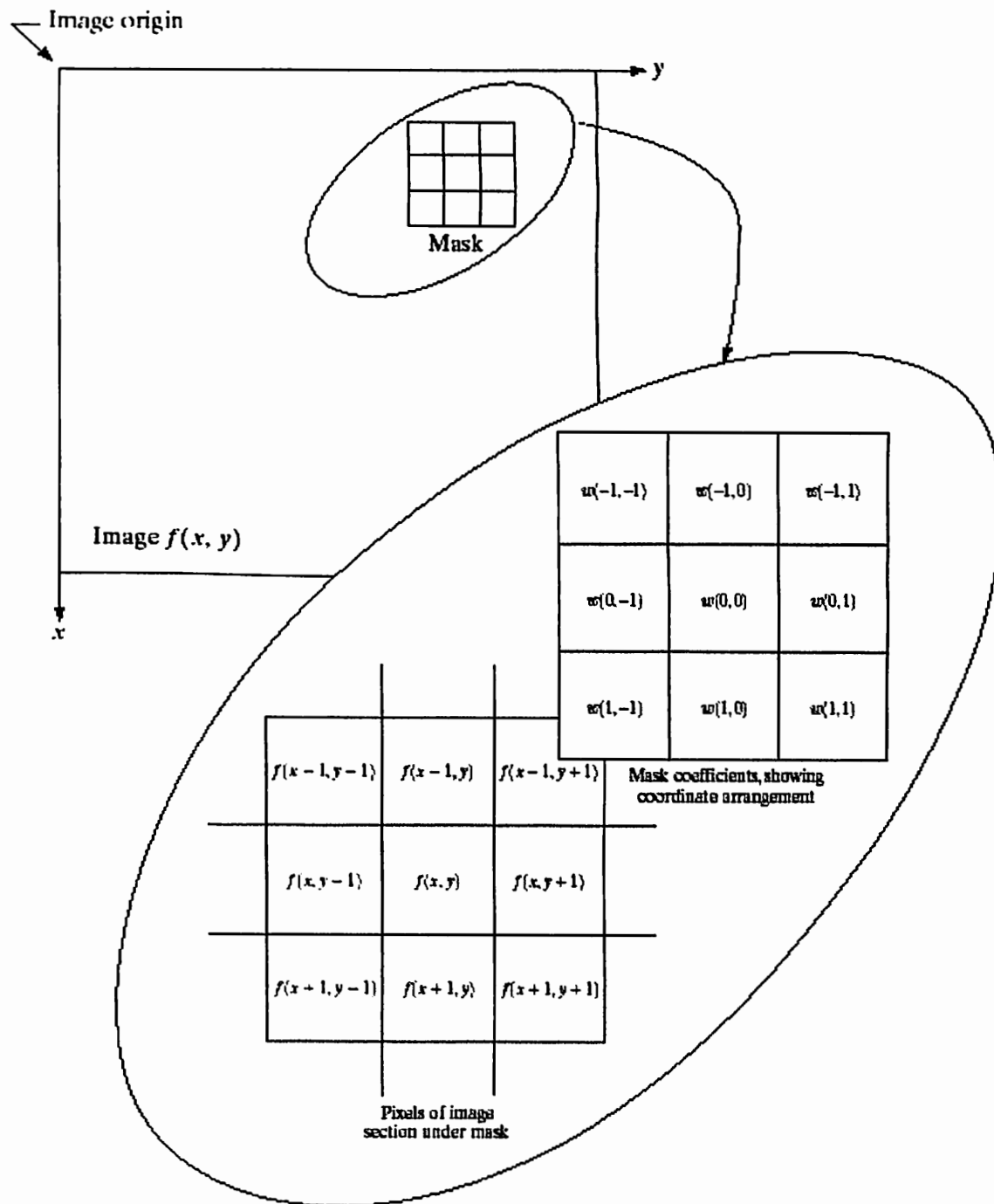
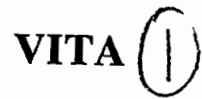


Figure A1. Illustration of convolution (Adapted from Gonzalez and Woods, 2002).



Latha Thiyagarajan

Candidate for the degree of

Master of Science

Thesis:       SIMILARITY MEASURES FOR RETRIEVAL OF  
TEXTURAL IMAGES

Major field: Computer Science

Biographical:

Personal Data: Born in Madurai, India on January 24, 1978, the daughter of Thiyagarajan and Valliammai. Married to Jeyamkondan in India on December 7, 2000. Son, Vishal born in Stillwater Medical Center, Oklahoma on September 29, 2003.

Education: Higher Secondary School Certificate from Sitalakshmi Girls Higher Secondary School, Madurai, India (1995), received Bachelor of Engineering in Computer Science Engineering from Madurai Kamaraj University, India (1999). Completed the requirements for the degree of Master of Science with major in Computer Science at Oklahoma State University in May, 2004.