Texture analysis using the Trace transform

Alireza Talebpour

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

May 2004

ALL MISSING PAGES ARE BLANK

IN

ORIGINAL

Abstract

Texture analysis plays an important role in image analysis and pattern recognition. Also feature extraction is one of the most important tasks for efficient and accurate image retrieval purpose. This work concerns the analysis of textures and feature extraction. It deals with textures which are regularly shaped and sampled, irregularly sampled, and irregularly shaped. The Trace transform is used to extract thousands of features which can be investigated to identify those which are most relevant to the task.
Also texture is widely used in content based image retrieval and there have been a number of studies over the years to establish which features are perceptually significant. However it is still difficult to retrieve reliably images that the human user would agree that are similar. In this work perceptual grouping and finding the most related features to human texture ranking are discussed. The results of using the Trace transform are compared with 10 other methods mainly based on Co-occurrence matrices and the Sum and difference histograms.
Most image processing techniques assume that the image is represented by a rectangular grid of sampling points. This, however, need not be the case. The regularity of sampling is particularly important for texture analysis, where the relative spatial arrangement of the pixels is of paramount importance. In this work we investigate the way of using the Trace transform to recognise textures from irregularly sampled data. The Hough transform is used as an interface that allows us to identify tracing lines in the image and normalise convolution allows us to deal with the irregularly placed samples along the tracing lines in order to compute the trace functionals.
In all cases we investigate and develop the use of the Trace transform and its functionals.

Key words: trace transform, texture analysis, irregular sampling, irregular shapes, normalised convolution,

Email: a.talebpour@ee.surrey.ac.uk
WWW: http://www.eps.surrey.ac.uk/

Acknowledgements

After praying to God and thanks to Him for providing this opportunity for me, I would like to thank my supervisor professor Maria Petrou, who supported me all the times by understanding my situation and by her guidance and providing the ideas and motivation that led to this work.
My wife and children have main role in this work, who were patient while I was studying and I was far from them.
I would like to express my gratitude to all members in the CVSSP, especially Dr. Alex Kadyrov who really encouraged me to move forward and helped me by the discussions we had.
Finally I would like to thank my parents. What I have, I have because of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most of what we get as data from the world comes from our eyes. This data easily are being classified regardless of their sizes, directions and locations and other changes like their irregular shapes. Also, the brain can recognise images, even when only a few pixels of them are shown. There are many works trying to do this kind of operation and recognition automatically [87, 68, 55, 19, 32]. The results show that it is possible to improve this kind of machine recognition. Here we try to find and develop a method by which we can recognise regular and irregular sampling textures as well as irregularly shaped textures [21]. It is interesting because we may find a clue of how the human brain works in finding similarity [78] between textures.

1.1 Motivation

The wealth of objects around us requires a wealth of descriptors. It is very unlikely that a few characteristics measured from the images of these objects will suffice to allow us to discriminate all objects we see. And yet our brain recognises thousands of objects, working mostly at the sub-conscious level. That is the reason knowledge engineering is very difficult: it is very hard to identify the characteristics, which allow us to identify easily so many different faces, objects, materials, textures etc. Restricting ourselves, therefore, in computer vision to features that we can *consciously* identify as characterising our cognition, excludes the vast number of features that our *sub-conscious* uses and

which we cannot usually identify. We may, however, replace the mechanism of our subconscious with a Mathematical tool that allows us to construct thousands of features that do not have physical or other meaning: we may use the Trace transform, which is an alternative image representation and from which we can construct the so called triple features [36]. The Trace transform is a generalisation of the Radon transform [79] that allows one to construct image features that do not necessarily have meaning in terms of human perception, but they measure different image characteristics. The ability of producing thousands of features from an image allows one to be selective as to which are appropriate for a particular task. This task may be any kind of pattern recognition like face recognition [87], finger print recognition [32], character recognition (OCR) [38, 19] etc.

On the other hand, most image processing techniques assume that the image is represented by a rectangular grid of sampling points. This, however, need not be the case. For example, the human eye has its receptors distributed in an irregular pattern. The regularity of sampling is particularly important for texture analysis, where the relative spatial arrangement of the pixels is of paramount importance. For this purpose we use the Trace transform [36]. The Trace transform operates along straight lines criss-crossing the image. There is some physiological evidence to indicate that the human eye performs a similar scanning, known as the "tremor" of the eye [15], being performed with 40-120 cycles per second and amplitude of about 1 degree. Over such short periods of time, the receptors in the human eye scan along straight lines. Although it is debatable, it has been demonstrated that if the tremor is suppressed, the person cannot see [89]. The eye, therefore, looks like a special device that may perform a Trace transform of the viewed scene, in the fovea area.

Finally, it is not possible to give linguistic names to all features we use to identify the different patterns around us. A lot of such features might be computed in the subconscious level without the viewer even being aware of what it is that makes a pattern distinct from all other patterns. We believe, therefore, that if one wishes to develop a system that imitates the human ranking of textures, all one needs to do is to compute thousands of features from the textures and then perform feature selection that allows one to choose those features that produce results that correlate with the human rank-

ing. The Trace transform is a relatively new tool in image processing that allows one to compute thousands of features from an image, which with the appropriate choice of the functionals used, can be made to be rotation, translation and scale invariant [66]. We then shall modify the scheme to select those features, which allow the classifier to imitate the ranking of human subjects and identify the features, which achieve that, without any prejudice on the nature of those features. This is a form of reverse engineering the human vision system in a way that is not restricted by what the subject can consciously identify as a feature he/she uses to classify a texture.
For all of these reasons, we investigate and develop in this thesis, the Trace transform to be used with regularly sampled data, irregularly sampled data and irregularly shaped data and compare the results with the benchmarks of this field, which are the Co-occurrence matrix method [23] and the Sum and difference histograms.

1.2 Approaches used here

For regularly sampled data we are simply trying to construct many features, each of which captures some aspect of the image. We then use a simple form of training that allows us to give relative importance to each feature with respect to the task we have. In the specific example presented, the task in question is texture discrimination.

One of the most well established methods for texture discrimination is based on the use of Co-occurrence matrices, and in particular on the use of features extracted from them [26]. The Co-occurrence matrix captures the second order statistics of a stationary texture and computes some quantities from them that have perceptual meaning: contrast, homogeneity, etc. As Co-occurrence matrices are considered by many a benchmark for texture analysis [2, 14, 53], we are going to use them here to discriminate textures from the Brodatz album [29] and compare their performance with the results produced by the Trace transform method. Like [56, 46, 7], we used all textures in the Brodatz album.
After that, we present a texture classification scheme based on features constructed from the Trace transform so that it ranks the textures according to human visual perception. The system is trained using perceptual classes created by 11 subjects. The

selected features are then used to classify a totally different set of textures.
Then for irregularly sampled data, to be able to apply the Trace transform, we need an intermediate step, which will lead from the irregularly distributed sample points to straight lines. Such a mechanism is provided by the Hough transform [74]. To demonstrate our ideas, we first create an irregular pattern of sampling points using three methods of distribution for random points:

- Gaussian distribution [54]
- Log-polar model [73]
- Retinal distribution [15]

This way we can represent our images by a collection of sample points, which are densely distributed towards the centre, and sparsely distributed in the periphery. Then we use the Hough transform to identify sets of pixels that constitute straight lines. Each identified line can be used as a tracing line in the Trace transform. So we can classify these sets of data after extracting features by the Trace transform. Finally we use irregularly shaped masks, to make irregularly shaped textured regions from the Brodatz album [5], to be analysed by the Trace transform. In this case we again adapt the method to be suitable for this purpose.

1.3 Structure of the thesis

In chapter 2 we describe some methods used for texture recognition and perceptual grouping. Chapter 3 explains the Trace transform in detail and shows how this method can be used to produce features invariant to scaling, shifting and rotation. In chapter 4, first we use the Brodatz album to investigate which of the 12 methods we implemented, mainly based on the Co-occurrence matrix, can classify the textures better. Then we try to use these methods for perceptual grouping of textures. Chapter 5 is concerned with the recognition of irregularly sampled data. First we generate sampling masks by using three kinds of distributions: the Gaussian distribution, Log-polar and Retinal model. After that we use the Hough transform as an interface that allows us to identify tracing

lines in the image. Normalised convolution [41] allows us to deal with the irregularly placed samples along the tracing lines in order to compute the trace functionals. Finally we obtain the results and compare them with the results of using the Trace transform in conjunction with regularly sampled data. Irregularly shaped textured regions are created in chapter 6 and classified by the Trace transform after adapting it to remove its shape recognition capability. Finally in chapter 7 we present our conclusions and future work.

1.4 Thesis achievements

- We demonstrated that features could be selected from among those produced by the Trace transform appropriate for ranking textures like human do, much better than by the other methods discussed here.
- We demonstrated that it is possible to recognise textures from irregularly sampled data.
- We demonstrated that textures can be recognised from irregularly shaped samples.
- We demonstrated the usefulness of the Trace transform in discriminating textures by extracting thousands of features.

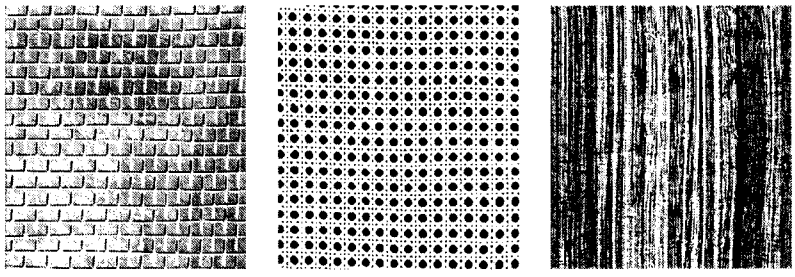
Chapter 2

Literature survey on texture classification

2.1 Introduction

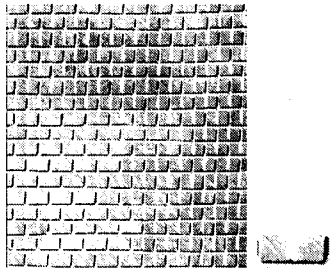
Texture means the repetition of regular patterns of pixels or as in [76] is something consisting of mutually related elements. Some example textures are shown in Figure 2.1. People usually describe texture as fine, coarse, grained, smooth, etc., i.e. they use qualitative terminology. This implies that some more precise features must be defined to make machine recognition possible. Such features can be found in the tone and structure of a texture [25]. Tone is based mostly on pixel intensity properties in the primitive, while structure is the spatial relationship between primitives. Each pixel can be characterised by its location and tonal properties. A texture primitive is a set of pixels with some tonal and/or regional property, and can be described by its average intensity, maximum intensity, size, shape, etc. The spatial relationship of primitives can be random, or they may be pairwise dependent, or some number of primitives can be mutually dependent. Image texture is then described by the number and types of primitives and by their relationship. Figure 2.2 shows a texture and its primitive pattern. Leu [47] suggested a method to calculate the periodicity of a texture which can be useful for finding the primitive pattern.

Automatic image analysis results in two fundamentally different approaches to texture



(a) (b) (c)

Figure 2.1: Three samples of different textures. It is clear in some of them that some primitive pattern is repeated regularly.



(a) (b)

Figure 2.2: This figure shows a texture and its primitive pattern (Texel) in magnification.

analysis: Generating parameters to characterise the spatial distribution of grey levels in a texture, the so called statistical approach. Using Co-occurrence matrix is one solution of this approach, which will be discussed more in section 2.2.1.2. The second approach is the structural approach. It analyses a visual scene in terms of the structure of the texture [90]. In this approach we have two concepts: some primitive pattern of one or more pixels exists, and this primitive pattern is repeated at partially predictable intervals [51].

There are the following main issues in texture analysis:

1- Having a textured region, find its class.

2- Having a textured region, find a model for it.

3- Having an image which contains several textured areas, find the different regions.

Here we work on issue one, and in two ways: first, we review methods for texture discrimination, and then we review the efforts that have been done for perceptual grouping. Our method, which is using the Trace transform to classify textures, is neither a statistical nor a structural approach. It is a new method, which generates features which characterise a particular texture.

2.2 Review on texture discrimination

As we said there are two fundamentally different approaches to texture analysis: Statistical and syntactic. Next we review the most important methods in each category.

2.2.1 Statistical Approaches to Texture Analysis

In statistical texture description, each texture is described by a feature vector of properties which is a point in a feature space. The aim is to find a decision rule to assign a texture to some specific class.

2.2.1.1 Spatial Frequencies Method

A large group of texture recognition methods is based on measuring spatial frequencies. One of them is the autocorrelation function of a texture. Consider two copies of an image printed on transparencies. If we put one of them on the top of the other and slowly shift it and measure the average light transmitted through the overlapping region, a graph of these measurements as a function of the shifting vector (x, y) and normalised with respect to the $(0, 0)$ translation, depicts the two-dimensional autocorrelation function of the image.

In this method each pixel is a primitive texture element and the primitive property is its grey-level. Texture spatial organisation is described by the correlation coefficient that evaluates linear spatial relationships between primitives. Small primitives act exactly in

the opposite way of large primitives which make the autocorrelation function decrease slowly with increasing distance. Equation 2.1 shows how the autocorrelation coefficient can be calculated.

$$C_{ff}(p,q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p}\sum_{j=1}^{N-q} f(i,j)f(i+p,j+q)}{\sum_{i=1}^{M}\sum_{j=1}^{N} f^2(i,j)} \tag{2.1}$$

where $f(i,j)$ is the image grey value, (p,q) is the relative shift between two pixels, the first of which is at position (i,j) and M,N are the image dimensions.

2.2.1.2 Co-occurrence Matrix

Given an image $I(x,y)$ we define the Co-occurrence matrix $S_{d,\theta}(i,j)$, each element of which is the number of times grey level i and grey level j are found in relative distance d and relative orientation θ. So, a pair of pixels (x_1,y_1) and (x_2,y_2) contribute to element $S_{d,\theta}(i,j)$ if:

$$f(x_1,y_1) = i, \quad f(x_2,y_2) = j \tag{2.2}$$

and

$$x_2 = x_1 + d\cos\theta, \quad y_2 = y_1 + d\sin\theta \tag{2.3}$$

where $f(x,y)$ is the grey level at position (x,y) of image f, d is the distance between two pixels of the image and θ is the angle between the line joining the two pixels with the horizontal line [51]. It is possible that we may not wish to characterise a texture using directional information. In that case we construct the rotationally invariant co-occurrence matrix by considering pairs of points at a particular distance from each other, irrespective of orientation θ. Let us explain this, with an example. Table 2.1 shows a grey level image and tables 2.2 and 2.3 flag the pixels that are at distance $d = 2$ and $d = 3$ from the pixel marked with X. When we compute the rotationally invariant co-occurrence matrix, we scan the image from left to right and from top to bottom, to identify all pairs of points that are at distance d from each other. However, we must count each pair only once. So, when we scan the image we use the masks of

1	1	5	5	7	4
1	2	5	5	7	2
2	1	3	4	7	1
1	1	5	5	7	1
5	3	5	5	3	1

Table 2.1: This is an example grey level image.

0	1	1	1	0
1	0	0	0	**1**
1	0	X	0	**1**
1	0	0	0	**1**
0	**1**	**1**	**1**	0

Table 2.2: The pixels marked with 1 are at distance 2 from the pixel marked with X. Only elements in bold are included in the mask, for avoiding repetition.

tables 2.2 and 2.3, but pair the central pixel only with the pixels marked with bold 1. Now we describe the algorithm of building the co-occurrence matrix.

First we create a square matrix of size G by G where G is the number of distinct grey levels in the image, with zero elements. For this example $G = 8$. Then we begin from the first pixel of image located at $(1, 1)$ and use the corresponding mask to see which pixels must be paired with it. Table 2.4 marks the pixels which will be paired with pixel $(1, 1)$ for the mask with $d = 3$.

0	0	1	1	1	0	0
0	1	0	0	0	1	0
1	0	0	0	0	0	1
1	0	0	X	0	0	**1**
1	0	0	0	0	0	**1**
0	**1**	0	0	0	**1**	0
0	0	**1**	**1**	**1**	0	0

Table 2.3: The pixels marked with 1 are at distance 3 from the pixel marked with X.

1	0	0	5	0	0
0	0	0	5	0	0
0	0	3	0	0	0
1	1	0	0	0	0
0	0	0	0	0	0

Table 2.4: Using the distance mask with $d = 3$ on our image for the first pixel

Grey level	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	4	0	3	2	12	0	5
2	0	1	0	3	1	6	0	2
3	0	2	0	2	0	1	0	0
4	0	3	0	1	1	1	0	1
5	0	11	4	2	1	10	0	3
6	0	0	0	0	0	0	0	0
7	0	4	1	3	0	4	0	1

Table 2.5: The co-occurrence matrix of our image (Table 2.1) for $d = 3$.

Then we increase the value of the co-occurrence matrix at the corresponding position indicated by the pair of the grey levels. For example the first pair is $(1, 5)$, so we add 1 to the value of element $(1, 5)$ of the co-occurrence matrix, and so on for element $(1, 5)$, $(1, 3)$, $(1, 1)$ and again $(1, 1)$.

Then we repeat all steps again for the next pixel and continue this process until all pixels of the image are processed.

Finally we have the square matrix shown in table 2.5.

From each co-occurrence matrix we compute the following 5 features:

- Energy:

$$\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j)^2 \tag{2.4}$$

- Entropy:

$$-\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} P(i,j)\log P(i,j) \tag{2.5}$$

- Contrast:

$$\sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-j)^2 P(i,j) \tag{2.6}$$

- Correlation:

$$\frac{\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} ijP(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y} \tag{2.7}$$

where

$$\mu_x = \sum_{i=0}^{G-1} i \sum_{j=0}^{G-1} P(i,j) \qquad \mu_y = \sum_{j=0}^{G-1} j \sum_{i=0}^{G-1} P(i,j) \tag{2.8}$$

$$\sigma_x = \sum_{i=0}^{G-1}(i-\mu_x)^2 \sum_{j=0}^{G-1} P(i,j) \qquad \sigma_y = \sum_{j=0}^{G-1}(j-\mu_y)^2 \sum_{i=0}^{G-1} P(i,j) \tag{2.9}$$

- Homogeneity:

$$\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} \frac{P(i,j)}{1+|i-j|} . \tag{2.10}$$

where $P(i,j)$ is the fraction of pairs of pixels that are at the particular distance d from each other and one of them has grey value i while the other has grey value j.

These features can be used to characterise the texture. Co-occurrence matrices give very good results in discrimination between textures. But, the method is computationally expensive. A fast algorithm for it is given in [4, 10, 11].

2.2.1.3 The Sum and Difference Histograms Method

The sum and difference of two random variables with the same variances can be the principal axes of their associated joint probability density function. So, the Sum and Difference histogram can be a good alternative of the co-occurrence matrix method

for texture recognition. It runs faster and needs less memory than the co-occurrence method. In [84] it is shown that the sum and difference define the principal axes of the second-order PF (probability function) of a stationary process and can be estimated directly from the image, so it is suggested to use them instead of the co-occurrence matrix method.

Here we find the number of pairs with the sum of their grey levels equal to i and the difference of their grey levels equal to j, exactly as we have done in the calculation of the co-occurrence matrix where we had to find the number of pairs with grey level i and j.

The normalised sum and differences histograms are:

$$\hat{P}_s(i) = \frac{h_s(i)}{N}; \quad (i = 2, ..., 2G - 2) \tag{2.11}$$

$$\hat{P}_d(j) = \frac{h_d(j)}{N}; \quad (j = -G + 1, ..., G - 1) \tag{2.12}$$

where $h_s(i)$ is the sum histogram, $h_d(j)$ is the difference histogram, $P_s(i)$ is the normalised sum histogram, $P_d(j)$ is the normalised difference histogram of our image and $N = \sum_i h_s(i) = \sum_d h_d(j)$.

The normalised histograms play the role of probability density functions. So, if we assume uncorrelatedness of the sum and difference probability density functions, we can estimate the second order probability density function, (i.e. normalised entries of the co-occurrence matrix) by the multiplication of:

$$\hat{P}_s(i)\hat{P}_d(j) = P\left(\frac{i+j}{2}, \frac{i-j}{2}\right) \tag{2.13}$$

where $P(\frac{i+j}{2}, \frac{i-j}{2})$ corresponds to $P(i, j)$ in formulae 2.4 to 2.10. This simplification makes the calculation faster and easier as instead of working with matrices we work with 1D histograms. In [6, 21, 30] are examples of using sum and difference histograms in comparison with other methods.

2.2.1.4 Edge Frequency

The comparison of edge frequencies in texture can be used as a method for texture recognition. The distance-dependent texture description function $g(d)$ can be computed

for any sub-image f defined in a neighbourhood for variable distance d:

$$g(d) = |f(i,j) - f(i+d,j)| + |f(i,j) - f(i-d,j)| \\ +|f(i,j) - f(i,j+d)| + |f(i,j) - f(i,j-d)| \quad (2.14)$$

So we must compute a gradient $g(d)$ for all pixels of the texture and then evaluate texture features as average values of this gradient. Several texture properties can be derived from first-order and second-order statistics of edge distributions like:
Coarseness: The finer the texture, the higher is the number of edges present in the texture edge image.
Contrast: Large edge magnitude is a sign for high contrast textures.
Randomness may be measured as entropy of the edge magnitude histogram, and so on. Several other properties may be derived from first-order and second-order statistics of edge distributions [83].

2.2.1.5 Wavelet based Methods

In the wavelet transform the time information of the signal is not completely lost and the signal is represented in a two-dimensional time-frequency space. This transform is a relatively new technique for studying signals, and has an edge over the conventional transform viz. Fourier transforms etc. The bases for this transform are finite length waves called wavelets, hence the name. The following is the general form of the basic two-dimensional wavelet [75]:

$$\psi_{a,b_x,b_y}(x,y) = \frac{1}{|a|}\psi\left(\frac{x-b_x}{a}, \frac{y-b_y}{a}\right) \quad (2.15)$$

where $\psi(x,y)$ is a suitable function called the mother wavelet. We first select a suitable mother wavelet, usually by experimentation, and then the input signal is finitely windowed and cross-correlated with stretched and scaled versions of this mother wavelet. Wisely selecting the wavelet shape so as to suit the signal being studied is very important. By decomposition of the signals in terms of wavelets, we can extract features that

can be used to discriminate among different textures. For example in [75] the wavelet decomposition is used to obtain 10 different sets of coefficients i.e. three (vertical, horizontal and diagonal) coefficients at each of the three levels of wavelet decomposition and the low pass coefficient at the third level of decomposition. Then they used mean and variance of each of these vectors as features to classify the textures with some classifier. Some other examples of using wavelets in texture classification can be found in [43, 70, 69].

Some other methods are: Law's texture features [45], fractal based features [57, 22], primitive length texture features [18], Markov Random field parameters [39, 9], multi channel filtering features [71], support vector machines [49], boolean model [20], local Fourier transform [92] and adaptive Gabor filtering [8, 52, 80].

2.2.2 Syntactic approaches to texture Analysis

The idea that textures consist of primitive patterns located in almost regular relationships is the basis of syntactic texture description models. Primitive description and rules of primitive placement must be determined to describe a texture. One of the most efficient ways to describe the structure of primitive relationships is using a grammar which shows how to build a texture from primitives by applying transformation rules to a limited set of symbols, which represent the texture primitives. In the real world it is difficult to find strict rules and we must use variable rules incorporated into the description grammars. We discuss here as an example a chain grammar. Other grammars suitable for texture description like tree, matrix and graph grammars can be found in [85].

2.2.2.1 Shape Chain Grammar

This grammar is the simplest one that can be used for texture description. First we begin with a starting symbol, then the application of transform rules and generating processes continues until no further transform rule can be applied.

Figure 6.1 shows how by using such a grammar we can recognise a texture (Figure 2.4). V_n is a set of non-terminal symbols, and V_t is a set of terminal symbols, R is a set of

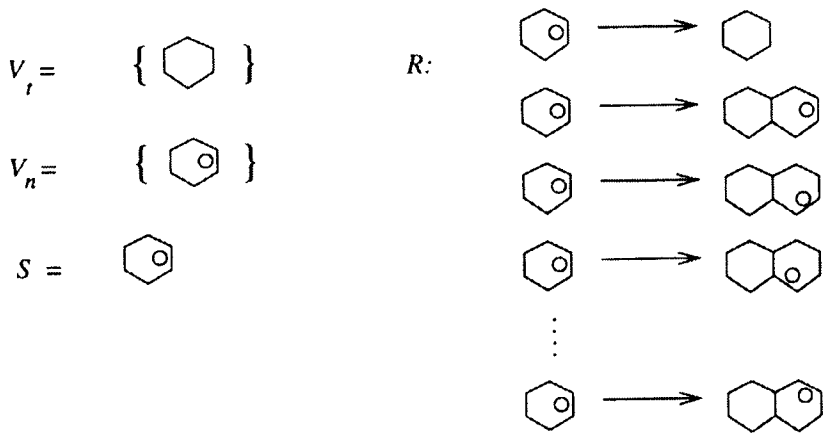


Figure 2.3: Grammar generating hexagonal textures. Picture taken from [76].

rules and S is the start symbol. The rules of the grammar are shown in figure 6.1. As we can see according to this grammar, the pattern shown in figure 2.4a is acceptable and the pattern shown in figure 2.4b is not, because by the set of rules R in figure 6.1 we can build pattern 2.4(a) but we can not build pattern 2.4(b).

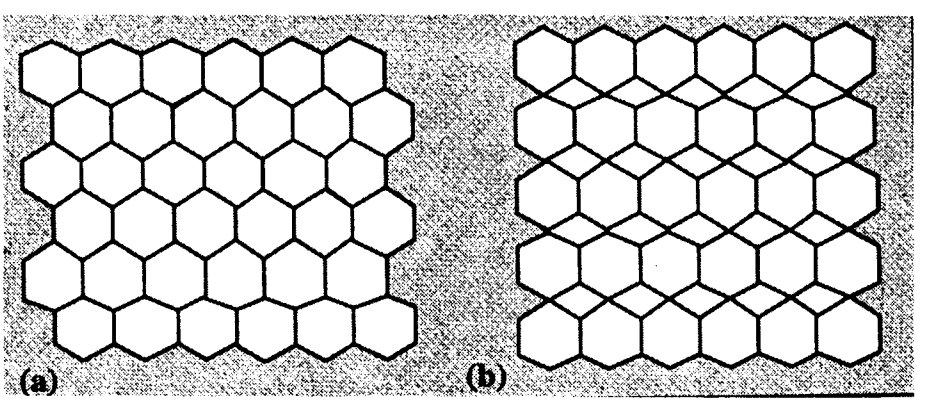


Figure 2.4: Hexagonal textures: (a) accepted (b) rejected, because by the set of rules R in figure 6.1 we can build pattern (a) but we can not build pattern (b). Picture taken from [76].

2.3 Perceptual grouping

Texture is an important visual feature for computer vision tasks. Texture similarity in many applications such as image retrieval and computer image understanding, should be measured in a manner that is invariant to texture scale and orientation, as well as consistent with human perception. Although there exist scale and orientation invariant texture features and similarity measures [12, 24, 81], they are not necessarily perceptually consistent[61]. They are statistical methods (e.g., convolution filters that measure variance, inertia, entropy and energy) and perceptual techniques (e.g., identifying an underlying direction, orientation and regularity[72].

Different researchers used different perceptual features as well as different methods to implement them. Abbadeni et al [1] presented a new method to estimate coarseness, contrast and direction, by using the auto covariance function. Healy and Enns [27], used height, density and regularity based on the results from computer graphics, computer vision and cognitive psychology which have identified these characteristics as playing important role in the formation of perceptual texture patterns. They showed the effectiveness of their technique by applying it to a real-word visualisation environment: tracking typhoon activities in southeast Asia, and analysing ocean conditions in the northern Pacific. Iqbal and Aggarwal [31] used perceptual grouping for image structure extraction for both retrieval and classification. They extracted the following features hierarchically in an unconstrained environment, i.e., with no constraints on the viewing angle and depth, using the approach detailed in [31]: line segments, larger linear lines, co-terminations, "L" junctions, "U" junctions, parallel lines, parallel groups and polygons. They used perceptual grouping rules of similarity, continuity, parallelism and closure to extract these features. Long and Leow [50] presented a hybrid method, using a convolutional neural network and an SVM to perform the invariant and perceptual mapping. Hoogs et al [28] used a method of analysing macro textures by using perceptual observables. The typical geometric Gestalt [59] grouping criteria such as proximity and parallelism are extended with descriptive measures of topology and photometry enabled by region neighbourhood analysis. Orrite et al [58] proposed a novel approach for perceptual grouping that takes into account the sequential order

in the application of the different mechanisms for the grouping of primitives, so that, it changes in accordance to the relative importance of the perceptual mechanisms in each particular case. The use of perceptual organisation allowed Yow and Cipolla [91] to propose a face detection framework that groups image features into meaningful entities. Johansson et al [33] presented a new method to detect rotational symmetries, which describes complex curvature such as corners, circles, star and spiral patterns. Picard and Kabir [67] tried to find similar patterns in large image database and Payne and Stonham [63], and [62] showed that in general image content-based retrieval methods do not match human perception well.

Like as some others, we believe that it is not possible to give linguistic names to all features we use to identify the different patterns around us. A lot of such features are computed in the subconscious level without the viewer even being aware of what it is that makes a pattern distinct from all other patterns. We believe, therefore, that if one wishes to develop a system that imitates the human ranking of textures, all one needs to do is to compute thousands of features from the textures and then perform feature selection that allows one to chose those features that produce results that correlate with the human ranking. The Trace transform is a relatively new tool in image precessing that allows one to compute thousands of features from an image, which with the appropriate choice of the functionals used, can be made to be rotation, translation and scale invariant [66]. On the other hand, Co-occurrence matrices also allow the calculation of thousands of features, if one uses as features the elements of the matrix themselves [44]. In the sections that follow we shall use both perceptual-type features extracted from the co-occurrence matrix and features constructed from the Trace transform and the elements of the co-occurrence matrix to perform texture classification. We then shall modify the scheme to select those features which allow the classifier to imitate the ranking of human subjects and identify the features which achieve that, without any prejudice on the nature of those features. This is a form of reverse engineering the human vision system in a way that is not restricted by what the subject can consciously identify as a feature he/she uses to classify a texture.

Chapter 3

Trace Transform

3.1 Introduction

Searching for an image between a limited number of images is possible for one person, but when the number of images increases, the task gradually become impossible. If we use a computer to assist us, the task is very different from that of searching a word in a huge number of pages of data. The latter can be done easily by computer. For the former task however, the computer has difficulties; it can not find two copies of the same image with small differences, unless we can tell it somehow to ignore that difference and look for some similarities. Figure 3.1 shows two images with a small difference. Our brain can put both of them in the same class, but how can we make the computer to do the same? There are several ways to extract some features from images which are similar because they belong to the same class. We shall explain one of them based on the so called Trace transform, and we shall show that it can be used so that rotation, translation and also scaling of an image does not affect the features which are extracted. Logically, if we consider one image and all its rotations and extract a feature from them, there is no reason to expect different results for differently rotated versions of the image. Figure 3.2 shows one image and its rotated version by 20 degrees. If we define a process on these two images which considers all rotations, it is clear that the result should be the same for both images. Figure A1 can fit figure A2 after 20 degrees of rotation and figure A2 can be fitted on A1 after 340 degrees of rotation. This means

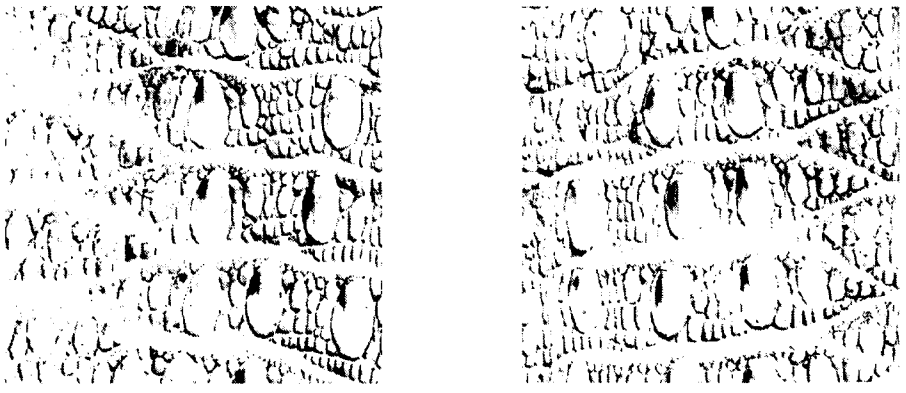


Figure 3.1: Our brain can easily realize that these two textures belong to the same class, but how can a computer do it?

that the features computed on the two images must be the same. We shall see that when computing the Trace transform, we consider all rotations of an image, that is why by using a special functional applied to this transformation, we can remove the variation due to rotation and identify the original image and its rotated version as being the same image. On the other hand, by normalising the images, or by normalising the features and finding the ratio of two different features of the same image, we can remove the effect of scale. Finally by using functionals invariant to shifting, we can make the features be invariant to translation. The mathematical proof of this statement will be shown in a subsequent section.

3.2 Overview of the production of features from the Trace transform

3.2.1 Trace Matrix

Consider an image like A1 of figure 3.2. It has 232 rows and 330 columns, so the center of this image is at O(116,165). If we use this point as our polar coordinate centre (figure 3.3), we can plot 360 lines crossing the center with one degree rotation from each other. On the other hand, we can shift each line from its position (distance to

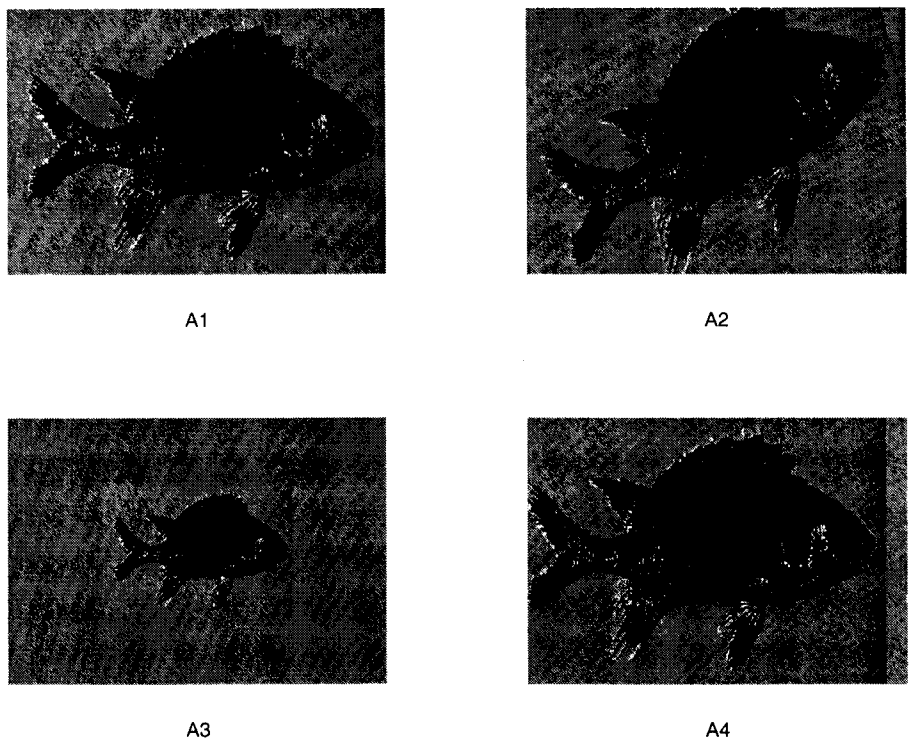


Figure 3.2: By using the Trace transform with special functionals to compute features that describe these images, we can remove dependence on rotation (A2), scaling (A3), translation (A4) and we can easily put all of these four images in the same class.

center= 0) to a maximum position $\sqrt{116^2 + 165^2} = 202$. We can define a functional T computed from the image along each line and write the result in a new matrix at position (P =distance to center,ϕ =its angle with respect to the horizontal line). This matrix is named Trace matrix, and as we mentioned, it has two dimensions, P and ϕ. Figure 3.4(b) shows the trace matrix of image 3.4(a) after using the integral function along each line. It is clear that P is maximum for $\phi = 45, 135, 225, 315^o$, because the original image is square. Also P is minimum when $\phi = 0, 90, 180, 270^o$.

Generally we have maximum value of P when:

$\phi = k\pi \pm \arctan \frac{b}{a}$, where:

a=length of the image

b=width of the image and

$k = 1, 2$

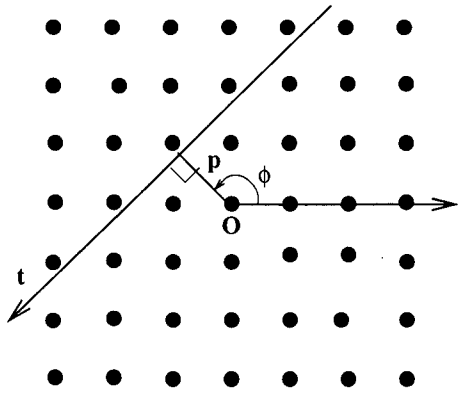


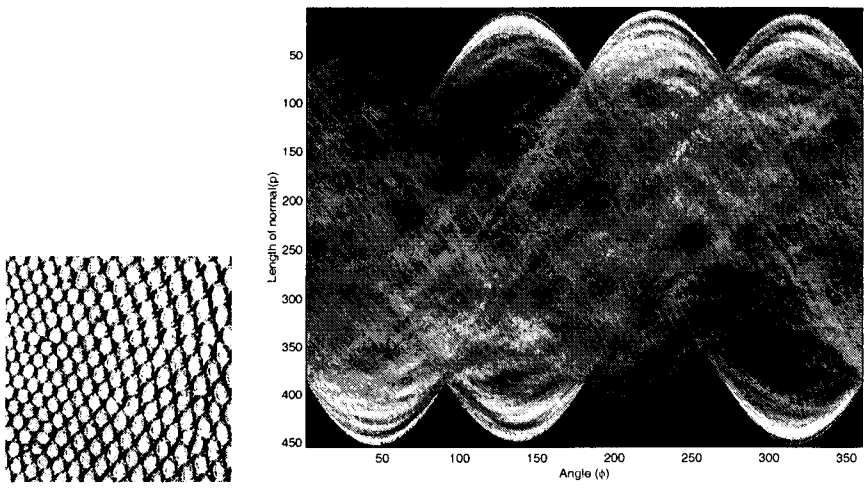
Figure 3.3: Definition of the parameters of an image tracing line.

and the minimum value of P is for:

$\phi = k\frac{\pi}{2}$, where:

$k = 0, 1, 2, 3$

At this step we obtain the trace matrix, the size of which depends on the number of



(a) (b)

Figure 3.4: (a) An original texture from the Brodatz album [29] and (b) its trace transform computed with the first functional of table 1.

points we use to sample the angle of rotation and the distance between the tracing lines used. For example, if the step in ϕ is ϕ_1, we have $\frac{360}{\phi_1}$ columns in our trace matrix, and if the distance of two successive lines is d_1, the number of rows of the trace matrix is:

$\frac{P_{max}}{d_1}$. In the next step we convert this matrix to a row vector by using the so called diametrical functional.

3.2.2 Diameter Vector

Column n of the trace matrix is obtained by tracing the image with parallel lines at an angle ϕ_n (Figure 3.5). So to ensure invariance to rotation, we should not treat the columns of the Trace transform together, but we must first process each column, separately. We process each column by using another functional, called diametric func-

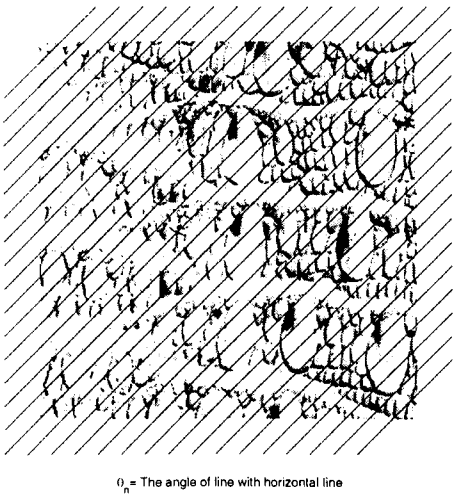


Figure 3.5: Column n of the trace matrix is obtained by looking the image at an angle ϕ_n.

tional. The size of the vector created this way is equal to the number of columns of the Trace transform, which is the number of angles we used to create the Trace transform.

3.2.3 Construction of a Triple Feature

This is the last step. Here we find a number, the so called triple feature, which is the result of using the so called circus functional on the vector produced after the application of the diametric functional. Figure 3.6 shows one such vector. We plot it in order to see how these numbers differ along the different directions of the image. As

we can see this vector is periodic with period π. Because after rotating a line by π, we will have the same line again. It is very important that the functional we apply in this step is invariant to shift. Otherwise we will not have features invariant to rotation. This is an image and its rotated version just differ by a shift of this vector.

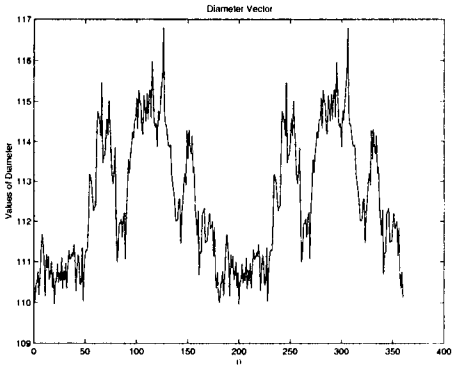


Figure 3.6: This figure shows the vector produced from the Trace transform after the application of the diametric functional. As we can see this vector is periodic with period π, because after rotating a line by π we get the same line again.

3.2.4 Normalisation

In most cases, we need more than one feature to feed them to a classifier in order to classify the objects correctly. So we define several functionals to be used as the Trace, Diametrical and Circus functionals. For example Table 3.1, lists 31 functionals which may be used for step one, Table 3.3 lists 10 functionals which may be used for step two and table 3.4 shows 18 functionals which may be used for step three. Each combination of these functionals can be used to produce a feature. The maximum number of features which can be obtained from these functionals is: $31 \times 10 \times 18 = 5580$ features. When we want to combine these features together, features which take large values will dominate. That is why we must normalise them and ensure that each feature takes values in a certain range, for example [0 1]:

$feature_{i_{norm}} = \frac{feature_i - Min_i}{Max_i - Min_i}$

Where:

Min_i is the minimum value of the i^{th} feature,

Max_i is the maximum value of the i^{th} feature,

$feature_i$ is the i^{th} feature,

and $feature_{i_{norm}}$ is the normalised version of the i^{th} feature,

Rotation does not affect the result, since we consider lines along many orientations but change in scale does. How can we remove the dependence on scale? We can do that by normalising the features produced. We discuss this in the next section.

3.3 The Theory of Triple Feature Extraction

In this section we follow Kadyrov and Petrou [36, 34]. We suppose that our image is subject to linear distortions: rotation, scaling and translation. We can consider that the image remains the same but we change our coordinate system. Let us name the original coordinate system C_1 and the distored one C_2. θ is the angle of rotation, ν^{-1} is the value of scaling and the translation vector is: $[s_0 \cos\psi \ s_0 \sin\psi]$. So if we have image $F_1(x, y)$ in coordinate system C_1, $F_2(\hat{x}, \hat{y})$ is constructed from $F_1(x, y)$ by rotation by θ, scaling by ν^{-1} and translation by $[s_0 \cos\psi \ s_0 \sin\psi]$. Because we assume linear transformations, we can show [36] that a straight line with parameters (p_{old}, ϕ_{old}) will be straight line in the new sytem, with parameters (p, ϕ), such that:

$$\phi_{old} = \phi - \theta \tag{3.1}$$

$$p_{old} = \nu[p - s_0 \cos(\psi_0 - \phi)] \tag{3.2}$$

$$t_{old} = \nu[t - s_0 \sin(\psi_0 - \phi)] \tag{3.3}$$

We collectively refer to all scanning lines in all directions by Λ. We know that in the first step the Trace transform is a function g on Λ with trace functional T. It can be written as:

$g(F; C_1; \phi, p) = T(F(C_1 : \phi, p, t))$,

where $F(C_1; \phi, p, t)$ means the value of the trace functional along the line (ϕ, p) in C_1. After using this function, we eliminate t, and the new image representation on Λ with two coordinates ϕ and p is obtained.

Number	Functional
1	$\sum_{i=0}^{N} x_i$
2	$\sum_{i=0}^{N} i x_i$
3	$\sqrt{\frac{\sum_{i=0}^{N}(x_i-m)^2}{N}}$
4	$\sqrt{\sum_{i=0}^{N} x_i^2}$
5	$Max_{i=0}^{N} x_i$
6	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$
7	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert^2$
8	$\sum_{i=3}^{N-3} \lvert x_{i-3} + x_{i-2} + x_{i-1} - x_{i+1} - x_{i+2} - x_{i+3} \rvert$
9	$\sum_{i=0}^{N-2} \lvert x_{i-2} + x_{i-1} - x_{i+1} - x_{i+2} \rvert$
10	$\sum_{i=4}^{N-4} \lvert x_{i-4} + x_{i-3} + ... + x_{i-1} - x_{i+1} - ... - x_{i+3} - x_{i+4} \rvert$
11	$\sum_{i=5}^{N-5} \lvert x_{i-5} + x_{i-4} + ... + x_{i-1} - x_{i+1} - ... - x_{i+4} - x_{i+5} \rvert$
12	$\sum_{i=6}^{N-6} \lvert x_{i-6} + x_{i-5} + ... + x_{i-1} - x_{i+1} - ... - x_{i+5} - x_{i+6} \rvert$
13	$\sum_{i=7}^{N-7} \lvert x_{i-7} + x_{i-6} + ... + x_{i-1} - x_{i+1} - ... - x_{i+6} - x_{i+7} \rvert$
14	$\sum_{i=4}^{N-4} \sum_{k=0}^{4} \lvert x_{i-k} - x_{i+k} \rvert$
15	$\sum_{i=5}^{N-5} \sum_{k=0}^{5} \lvert x_{i-k} - x_{i+k} \rvert$
16	$\sum_{i=6}^{N-6} \sum_{k=0}^{6} \lvert x_{i-k} - x_{i+k} \rvert$
17	$\sum_{i=7}^{N-7} \sum_{k=0}^{7} \lvert x_{i-k} - x_{i+k} \rvert$
18	$\sum_{i=10}^{N-10} \sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert$
19	$\sum_{i=15}^{N-15} \sum_{k=0}^{15} \lvert x_{i-k} - x_{i+k} \rvert$
20	$\sum_{i=20}^{N-20} \sum_{k=0}^{20} \lvert x_{i-k} - x_{i+k} \rvert$
21	$\sum_{i=25}^{N-25} \sum_{k=0}^{25} \lvert x_{i-k} - x_{i+k} \rvert$
22	$\sum_{i=10}^{N-10}((1 + \sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert)/(1 + \sum_{k=-10}^{9} \lvert x_{i-k} - x_{i+k} \rvert))$
23	$\sum_{i=10}^{N-10} \sqrt{(\sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert)^2/(1 + \sum_{k=-10}^{9} \lvert x_{i-k} - x_{i+k} \rvert)}$

Table 3.1: The trace functionals T used in the experiments. N is the total number of points along the line and x_i is the ith sample along the tracing line. Continued ...

Number	Functional
24	$\sum_{i=0}^{N-2} \lvert x_i - 2x_{i+1} + x_{i+2} \rvert$
25	$\sum_{i=0}^{N-3} \lvert x_i - 3x_{i+1} + 3x_{i+2} - x_{i+3} \rvert$
26	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert$
27	$\sum_{i=0}^{N-5} \lvert x_i - 5x_{i+1} + 10x_{i+2} - 10x_{i+3} + 5x_{i+4} - x_{i+5} \rvert$
28	$\sum_{i=0}^{N-2} \lvert x_i - 2x_{i+1} + x_{i+2} \rvert x_{i+1}$
29	$\sum_{i=0}^{N-3} \lvert x_i - 3x_{i+1} + 3x_{i+2} - x_{i+3} \rvert x_{i+1}$
30	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert x_{i+2}$
31	$\sum_{i=0}^{N-5} \lvert x_i - 5x_{i+1} + 10x_{i+2} - 10x_{i+3} + 5x_{i+4} - x_{i+5} \rvert x_{i+2}$

Table 3.2: The trace functionals T used in the experiments. N is the total number of points along the line and x_i is the ith sample along the tracing line.

Number	Functional
1	$Max_{i=0}^{N} x_i$
2	$Min_{i=0}^{N} x_i$
3	$\sqrt{\sum_{i=0}^{N} x_i^2}$
4	$\frac{\sum_{i=0}^{N} i x_i}{\sum_{i=0}^{N} x_i}$
5	$\sum_{i=0}^{N} i x_i$
6	$\frac{1}{N} \sum_{i=0}^{N} (x_i - \hat{x})^2$
7	c so that: $\sum_{i=0}^{c} x_i = \sum_{i=c}^{N} x_i$
8	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$
9	c so that: $\sum_{i=0}^{c} \lvert x_{i+1} - x_i \rvert = \sum_{i=c}^{N-1} \lvert x_{i+1} - x_i \rvert$
10	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert$

Table 3.3: The diametric functionals P used in the experiments.

Number	Functional
1	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert^2$
2	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$
3	$\sqrt{\sum_{i=0}^{N} x_i^2}$
4	$\sum_{i=0}^{N} x_i$
5	$Max_{i=0}^{N} x_i$
6	$Max_{i=0}^{N} x_i$- $Min_{i=0}^{N} x_i$
7	i so that $x_i = Min_{i=0}^{N} x_i$
8	i so that $x_i = Max_{i=0}^{N} x_i$
9	i so that $x_i = Min_{i=0}^{N} x_i$ without first harmonic
10	i so that $x_i = Max_{i=0}^{N} x_i$ without first harmonic
11	Amplitude of the first harmonic
12	Phase of the first harmonic
13	Amplitude of the second harmonic
14	Phase of the second harmonic
15	Amplitude of the third harmonic
16	Phase of the third harmonic
17	Amplitude of the fourth harmonic
18	Phase of the fourth harmonic

Table 3.4: The circus functionals Φ used in the experiments

We define a triple feature with the help of two more functionals designated by letters P and Φ and called diametrical and circus functionals, respectively. P is a functional of the Trace transform function when it is considered as a function of the length of the normal to the line only. Φ is a functional operating on the orientation variable after the previous two operations have been performed. Thus, the triple feature Π is defined as:

$$\Pi(F, C_1) = \Phi(P(T(F(C_1; \phi, p, t)))) \tag{3.4}$$

We use 3.1, 3.2 and 3.3 to obtain:

$$\Pi(F, C_2) = \Phi(P(T(F(C_1; \phi - \theta, \nu[p - s_0 \cos(\psi_0 - \phi)], \nu[t - s_0 \sin(\psi_0 - \phi)])))) \tag{3.5}$$

We select T so that it is invariant to displacement, i.e $T(\xi(t + b)) = T(\xi(t))$ and also obeys the property $T(\xi(at)) = \alpha_T(a)T(\xi(t))$ for $a \geq 0$, So we have:

$$\Pi(F, C_2) = \Phi(P(\alpha_T(\nu)T(F(C_1; \phi - \theta, \nu[p - s_0 \cos(\psi_0 - \phi)], t)))) \tag{3.6}$$

Again we can define P so that, it is invariant to displacement and has the property that $P(c\dot{\xi}(p)) = \gamma_P(c)P(\xi(p))$. Then we have:

$$\Pi(F, C_2) = \Phi(\gamma_P(\alpha_T(\nu))(P(T(F(C_1; \phi - \theta, \nu[p - s_0 \cos(\psi_0 - \phi)], t)))) \tag{3.7}$$

and because of its scaling property and invariance to displacement we have:

$$\Pi(F, C_2) = \Phi(\gamma_P(\alpha_T(\nu))\alpha_P(\nu)(P(T(F(C_1; \phi - \theta, p, t))))) \tag{3.8}$$

Finally if Φ is invariant to displacement and also has the property $\Phi(c\xi(\phi)) = \gamma_\phi(c)\Phi(\xi(\phi))$, we can write:

$$\Pi(F, C_2) = \gamma_\Phi(\gamma_P(\alpha_T(\nu))\alpha_P(\nu)(P(T(F(C_1; \phi, p, t))))) \tag{3.9}$$

It can be shown that for a functional Ξ [36] $\alpha_\Xi(a) = a^{\kappa_\Xi}$ and $\gamma_\Xi(c) = a^{\lambda_\Xi}$, so finally we have:

$$\Pi(F, C_2) = \nu^{\lambda_\Phi(\kappa_T\lambda_P + \kappa_P)}\Pi(F, C_1) \tag{3.10}$$

Here we have two ways to remove dependence on scaling and in both of them first we must normalise the features: We may write:

$$\Pi(F, C_2) = \nu^{-\omega}\Pi(F, C_1) \tag{3.11}$$

where:

$\omega = -\lambda_\Phi(\kappa_T\lambda_P + \kappa_P)$

If we take the ω^{th} root of both sides:

$$\sqrt[\omega]{\Pi(F, C_2)} = \nu^{-1}\sqrt[\omega]{\Pi(F, C_1)} \tag{3.12}$$

Or:

$$\Pi_{norm}(F, C_2) = \nu^{-1}\Pi_{norm}(F, C_1) \tag{3.13}$$

1- If we normalise the features as above and then find the ratio of two different features of the same image, we shall produce a feature that does not depend on ν, and is constant for all images which belong to the same class, provided they differ from each other by rotation, translation and scaling:

$$\frac{\Pi_{norm_i}(F, C_2)}{\Pi_{norm_j}(F, C_2)} = \frac{\Pi_{norm_i}(F, C_1)}{\Pi_{norm_j}(F, C_1)} \tag{3.14}$$

So in this way we produce features that can be used to classify images which belong to the same class. First we must find the ratio of two features of one image and then compare it with the corresponding ratio of other images. If they have the same value, we can put the two images in one class.

2- If we find the ratios of the values of the corresponding features computed from two images, they must all be the same, equal to the scaling coefficient between the two images. In other words, if we want to verify whether, for example images 1 and 2 are of same class or not, we must find the ratios of all features of them, and check whether they are the same.

$$\frac{\Pi_{norm_i}(F, C_2)}{\Pi_{norm_i}(F, C_1)} = \frac{\Pi_{norm_j}(F, C_2)}{\Pi_{norm_j}(F, C_1)} \tag{3.15}$$

3.4 Applications of the Trace Transform

The Trace transform can be used in many applications.

3.4.1 Two Dimensional Object Recognition

When we have a database of objects, and want to find a rotated and/or translated and/or scaled one, we can use the Trace transform. For example in [36] Kadyrov and Petrou used the Trace transform to find a fish between 94 images of several fish. In such a case, the features extracted are important and can be unique for each object. By simple combination of a collection of these features, it is possible to classify any object including affine version of it [37, 65].

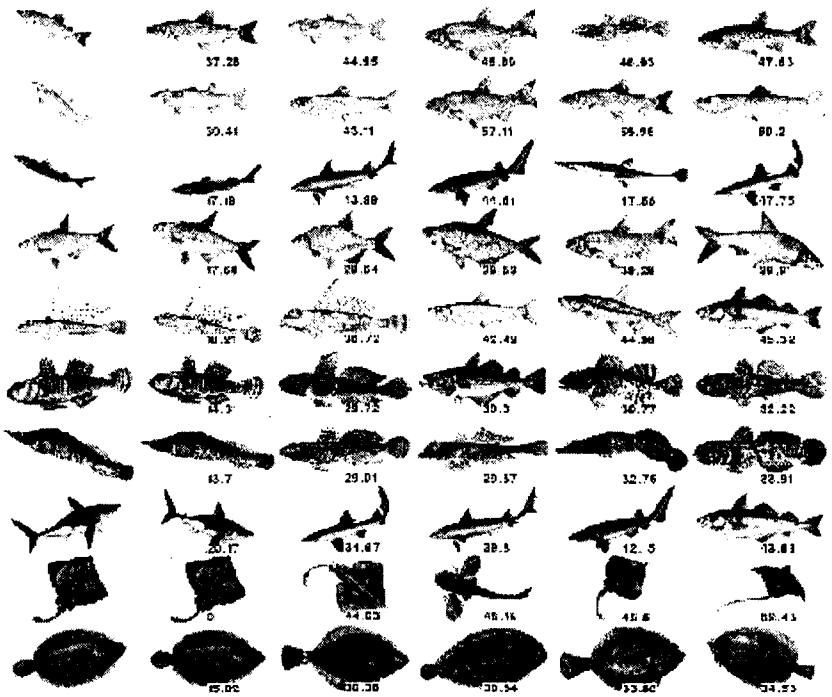


Figure 3.7: Querying the database with the images in the first column produces the answers in all other columns arranged in order of similarity. The values of the similarity measure are given next to each object retrieved. Figure taken from [36].

3.4.2 Change Detection

By using all functionals of tables 1,2 and 3, we can produce 5580 features for any image. We can check each one separately to see which of them best correlates with a particular

phenomenon we wish to monitor. Kadyrov and Petrou in [36] used this approach to detect the level of change in car park use.

3.4.3 Any Image Recognition

As we have seen, the Trace transform can be used to build a large number of features which characterise an image. So it is quite logical to use this method in all cases, to produce many features and then use a feature selection algorithm to identify the features which are most appropriate to identify an image. So we can use this method for face recognition [77], finger print recognition, character recognition (OCR)[38], etc [17]. And also we can use this method in conjunction with sparse and irregularly sampled data.

Chapter 4

Texture Classification with thousands of features

4.1 Introduction

The wealth of objects around us requires a wealth of descriptors. It is very unlikely that a few characteristics measured from the images of these objects will suffice to allow us to discriminate all objects we see. And yet our brain recognises thousands of objects, working mostly at the sub-conscious level. That is the reason knowledge engineering is very difficult: it is very hard to identify the characteristics which allow us to identify easily so many different faces, objects, materials, textures etc. Restricting ourselves, therefore, in computer vision to features that we can *consciously* identify as characterising our cognition, excludes the vast number of features that our *sub-conscious* uses and which we cannot usually identify. We may, however, replace the mechanism of our sub-conscious with a Mathematical tool that allows us to construct thousands of features that do not have physical or other meaning: we may use the Trace transform which is an alternative image representation and from which we can construct the so called triple features [36]. In [36] it was shown how one can construct such features invariant to rotation, translation and scaling, while in [35] it was shown how to construct object signatures invariant to affine transforms. In this chapter we are not trying to construct invariant features. Instead we are simply trying to construct

many features, each of which captures some aspect of the image. We then train the system to allow us to give relative importance to each feature with respect to the task we have. In the specific example presented, the task in question is texture discrimination.

One of the most well established methods for texture discrimination is based on the use of Co-occurrence matrices, and in particular on the use of features extracted from them [26]. The Co-occurrence matrix captures the second order statistics of a stationary texture and computes some quantities from them that have perceptual meaning: contrast, homogeneity, etc. As Co-occurrence matrices are considered by many a bench mark for texture analysis, we are going to use them here to discriminate textures from the Brodatz album [29] and compare their performance with the results produced by the Trace transform method.

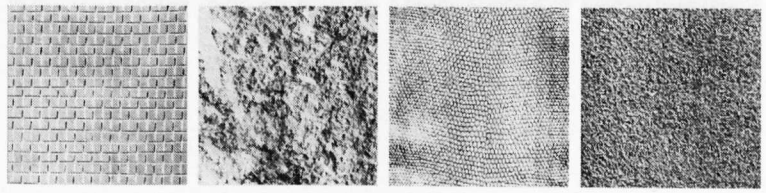
This chapter is organised as follows: In section 2 we present our methodology for texture classification based on the trace transform. In sections 3 we present results from some texture discrimination experiments. In section 4 we see how the Trace transform can be used for perceptual grouping. Finally in section 5 we present our conclusions.

4.2 Texture features from the trace transform

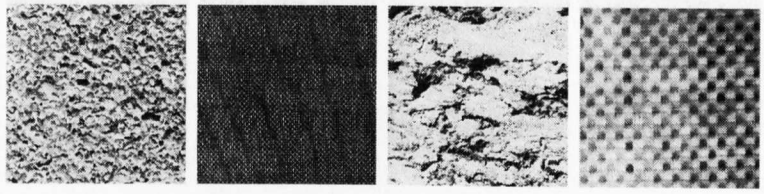
Using the triple feature construction method described in chapter 3, we can produce many features that characterise an image. To demonstrate our ideas we consider a database consisting of 112 images of texture from the Brodatz album, obtained from [29] and shown in figures 4.1 to 4.5.

These images are 640×640 in size. From each image four sub-images were created by dividing it into four quadrants. These sub-images are 320×320 pixels and they constitute a database of 448 textures of 112 classes. Let us call the set of 112 different textures made up from the top left quadrants, set TL, that made up from the top right quadrants, set TR, that made up from the bottom left quadrants, set BL and that made up from the bottom right quadrants set BR. We use two sets namely sets TR and TL for training and the other sets for testing.

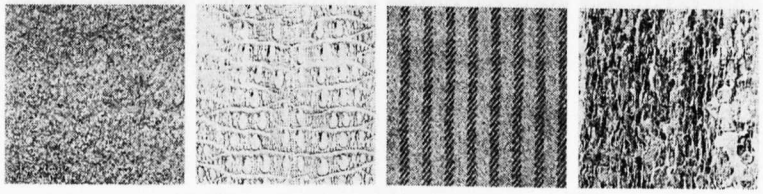
Let us consider a feature f_{klm} computed by combining the k^{th} trace functional, with



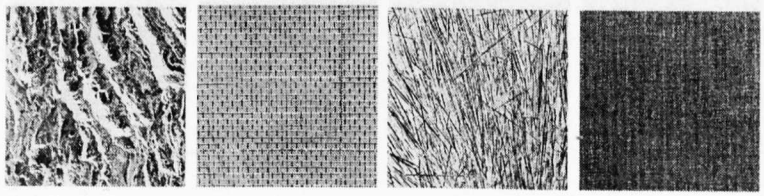
(a) T1 (b) T2 (c) T3 (d) T4



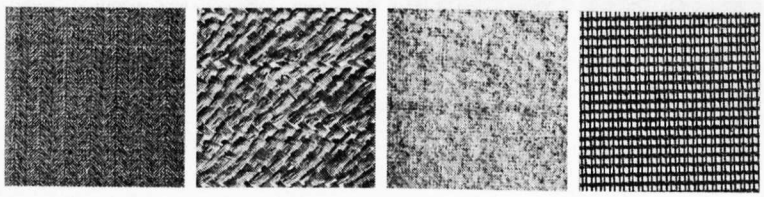
(e) T5 (f) T6 (g) T7 (h) T8



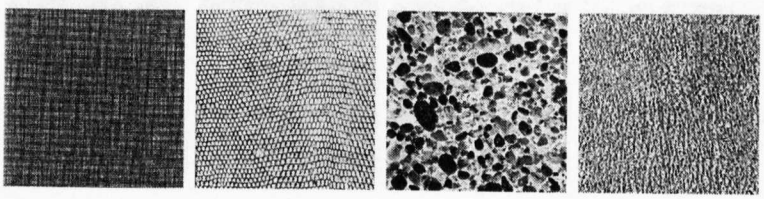
(i) T9 (j) T10 (k) T11 (l) T12



(m) T13 (n) T14 (o) T15 (p) T16

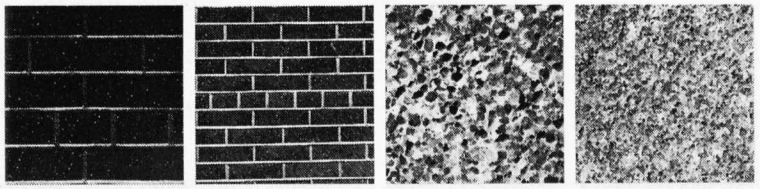


(q) T17 (r) T18 (s) T19 (t) T20

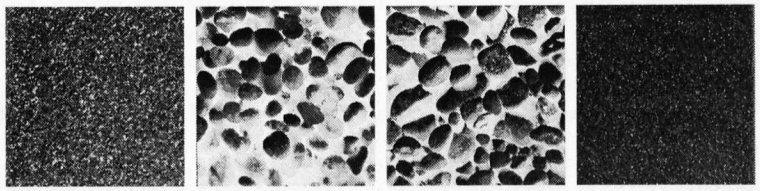


(u) T21 (v) T22 (w) T23 (x) T24

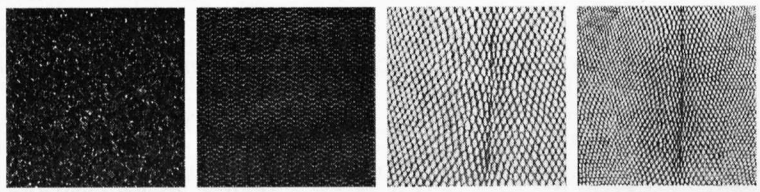
Figure 4.1: 112 textures of the Brodatz Album [29]. Continued ...



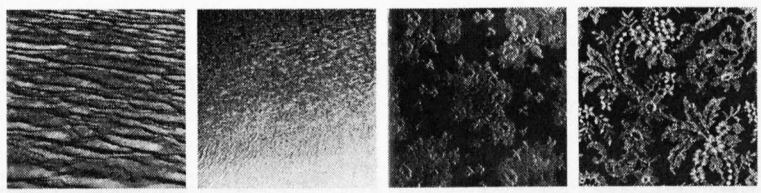
(a) T25 (b) T26 (c) T27 (d) T28



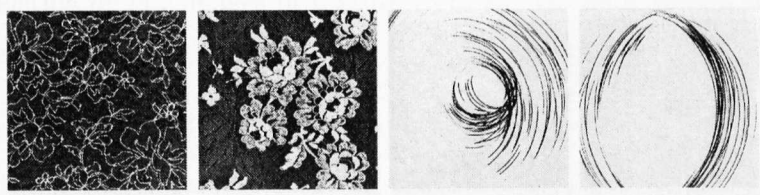
(e) T29 (f) T30 (g) T31 (h) T32



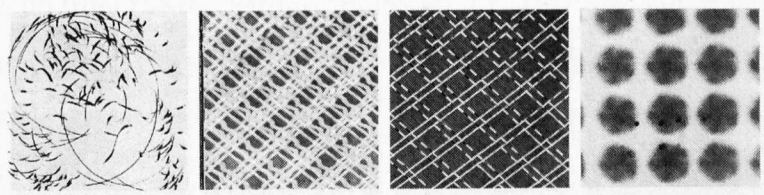
(i) T33 (j) T34 (k) T35 (l) T36



(m) T37 (n) T38 (o) T39 (p) T40

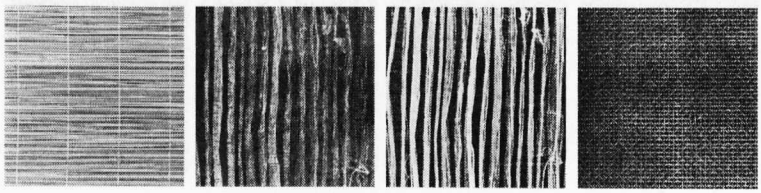


(q) T41 (r) T42 (s) T43 (t) T44

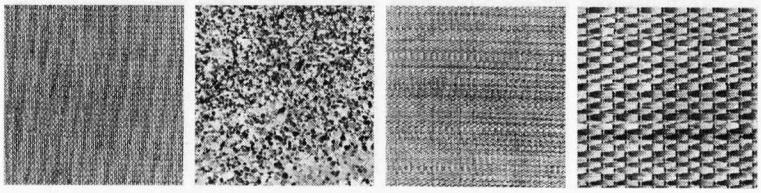


(u) T45 (v) T46 (w) T47 (x) T48

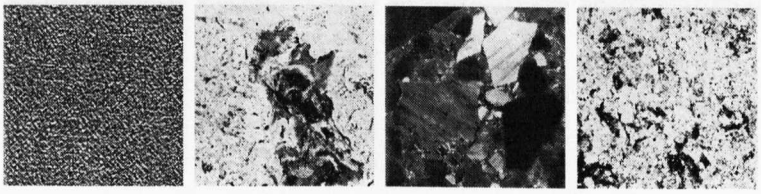
Figure 4.2: 112 textures of the Brodatz Album [29]. Continued ...



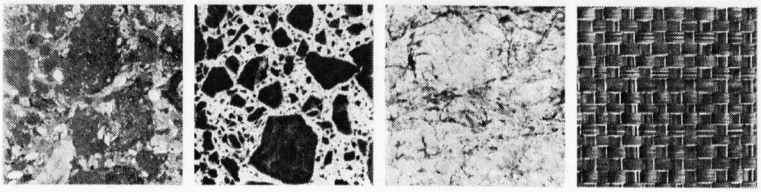
(a) T49 (b) T50 (c) T51 (d) T52



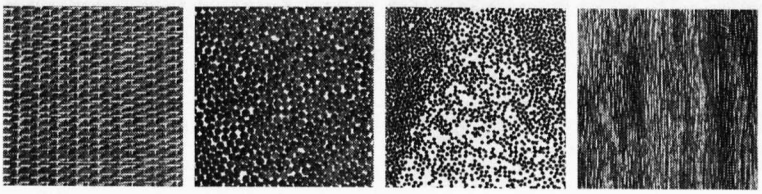
(e) T53 (f) T54 (g) T55 (h) T56



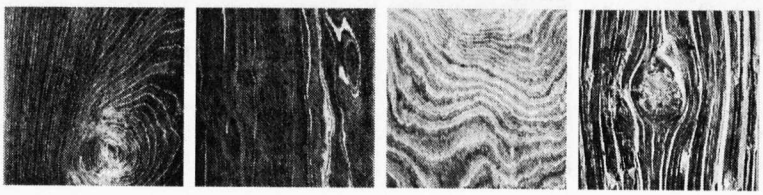
(i) T57 (j) T58 (k) T59 (l) T60



(m) T61 (n) T62 (o) T63 (p) T64

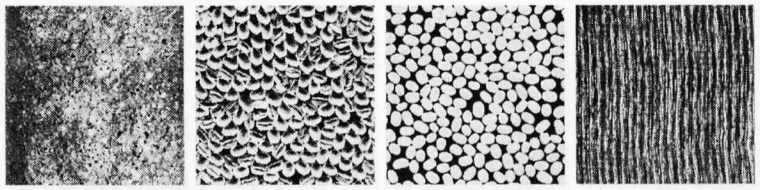


(q) T65 (r) T66 (s) T67 (t) T68

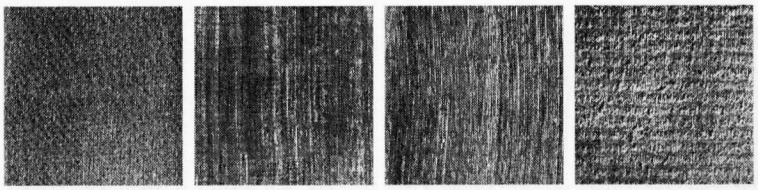


(u) T69 (v) T70 (w) T71 (x) T72

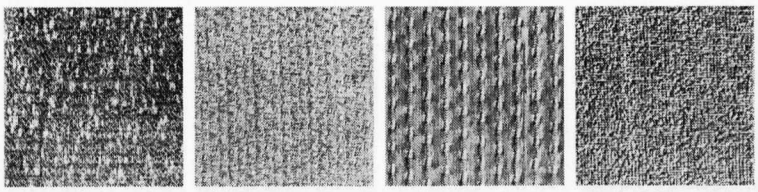
Figure 4.3: 112 textures of the Brodatz Album [29]. Continued ...



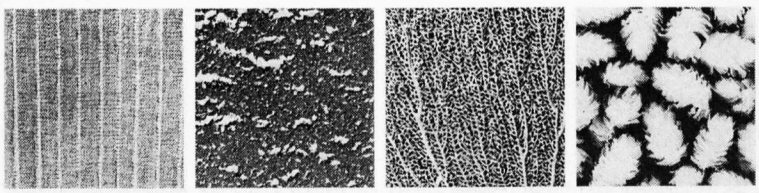
(a) T73 (b) T74 (c) T75 (d) T76



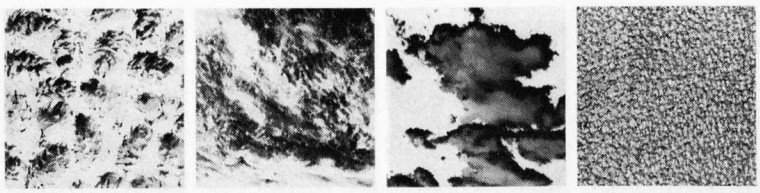
(e) T77 (f) T78 (g) T79 (h) T80



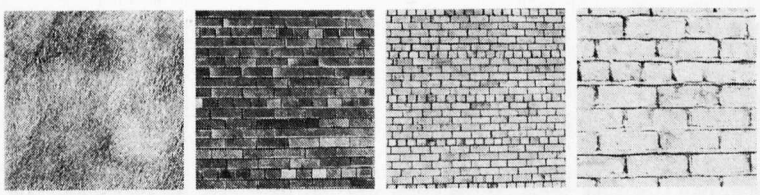
(i) T81 (j) T82 (k) T83 (l) T84



(m) T85 (n) T86 (o) T87 (p) T88

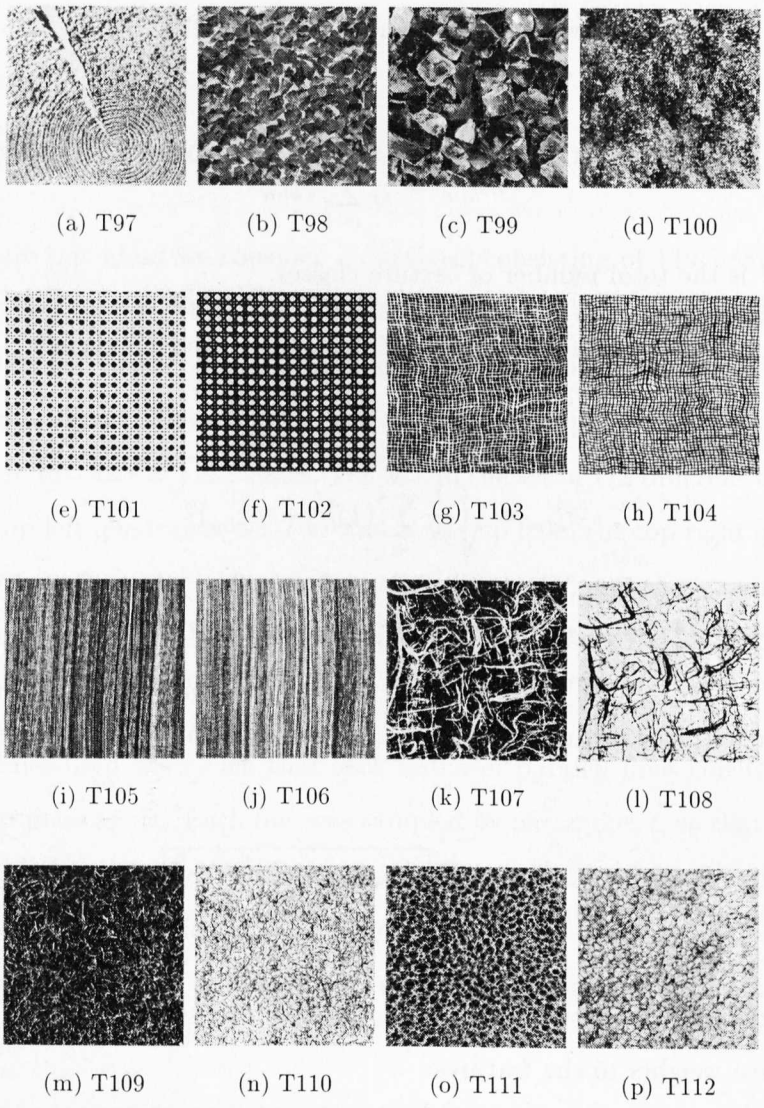


(q) T89 (r) T90 (s) T91 (t) T92



(u) T93 (v) T94 (w) T95 (x) T96

Figure 4.4: 112 textures of the Brodatz Album [29]. Continued ...



(a) T97 (b) T98 (c) T99 (d) T100

(e) T101 (f) T102 (g) T103 (h) T104

(i) T105 (j) T106 (k) T107 (l) T108

(m) T109 (n) T110 (o) T111 (p) T112

Figure 4.5: 112 textures of the Brodatz Album [29].

the l^{th} diametrical functional and the m^{th} circus functional. Let us say that its value for texture class c, and instantiation of this class s is denoted by f_{klm}^{cs}. Since we have two instantiations of each class in our training set, one from set TL and one from set TB, $s = 1$ or $s = 2$. We compute the average value of this feature over all $s = 1$ substantiations of the training texture samples we have. (Note that it does not really matter which instantiation we call $s = 1$ and which we call $s = 2$.):

$$m_{klm}^{1} = \frac{1}{N} \sum_{c=1}^{N} f_{klm}^{c1} \tag{4.1}$$

where here N is the total number of texture classes.

We also compute the standard deviation of this feature over all classes:

$$\sigma_{klm}^{1} = \sqrt{\frac{1}{N} \sum_{c=1}^{N} (f_{klm}^{c1} - m_{klm}^{1})^2} \tag{4.2}$$

A feature is useful in characterising textures, if its value is stable when the instantiation of a texture changes. So we define an average stability measure for each feature and scale it by the variance of the values of the feature over the whole database:

$$q_{klm} = \frac{1}{\sigma_{klm}^{1}} \sqrt{\frac{1}{N} \sum_{c=1}^{N} (f_{klm}^{c1} - f_{klm}^{c2})^2} \tag{4.3}$$

The smaller q_{klm} is, the better feature f_{klm} is. We may set a threshold Q which will allow us to give weights to the features:

$$w_{klm} = \begin{cases} Q - q_{klm} & \text{if } q_{klm} < Q, \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Finally, the "distance" *dist* between a test sample *test* and any reference sample *ref* can be obtained by using:

$$dist(test, ref) \equiv \sum_{klm} w_{klm} \frac{1}{\sigma^1_{klm}} |f^{test}_{klm} - f^{ref}_{klm}| \tag{4.5}$$

Note that as the value of Q increases, more features are included in the computation of $dist$.

4.3 Experiments on texture recognition

To demonstrate our ideas we consider a database consisting of 112 images of texture from the Brodatz album, obtained from [29] and shown in figures 4.1 to 4.5. These images are 640×640 in size. From each image four sub-images were created by dividing it into four quadrants. These sub-images are 320×320 pixels and they constitute a database of 448 textures of 112 classes. Let us call the set of 112 different textures made up from the top left quadrants, set TL, that made up from the top right quadrants, set TR, that made up from the bottom left quadrants, set BL and that made up from the bottom right quadrants set BR. We use two sets, namely sets TR and TL, for training and the other sets for testing.

The tracing lines used were such that each batch of parallel lines consisted of lines 2 inter-pixel distances apart. Each line was sampled by parameter t, so that the sampling points were also 2 inter-pixel distances apart. For each value of p, 20 different orientations were used, ie the orientations of the lines with the same p differed by 18^o. We only considered the part of each image that was inside the maximum inscribed circle in each 320×320 pixels square.

The significance of each feature was extracted from the training samples, and subsequently each one of the test samples was associated with the reference texture from which the distance value computed by equation (4.5) was least.

Table 4.1 presents the results of this approach for identifying the correct class of a texture as the most similar one, the second most similar one, the third most similar one, the fourth most similar one, and beyond, presenting the numbers under the corresponding columns. All these numbers are out of 112, as we present the results of

testing separately for each set of data. In all cases the reference set was the TL set of images. Each row of results corresponds to a different choice of threshold Q. Note that after a certain value of Q (which is about 0.3) all features are used in the calculation and the performance of the system stabilises.

The features computed from the Trace transform may also be used in a different way: First we train the system, by for example, 30 textures and find all necessary parameters for the distance formula, and then use all remaining textures for testing. So, this method is almost an unsupervised texture classification method, as the system is called to classify textures it has not encountered before. We call this method "Blind Trace Transform" method (BTT) to distinguish it from the previous one which we call "Trace Transform" method (TT). The results of BTT are shown in table 4.2.

Further, we apply the classification method described in section 4.2 using as features those extracted by the following methods:

- Using the following 5 features obtained from the co-occurrence matrix, which have some perceptual meaning. We call this method "Features from the Co-occurrence Matrix" and denote it by FCM. These features are: Energy, Entropy, Contrast, Correlation and Homogeneity which are defined as [26]. The results are in tables 4.3 and 4.4.

- Using all elements of the co-occurrence matrix as features in the distance formula. We used here the co-occurrence matrix computed with 4-bit resolution (i.e. co-occurrence matrix of size 16×16) to have to deal with fewer elements. We name this method CM16. The results are in tables 4.5 to 4.7.

- We augment the set of features of the previous method by also including the 5 perceptual features extracted from it, to produce CM16+F. The results are in tables 4.8 to 4.10.

- Using all the elements of the co-occurrence matrix computed with 8-bit resolution (i.e. co-occurrence matrix of size 256×256). We call this method CM256. The results are in table 4.11.

- We augment the previous method by adding the 5 extracted features from the

co-occurrence matrix. This method is CM256+F. The results are in table 4.12.

- Using the 5 perceptual features extracted from the sum and difference histograms. This method is called "Features from the Sum and Difference Histograms" or FSDH [84]. The results are in table 4.13.

- Using all the elements of the sum and difference histograms computed with 4-bit resolution as features in the distance formula. We call this method SDH16. The results are in tables 4.14 to 4.16.

- We augment the previous method by adding to the elements of the histograms the 5 perceptual features extracted from them, and so we have method SDH16+F. The results are in tables 4.17 to 4.19.

- Using as features all the elements of the sum and difference histograms computed with 8-bit resolution. We call this method SDH256. The results are in table 4.20.

- We augment the previous method by adding the 5 extracted features from the histograms. This method is SDH256+F. The results are in table 4.21.

Table 4.30 shows the best results of each one of the methods obtained as the value of threshold Q and the value of distance for which the co-occurrence matrix was computed, changed. We observe that methods which rely only on the use of perceptually meaningful features (ie contrast, homogeneity etc) perform worse than the methods which rely on the use of many features, even if these features have no perceptual meaning. The best method of all is the one based on using all elements of the co-occurrence matrix with 8 bit grey scale resolution. We note that BTT performs remarkably well given that it is in practice an unsupervised method.

Q	Test Set BL					Test Set BR				
	1	2	3	4	R	1	2	3	4	R
0.1	1	1	1	1	108	1	1	1	1	108
0.2	38	14	6	6	48	39	11	4	5	53
0.3	84	10	2	0	16	70	12	6	5	19
0.4	89	10	2	1	10	80	9	5	3	15
0.5	90	11	1	1	9	81	11	3	3	14
0.6	89	11	3	0	9	80	11	3	5	13
0.7	89	12	2	0	9	80	10	4	4	14
0.8	89	13	1	0	9	79	10	5	4	14
0.9	89	14	0	0	9	79	11	4	3	15
1.0	92	10	1	0	9	81	8	6	4	13
1.1	93	8	1	0	10	80	11	6	2	13
1.2	95	6	1	0	10	78	16	4	2	12
1.3	95	6	0	1	10	82	11	3	4	12
1.4	93	7	1	1	10	82	11	2	2	15
1.5	90	10	1	1	10	80	13	2	2	15
1.7	88	10	2	2	10	78	14	3	2	15
2.3	85	11	4	2	10	75	12	4	5	16
2.6	84	12	3	2	11	74	11	6	4	17
3.0	83	12	4	2	11	74	8	9	2	19

Table 4.1: Texture classification results using the trace transform method (TT). Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. All numbers are out of 112.

Q	Test Set BL					Test Set BR				
	1	2	3	4	R	1	2	3	4	R
0.5	65	6	3	2	6	52	9	3	4	14
1.0	66	7	2	0	7	51	9	3	5	14
1.5	62	10	3	0	7	49	9	4	4	16
2.0	59	13	2	2	6	47	9	5	4	17

Table 4.2: Texture classification results using the blind trace transform method ($\mathrm{BT\dot{T}}$). Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. Here the total number of testing set textures is 82 and the training set consists of completely different textures from the testing set.

Q	Test Set BL					Test Set BR				
	1	2	3	4	R	1	2	3	4	R
0.1	1	1	1	1	108	1	1	1	1	108
0.2	1	1	1	1	108	1	1	1	1	108
0.3	46	21	12	3	30	33	18	8	6	47
0.4	61	17	7	5	22	38	19	8	5	42
0.5	60	19	8	2	23	40	17	6	6	43
0.6	65	14	7	2	24	39	17	6	10	40
0.7	65	14	9	3	21	39	17	7	10	39
0.8	66	13	8	3	22	41	14	10	9	38
0.9	65	18	6	2	21	40	17	8	9	38
1.0	64	19	7	1	21	41	16	7	10	38
1.1	64	19	7	1	21	41	16	6	10	39
1.2	63	20	6	2	21	41	16	7	9	39
1.3	64	19	6	3	20	41	16	7	8	40
1.4	64	19	7	2	20	41	15	8	8	40
1.5	64	19	6	3	20	41	15	9	7	40
1.6	63	20	6	2	21	42	14	9	6	41
1.7	63	20	6	2	21	42	14	8	7	41
1.8	63	20	6	2	21	42	14	8	7	41
1.9	63	20	6	2	21	42	14	8	7	41
2.0	63	20	6	1	22	41	15	9	6	41

Table 4.3: Texture classification results using features extracted from the co-occurrence matrix (FCM) constructed for $d = 2$. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. All numbers are out of 112.

Q	Test Set BL					Test Set BR				
	1	2	3	4	R	1	2	3	4	R
0.1	1	1	1	1	108	1	1	1	1	108
0.2	1	1	1	1	108	1	1	1	1	108
0.3	43	21	13	6	29	33	17	8	9	45
0.4	58	15	10	4	25	39	15	7	9	42
0.5	59	18	10	4	21	39	19	7	3	44
0.6	59	20	9	4	20	39	21	6	4	42
0.7	60	18	10	5	19	42	18	4	7	41
0.8	60	18	10	4	20	42	17	9	6	38
0.9	60	17	10	5	20	42	18	8	6	38
1.0	60	17	9	5	21	42	17	8	7	38
1.1	60	18	8	5	21	42	17	8	8	37
1.2	61	17	8	5	21	42	17	8	7	38
1.3	62	16	8	6	20	42	17	8	7	38
1.4	62	16	8	6	20	42	17	8	7	38
1.5	63	15	8	6	20	42	16	8	8	38
1.6	64	14	8	6	20	43	15	8	8	38
1.7	64	14	8	6	20	43	15	8	8	38
1.8	64	14	8	6	20	44	14	8	8	38
1.9	64	14	8	6	20	44	14	8	8	38
2.0	65	13	8	7	19	44	14	8	8	38

Table 4.4: Texture classification results using features extracted from the co-occurrence matrix (FCM) constructed for $d = 5$. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	100	6	3	0	3	83	11	5	2	11
1	1	100	7	2	0	3	82	13	2	4	11
1	1.5	98	8	4	0	2	84	11	0	3	14
1	2	99	7	4	0	2	84	9	2	2	15
1	2.5	99	8	3	0	2	84	8	3	2	15
2	0.5	100	9	0	0	3	79	9	11	3	10
2	1	101	5	3	0	3	82	11	4	1	14
2	1.5	101	5	3	0	3	79	13	3	4	13
2	2	100	6	3	0	3	79	12	3	3	15
2	2.5	99	7	3	0	3	77	13	4	3	15
3	0.5	100	7	2	0	3	75	13	6	3	15
3	1	101	6	1	1	3	77	13	5	2	15
3	1.5	99	6	2	2	3	73	15	6	2	16
3	2	98	7	2	1	4	71	14	8	2	17
3	2.5	98	7	2	1	4	70	14	9	2	17
4	0.5	101	6	1	1	3	72	17	5	2	16
4	1	99	8	1	1	3	74	16	5	2	15
4	1.5	97	8	3	1	3	69	18	6	3	16
4	2	97	7	4	0	4	69	16	9	2	16
4	2.5	96	7	5	0	4	67	18	7	3	17

Table 4.5: This table shows the results of using all elements of the co-occurrence matrix as features in texture classification with 16 grey levels (CM16), for distances $1-4$. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
5	0.5	94	11	3	0	4	69	16	7	4	16
5	1	97	10	2	0	3	71	16	7	3	15
5	1.5	95	10	2	2	3	67	16	10	2	17
5	2	95	9	3	1	4	64	19	8	2	19
5	2.5	95	9	3	1	4	63	19	8	3	19
6	0.5	93	10	4	1	4	64	21	7	5	15
6	1	93	12	3	1	3	66	18	7	5	16
6	1.5	94	10	3	2	3	64	16	8	4	20
6	2	93	11	3	2	3	63	16	8	3	22
6	2.5	91	12	4	2	3	63	16	8	3	22
7	0.5	93	8	5	2	4	65	18	7	5	17
7	1	92	11	4	2	3	63	20	8	3	18
7	1.5	92	13	3	1	3	61	15	9	6	21
7	2	91	12	5	1	3	63	13	7	7	22
7	2.5	91	12	4	2	3	62	14	7	5	24
8	0.5	95	6	5	2	4	64	17	9	6	16
8	1	94	10	3	2	3	61	19	8	6	18
8	1.5	92	12	4	1	3	60	14	11	6	21
8	2	90	12	5	2	3	62	10	12	3	25
8	2.5	90	11	5	2	4	59	14	11	4	24

Table 4.6: This table shows the results of using all elements of the co-occurrence matrix as features in texture classification with 16 grey levels (CM16), for distances $5-8$. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
9	0.5	89	12	6	2	3	63	17	9	4	19
9	1	91	12	4	2	3	61	16	10	3	22
9	1.5	88	15	3	3	3	56	18	8	8	22
9	2	89	12	5	3	3	59	15	6	7	25
9	2.5	88	12	3	5	4	58	15	8	5	26
10	0.5	89	12	5	0	6	65	14	7	5	21
10	1	93	10	3	3	3	61	16	7	7	21
10	1.5	90	13	3	3	3	57	15	10	7	23
10	2	89	11	5	4	3	57	15	8	5	27
10	2.5	89	11	5	3	4	57	15	7	7	26

Table 4.7: This table shows the results of using all elements of the co-occurrence matrix as features in texture classification with 16 grey levels (CM16) for distances $9 - 10$. The first column indicates the distance for which the co-occurrence matrix has been computed. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	101	6	2	0	3	84	11	4	2	11
1	1	100	7	2	0	3	84	11	3	2	12
1	1.5	99	8	3	0	2	85	9	1	3	14
1	2	99	8	3	0	2	85	8	1	3	15
1	2.5	99	8	3	0	2	85	8	1	3	15
2	0.5	101	8	0	0	3	81	9	10	2	10
2	1	101	5	3	0	3	83	9	5	2	13
2	1.5	100	6	3	0	3	82	10	3	4	13
2	2	101	5	3	0	3	78	13	3	3	15
2	2.5	99	7	3	0	3	76	14	4	3	15
3	0.5	100	7	2	0	3	77	11	6	3	15
3	1	101	6	1	1	3	80	10	5	2	15
3	1.5	99	8	1	1	3	77	12	5	2	16
3	2	98	7	3	1	3	74	12	7	3	16
3	2.5	98	7	2	2	3	72	13	8	3	16
4	0.5	100	8	0	1	3	72	18	4	3	15
4	1	99	8	1	1	3	74	16	5	2	15
4	1.5	97	8	4	0	3	69	18	6	3	16
4	2	97	7	4	0	4	69	18	7	2	16
4	2.5	97	6	5	0	4	67	18	7	2	18

Table 4.8: This table shows the results of using all elements of the co-occurrence matrix with 16 grey levels for distances $1-4$ in addition to the five features extracted from it, as features for texture classification (CM16+F). 5appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
5	0.5	95	12	1	0	4	72	14	10	0	16
5	1	97	10	2	0	3	73	13	7	4	15
5	1.5	96	9	2	2	3	68	16	7	5	16
5	2	96	9	2	2	3	65	18	7	3	19
5	2.5	95	10	2	1	4	65	17	7	4	19
6	0.5	93	11	3	1	4	67	20	7	3	15
6	1	95	10	3	1	3	70	15	6	5	16
6	1.5	95	11	2	1	3	65	15	7	5	20
6	2	94	10	3	2	3	63	16	8	4	21
6	2.5	93	11	3	2	3	63	16	6	6	21
7	0.5	95	7	4	2	4	65	21	4	6	16
7	1	92	10	5	2	3	64	21	6	3	18
7	1.5	92	12	3	2	3	61	17	6	8	20
7	2	91	13	4	1	3	62	14	8	6	22
7	2.5	92	11	5	1	3	62	15	4	8	23
8	0.5	95	7	4	2	4	65	21	4	6	16
8	1	94	10	3	2	3	62	19	8	5	18
8	1.5	93	11	4	1	3	61	13	11	6	21
8	2	90	14	4	1	3	60	12	13	3	24
8	2.5	90	12	6	1	3	60	12	12	3	25

Table 4.9: This table shows the results of using all elements of the co-occurrence matrix with 16 grey levels for distances $5-8$ in addition to the five features extracted from it, as features for texture classification (CM16+F). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
9	0.5	91	11	5	1	4	64	17	8	5	18
9	1	92	10	5	2	3	63	15	10	3	21
9	1.5	90	13	3	3	3	57	16	10	6	23
9	2	89	12	4	4	3	58	16	7	6	25
9	2.5	88	12	4	5	3	58	15	9	5	25
10	0.5	91	11	4	1	5	66	15	6	5	20
10	1	94	9	3	3	3	63	14	10	4	21
10	1.5	92	10	4	3	3	59	16	9	5	23
10	2	90	11	4	4	3	57	15	9	5	26
10	2.5	90	10	4	5	3	56	17	7	6	26

Table 4.10: This table shows the results of using all elements of the co-occurrence matrix with 16 grey levels for distances $9-10$ in addition to the five features extracted from it, as features for texture classification (CM16+F). The first column indicates the distance for which the co-occurrence matrix has been computed. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	109	2	0	0	1	104	4	2	1	1
1	1	109	2	0	0	1	105	5	1	0	2
1	1.5	109	2	0	0	1	104	4	1	0	3
1	2	109	2	0	0	1	104	4	1	0	3

Table 4.11: This table shows the results of using all elements of co-occurrence matrix with 256 grey levels and $d = 1$ as our features (CM256). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	109	2	0	0	1	104	4	2	1	1
1	1	109	2	0	0	1	105	4	2	0	2
1	1.5	109	2	0	0	1	104	4	1	0	3
1	2	109	2	0	0	1	104	4	1	0	3

Table 4.12: This table shows the results of using all elements of co-occurrence matrix with 256 grey levels and $d = 1$ in addition to the five features extracted from it as our features (CM256+F). Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
2	0.5	63	17	10	4	18	42	15	12	4	39
2	1	67	15	8	5	17	45	13	10	9	35
2	1.5	68	14	8	6	16	44	13	11	9	35
2	2	69	14	6	7	16	44	14	9	11	34
2	2.5	70	13	6	8	15	44	14	9	12	33
5	1	59	18	7	8	20	44	15	6	5	42
5	1.5	60	18	6	7	21	43	15	8	5	41
5	2	60	19	5	6	22	43	15	8	6	40
5	2.5	61	18	5	6	22	43	15	8	7	39

Table 4.13: Texture classification results using features extracted from the sum and difference histograms constructed for *distance* = 2 and *distance* = 5 (FSDH). Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	98	8	3	0	3	78	16	6	3	9
1	1	98	7	3	1	3	75	16	7	5	9
1	1.5	97	9	3	0	3	78	13	6	5	10
1	2	97	9	2	0	4	78	12	6	5	11
1	2.5	97	9	2	0	4	78	12	6	5	11
2	0.5	95	12	2	1	2	73	15	10	3	11
2	1	92	14	4	0	2	76	15	5	6	10
2	1.5	95	11	4	0	2	77	11	6	5	13
2	2	95	11	4	0	2	77	11	6	5	13
2	2.5	96	10	4	0	2	77	11	6	6	12
3	0.5	102	4	2	2	2	74	13	7	5	13
3	1	97	11	2	0	2	71	16	7	6	12
3	1.5	96	10	4	0	2	72	14	6	6	14
3	2	97	9	3	1	2	72	14	6	6	14
3	2.5	97	9	3	1	2	72	14	6	6	14
4	0.5	100	6	2	1	3	75	9	7	8	13
4	1	103	4	2	1	2	72	15	6	8	11
4	1.5	103	2	4	1	2	72	15	4	8	13
4	2	101	4	4	1	2	70	17	4	8	13
4	2.5	101	4	5	0	2	71	16	4	8	13

Table 4.14: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $1-4$ as features in texture classification (SDH16). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
5	0.5	101	3	4	1	3	75	12	6	3	16
5	1	98	7	4	1	2	70	14	8	9	11
5	1.5	98	6	5	1	2	67	18	6	7	14
5	2	97	7	5	1	2	66	17	8	7	14
5	2.5	97	7	5	1	2	67	16	8	7	14
6	0.5	95	8	5	1	3	69	13	12	3	15
6	1	94	7	6	3	2	64	15	12	8	13
6	1.5	94	7	5	4	2	65	13	13	7	14
6	2	94	7	5	4	2	65	14	11	7	15
6	2.5	92	9	5	4	2	65	14	11	7	15
7	0.5	96	7	3	2	4	66	13	13	5	15
7	1	93	8	5	2	4	62	17	11	8	14
7	1.5	92	9	5	2	4	62	16	13	6	15
7	2	93	8	4	3	4	62	17	12	6	15
7	2.5	93	7	4	4	4	62	17	12	7	14
8	0.5	96	5	3	4	4	63	18	9	6	16
8	1	94	5	5	4	4	63	17	15	2	15
8	1.5	93	6	5	4	4	59	20	13	4	16
8	2	93	6	5	4	4	59	19	14	4	16
8	2.5	93	6	5	5	3	59	19	14	4	16

Table 4.15: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $5 - 8$ as features in texture classification (SDH16). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
9	0.5	97	7	2	2	4	61	20	8	5	18
9	1	92	9	4	3	4	58	18	14	5	17
9	1.5	94	7	3	4	4	56	20	11	6	19
9	2	93	8	3	4	4	56	20	11	5	20
9	2.5	92	9	3	4	4	56	20	11	5	20
10	0.5	95	6	2	4	5	63	16	7	9	17
10	1	93	7	4	4	4	59	19	11	3	20
10	1.5	92	8	4	4	4	58	19	11	4	20
10	2	92	9	3	4	4	58	19	11	2	22
10	2.5	92	9	3	4	4	58	18	12	1	23

Table 4.16: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $9 - 10$ as features in texture classification (SDH16). The first column indicates the distance for which the co-occurrence matrix has been computed. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	97	9	3	0	3	79	14	6	2	11
1	1	97	8	3	1	3	76	14	11	1	10
1	1.5	96	10	3	0	3	79	12	7	4	10
1	2	97	9	2	1	3	79	12	6	4	11
1	2.5	97	9	2	1	3	79	11	7	4	11
2	0.5	95	12	2	1	2	75	15	7	4	11
2	1	93	15	2	0	2	74	17	5	8	8
2	1.5	95	11	4	0	2	76	13	6	7	10
2	2	95	11	4	0	2	75	13	7	6	11
2	2.5	95	11	4	0	2	75	13	7	5	12
3	0.5	99	7	2	2	2	74	13	7	3	15
3	1	97	11	2	0	2	71	16	7	5	13
3	1.5	97	10	2	1	2	70	18	5	5	14
3	2	96	10	3	1	2	72	16	5	4	15
3	2.5	96	10	3	1	2	72	16	5	3	16
4	0.5	100	5	2	2	3	76	8	6	8	14
4	1	102	5	3	0	2	71	16	5	8	12
4	1.5	102	4	4	0	2	72	15	6	5	14
4	2	102	4	4	0	2	72	16	4	5	15
4	2.5	102	4	4	0	2	71	17	4	5	15

Table 4.17: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $1-4$ in addition to the five features extracted from them as features in texture classification (SDH16+F). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
5	0.5	101	3	4	1	3	73	12	4	6	17
5	1	99	7	4	0	2	70	14	6	9	13
5	1.5	98	6	6	0	2	67	17	8	6	14
5	2	97	7	5	1	2	67	17	8	5	15
5	2.5	95	9	5	1	2	67	17	8	4	16
6	0.5	96	7	4	2	3	71	12	9	7	13
6	1	94	9	4	3	2	64	14	13	7	14
6	1.5	92	10	5	3	2	64	16	10	7	15
6	2	92	10	5	3	2	64	16	11	6	15
6	2.5	92	9	6	3	2	64	16	11	5	16
7	0.5	96	4	8	1	3	67	12	13	5	15
7	1	94	7	6	1	4	62	16	14	5	15
7	1.5	93	7	7	1	4	61	17	14	5	15
7	2	92	8	5	3	4	61	17	14	4	16
7	2.5	92	8	5	3	4	61	19	12	4	16
8	0.5	96	5	3	5	3	63	16	13	6	14
8	1	94	5	7	3	3	60	20	12	4	16
8	1.5	94	5	5	5	3	60	19	13	3	17
8	2	93	6	5	5	3	60	19	12	4	17
8	2.5	93	6	5	5	3	60	20	11	4	17

Table 4.18: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $5-8$ in addition to the five features extracted from them as features in texture classification (SDH16+F). All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
9	0.5	95	7	4	3	3	63	17	9	8	15
9	1	94	8	2	4	4	57	19	13	7	16
9	1.5	92	10	2	4	4	58	19	13	4	18
9	2	92	8	4	4	4	59	18	12	3	20
9	2.5	93	7	4	4	4	59	18	12	3	20
10	0.5	93	8	3	4	4	62	18	8	5	19
10	1	93	7	4	4	4	59	19	9	5	20
10	1.5	93	7	4	4	4	59	20	9	1	23
10	2	92	8	3	5	4	60	19	9	1	23
10	2.5	91	8	4	5	4	60	18	10	1	23

Table 4.19: This table shows the results of using all elements of the sum and difference histograms with 16 grey levels for distance $9 - 10$ in addition to the five features extracted from them as features in texture classification (SDH16+F). The first column indicates the distance for which the co-occurrence matrix has been computed. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	98	8	3	0	3	77	17	6	3	9
1	1	98	7	3	1	3	75	16	7	5	9
1	1.5	97	9	3	0	3	77	14	6	5	10
1	2	97	9	2	0	4	77	13	7	5	10

Table 4.20: This table shows the results of using all elements of the sum and difference histograms with 256 grey levels and $d = 1$ (SDH256) as our features. All numbers are out of 112.

Distance	Q	Test Set BL					Test Set BR				
		1	2	3	4	R	1	2	3	4	R
1	0.5	97	9	3	0	3	79	16	4	2	11
1	1	97	8	4	0	3	75	16	8	5	8
1	1.5	97	9	3	0	3	76	15	5	6	10
1	2	97	9	2	1	3	76	16	4	7	9

Table 4.21: This table shows the results of using all elements of the sum and difference histograms with 256 grey levels and $d = 1$ in addition to the five extracted features from them (SDH256+F) as our features. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. All numbers are out of 112.

4.4 Perceptual grouping

To indicate how "gracefully" each method moves away from the correct textures when classifying textures, we show an example of 3 textures chosen at random from the

Method	Texture T31				Texture T41				Texture T51			
	1	2	3	4	1	2	3	4	1	2	3	4
TT	T31	T88	T27	T30	T41	T109	T86	T57	T51	T72	T95	T35
BTT	T31	T88	T46	T54	T109	T41	T86	T57	T51	T72	T35	T95
FCM	T31	T90	T62	T99	T41	T42	T68	T3	T74	T42	T51	T72
CM16	T31	T90	T30	T88	T41	T20	T67	T42	T51	T74	T62	T60
CM16+F	T31	T90	T88	T99	T41	T20	T107	T42	T51	T74	T26	T60
CM256	T31	T59	T90	T30	T41	T33	T102	T101	T51	T75	T25	T45
CM256+F	T31	T59	T90	T30	T41	T33	T102	T101	T51	T75	T25	T45
FSDH	T31	T90	T62	T58	T41	T109	T3	T10	T74	T51	T60	T42
SDH16	T31	T90	T30	T58	T41	T42	T109	T6	T42	T51	T74	T62
SDH16+F	T31	T30	T90	T89	T41	T6	T65	T42	T51	T62	T74	T23
SDH256	T31	T90	T58	T30	T41	T42	T20	T6	T74	T51	T42	T60
SDH256+F	T31	T30	T90	T99	T41	T6	T20	T109	T51	T72	T74	T62

Table 4.22: At the top row we have three examples of the texture database. Underneath each texture we show the code number of textures picked up by each method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.

database and we present the four most similar textures each method produced, in decreasing order of similarity, in table 4.22. we present in appendix A in figures A.1 to A.12 some sample results. We chose 3 textures from the database (Shown at the top row of each figure) and we show underneath the four most similar textures each method produced, in decreasing order of similarity from top to bottom. Table 4.22 summarises these results.

We note that none of the methods appears to pick up very similar to each other textures in the first four positions. This somehow indicates that none of the methods imitates how the human vision system groups textures. To confirm this, we asked 11 subjects to pick up the 4 most similar textures for each one of our test textures. Their choices are shown in table 4.23. The consensus choice (i.e. the one which was picked by most subjects in each position) is shown in figure 4.6. We can see from these results that humans tend to pick up patterns not only on the basis of grey values, but also on the basis of semantics. For example, the four most similar textures picked up for texture $T41$, tend to be all lace patterns, independent of the tones of grey they are made up

Person	Texture T31				Texture T41				Texture T51			
	1	2	3	4	1	2	3	4	1	2	3	4
Alex	T31	T30	T23	T99	T41	T108	T43	T42	T51	T72	T76	T105
Ali	T31	T30	T99	T98	T41	T40	T42	T39	T51	T50	T72	T76
Bagher	T31	T23	T30	T54	T41	T40	T42	T39	T51	T50	T76	T46
Leila	T31	T30	T23	T99	T41	T39	T40	T42	T51	T76	T50	T72
Maria	T31	T30	T99	T62	T41	T42	T40	T39	T51	T72	T105	T50
Marvi	T31	T30	T23	T98	T41	T42	T39	T40	T51	T50	T105	T76
Matt	T31	T23	T30	T27	T41	T39	T42	T40	T51	T50	T72	T76
Mohammad	T31	T23	T30	T99	T41	T42	T40	T45	T51	T72	T50	T76
Naghi	T31	T23	T30	T61	T41	T42	T40	T39	T51	T50	T72	T76
Saied	T31	T30	T62	T99	T41	T39	T40	T42	T51	T11	T68	T72
Zeinab	T31	T23	T30	T27	T41	T40	T39	T42	T51	T50	T76	T79

Table 4.23: At the top row we have three examples of the texture database. Underneath each texture we show the code number of textures picked up by 11 persons in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.

from. The same goes for the other two textures. So, the question we ask next is: Is it possible to choose functionals for the trace transform which will pick up preferentially the same textures as the humans do?

To be able to achieve this, we must train the classifier to give more weight to the features which produce a trend in texture choice which agrees with human ranking of textures.

We arbitrarily divided the 112 textures into two sets, the first 56 in one set and the other 56 in another set. The question we are trying to answer is the following: Can we choose feature weights by using the first 56 textures such that the textures are ranked according to the perceptual group to which they belong, and then use the same features and the same weights for the remaining 56 textures never seen by the system before, to check whether they will be ranked in a perceptually consistent way?

First we asked 5 individuals to group the 56 training textures in perceptual groups. The perceptual groups identified by consensus are given in table 4.24. We asked the same individuals to group also the 56 test textures. The consensus perceptual grouping of these textures is shown in table 4.25.

Classes	Texture Number					Classes	Texture Number				
1	T1	T26	T26	T6	T14	T12	T21	T16	T53	T49	
2	T2	T28	T5	T7		T13	T23	T27	T30	T31	T54
3	T3	T22	T35	T36		T14	T32	T33			
4	T4	T9	T24	T29		T15	T34				
5	T10					T16	T37				
6	T11	T17				T17	T38				
7	T12	T13				T18	T39	T40	T41	T42	
8	T15					T19	T43	T44	T45		
9	T18	T56				T20	T46	T47			
10	T19	T55				T21	T48	T8			
11	T20	T52				T22	T50	T51			

Table 4.24: Here we show the first 56 (training) textures grouped in perceptual classes.

Classes	Texture Number							Classes	Texture Number		
1	T57	T92	T80	T84	T81	T100		T14	T89		
2	T58	T62	T60	T61	T59			T15	T90		
3	T63	T73	T71					T16	T91		
4	T64	T65						T17	T93		
5	T66	T67						T18	T94	T95	T96
6	T68	T69	T70	T72	T76	T105	T106	T19	T97		
7	T74	T75						T20	T98	T99	
8	T77	T78	T79					T21	T101	T102	
9	T82	T85						T22	T103	T104	
10	T83							T23	T107	T108	
11	T86							T24	T109	T110	
12	T87							T25	T111	T112	
13	T88										

Table 4.25: Here we show the second 56 (testing) textures grouped in perceptual classes.

Each texture is represented by 4 images (the 4 quadrants of the original image) and therefore each perceptual group is represented by $4N_c$ images, where N_c is the number of textures in perceptual class c.

To use the perceptual grouping in defining the different weights for different features, we followed these steps:

- Instead of using one set of weights for our features like we did in section 4.2, we define here 22 sets of weights, one for each perceptual class.
- We obtain the distance of two textures by using a new metric (given by equation 4.10 below). With the 22 sets of weights we compute 22 different distances between any two textures.
- We select the minimum value of these distances between a pair of textures. For example, for finding the distance value between texture T3 and T6, we compute 22 different distances between them by using all 22 sets of weights defined from the perceptual classes, and then we get the minimum of these 22 values to indicate the distance between texture T3 and T6. We repeat this for all texture pairs we have.

In particular: Let us consider a feature f_{klm} computed by combining the k^{th} trace functional, with the l^{th} diametrical functional and the m^{th} circus functional. We follow the following procedure using the training set of images: Let us say that the value of this feature for texture t is denoted by f^t_{klm}. We can say that: the average value of this feature over all textures in the training set is:

$$m_{klm} = \frac{1}{N}\sum_{t=1}^{N} f^t_{klm} \tag{4.6}$$

where here N is the number of images in the training set, which is $56 \times 4 = 224$. The standard deviation of this feature is:

$$\sigma_{klm} \equiv \sqrt{\frac{1}{N}\sum_{t=1}^{N}(f^t_{klm} - m_{klm})^2} \tag{4.7}$$

We define:

$$q^c_{klm} = \frac{1}{\sigma_{klm}}\sqrt{\frac{1}{4N_c(4N_c-1)}\sum_{t_1=1}^{4N_c}\sum_{t_2=1}^{4N_c}(f^{ct_1}_{klm} - f^{ct_2}_{klm})^2} \tag{4.8}$$

where N_c is the number of textures in class c. In other words, we compute the average square distance between any two images which are assumed to belong to the same perceptual class. Because we must not try to compare one image with itself, the number of all possible pairwise combinations we may consider is $4N_c(4N_c - 1)$. The

smaller the q^c_{klm}, the more stable f_{klm} is, and we must give it bigger weight. So we define:

$$w^c_{klm} \equiv \begin{cases} Q - q^c_{klm} & \text{if } q^c_{klm} < Q, \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

where Q is a threshold which in most of cases is between 1 and 2.

Then, when we want to group together into perceptual classes an unknown set of textures, we use the following metric to compute the distance between any two textures.

$$dist_c(t_1, t_2) \equiv \frac{1}{N_f} \sum_{klm} w^c_{klm} \frac{1}{\sigma_{klm}} |f^{t_1}_{klm} - f^{t_2}_{klm}| \tag{4.10}$$

where in the above formula the value of index c changes from 0 to the maximum number of perceptual classes which in this case is 22. N_f is the number of features with weight $w^c_{klm} \neq 0$. It is very important to use such a normalisation, because features with zero weight are not stable and they should be entirely ignored. Such a normalisation was not necessary for the experiments reported in section 4.2, because we used there only one set of weights and therefore the number of features with non-zero weight was a constant. Division by a constant does not affect the ranking of distances.

Then we take the minimum over these distances to indicate the distance between textures t_1 and t_2:

$$dist(t_1, t_2) \equiv \min_c(dist_c(t_1, t_2)) \tag{4.11}$$

Then we rank the textures in terms of similarity with respect to any texture in the set. In other words, for any texture in the set we can choose which texture is the nearest one, which is the second most nearest one and so on. Tables 4.26 and 4.27 show the results of using this method on both the training and testing set. In the case when we have only one texture in the perceptual group, this approach is not meaningful. In the training set we have 5 textures without any other similar texture. In addition, one texture should not be compared with its 3 other versions (produced from the other 3 quadrants of the original image) because we wish here to test perceptual similarity rather than exact texture identification. This means that we have only $(56 - 5) \times 4 - 3 = 201$ textures to compare a texture with in the training set, and $(56 - 9) \times 4 - 3 = 185$ textures in the test set. The classification results of the training set serve as the benchmark. We do

Method	Training Set							
	N=1	N=2	N=3	N=4	N=5	N=6	N=7	N=8
TT	**109**	**125**	**135**	**146**	**152**	**157**	**160**	**162**
FCM	68	90	96	100	107	109	112	114
CM16	63	73	85	91	96	101	104	108
CM16+F	66	77	90	94	104	109	113	119
CM256	68	83	89	94	106	110	113	119
CM256+F	68	83	89	94	106	111	114	121
FSDH	45	62	73	80	85	95	101	104
SDH16	88	105	117	122	127	132	138	140
SDH16+F	87	103	113	123	128	133	140	142
SDH256	80	94	110	114	121	125	130	135
SDH256+F	78	90	102	110	119	125	131	136

Table 4.26: The results of the various methods, when we use the perceptual classes in table 4.24 to train the system and then classify the first 56 textures, which appear in the same table. In column $N = 1$ to $N = 8$ we show the total number of correct classification until the N^{th} choice. In all cases TT has the best performance and CM256 has the worst one. All numbers are out of 201.

not expect to be able to do better than that with the test set. Further, if these results are bad, we must conclude that it is not possible to train the classifier to imitate human perception.

The results on the test set tell us how stable and useful the identified features are, i.e. how generalisable they are, as opposed to giving us good results by chance. As we can see the TT has the best performance especially for the test set.

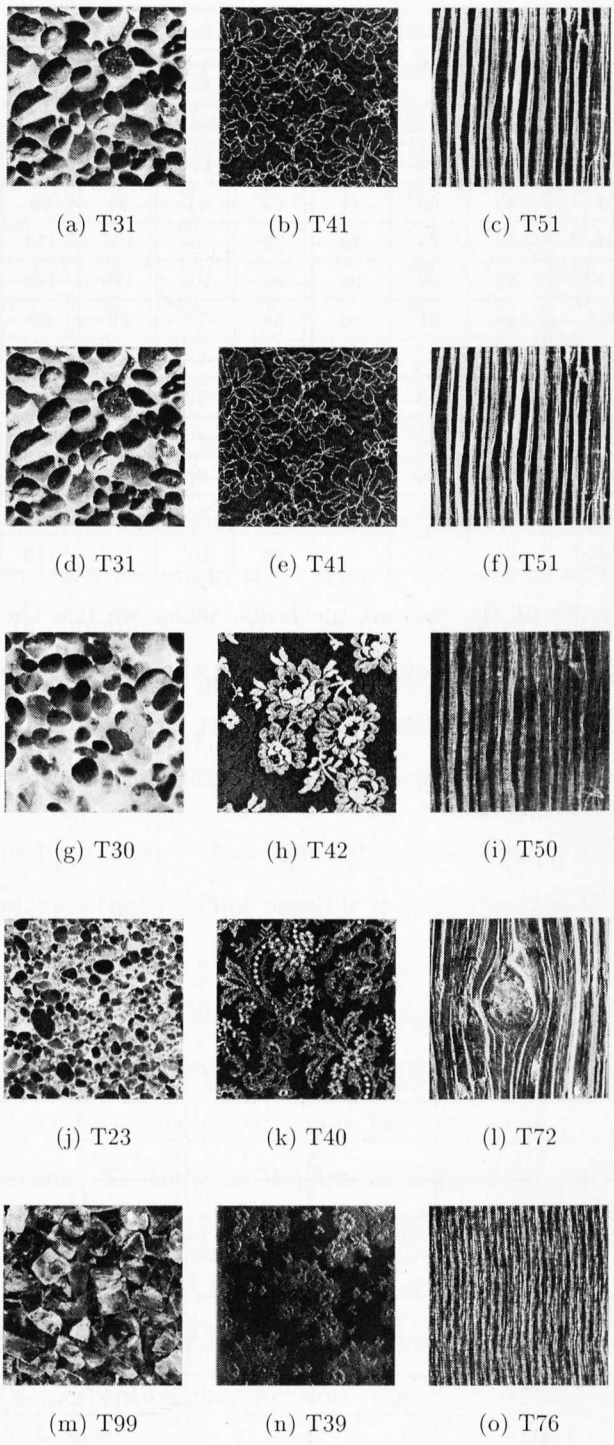
For each feature we have 22 different weights as defined by equation 4.9. We sum up these weights, and so we come up with the single number per feature. We use these numbers to rank the features according to their significance in perception. Table 4.28 presents the combinations of functionals which produced the features with the highest weights for the perceptual classification. The codes of the individual functionals correspond to the codes used to identify them in tables 3.1-3.4.

Method	Testing Set							
	N=1	N=2	N=3	N=4	N=5	N=6	N=7	N=8
TT	**76**	**90**	**103**	**112**	**120**	**126**	**131**	**133**
FCM	49	60	71	82	91	95	99	105
CM16	52	73	89	99	103	107	113	116
CM16+F	57	80	95	99	105	110	113	114
CM256	43	51	56	64	77	80	89	92
CM256+F	44	51	57	65	78	81	89	91
FSDH	44	53	60	71	76	81	84	86
SDH16	44	60	76	83	89	102	105	111
SDH16+F	45	66	76	87	96	98	105	110
SDH256	57	74	82	89	95	105	110	116
SDH256+F	61	82	88	96	101	107	113	117

Table 4.27: The results of the various methods, when we use the perceptual classes in table 4.24 to train the system and then classify the second 56 textures. In column $N = 1$ to $N = 8$ we show the total number of correct classification until the N^{th} choice. In all cases TT has the best performance and CM256 has the worst one. All numbers are out of 185.

Trace Functionals	Diametric Functionals	Circus Functionals
6	1	1
6	2	1
6	2	17
6	5	1
14	10	13
23	6	13
24	2	1
25	5	1
31	2	13

Table 4.28: This table shows the best features in TT. These were identified by training the system using the perceptual classes showed in table 4.24.



(a) T31 (b) T41 (c) T51

(d) T31 (e) T41 (f) T51

(g) T30 (h) T42 (i) T50

(j) T23 (k) T40 (l) T72

(m) T99 (n) T39 (o) T76

Figure 4.6: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the majority of humans subjects in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.

4.5 Discussion and Conclusions

We used 12 methods for feature extraction in our experiments. The co-occurrence matrix, when constructed for 256 grey levels, is very time consuming. The huge size of this matrix is the reason. We have 112 matrices 256×256 and for four sets of images (TL, TR, BL and BR). This means that we need nearly 240 million bytes of memory and a huge number of operations. That is why we repeated our experiments with a reduced number of grey levels, i.e. with 4-bit resolution. Table 4.29 shows the size of memory needed and also the number of operations which must be done for each method. It is obvious that CM256+F and CM256 are the worst methods regarding their need of memory and CPU operations. Figure 4.7 presents the summary results.

We can see that by increasing the number of features the results improve. This proves our first idea that using thousands of features, may help us classify textures more accurately. These features do not need to make sense to the human conscious perception, and therefore their number can be very large. The relevance of these features to the task we wish to solve can be assessed in a training phase, and then these features can be combined with their appropriate weights to form a similarity measure between two images.

It is believed that the incorporation of the thresholding operation in the feature weighting vector effectively constitutes a feature selection process thereby avoiding the potential problem of overfitting poor generalisation in the co-occurrence-based studies.

The Trace transform method does not have to be trained with representations of all textures we wish to identify. As we saw in the experiments performed with 30 training textures different from those in the test set, even texture classes that were not represented in the training set used to decide the relative importance of the features, could be classified correctly. The results were only slightly worse than the results obtained with using all 112 textures for training. For example, in the reported experiments, when all 112 textures were represented in the training set, the best recognition rate was about 85%. In the experiments with the limited training, the best recognition rate was about 80.5%.

The use of perceptually meaningful features on their own did not produce good results.

The supplement of co-occurrence matrix features with perceptually meaningful features did not change the result, which was already very good when the full co-occurrence matrix was used.

So, our first conclusion is that one does not need perceptually meaningful features (i.e. features which have linguistic names) in order to classify textures. Simply, one needs *many* features.

If then one wishes to classify textures in a perceptually meaningful way, i.e. in a way that imitates the human ranking of textures in terms of similarity, then one may select from these many features those which rank the textures as the humans do. We demonstrated that for this task features computed from the Trace transform are the most appropriate (see figures 4.8 and 4.9). Identifying which features were the most useful in this task allows us to reverse engineer to some extent the human vision system and single out some features which may be used in the *subconscious* level to analyse textures. These are features which do not have linguistic terms to describe them. It is interesting to note that from the trace functionals, the functional which contributed to the construction of the most useful features is a simple differentiator, and from the diametric functionals, the functional which contributed to the construction of the most useful feature is taking the minimum. From the circus functionals the most useful functional is functional 1 which again is based on a simple differentiator.

Method of Classification (for Textures)	Storages Needed (Mega Bytes)	Operations Needed (Mega Operations)
TT	20	71
BTT	14.6	52
FCM	0.018	0.063
CM16	0.89	3.13
CM16+F	0.91	3.2
CM256	240	836
CM256+F	240	836
FSDH	0.018	0.063
SDH16	0.096	0.34
SDH16+F	0.114	0.4
SDH256	1.78	6.43
SDH256+F	1.8	6.5

Table 4.29: This table shows for each of the 12 methods we discussed here, the volume of storage it needs and the number of operations it performs. The storage value is in Mega bytes and the operation number is in Mega operations.

Method	Dist	Q	Test Set BL					Test Set BR				
			1	2	3	4	R	1	2	3	4	R
TT		1.3	85	4.9	0	0.8	8.2	73	9.1	2.4	3.3	9.9
BTT		1.0	80	8.6	2.4	0	8.6	62	11	3.7	6.1	17.2
FCM	2	0.8	59	10.7	6.6	2.4	18.2	36	11.5	8.2	7.4	31.4
CM16	1	0.5	89	4.9	2.4	0	2.4	74	9.1	4.1	1.6	9.1
CM16+F	2	0.5	90	6.6	0	0	2.4	72	7.4	8.2	1.6	8.2
CM256	1	1	97	1.6	0	0	0.8	94	3.3	1.6	0	0.8
CM256+F	1	1	97	1.6	0	0	0.8	94	3.3	1.6	0	0.8
FSDH	2	2.5	62	10.7	4.9	6.6	12.4	39	11.5	7.4	9.9	27.5
SDH16	4	1	92	3.3	1.6	0.8	1.6	64	12.4	4.9	6.6	9.1
SDH16+F	4	2	91	3.3	3.3	0	1.6	64	13.2	3.3	4.1	12.4
SDH256	1	0.5	87	6.6	2.4	0	2.4	69	14	4.9	2.4	7.4
SDH256+F	1	0.5	86	7.4	2.4	0	2.4	70	13.2	3.3	1.6	9.1

Table 4.30: This table shows the best results for the 12 methods we used. The name of each method and the value of its parameters are written in each row. *Dist* is the distance value for the co-occurrence matrix and Q is the value of the threshold used. All numbers are in percentages of test images classified correctly in the 1^{st}, 2^{nd}, 3^{rd}, 4^{th} or remaining positions.

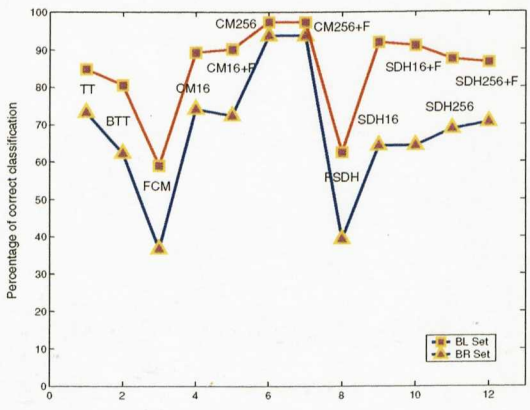


Figure 4.7: The best results for the 12 mentioned methods are shown here. All results are converted to percentages. The top line is for set BL and the bottom line is for set BR. The name of each method is written either next to the top or the bottom line.

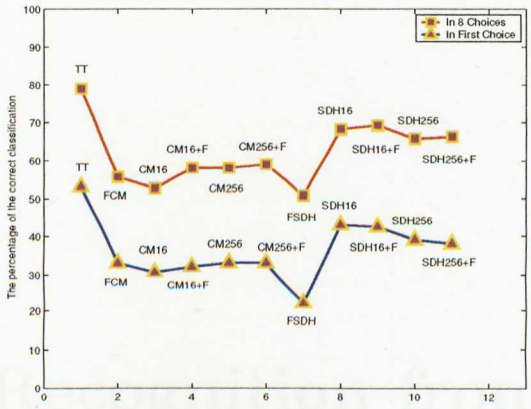


Figure 4.8: The best results for classification in perceptual classes for the 12 mentioned methods (Except BTT which is the same as TT in this experiment) are shown here. The top line is for the percentage of correct classification after 8 choices and the bottom line is for correct classification as the first choice. As we can see the TT method has the best performance with a large gap with the second method.

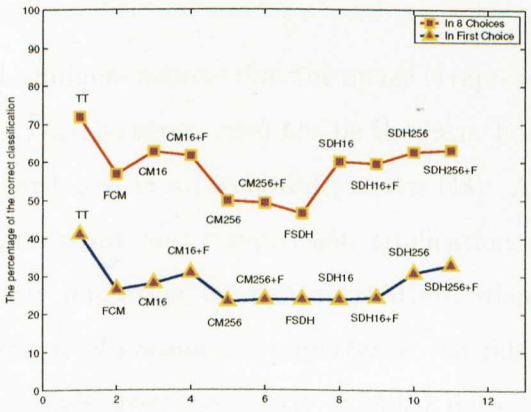


Figure 4.9: The best results for classification in perceptual classes for the 12 mentioned methods (Except BTT which is the same as TT in this experiment) are shown here. The top line is for the percentage of correct classification after 8 choices and the bottom line is for correct classification as the first choice. As we can see the TT method has the best performance with a large gap with the second method.

Chapter 5

Texture Recognition from Sparsely and Irregularly Sampled Data

5.1 Introduction

Most image processing techniques assume that the image is represented by a rectangular grid of sampling points. This, however, need not be the case. For example, the human eye has its receptors distributed in an irregular pattern [48]. Also in remote sensing [16] and other image processing and compression applications [86]. The regularity of sampling is particularly important for texture analysis, where the relative spatial arrangement of the pixels is of paramount importance. In this paper we investigate a way of recognising textures from irregularly sampled data. For this purpose we use the Trace transform [36]. The Trace transform operates along straight lines criss-crossing the image. There is some physiological evidence to indicate that the human eye performs a similar scanning, known as the "tremor" of the eye [15], being performed with 40-120 cycles per second and amplitude of about 1 degree. Over such short periods of time, the receptors in the human eye scan along straight lines. If the tremor is suppressed, the person cannot see [89]. The eye, therefore, looks like a special device that may perform a Trace transform of the viewed scene, in the fovea area. The

importance of the tremor of the eye has not been understood yet in image processing terms, as there is no adequate theory utilising the information obtained by the tremor. Trace transform appears to offer a possible mathematical model for this mechanism. To be able to apply the Trace transform to irregularly sampled data, we need an intermediate step which will lead from the irregularly distributed sample points to straight lines. Such a mechanism is provided by the Hough transform. To demonstrate our ideas, we first create an irregular pattern of sampling points using three methods of distribution for random points:

- Gaussian Distribution
- Log-polar model
- Retinal Distribution

This way we can represent our images by a collection of sample points which are densely distributed towards the centre, and sparsely distributed in the periphery. Then we use the Hough transform to identify sets of pixels that constitute straight lines. Each identified line can be used as a tracing line in the Trace transform. So we can classify these sets of data after extracting features by the Trace transform. The structure of this chapter is as follows: Section 2 explains the Hough transform method which will be used in finding the lines in a set of irregularly sampled points. In section 3 we deal with the problem of computing functionals along irregularly sampled lines. In section 4 we explain how we can make irregular sampling patterns and in the following section we compare the results of using three different sampling masks with the results obtained by using regularly sampled data in the recognition of textures. Finally in section 6 we discuss the results and present our conclusions.

5.2 The Hough transform method

The equation of a line is (See figure 5.2):

$$y = ax + b \tag{5.1}$$

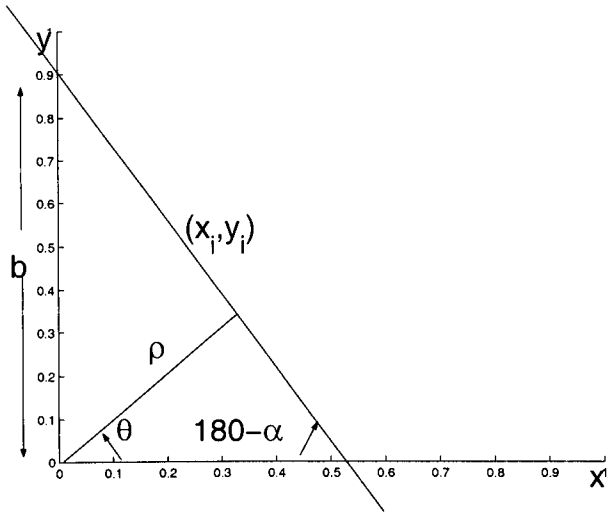


Figure 5.1: We can represent a line by the distance from the centre of the coordinate system to the line, ρ, and the angle of the normal to the line with the x axis.

where:

$a = \tan(\alpha)$

$b = y$ of the point with $x = 0$

On the other hand we can fully define a line by parameters ρ and θ, where:

$\rho =$ The distance from the centre of the coordinate system to the line.

$\theta =$ The angle of the normal to the line from the center of the axes with the x axis.

We can write:

$\cos(90 - \theta) = \frac{\rho}{b} \Rightarrow b = \frac{\rho}{\cos(90-\theta)} = \frac{\rho}{\sin(\theta)}$

And also:

$a = -\tan(90 - \theta) = -\frac{\cos(\theta)}{\sin(\theta)}$

By using (5.1) we can write:

$y = -\frac{\cos(\theta)}{\sin(\theta)}x + \frac{\rho}{\sin(\theta)}$

This yields:

$$\rho = y\sin(\theta) + x\cos(\theta) \tag{5.2}$$

This equation is true for all points in the line, so all possible lines passing through point (x_i, y_i) can be represented by:

$$\rho = y_i\sin(\theta) + x_i\cos(\theta) \tag{5.3}$$

Consider a set of points in one line. If we plot for each one of them the curve defined by

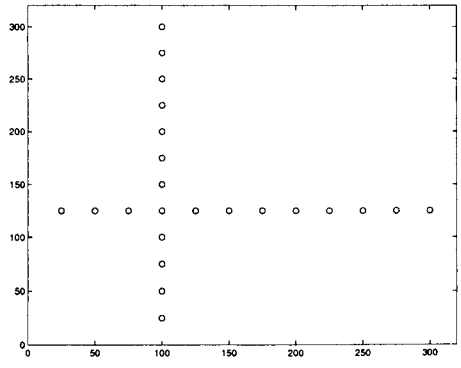


Figure 5.2: This image has 10 distinct points in vertical line and 10 distinct points in horizontal line.

the above formula in axes (ρ, θ), it is obvious that they will all pass through a common point. This common point indicates the value of ρ and θ for the line passing through all those points. The example that follows makes this clearer.

Consider an image with 10 distinct points along a vertical line and 10 distinct points along a horizontal line as shown in figure 5.2. Figure 5.3 shows the Hough space of this image. As we see we plot for each point (x_i, y_i), one curve in (ρ, θ) space given by equation (5.3). Because we had 20 points, so we have here 20 curves. These 20 curves cross each other in two points: The first one has $\rho = 125, \theta = 0$ and the second one has $\rho = 100, \theta = 90$. In this way we can find all lines which can be formed from the points in the image. We can apply this method to the random sampling pattern to identify straight lines made up by the sampling points. We can then use these lines as our tracing lines and apply the algorithm of texture classification based on the Trace transform.

5.3 Computing the functionals for irregularly sampled data points

The tracing lines identified by the Hough transform consist of samples which are irregularly placed along the line. It is obvious, therefore, that the tracing functionals which

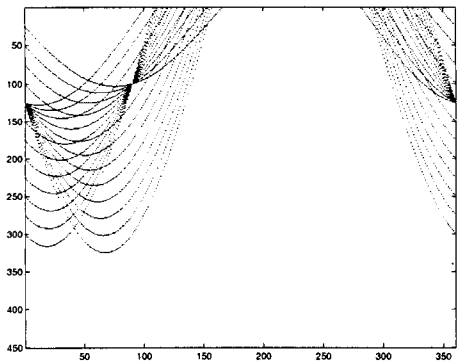


Figure 5.3: These 20 curves cross each other in two points: The first one has $\rho = 125, \theta = 0$ and the second one has $\rho = 100, \theta = 90$. These two points define the two lines in figure 5.2

require, for example, some form of differentiation, can not be computed in a straightforward way. These functionals are entries 6-31 from table 3.1. In addition functionals 1-4 which perform some sort of integration have also to be computed by taking into consideration the irregular spacing of the samples. We discuss in this section first how to insert missing points in the collection of the irregularly sampled points and then the methodologies we use in order to compute the functionals we need along the tracing lines with the missing points.

5.3.1 Identifying the missing points along the tracing lines

By using the Hough transform, we obtain some points which are approximately in one line. We know the coordinates of all these points in the image. Now we want to build the line joining all these points including missing points between some of them. First of all we can find the equation of this line by having ρ and θ:

$$\rho = y\cos\theta + x\sin\theta \Rightarrow \tag{5.4}$$

$$y = -\frac{\sin\theta}{\cos\theta}x + \frac{\rho}{\cos\theta} \tag{5.5}$$

By using this equation, we can add all missing point between the irregularly sampled points. Suppose $x_n(i_n, j_n)$ and $x_{n+1}(i_{n+1}, j_{n+1})$ are two successive points along a line

identified by the Hough transform. We want to insert the missing points between these points as if the line had been regularly sampled. To do that we must find d_{max}:

$$d_{max} = \max\left(|y_{n+1} - y_n|, |x_{n+1} - x_n|\right) - 1 \tag{5.6}$$

Then we add between these two points d_{max} zeros, to indicate the missing points. For example suppose the equation of the line is:

$$y = 2x + 1 \tag{5.7}$$

and the sampled points are:

$$X = [x_1(5, 11), x_2(8, 17), x_3(9, 19), x_4(11, 23)] \tag{5.8}$$

We must insert $17 - 11 - 1 = 5$ points between x_1 and x_2, $19 - 17 - 1 = 1$ points between x_2 and x_3 and $23 - 19 - 1 = 3$ points between x_3 and x_4. The final line is:

$$L = [x_1, 0, 0, 0, 0, 0, x_2, 0, x_3, 0, 0, 0, x_4] \tag{5.9}$$

5.3.2 Computing the functionals with missing data

In order to compute the functionals we need we must take into consideration the spacing of the points that constitute each tracing line. A method that does this is the so called normalised convolution proposed by Knutsson and Westin [41]. Normalised convolution can be used to estimate the value of a signal or any of its derivatives at regularly placed points from its irregularly sampled values. In the subsections that follow we shall examine if and how normalised convolution can help estimate the functionals we need from the samples that make up the digital lines identified by the Hough transform. First, however, we shall give a brief introduction to normalised convolution and its derivatives.

5.3.2.1 Normalised convolution

Let us consider an example of a line consisting of 7 regular samples, 3 of which are missing:

$$f(t) \equiv [x_1, x_2, 0, 0, x_5, 0, x_6, x_7] \tag{5.10}$$

where 0 indicates a missing value. Also consider a simple smoothing filter:

$$g(t) = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right] \tag{5.11}$$

If we convolve these two functions we can see that the gaps of the missing samples will be filled with the available information:

$$f(t) * g(t) = \left[\frac{x_1 + x_2}{3}, \frac{x_1 + x_2}{3}, \frac{x_2}{3}, \frac{x_5}{3}, \frac{x_5}{3}, \frac{x_5 + x_6}{3}, \frac{x_6 + x_7}{3}, \frac{x_6 + x_7}{3}\right] \tag{5.12}$$

We can define a sequence $c(t)$ with dimensions equal to the dimensions of the signal, such that it indicates which sample is missing and which sample is available. This can be done by putting zero for the missing points and one for the available ones. For signal $f(t)$ we can write:

$$c(t) = [1, 1, 0, 0, 1, 0, 1, 1] \tag{5.13}$$

and:

$$c(t) * g(t) = \left[\frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3}\right] \tag{5.14}$$

We call $c(t)$ the certainty sequence as it is nothing more than the sequence of locations where samples are to be found. The Normalised Convolution of f with g is defined by:

$$\hat{f}(t) \equiv \frac{f(t) * g(t)}{c(t) * g(t)} \tag{5.15}$$

where $\hat{f}(t)$ means the reconstruction of $f(t)$. $\hat{f}(t)$ in the case of this example is:

$$\hat{f}(t) = \left[\frac{x_1 + x_2}{2}, \frac{x_1 + x_2}{2}, x_2, x_5, x_5, \frac{x_5 + x_6}{2}, \frac{x_6 + x_7}{2}, \frac{x_6 + x_7}{3}\right] \tag{5.16}$$

It is possible to say that the second convolution is a weight for the first one and that is why we divide the first by the second convolution point by point. So for performing normalised convolution we must follow these steps:

- Convolve the signal with the filter, putting 0 at the positions of missed samples.
- Define $c(t)$, the certainty sequence of the signal, consisting of 1s at the positions where we have data and 0s at the positions of missing data.
- Convolve the certainty sequence with the filter.

- Divide these two convolutions point by point.

The result is a reconstruction of the signal with values assigned at the positions of the missing data. In these experiments we use as our smoothing filter the filter suggested by Knutsson and Westin [41] in their original paper, defined by:

$$g(r) = \begin{cases} r^{-\alpha} \cos^{\beta}(\frac{\pi r}{2r_{max}}) & \text{if } r < r_{max}, \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

where:

r denotes the distance from the neighbourhood center, and

α and β are positive integers.

We use the reconstruction of the signal by filter (5.17) along each line before we compute functionals $1-5$ of table 3.1. For the other functionals which involve differentiation, we use the corresponding smoothing filter to that used for the differentiation.

5.3.2.2 Estimating the derivative of a signal with missing data

In order to compute the derivative of a signal with missing data, we may differentiate its normalised convolution:

$$D\left(\frac{C}{NC}\right) = \frac{D(C) \times NC - D(NC) \times C}{NC^2} \tag{5.18}$$

where

$$C \equiv f * g_2 \tag{5.19}$$

$$NC \equiv c * g_2 \tag{5.20}$$

f is the signal, g_2 is a smoothing filter and c is the certainty sequence of the signal. We may write:

$$D(C) = D(f * g_2) = f * D(g_2) = f * g_1 \tag{5.21}$$

where $g_1 = D(g_2)$ and:

$$D(NC) = D(c * g_2) = c * D(g_2) = c * g_1 \tag{5.22}$$

So we have:

$$D\left(\frac{C}{NC}\right) = \frac{f * g_1 \times NC - c * g_1 \times C}{NC^2} \tag{5.23}$$

5.3.3 Using normalised convolution to estimate the values of the functionals

In this section we shall examine whether normalised convolution may help us estimate ([42, 3, 88]) the functionals of tables 3.1 to 3.4 when we can use only irregularly placed samples along the digital lines more accurately. For this purpose, we shall divide the functionals in two categories, namely those that do not involve convolution of the data and those that involve some sort of convolution. In all cases we shall compare the estimated values of the functionals with or without normalised convolution with the values obtained from the corresponding digital lines made up from all the samples in the original images represented by rectangular grids. For this study we shall use 50 indicative tracing lines, each of which will contain at least 200 points when irregularly sub-sampled.

5.3.3.1 The functionals which do not involve convolution

By inspecting tables 3.1 to 3.4 we can see that functionals 1 to 5, and 22 and 23 from table 3.1, functionals 1 to 7 from table 3.3 and functionals 3 to 18 from table 3.4 are functionals which can not be expressed as convolutions. For this set of functionals we first reconstruct the signal using normalised convolution with the filter defined in (5.17) and then use the functionals. We shall demonstrate the process from the beginning to the end using an example, and we shall investigate whether this method improves the result obtained if the irregular spacing of the data is ignored.

Let us consider a sequence of values from which these functionals have to be computed:

$$f(t) = [75, 148, 148, 148, 179, 135, 135, 135, 148, 133, 148, 203] \tag{5.24}$$

where:

$f(t)$ is one of our original tracing lines with regularly spaced samples. Consider a sub-sampled version of it where 0 stands for a missing sample:

$$f'(t) = [75, 148, 0, 0, 179, 0, 135, 0, 0, 133, 148, 0] \tag{5.25}$$

where:

$f'(t)$ is the sequence of the irregularly sampled data. $c(t)$, the certainty sequence, is:

$$c(t) = [1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0] \tag{5.26}$$

The filter defined in (5.17) with $\alpha = 0$, $\beta = 2$ and $r_{max} = 8$ is:

$$g(t) = [0.05, 0.35, 0.8, 1, 0.8, 0.35, 0.05] \tag{5.27}$$

We may divide it by the sum of its elements to normalise it. Now we shall compute the result of applying each functional on both the original and the irregularly sampled sequence:

$T_1 = \sum_{i=1}^{N} x_i$ (Figure B.1)

For this functional we have these results:

$T_1(f) = 5205$

$T_1(f') = 2454$

$T_1(\hat{f}) = 4581$ where $\hat{f}$ is the reconstructed signal by normalised convolution.

We can see that the result of the reconstructed signal is much closer to the original than the result of the irregularly sampled signal. To verify this, we repeat this calculation for 50 lines taken from the original set of tracing lines. The result is shown in figure 5.4. The result shows a very clear improvement when normalised convolution is used. In what follows we perform similar calculation for all other functionals in this set (All referenced figures are in appendix B.). Figures B.2 to B.4 show the corresponding results for functionals T_2, T_3 and T_4 respectively.

When the functional uses the Max or Min operator, signal reconstruction does not help. This is because of the sensitivity of these operators to individual numbers. For all these functionals, therefore, it is better to deal with the irregularly sampled data instead of trying to reconstruct them (and thus smooth them).

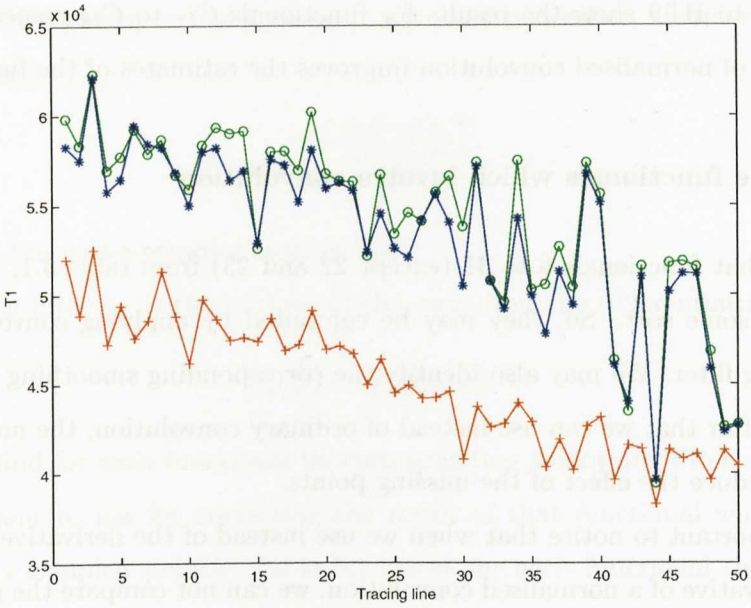


Figure 5.4: The values of trace functional T_1 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

Figure B.5 shows the result for functional T_5. Figures B.22 and B.23 show the results for functionals T_{22} and T_{23} respectively. For these functionals we obtain approximately the same results in all three cases.

Figures B.32 and B.33 show the results for functionals D_1 and D_2 of table 3.3, respectively. These functionals involve operators sensitive to smoothing effects, so it is better to use for their calculation the raw data. Figure B.34 shows the results for functional D_3 which indicate that the use of normalised convolution is recommended. The results for functional D_4 are shown in figure B.35 and they indicate indifference to whether we use regularly or irregularly sampled data. Figures B.36 to B.38 and B.44 and B.45 show the results for functionals D_5, D_6 and D_7 from table 3.3 and C_3 and C_4 from table 3.4 respectively. In all cases the use of normalised convolution is recommended.

Figures B.46 to B.51 show the results for functionals C_5 to C_{10} respectively. In all cases the Min or Max operators are involved and one should clearly use the raw data to estimate the values of the functionals.

Figures B.52 to B.59 show the results for functionals C_{11} to C_{18} respectively. In all cases the use of normalised convolution improves the estimates of the functionals.

5.3.3.2 The functionals which involve convolution

We can see that functionals 6 to 31 (except 22 and 23) from table 3.1, involve differentiations of some sort. So, they may be computed by applying convolution with a differentiating filter. We may also identify the corresponding smoothing filter for each functional. After that we can use instead of ordinary convolution, the normalised convolution to reduce the effect of the missing points.

It is very important to notice that when we use instead of the derivative of a convolution, the derivative of a normalised convolution, we can not compare the result directly with the one obtained from the derivative of convolution of regularly sampled points. In mathematical terms, let us say that $T_i(t)$ is one of the trace functionals:

$$T_i(t) = \frac{d\left(f(t) * g_{2_i}(t)\right)}{dt} \tag{5.28}$$

where as earlier $f(t)$ is a tracing line and $g_{2_i}(t)$ is the smoothing filter which corresponds to that functional. We can divide and multiply inside the derivative operator by $c(t) * g_{2_i}(t)$:

$$T_i(t) = \frac{d\left(\frac{f(t)*g_{2_i}(t)}{c(t)*g_{2_i}(t)} \times c(t) * g_2(t)\right)}{dt} \tag{5.29}$$

We know that all elements of $c(t)$ in regularly sampled points are 1. So if we use wrap round boundary conditions for the convolutions, we have:

$$c(t) * g_{2_i}(t) = \left[\sum g_{2_i}(t), \sum g_{2_i}(t), ..., \sum g_{2_i}(t)\right] \tag{5.30}$$

This means that:

$$T_i(t) = \frac{d\left(\frac{f(t)*g_{2_i}(t)}{c(t)*g_{2_i}(t)}\right)}{dt} \times C_{f_i} \tag{5.31}$$

where C_{f_i} is the coefficient of the smoothing filter for the i^{th} functional which is equal to:

$$C_{f_i} = \sum_t g_{2_i}(t) \tag{5.32}$$

When we use irregularly sampled points, we must remember, therefore, to use the normalising factor C_{f_i}:

$$\hat{T}_i(t) = \frac{d\left(\frac{f'(t)*g_{2_i}(t)}{c(t)*g_{2_i}(t)}\right)}{dt} \times C_{f_i} \tag{5.33}$$

where $f'(t)$ is the under-sampled tracing line.

The smoothing filter g_{2_i} is the integral of the corresponding differentiating filter g_{1_i}

$$g_{2_i}(t) = \int g_{1_i}(t)dx \tag{5.34}$$

Now we can find for each functional its corresponding smoothing filter and its appropriate coefficient to use for correcting the result of that functional when applied to the irregularly sampled points. Table 5.1 shows for each functional the corresponding smoothing filter. We describe, as an example, one of them and the others can be obtained in exactly the same way. Let us consider functional T_9 of table 3.1:

$$T_9 = \sum_{i=3}^{N-2} |x_{i-2} + x_{i-1} - x_{i+1} - x_{i+2}| \tag{5.35}$$

This can be written as taking the absolute value of the convolution of a signal by filter:

$$g_1(t) = [1, 1, 0, -1, -1] \tag{5.36}$$

To compute $g_2(t)$ we must integrate $g_1(t)$. We use numeric integration as follows:

$$g_2(1) = 0 + g_1(1) = 0 + 1 = 1 \tag{5.37}$$

$$g_2(2) = g_2(1) + g_1(2) = 1 + 1 = 2 \tag{5.38}$$

$$g_2(3) = g_2(2) + g_1(3) = 2 + 0 = 2 \tag{5.39}$$

$$g_2(4) = g_2(3) + g_1(4) = 2 + (-1) = 1 \tag{5.40}$$

$$g_2(5) = g_2(4) + g_1(5) = 1 + (-1) = 0 \tag{5.41}$$

so:

$$g_2(t) = [1, 2, 2, 1] \tag{5.42}$$

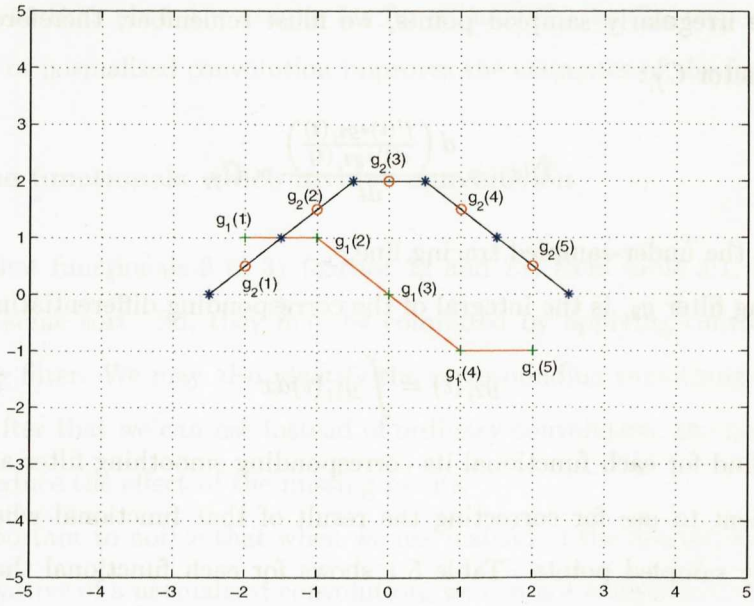


Figure 5.5: How can we obtain g_2 from g_1? In this figure we can see that by numerical integration of g_1 we can build filter g_2.

If we accept that every sample of filter $g_1(t)$ is the mean value of the filter in the range $[t-\frac{1}{2}, t+\frac{1}{2}]$, the integral $g_2(t)$ represents the integral up to point $t+\frac{1}{2}$. So, the values of $g_2(t)$, computed above, correspond to the inter-sample positions of filter $g_1(t)$ indicated by a star ($\star$) in figure 5.5. From this figure we can see that the smoothing filter we should use for convolution is:

$$g_2(t) = [0.5, 1.5, 2, 1.5, 0.5] \tag{5.43}$$

All other filters in table 5.1 can be obtained in the same way. In appendix B, figures B.6 to B.13 show the results of computing functionals T_6 to T_{13} of table 3.1 for 50 regularly sampled tracing lines, sub-sampled versions of them, while ignoring the irregular spacing of the data, and sub-sampled versions of them while taking into consideration the irregular placing of the data and using formula (5.23) with the corrective factor as shown by equation (5.33).

Functionals T_{14} to T_{21} from table 3.1, involve the sum of some convolutions. We discuss here in detail functional T_{14} while similar analysis may be performed for the

other functionals in this category:

$$T_{14} = \sum_{i=5}^{N-4} \sum_{k=1}^{4} |x_{i-k} - x_{i+k}| \tag{5.44}$$

We may exchange the order of the two summations:

$$T_{14} = \sum_{k=1}^{4} \sum_{i=5}^{N-4} |x_{i-k} - x_{i+k}| \tag{5.45}$$

So:

$$T_{14} = Con_1 + Con_2 + Con_3 + Con_4 \tag{5.46}$$

where:

$$Con_k \equiv \sum_{i=5}^{N-4} |x_{i-k} - x_{i+k}| \tag{5.47}$$

The convolution filter used in Con_4 is:

$$g_1 = [1, 0, 0, 0, 0, 0, 0, 0, -1] \tag{5.48}$$

This means that all these functionals, from T_{14} to T_{21} involve the sum of convolutions with filters $[1, 0, -1]$ to $[1, zeros(2 \times k - 1), -1]$, where $zeros(2 \times k - 1)$ indicates that there are $(2 \times k - 1)$ zeros there and k is the upper bound of the second summation in each functional.

Apart from functionals T_{20} and T_{21}, the values of the others show improvement after using normalised convolution. We can see their results in appendix B, figures B.14 to B.21. The results for T_{20} and T_{21} are neither better nor worse when using normalised convolution, so we may use normalised convolution for these two functionals as well. The size of the convolution filter used in T_{20} and T_{21} is the reason of the lack of improvement in this case.

Functionals T_{24} and T_{28} involve the calculation of the second derivative of the data, while the third, fourth and fifth derivatives are computed for pairs (T_{25}, T_{29}), (T_{26}, T_{30}), (T_{27}, T_{31}) respectively. So for functional T_{24} and T_{28}, we must use $[0.5, 0.5]$ as g_2 and take the second derivative of the result by convolving it with [1,-2,1]. The result from the comparative study over the 50 lines are shown in figure B.24. For T_{25} and T_{29}, we

need to use the same g_2, but compute the third derivative afterwards, by convolving the results with [1,-3,3,-1]. In the same way, we use the same g_2 for T_{26} and T_{30}, and then compute the fourth derivative by convolving them with [1,-4,6,-4,1]. Finally, for T_{27} and T_{31}, we use again the same g_2, but estimate the fourth derivative by convolving the result with [1,-5,10,-10,5,-1]. The corresponding results are shown in figures B.24 to B.31. Figures B.39 to B.43 show the results for functionals D_8 to D_{10} from table 3.3, and C_1 and C_2 from table 3.4. All of them show that better estimates may be obtained when normalised convolution is used.

Table 5.1 shows both g_1 and g_2 for trace functionals 6 to 31 (except 22 and 23) in addition to their C_{f_i} and I, the number of times the functional filter has to be integrated to produce the smoothing filter g_2. Using these filters and their coefficients we can calculate the values of functionals from 6 to 31, for irregularly placed points along the tracing lines identified by the Hough transform, using normalised convolution. This method will be used in all of our experiments for irregularly sampled data.

5.4 Creating irregularly sampled images

We create irregular patterns of sampling points using three different methods of distribution for the random points:

- Gaussian Distribution
- Log-polar model
- Retinal Distribution

This way we may represent our images by a collection of sampling points which are densely distributed towards the centre, and sparsely distributed in the periphery. In the next three subsections we explain the methods of making these sampling masks.

Funct	Functional Filter $g_1(t)$	$g_2(t)$	C_{f_i}	I
T_6	[1,-1]	[0.5,0.5]	1	1
T_7	[1,-1]	[0.5,0.5]	1	1
T_8	[1,1,1,0,-1,-1,-1]	[0.5,1.5,2.5,3,2.5,1.5,0.5]	12	1
T_9	[1,1,0,-1,-1]	[0.5,1.5,2,1.5,0.5]	6	1
T_{10}	[1,1,1,1,0,-1,-1,-1,-1]	[0.5,1.5,2.5,3.5,4,3.5,2.5,1.5,0.5]	20	1
T_{11}	[1,1,1,1,1,0,-1,-1,-1,-1,-1]	[0.5,1.5,2.5,3.5,4.5,5,4.5,3.5,2.5,1.5,0.5]	30	1
T_{12}	[1,1,1,1,1,1,0,-1,-1,-1,-1,-1,-1]	[0.5,1.5,2.5,3.5,4.5,5.5,6,5.5,4.5,3.5,2.5,1.5,0.5]	42	1
T_{13}	[1,1,1,1,1,1,1,0,-1,-1,-1,-1,-1,-1,-1]	[0.5,1.5,2.5,3.5,4.5,5.5,6.5,7,6.5,5.5,4.5,3.5,2.5,1.5,0.5]	56	1
$T_{14}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{14}(last)$	[1,0,0,0,0,0,0,0,-1]	[0.5,1,1,1,1,1,1,1,0.5]	8	1
$T_{15}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{16}(first)$	[1.0,-1]	[0.5,1,0.5]	2	1
$T_{17}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{18}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{19}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{20}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{21}(first)$	[1,0,-1]	[0.5,1,0.5]	2	1
$T_{21}(last)$	[1,zeros(49),-1]	[0.5,ones(49),0.5]	50	1
T_{24}	[1,-2,1]	[0.5,0.5]	1	2
T_{25}	[1,-3,3,-1]	[0.5,0.5]	1	3
T_{26}	[1,-4,6,-4,1]	[0.5,0.5]	1	4
T_{27}	[1,-5,10,-10,5,-1]	[0.5,0.5]	1	5
T_{28}	[1,-2,1]	[0.5,0.5]	1	2
T_{29}	[1,-3,3,-1]	[0.5,0.5]	1	3
T_{30}	[1,-4,6,-4,1]	[0.5,0.5]	1	4
T_{31}	[1,-5,10,-10,5,-1]	[0.5,0.5]	1	5
D_8	[1,-1]	[0.5,0.5]	1	1
D_9	[1,-1]	[0.5,0.5]	1	1
D_{10}	[1,-4,6,-4,1]	[0.5,0.5]	1	4
C_1	[1,-1]	[0.5,0.5]	1	1
C_2	[1,-1]	[0.5,0.5]	1	1

Table 5.1: The list of filters which are used in the trace functionals and their corresponding smoothing filters. Zeros(n) means insert n zeros and ones(n) means insert n ones. C_{f_i} means the coefficient of the smoothing filter of the i^{th} functionals. For functionals 24 to 31 we have the same g_2, but with different I.

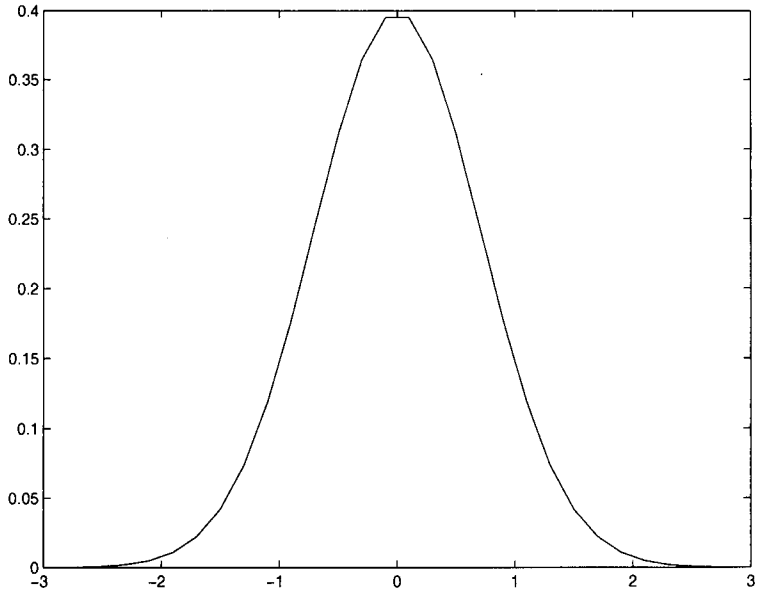


Figure 5.6: Gaussian distribution function with $mean = 0$ and standard deviation $\sigma = 1$.

5.4.1 The Gaussian sampling pattern

The first mask we want to use is based on the Gaussian distribution. Figure 5.6 shows a one dimensional Gaussian distribution with mean value $m = 0$ and standard deviation $\sigma = 1$. We know that:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-m)^2}{2\sigma^2}} \tag{5.49}$$

where $g(x)$ is the Gaussian probability density function, x is the independent variable and m is its average. In this distribution we know that the probability of finding a point between -3σ and 3σ is about 99%. We use this property to define a Gaussian distribution which produces numbers which cover all the image. The size of our image is 320×320, so if we use the point at the $(160, 160)$ as the image center, we must spread out the points along the two axes between -160 and $+160$. So σ must be:

$$\sigma = \frac{160}{3} = 53.3 \tag{5.50}$$

The mean value is still zero. To identify a random Gaussian sampling pattern we follow these steps:

- By using a uniform distribution, we draw a number between 0 and 1.

- By using a look up [60] table (table 5.2), we identify the corresponding number drawn from a Gaussian distribution.

- We multiply the value obtained with the standard deviation of the Gaussian distribution which we need (equation 5.50).

If the image is square, we draw twice as many random numbers as points we need and pair them to form the sampling patterns. If the image has different dimensions along the different axes, we must use a Gaussian with different σ to draw the random positions along the two axes.

Figure 5.7 shows such a distribution which covers a square image with 320 pixels along each side. Because we restrict ourselves to sampling at integer positions, a new generated point may have exactly the same coordinates as a point generated before. So, if we repeat the process N times it does not mean that we shall have N distinct sampling points. We actually have $M \leq N$ points. To reach a certain number of points for our mask, say $M = 20,000$ points, we need to repeat the process more than $20,000$ times. It is clear then that the final mask will not be exactly Gaussian. This is an inevitable necessity of using integer sampling positions. We could avoid the situation either by using much fewer points for the mask, or by allowing non-integer sampling positions. In the first case, we would severely under sample the image while in the second case, we would have to oversample the image in the central regions by using many close-by sampling points, the values of which would have been extracted from the same near-by integer positions, thus necessarily repeating the same information. For these reason, we decided to ignore the problem and accept that all sampling patterns we use will only be approximately what their name indicates and in practice they will be limited by the finite regular grid of the original images.

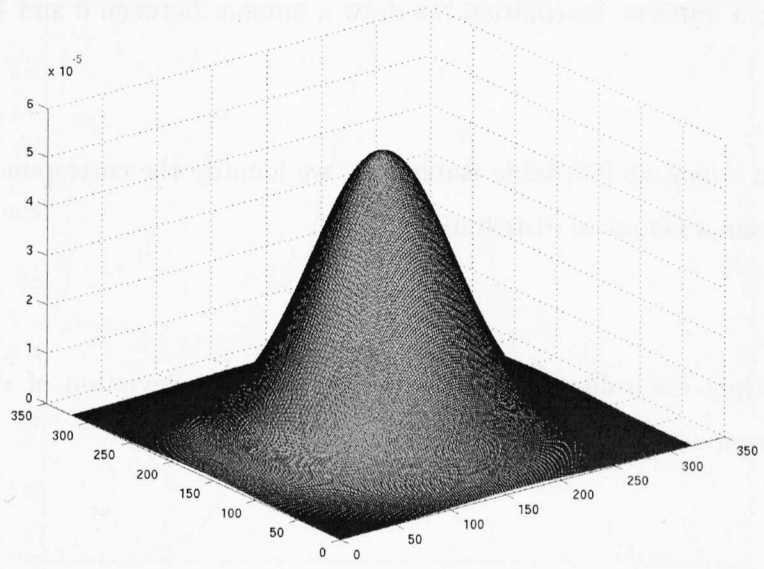


Figure 5.7: Gaussian distribution for two dimensional sampling.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0.500	0.519	0.559	0.598	0.636	0.673	0.708	0.741	0.772	0.801
1	0.828	0.852	0.874	0.893	0.911	0.926	0.939	0.949	0.959	0.967
2	0.974	0.979	0.983	0.987	0.989	0.992	0.994	0.995	0.996	0.997

Table 5.2: Look up table for drawing random numbers according to a Gaussian probability density function. The entries of the table are the values of the distribution function $P(X < z)$. The numbers in the first row give the decimal part of the answer and the values in the first column give the integer part of the answer. If we draw a random number 0.988 from a uniform distribution in the range [0,1], then by looking at this look up table we can see that z must be between 2.3 and 2.4 say 2.35. A more accurate value is obtained by interpolation.

Figure 5.8 and 5.9 show two Gaussian sampling patterns, one with 20,000 points and another with 10,000 points, and figure 5.10 shows both the Gaussian distribution we used to produce it and the actual distribution of the points of the mask. The density

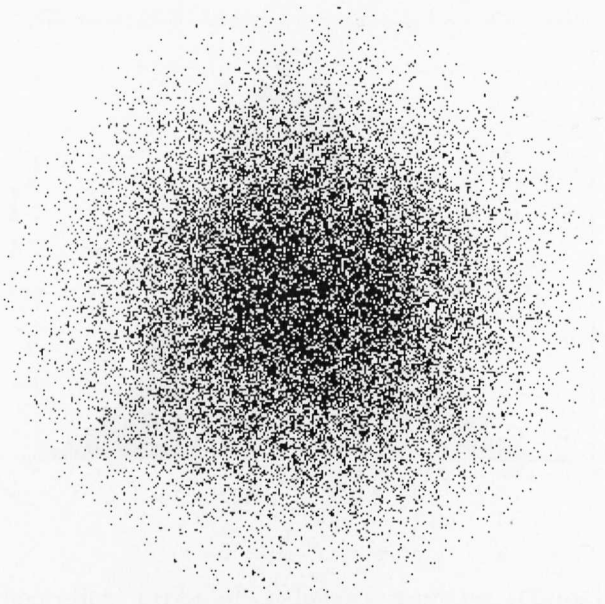


Figure 5.8: Gaussian sampling mask with 20,000 points.

in the center of the mask is less than that of the Gaussian and away from the center it is more than that of the Gaussian. The difference between the theoretical and the true mask is less pronounced when we use fewer sampling points as shown in figure 5.11 where we plot the mask with 10,000 points. We use these masks to convert all 112 textures which are sampled regularly to irregularly sampled data. After that we use the Hough transform and the trace transform to recognise the textures.

5.4.2 The Log-Polar Sampling Pattern

To compute the radii of rings in log-polar coordinates, we use:

$$r_n = e^{\frac{n \ln R}{N}} \tag{5.51}$$

where r_n is the radius of the n^{th} ring, R is the maximum radius, and N is the number of rings. We used this formula to find the radius for each ring from 1 to N. After finding the radius for each ring, we must distribute N_n points around that ring. To do that, we choose for each point an angle ϕ with integer value in the range [0,359]. N_n is

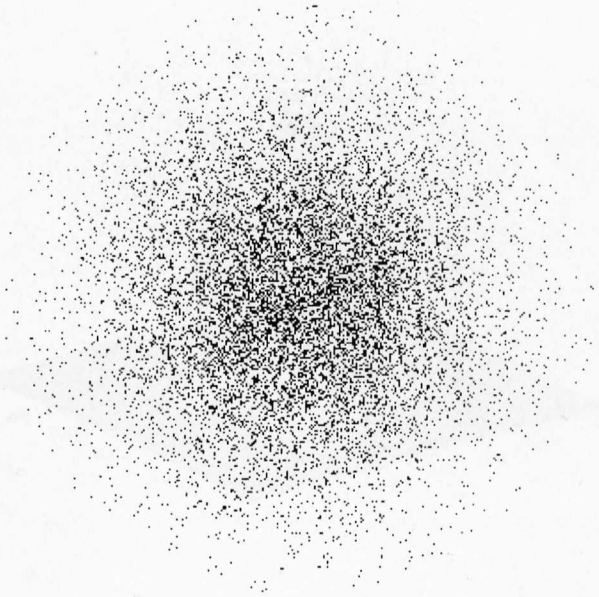


Figure 5.9: Gaussian sampling mask with 10,000 points.

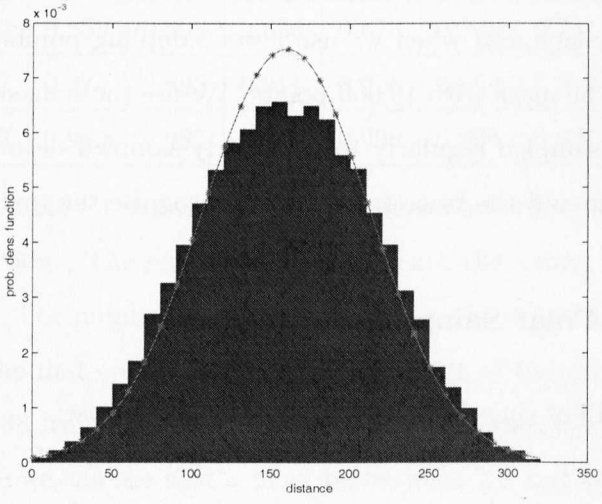


Figure 5.10: The theoretical probability density function (Gaussian) we use to draw the sampling points and the true probability density function we produce due to the repeated trials we allow (printed as bars). The number of points in the mask is 20,000 points.

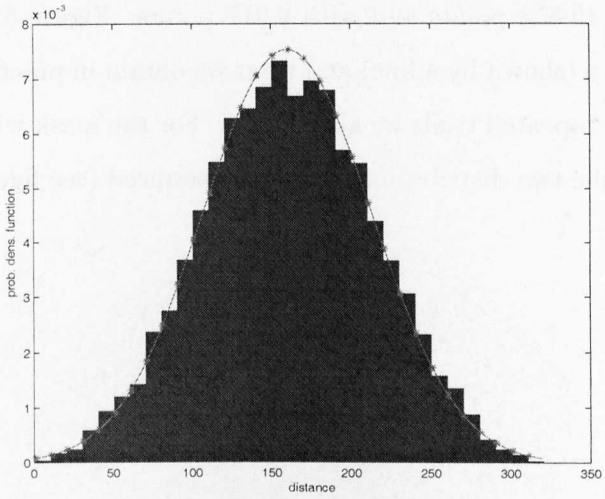


Figure 5.11: The theoretical probability density function (Gaussian) we use to draw the sampling points and the true probability density function we produce due to the repeated trials we allow (printed as bars). The number of points in the mask is 10,000 points.

the numbers of points on each ring and it can be obtained by dividing the total number of points we want to have (N_t), by the number of rings (N).

$$N_n = \frac{N_t}{N} \tag{5.52}$$

The points on each ring must then be spread uniformly to occupy an annulus with radii in the range:

$$\left[\frac{r_n + r_{n-1}}{2}, \frac{r_n + r_{n+1}}{2}\right] \tag{5.53}$$

So, we draw random numbers uniformly distributed in this range. So, if X is the random number we drew in the range [0,1], the point of the ring with radius r_n is shifted to:

$$X \times \left(\frac{r_n + r_{n+1}}{2} - \frac{r_n + r_{n-1}}{2}\right) + \frac{r_n + r_{n-1}}{2} \tag{5.54}$$

It must be mentioned that for the last ring we will have some points shifted outside the ring. In building the mask we ignore these points in order to avoid the creation of a rim for our mask. Figure 5.12 and 5.13 show the masks obtained by using $R = 160$,

$N = 64$ containing 19,852 points and with 9,917 points. Figure 5.14 shows both the log-polar distribution (shown by a line) and what we obtain in practice (the graph with the bars) due to the repeated trials we allow again. For the mask with 9,917 points the difference between the two distributions is less pronounced (see figure 5.15).

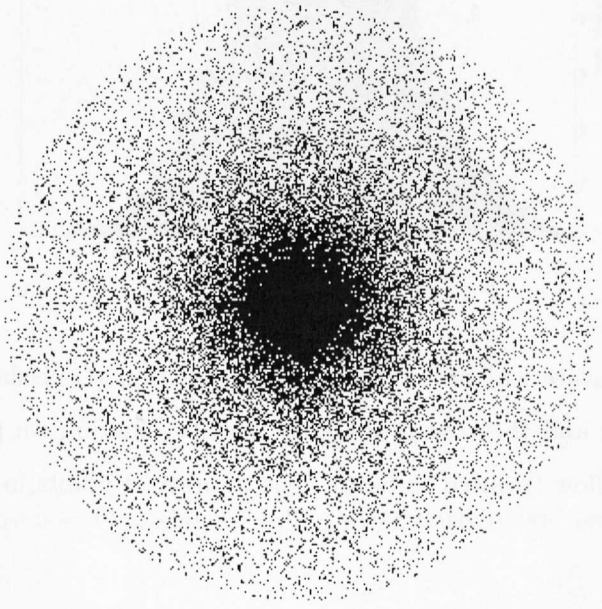


Figure 5.12: This figure shows the mask obtained by using $R = 160$, $N = 64$ containing 19,852 points.

5.4.3 Retina Sampling Pattern

In this section we use the retinal sampling model of [48] for building a sampling pattern. Let us say that $p(r)dr$ is the probability density function of the distribution of cones as a function of the polar radius r. By the approximation used in [48], this is equal to $\frac{Adr}{ar+b}$, or by dividing numerator and denominator by A, we have: $\frac{dr}{a'r+b'}$ where A, a, b, a' and b' are model parameters. To find values of the parameters, we do the following: Suppose we know that there are c_{r_1,r_2} cones between radii r_1 and r_2, and that the total number of cones is N. First we compute the probability of finding a cone between radii r_1 and r_2. This is the integral of $p(r)$ between these two radii:

$$P_{1,2} = \int_{r_1}^{r_2} p(r)dr = \int_{r_1}^{r_2} \frac{1}{a'r + b'}dr = \frac{\ln(a'r_2 + b') - \ln(a'r_1 + b')}{a'} \tag{5.55}$$

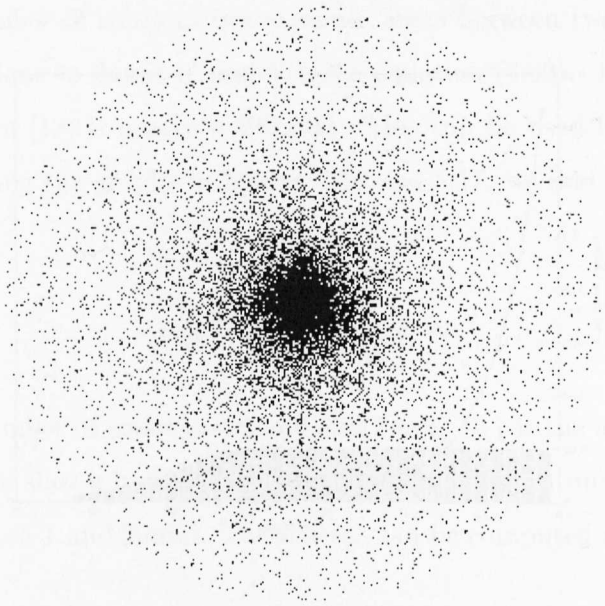


Figure 5.13: This figure shows the mask obtained by using $R = 160$, $N = 64$ containing 9,917 points.

Then we multiply this probability with the total number of cones that exist, ie with N. This then is equal to c_{r_1,r_2}:

$$c_{r_1,r_2} = N \times P_{1,2} \tag{5.56}$$

By using equations (5.55) and (5.56), we can write:

$$c_{r_1,r_2} = \frac{N}{a'}(\ln(a'r_2 + b') - \ln(a'r_1 + b')) \tag{5.57}$$

So:

$$\frac{c_{r_1,r_2}a'}{N} = \ln(a'r_2 + b') - \ln(a'r_1 + b') \tag{5.58}$$

By taking the exponential function of both sides, we have:

$$e^{\frac{c_{r_1,r_2}a'}{N}} = \frac{a'r_2 + b'}{a'r_1 + b'} = \frac{a'(r_2 - r_1)}{a'r_1 + b'} + 1 \tag{5.59}$$

So:

$$b' = \frac{a'(r_2 - r_1)}{e^{\frac{a'c_{r_1,r_2}}{N}} - 1} - a'r_1 \tag{5.60}$$

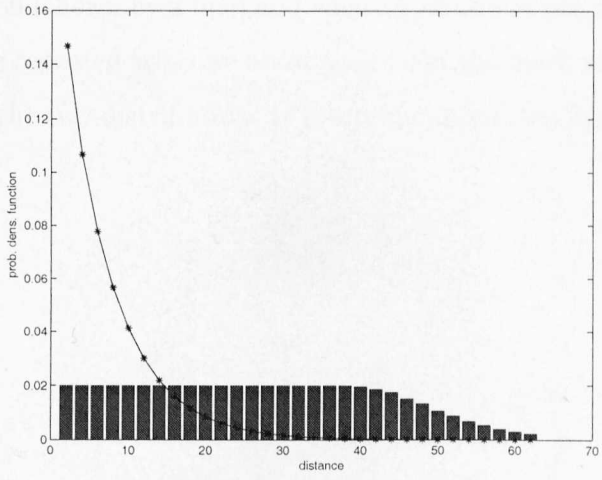


Figure 5.14: The log-polar distribution used to produce the mask in figure 5.12 (the graph with the line) and what we obtain in practice (the graph with the bars). The number of points in the mask is 19,852.

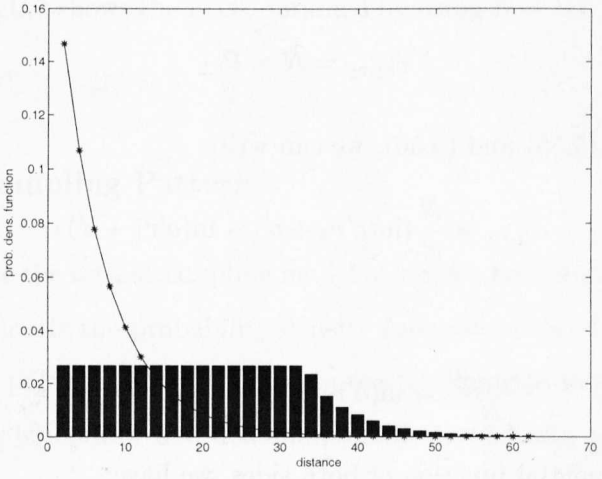


Figure 5.15: The log-polar distribution used to produce the mask in figure 5.12 (the graph with the line) and what we obtain in practice (the graph with the bars). The number of points in the mask is 9,917.

If we know the number of cones in two separate areas between two different radii, we can find two equations to define a' and b' using equation (5.60). The total number of cones as reported in [13] is nearly $4,500,000$. This can be used for $r_2 = 20mm$ and $r_1 = 0$. Also by using the graphs in figures 5.16 and 5.17, we find that for $r_2 = 1$ and $r_1 = 0$ we have:

$$c_{1,0} = \frac{S_{0,1}}{S_{0,1} + S_{1,20}} \times N \tag{5.61}$$

where $c_{1,0}$ is the number of cones between 0 and $1mm$. $S_{0,1}$ is the area under the curve between 0 and 1mm shown in figure 5.16, and $S_{1,20}$ is the area under the curve shown in figure 5.17 between 1 and 20mm. These areas can be computed using the trapezium rule as follows:

$$S_{0,1} = \sum_{n=1}^{12} \frac{Y_n + Y_{n-1}}{2}(X_n - X_{n-1}) \tag{5.62}$$

$$S_{1,20} = \frac{Y_{14} + Y_{13}}{2}(X_{14} - X_{13}) \tag{5.63}$$

Table 5.3 lists the coordinate positions of the points plotted in figures 5.16 and 5.17. Using these numbers we find that:

$$S_{0,1} = 50.95 \tag{5.64}$$

$$S_{1,20} = 275.5 \tag{5.65}$$

So,

$$c_{1,0} = \frac{50.95}{50.95 + 275.5} \times 4,500,000 = 702,328 \tag{5.66}$$

So we have $702,328$ cones in the area between 0 to $1mm$. Now we can find a' and b'. We used equation (5.60) for two pairs of values, namely $(r_1 = 0, r_2 = 1, c_{r_1,r_2} = 702,328)$ and $(r_1 = 0, r_2 = 20, c_{r_1,r_2} = 4,500,000)$ to find a' and b', which turn out to be $a' = 2.24$ and $b' = 5.35$.

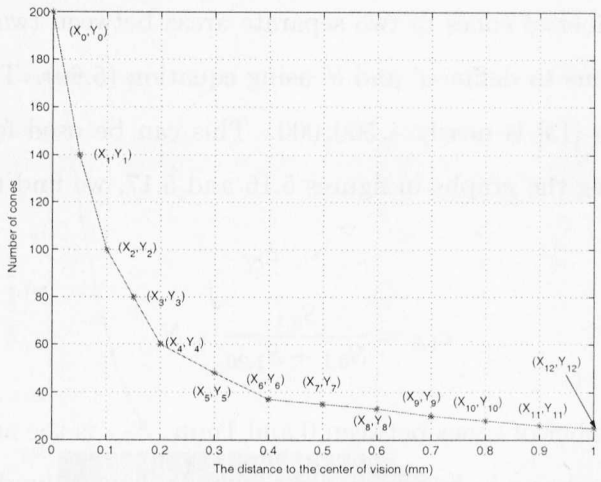


Figure 5.16: This figure is taken from [48] and shows the cone density versus distance from the fovea. It is a zoom in to the central part of the curve shown in figure 5.17.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X	0	0.05	0.1	0.15	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.0	20
Y	200	140	100	80	60	48	37	35	33	30	28	26	25	25	4

Table 5.3: The X_i and Y_i for some points in the Retinal model distribution. The values are taken from figures 5.16 and 5.17.

To compute the distribution function $P(z)$ which corresponds to this probability density function, we integrate it from 0 to z:

$$P(z) = \frac{\ln(a'z + b') - \ln(b')}{a'} \tag{5.67}$$

To choose numbers then according to this distribution, we make a look up table for $P(z)$ versus z. We choose random numbers in the range [0,1]. These are the values of $P(z)$. Then we read the corresponding value of z from the table. This is the polar radius r of the point. Again we choose an integer random number uniformly distributed between 0^o and 359^o to be its angle.

This function in our eyes is from 0 to $20mm$. If we want to simulate this method for our

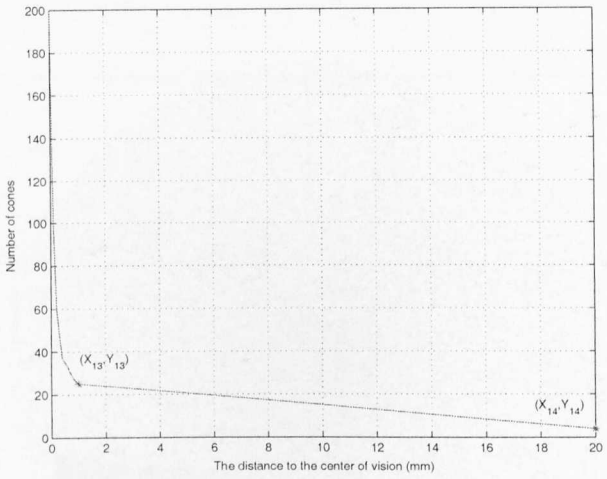


Figure 5.17: This figure is taken from [48] and shows the cone density versus distance from the fovea.

images which have maximum radius 160, we must multiply that interval by 8. Figure 5.18 and figure 5.19, show a retinal sampling mask with 20,000 points and one with 10,000 points, while figure 5.20 shows both the retinal distribution we used to produce the mask with the 20,000 points and what we obtained in practice due to the limited number of discrete points we can have in a digital image. For the mask with 10,000 points the difference between the two distributions is less pronounced (see figure 5.21).

5.5 Experiments

Both our training and test images are 320×320 pixels in size. This means that each texture contains $320 \times 320 = 102,400$ points. First we assume that we do not really have these images sampled with the regular grid, so we re-sample them using the sampling patterns described in the previous section.

The number of random points we consider is an important parameter. We performed experiments with 10,000 and 20,000 points in each sampling pattern to see the effect of this parameter on the results. The other important problem parameters are the parameters of the Hough space. We used 452 points to sample the values of parameter

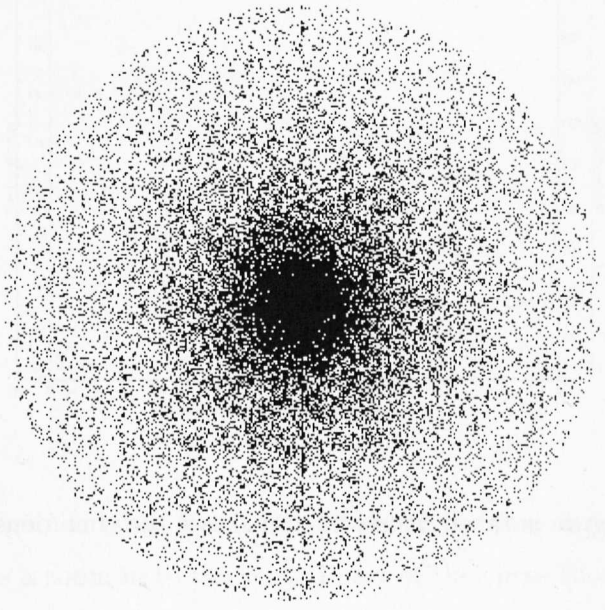


Figure 5.18: This figure shows the retinal sampling mask with 20,000 points.

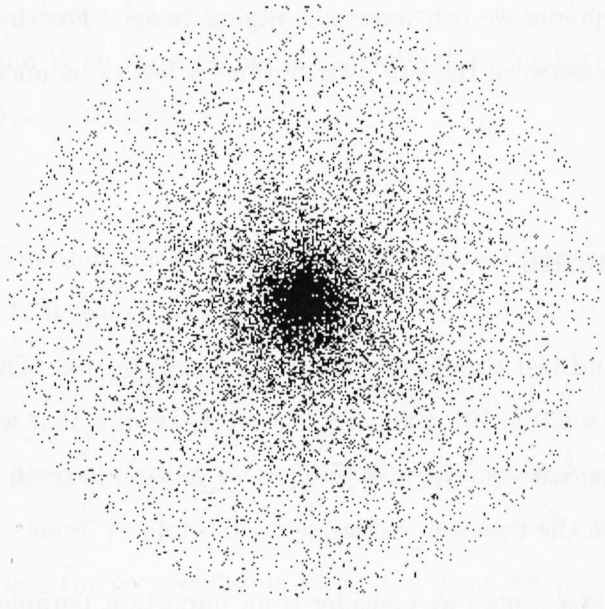


Figure 5.19: This figure shows the retinal sampling mask with 10,000 points.

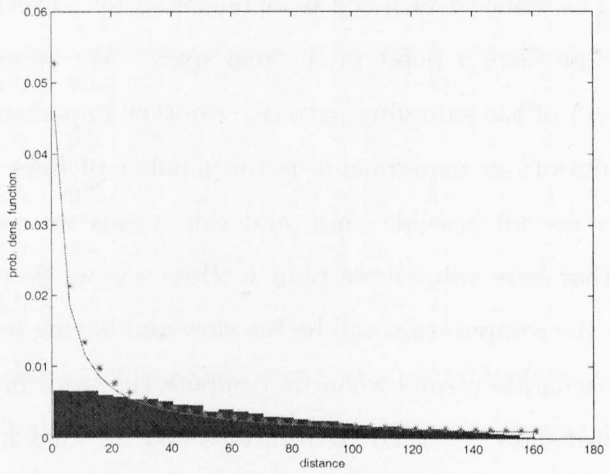


Figure 5.20: The theoretical retinal distribution we used to produce the mask in figure 5.18 (line) and the distribution of the actual sampling pattern (bars). The number of points in the mask is 20,000.

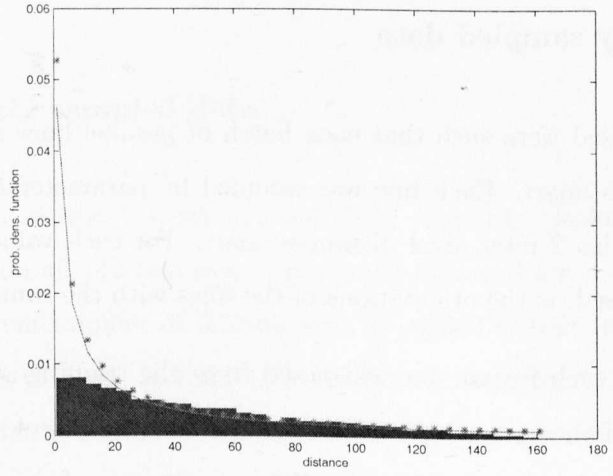


Figure 5.21: The theoretical retinal distribution we used to produce the mask in figure 5.18 (line) and the distribution of the actual sampling pattern (bars). The number of points in the mask is 10,000.

ρ and 360 points to sample θ. This way we created the accumulator array of the Hough space [74]. The value of each cell is incremented by 1 every time it is crossed by the curve that represents a point in the real space. We construct one such line for every point (x_i, y_i) of the sampling pattern. Another important parameter which may affect the results of our experiments is the number of lines we use for tracing the image. We can use all possible lines, and this means we use all points in the accumulator array that have value more than 1. However, we do not wish to use too many lines, because the computation will be too slow and having too sparsely sampled lines will produce inaccurate results when we compute the trace functional. When we had regularly sampled data, we used 20 rotations and 32 lines in each rotation for tracing the texture. So, we used $32 \times 20 = 640$ lines totally. Here, we expect to use more lines, to improve robustness to the errors introduced by the irregular positioning of the sample points along each line. So we used the first 1000 and 2000 lines with the most number of points, as identified by the Hough transform.
We are going to use the results of the regularly sampled data as a benchmark against which we shall evaluate the results obtained from the sub-sampled data.

5.5.1 Regularly sampled data

The tracing lines used were such that each batch of parallel lines consisted of lines 2 inter-pixel distances apart. Each line was sampled by parameter t, so that the sampling points were also 2 inter-pixel distances apart. For each value of p, 20 different orientations were used, ie the orientations of the lines with the same p differed by 18^o.

The significance of each feature was extracted from the training samples, and subsequently each one of the test samples was associated with the reference texture from which the distance value computed by the distance equation of chapter 3 was minimal. Table 5.4 presents the results of this approach for identifying the correct class of a texture as the most similar one, the second most similar one, the third most similar one, the fourth most similar one, and beyond, presenting the numbers under the corresponding columns. All these numbers are out of 112, as we present the results of testing separately for each set of data. In all cases the reference set was the TL set of

Q	Test Set BL					Test Set BR				
	1	2	3	4	R	1	2	3	4	R
0.5	90	11	1	1	9	81	11	3	3	14
1.0	92	10	1	0	9	81	8	6	4	13
1.5	**95**	**6**	**0**	**1**	**10**	**82**	**11**	**3**	**4**	**12**
2.0	88	10	2	2	10	78	14	3	2	15
2.5	84	12	3	2	11	74	11	6	4	17

Table 5.4: Texture classification results using the trace transform method (TT). Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q. All numbers are out of 112.

images. Each row of results corresponds to a different choice of threshold Q. The best results were obtained for $Q = 1.5$.

5.5.2 Irregularly sampled data

As we explained in section 5.4, we will use three sampling masks with two different numbers of points on all 112 textures. The results obtained for the different sampling patterns and different number of tracing lines identified by the Hough transform are shown in tables 5.5 to 5.13. In each table we give in bold the best results obtained. These tables include the results of dealing with the irregularly sampling lines carefully e.g. by using normalised convolution (NC), as well as the results obtained by simply treating the missing points along each tracing lines as zeros (MP) and the results obtained by ignoring the missing points (IM), i.e. ignoring the fact that the tracing lines are irregularly sampled. In all cases, training was done using the same irregular sampling mask. The idea behind this is that each human eye has its own sampling pattern which is always fixed.

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	61	14	10	6	21	50	12	10	9	34	0.5
1000	10000	75	14	6	3	14	58	13	6	4	32	1
1000	10000	71	15	6	5	15	53	15	6	5	32	1.5
1000	10000	62	16	10	6	18	46	19	10	2	37	2
1000	10000	57	14	13	8	20	44	13	13	5	38	2.5
1000	20000	71	13	5	5	18	54	12	5	7	33	0.5
1000	20000	80	13	3	3	13	59	12	3	4	31	1
1000	20000	78	11	6	1	16	56	17	6	4	33	1.5
1000	20000	69	15	7	3	18	53	14	7	4	37	2
1000	20000	67	11	6	9	19	52	14	6	3	40	2.5
2000	10000	72	15	4	4	17	57	9	4	4	36	0.5
2000	10000	84	8	3	2	15	61	8	3	3	35	1
2000	10000	82	9	5	4	12	61	8	5	2	38	1.5
2000	10000	76	10	7	3	16	57	10	7	2	38	2
2000	10000	71	15	5	2	19	53	12	5	6	38	2.5
2000	20000	76	13	6	4	13	60	11	6	3	31	0.5
2000	20000	**83**	**13**	**1**	**5**	**10**	**64**	**12**	**1**	**2**	**32**	1
2000	20000	79	16	1	2	14	65	10	1	3	31	1.5
2000	20000	73	17	3	2	17	60	13	3	3	32	2

Table 5.5: Texture classification results using the **Gaussian** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **normalised convolution was used in order to deal with the irregular sampling, where appropriate.**

Lines	*Points*	Test Set *BL*					Test Set *BR*					Q
		1	2	3	4	R	1	2	3	4	R	
1000	10000	25	14	8	5	60	18	3	8	4	76	0.5
1000	10000	33	11	5	6	57	17	10	5	5	73	1
1000	10000	33	11	11	5	52	18	13	11	5	71	1.5
1000	10000	33	10	9	10	50	19	8	9	5	71	2
1000	10000	31	13	6	11	51	18	7	6	6	72	2.5
1000	20000	31	17	5	5	54	23	8	5	7	70	0.5
1000	20000	41	13	13	7	38	24	14	13	2	63	1
1000	20000	46	11	10	5	40	25	15	10	5	60	1.5
1000	20000	42	15	9	6	40	27	11	9	5	61	2
1000	20000	43	11	10	8	40	28	7	10	7	64	2.5
2000	10000	31	11	3	6	61	16	10	3	3	79	0.5
2000	10000	37	11	13	4	47	19	13	13	8	69	1
2000	10000	43	10	13	6	40	20	10	13	4	66	1.5
2000	10000	44	15	8	3	42	19	10	8	6	67	2
2000	10000	45	15	6	4	42	20	10	6	4	69	2.5
2000	20000	36	14	8	6	48	25	10	8	2	70	0.5
2000	20000	48	13	6	6	39	26	13	6	3	63	1
2000	20000	**52**	**11**	**6**	**6**	**37**	**29**	**12**	**6**	**4**	**59**	1.5
2000	20000	50	13	9	4	36	27	15	9	3	59	2

Table 5.6: Texture classification results using the **Gaussian** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no normalised convolution was used, but missing sampling points along each tracing line were identified and their values were set to** 0.

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	55	15	9	3	30	47	8	9	7	44	0.5
1000	10000	72	8	10	3	19	51	15	10	4	36	1
1000	10000	73	10	6	4	19	52	10	6	8	35	1.5
1000	10000	62	19	2	4	25	49	9	2	7	38	2
1000	10000	57	16	9	4	26	47	9	9	6	41	2.5
1000	20000	70	18	3	6	15	49	13	3	7	37	0.5
1000	20000	73	17	6	1	15	59	7	6	3	33	1
1000	20000	76	14	5	3	14	57	11	5	3	35	1.5
1000	20000	75	12	3	3	19	53	13	3	7	35	2
1000	20000	64	18	5	4	21	53	11	5	5	37	2.5
2000	10000	67	11	5	5	24	43	17	5	4	39	0.5
2000	10000	80	9	4	1	18	54	15	4	6	31	1
2000	10000	78	9	4	3	18	53	12	4	7	34	1.5
2000	10000	77	9	2	2	22	51	12	2	5	38	2
2000	10000	70	14	2	1	25	48	13	2	5	41	2.5
2000	20000	74	14	8	1	15	56	10	8	4	33	0.5
2000	20000	**82**	**9**	**6**	**2**	**13**	**58**	**13**	**6**	**6**	**29**	1
2000	20000	84	8	2	1	17	55	11	2	6	32	1.5
2000	20000	76	9	6	1	20	54	15	6	3	36	2

Table 5.7: Texture classification results using the **Gaussian** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no action was taken to deal with the irregular sampling**. i.e the missing points were treated as if they were not there.

Lines	*Points*	Test Set *BL*					Test Set *BR*					Q
		1	2	3	4	R	1	2	3	4	R	
1000	10000	59	12	7	1	33	44	11	7	8	43	0.5
1000	10000	72	11	5	5	19	57	8	5	7	38	1
1000	10000	70	13	4	5	20	52	10	4	6	37	1.5
1000	10000	66	13	7	3	23	49	12	7	5	40	2
1000	10000	66	11	8	3	24	51	10	8	6	41	2.5
1000	20000	73	9	7	7	16	52	13	7	3	37	0.5
1000	20000	74	14	4	6	14	56	14	4	4	31	1
1000	20000	74	15	2	4	17	58	10	2	7	31	1.5
1000	20000	67	15	7	5	18	53	10	7	4	37	2
1000	20000	67	14	6	7	18	50	10	6	8	36	2.5
2000	10000	59	16	4	0	33	49	15	4	2	39	0.5
2000	10000	70	18	3	1	20	54	13	3	5	34	1
2000	10000	74	12	4	5	17	57	9	4	3	34	1.5
2000	10000	69	12	4	4	23	51	10	4	8	36	2
2000	10000	68	13	1	5	25	48	8	1	9	39	2.5
2000	20000	77	9	7	5	14	57	13	7	3	32	0.5
2000	20000	**80**	**14**	**6**	**0**	**12**	**64**	**11**	**6**	**2**	**30**	1
2000	20000	76	14	5	5	12	65	8	5	2	29	1.5
2000	20000	74	14	2	6	16	65	8	2	3	33	2

Table 5.8: Texture classification results using the **Logpolar** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **normalised convolution was used in order to deal with the irregular sampling, where appropriate.**

Lines	*Points*	Test Set *BL*					Test Set *BR*					Q
		1	2	3	4	R	1	2	3	4	R	
1000	10000	40	5	7	2	58	32	6	7	2	65	0.5
1000	10000	49	8	10	3	42	36	11	10	6	57	1
1000	10000	49	10	6	2	45	39	4	6	5	57	1.5
1000	10000	52	6	7	3	44	36	10	7	4	59	2
1000	10000	51	8	5	4	44	40	6	5	4	58	2.5
1000	20000	48	10	8	9	37	36	10	8	2	56	0.5
1000	20000	56	16	5	4	31	44	7	5	4	52	1
1000	20000	60	12	4	6	30	40	9	4	2	53	1.5
1000	20000	62	10	4	4	32	41	9	4	5	52	2
1000	20000	61	11	4	2	34	40	7	4	5	54	2.5
2000	10000	41	4	3	8	56	32	9	3	2	65	0.5
2000	10000	46	8	9	7	42	33	10	9	2	58	1
2000	10000	49	12	3	5	43	37	8	3	7	55	1.5
2000	10000	50	9	7	3	43	33	9	7	5	59	2
2000	10000	49	11	5	4	43	34	7	5	7	62	2.5
2000	20000	47	9	13	8	35	38	10	13	6	53	0.5
2000	20000	59	15	5	3	30	43	8	5	6	52	1
2000	20000	**61**	**13**	**2**	**4**	**32**	**44**	**5**	**2**	**11**	**48**	1.5
2000	20000	62	10	2	3	35	42	4	2	5	54	2

Table 5.9: Texture classification results using the **Logpolar** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no normalised convolution was used, but missing sampling points along each tracing line were identified and their values were set to 0.**

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	52	9	8	6	37	45	9	8	11	41	0.5
1000	10000	63	9	8	3	29	54	9	8	3	39	1
1000	10000	63	11	6	3	29	52	9	6	5	38	1.5
1000	10000	62	11	6	4	29	50	9	6	1	44	2
1000	10000	59	11	8	5	29	51	7	8	4	43	2.5
1000	20000	58	14	4	9	27	48	13	4	7	39	0.5
1000	20000	68	15	7	2	20	56	13	7	5	34	1
1000	20000	70	14	8	6	14	56	11	8	3	35	1.5
1000	20000	69	15	7	3	18	54	12	7	2	39	2
1000	20000	66	14	10	3	19	52	13	10	1	38	2.5
2000	10000	48	14	11	10	29	46	11	11	4	46	0.5
2000	10000	63	11	7	4	27	48	16	7	4	37	1
2000	10000	63	12	7	4	26	47	14	7	4	42	1.5
2000	10000	57	14	8	5	28	50	9	8	5	45	2
2000	10000	59	9	9	5	30	46	12	9	4	47	2.5
2000	20000	60	13	7	5	27	48	17	7	5	36	0.5
2000	20000	70	16	7	2	17	57	10	7	4	35	1
2000	20000	**74**	**15**	**5**	**3**	**15**	**56**	**11**	**5**	**7**	**36**	1.5
2000	20000	67	18	8	3	16	55	12	8	7	34	2

Table 5.10: Texture classification results using the **Logpolar** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no action was taken to deal with the irregular sampling**. i.e the missing points were treated as if they were not there.

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	51	13	10	3	35	35	14	10	13	43	0.5
1000	10000	72	9	5	6	20	54	5	5	4	39	1
1000	10000	71	12	5	5	19	54	6	5	6	41	1.5
1000	10000	65	17	2	7	21	49	9	2	8	42	2
1000	10000	64	15	5	5	23	46	11	5	7	42	2.5
1000	20000	68	12	8	4	20	51	10	8	6	36	0.5
1000	20000	75	14	2	4	17	60	11	2	3	36	1
1000	20000	74	11	5	5	17	57	8	5	5	33	1.5
1000	20000	69	14	8	4	17	55	9	8	8	35	2
1000	20000	64	15	9	4	20	54	10	9	11	33	2.5
2000	10000	64	13	1	2	32	48	12	1	8	34	0.5
2000	10000	76	10	5	3	18	53	14	5	2	36	1
2000	10000	72	11	5	6	18	55	9	5	9	34	1.5
2000	10000	66	14	3	7	22	50	11	3	6	40	2
2000	10000	62	17	5	6	22	47	14	5	4	43	2.5
2000	20000	70	16	7	3	16	56	13	7	3	35	0.5
2000	20000	75	18	6	1	12	59	16	6	2	29	1
2000	20000	**77**	**12**	**6**	**5**	**12**	**62**	**10**	**6**	**0**	**34**	1.5
2000	20000	66	21	5	4	16	56	12	5	3	36	2

Table 5.11: Texture classification results using the **Retinal** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **normalised convolution was used in order to deal with the irregular sampling, where appropriate.**

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	34	9	7	3	59	21	13	7	3	71	0.5
1000	10000	41	7	6	7	51	28	8	6	7	62	1
1000	10000	45	5	10	5	47	26	12	10	3	64	1.5
1000	10000	42	7	6	6	51	25	12	6	3	65	2
1000	10000	41	6	7	5	53	24	11	7	4	63	2.5
1000	20000	49	8	11	6	38	31	11	11	4	60	0.5
1000	20000	51	18	3	6	34	36	7	3	7	55	1
1000	20000	57	9	5	9	32	38	10	5	5	55	1.5
1000	20000	54	11	5	5	37	37	12	5	5	56	2
1000	20000	55	10	4	4	39	36	11	4	5	56	2.5
2000	10000	33	10	5	5	59	24	8	5	5	69	0.5
2000	10000	47	7	4	9	45	32	8	4	3	61	1
2000	10000	51	8	3	5	45	26	13	3	2	63	1.5
2000	10000	47	9	4	6	46	26	9	4	6	62	2
2000	10000	46	8	6	6	46	24	10	6	7	64	2.5
2000	20000	46	14	8	7	37	29	14	8	4	57	0.5
2000	20000	58	12	9	4	29	38	9	9	1	59	1
2000	20000	**60**	**11**	**5**	**6**	**30**	**40**	**7**	**5**	**7**	**54**	1.5
2000	20000	55	12	5	3	37	42	5	5	2	56	2

Table 5.12: Texture classification results using the **Retinal** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no normalised convolution was used, but missing sampling points along each tracing line were identified and their values were set to** 0.

Lines	*Points*	Test Set *BL*					Test Set *BR*					*Q*
		1	2	3	4	R	1	2	3	4	R	
1000	10000	51	13	7	4	37	40	12	7	10	44	0.5
1000	10000	58	13	5	3	33	48	15	5	5	41	1
1000	10000	62	11	4	5	30	51	11	4	7	39	1.5
1000	10000	59	8	5	6	34	50	10	5	5	42	2
1000	10000	56	6	6	7	37	47	10	6	4	43	2.5
1000	20000	56	15	7	8	26	48	10	7	5	40	0.5
1000	20000	72	8	11	3	18	55	12	11	5	36	1
1000	20000	72	12	5	4	19	52	12	5	6	38	1.5
1000	20000	69	14	4	3	22	49	8	4	4	45	2
1000	20000	66	16	4	2	24	47	12	4	4	45	2.5
2000	10000	53	17	8	2	32	41	14	8	7	40	0.5
2000	10000	62	12	3	4	31	51	13	3	7	39	1
2000	10000	64	7	9	3	29	52	12	9	5	41	1.5
2000	10000	59	11	5	4	33	48	13	5	3	45	2
2000	10000	56	11	7	4	34	47	12	7	3	46	2.5
2000	20000	59	14	7	8	24	43	17	7	4	38	0.5
2000	20000	70	15	8	2	17	56	10	8	4	37	1
2000	20000	**76**	**11**	**7**	**2**	**16**	**53**	**15**	**7**	**4**	**37**	1.5
2000	20000	70	16	4	2	20	52	11	4	5	39	2

Table 5.13: Texture classification results using the **Retinal** sampling pattern. Under the headings 1, 2, 3 and 4 we show how many times the correct texture appeared in the first, second, third and fourth position of the returned answer, respectively. Under heading R we show how many times it appeared in one of the remaining positions. The results are shown for different values of threshold Q, number of tracing lines and number of points in the sampling pattern. All numbers are out of 112 and **no action was taken to deal with the irregular sampling**. i.e the missing points were treated as if they were not there.

Position	Regular	Gaussian			Log-polar			Retinal		
		NC	MP	IM	NC	MP	IM	NC	MP	IM
1	**79**	**65**	36	62	**64**	46	58	**62**	44	57
2	**86**	**76**	46	72	**75**	54	69	**71**	52	69
3	**87**	**77**	56	**77**	**78**	56	74	**77**	57	75
4	**90**	**81**	60	**81**	**79**	63	78	**79**	62	78

Table 5.14: This table shows the best result from each method in terms of percentages and returns of the correct answer in the first, the first 2, the first 3 or the first 4 positions. The bold one for each mask is the best result between NC, MP and IM. In all cases the number of points is 20,000 and the number of tracing lines is 2,000.

Table 5.14 summarises the best result from each method in terms of percentages and returns of the correct answer in the first, the first 2, the first 3 or the first 4 positions, for both datasets. In figure 5.22 we show the best results for each sampling pattern. The best result of all cases was produced by the Gaussian sampling pattern when we used 20,000 sampling points and 2,000 tracing lines and is 65% return in the first position.

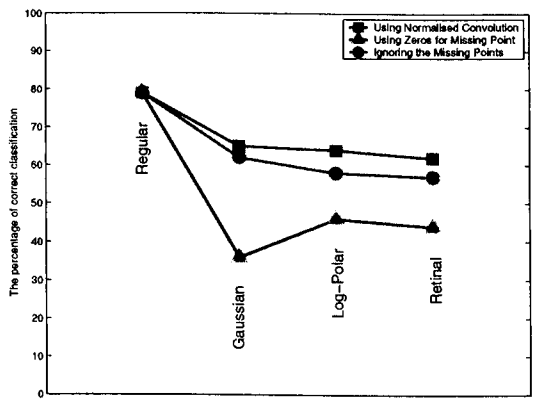


Figure 5.22: By using three sampling patterns which are Gaussian, Logpolar and Retinal, we can obtain the above results. In each sampling pattern we used three methods: NC, MP and IM. We can see that the best result in total experiments belongs to the Gaussian sampling pattern which has been obtained by the NC method.

5.6 Conclusions

We demonstrated in this chapter that by using the Trace transform method we may recognise textures from irregularly sampled images. The Hough transform was used as an interface that allowed us to identify tracing lines in the image and normalised convolution allowed us to deal with the irregularly placed samples along the tracing lines in order to compute the trace functionals.

From table 5.14 we note that the worse thing one might do is to assume that the missing points along the irregularly sampled tracing lines have 0 value. If we simply apply the trace transform ignoring the fact that the tracing lines we use are made up of irregularly placed points, we do only slightly worse than when we take extra care to deal with irregular sampling. (Compare columns NC and IM in table 5.14). However this is likely to yield very poor result if the sampling pattern changes between training and testing.

Overall the best results were produced by the Gaussian sampling pattern. This is because this pattern had its points more uniformly spread all over the image than either the log polar or the retinal pattern. However, none of the sampling patterns was what it was supposed to be. This is because in our experiments we are limited by the bandwidth of the already digitised images. The sampling patterns are supposed to be used to sample an analogue scene. In addition, human retina has 4,500,000 sampling points, as opposed to a few thousands we used here. So, given that we used less than 20% of the pixels in the original regularly sampled images, and we choose them at random positions, something that goes against the fundamental property of texture being a spatial property, the produced results are remarkably good in comparison with the results obtained by using regular grids.

One way to improve the classification accuracy of irregularly sampled data is to use more than one places in the image where we place the sampling mask. After all, our eyes often scan an image by foveating at several places in it in order to achieve recognition.

A further improvement in the results may be achieved by using a hierarchical system where textures are first classified into broad classes and at a second stage they are

reclassified inside each broad class.

Chapter 6

Texture recognition from irregularly shaped samples

The Trace transform describes objects by characterising both their shape and texture. When all textures we had were represented by similarly shaped samples, shape information did not play any role in discriminating them. What will happen, however, if we have irregularly shaped texture samples? Can we recognise them by normalising the results of the functionals in tables 3.1 to 3.4 so that shape information is removed? This is what we want to test in this chapter. So, first we make 112 irregularly shaped texture samples by using 20 irregularly shaped masks, and then we use the Trace transform to recognise these textures. Then we apply the method to the classification of some real pathology images.

6.1 Simulated experiments

6.1.1 Irregularly shaped texture samples

Figure 6.1 shows 20 irregularly shaped masks which we use to convert textures in the Brodatz album to irregularly shaped texture samples.

For doing that we randomly choose one of the masks and impose it on the original texture. The masks were created by picking 20 images at random from an image

database and segmenting them. The black part of the chosen mask is used to define the acceptable part of the texture. Figure 6.2 shows the first 20 textures from the Brodatz album after they have been masked.

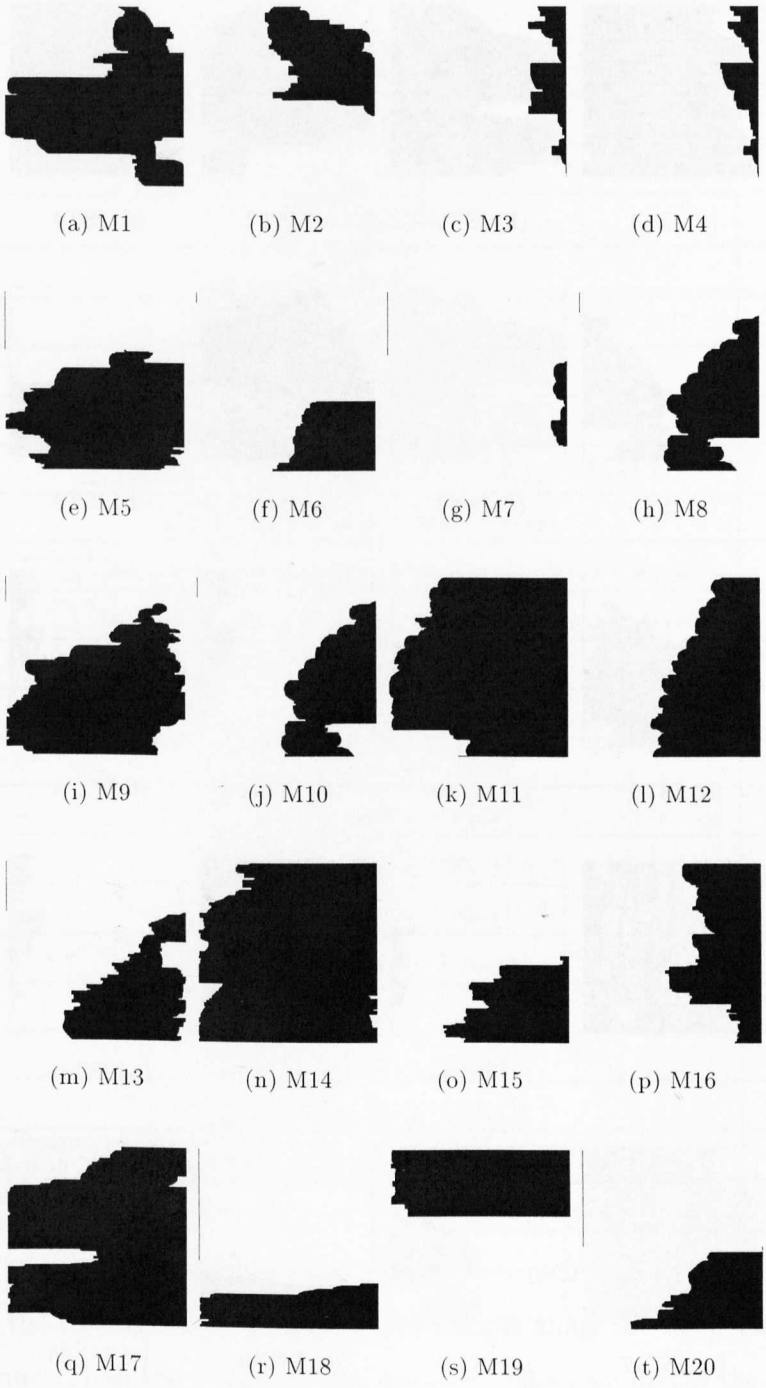
6.1.2 Adapting the trace transform to remove shape information

If we look at the functionals of the Trace transform in tables 3.1 to 3.4, we can see that some of them are invariant to the length of the tracing line, such as functionals which use the max or min operator. However, the values of some others change if the length of the tracing line changes. When we had fixed shaped texture samples, this did not matter for the classification result because for all textures we had lines with equal lengths. Here, we must divide the result of those functionals which are dependent on the length of the tracing line by the length of the line. Tables 6.1 to 6.4 show all functionals and whether it is necessary to divide their value by the length of the line or not.

6.1.3 Experiments

We used 112 irregularly shaped texture samples and then the Trace transform method described in chapter 3 to recognise these textures. To be able to use the features extracted by the Trace transform, we used tables 6.1 to 6.4 to see whether it is necassary to divide the result by the length of the tracing line or not. The results are shown in figure 6.3. These results are obviously worse than those obtained when using full images on rectangular grids. However, the correct answer is within the 8 first choices 86% of the time. One should realise that this experiment is a very severe test of the method as some of the masks used are very small as one can see from figure 6.1 and several of the textures in the database are macro textures, which means that using such a mask on them destroys the texture entirely.

The fact that the result curves in figure 6.3 converge as the rank of accepted choices increases, shows that the Trace transform method degrades gracefully i.e. when the method fails to recognise the correct textures in the first position of choice, it does not place it too far away from it.



(a) M1 (b) M2 (c) M3 (d) M4

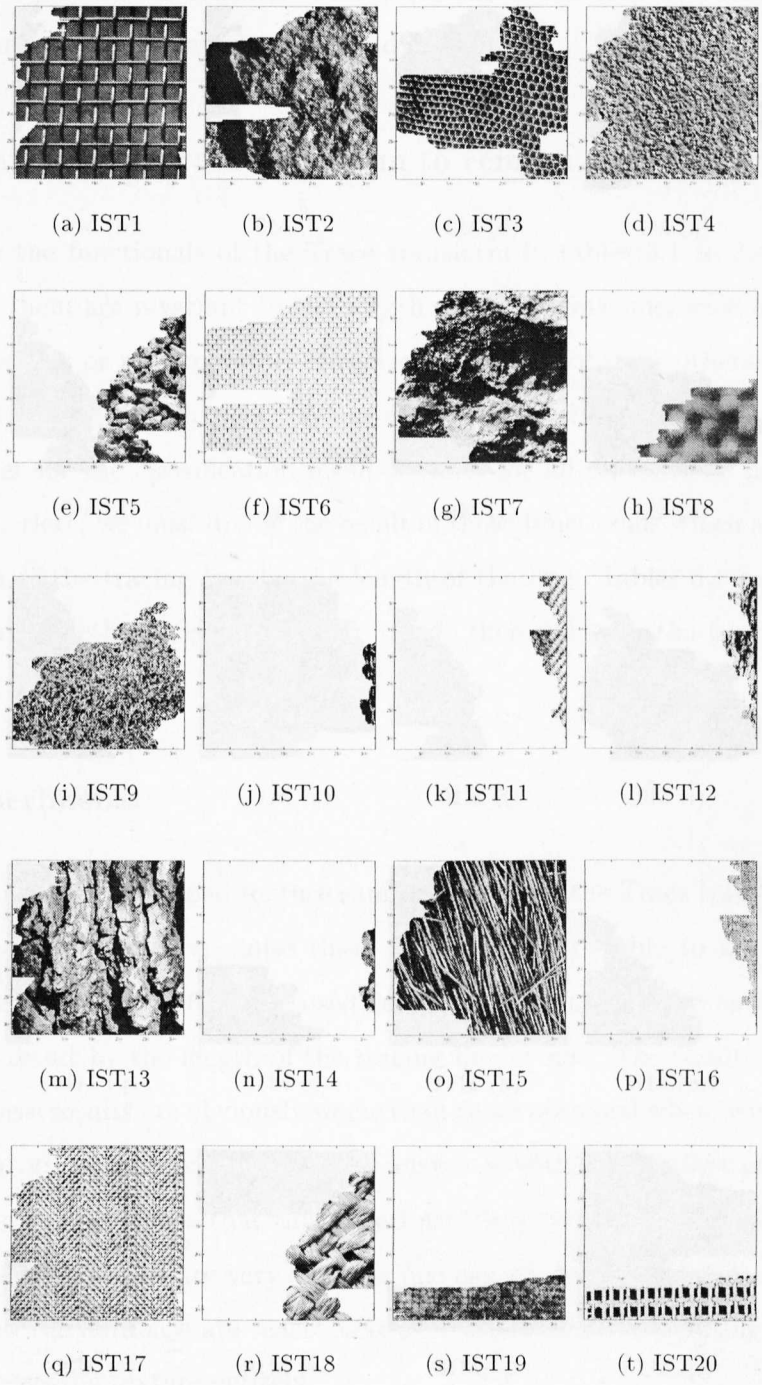
(e) M5 (f) M6 (g) M7 (h) M8

(i) M9 (j) M10 (k) M11 (l) M12

(m) M13 (n) M14 (o) M15 (p) M16

(q) M17 (r) M18 (s) M19 (t) M20

Figure 6.1: 20 masks used for converting the Brodatz album to irregularly shaped texture samples.



(a) IST1 (b) IST2 (c) IST3 (d) IST4

(e) IST5 (f) IST6 (g) IST7 (h) IST8

(i) IST9 (j) IST10 (k) IST11 (l) IST12

(m) IST13 (n) IST14 (o) IST15 (p) IST16

(q) IST17 (r) IST18 (s) IST19 (t) IST20

Figure 6.2: The first 20 textures from the Brodatz album after having been converted to irregularly shaped texture samples, by using masks from table 6.1 randomly chosen.

Number	Functional	Divide
1	$\sum_{i=0}^{N} x_i$	yes
2	$\sum_{i=0}^{N} i x_i$	yes
3	$\sqrt{\frac{\sum_{i=0}^{N}(x_i-m)^2}{N}}$	no
4	$\sqrt{\sum_{i=0}^{N} x_i^2}$	yes
5	$Max_{i=0}^{N} x_i$	no
6	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$	yes
7	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert^2$	yes
8	$\sum_{i=3}^{N-3} \lvert x_{i-3} + x_{i-2} + x_{i-1} - x_{i+1} - x_{i+2} - x_{i+3} \rvert$	yes
9	$\sum_{i=0}^{N-2} \lvert x_{i-2} + x_{i-1} - x_{i+1} - x_{i+2} \rvert$	yes
10	$\sum_{i=4}^{N-4} \lvert x_{i-4} + x_{i-3} + ... + x_{i-1} - x_{i+1} - ... - x_{i+3} - x_{i+4} \rvert$	yes
11	$\sum_{i=5}^{N-5} \lvert x_{i-5} + x_{i-4} + ... + x_{i-1} - x_{i+1} - ... - x_{i+4} - x_{i+5} \rvert$	yes
12	$\sum_{i=6}^{N-6} \lvert x_{i-6} + x_{i-5} + ... + x_{i-1} - x_{i+1} - ... - x_{i+5} - x_{i+6} \rvert$	yes
13	$\sum_{i=7}^{N-7} \lvert x_{i-7} + x_{i-6} + ... + x_{i-1} - x_{i+1} - ... - x_{i+6} - x_{i+7} \rvert$	yes
14	$\sum_{i=4}^{N-4} \sum_{k=0}^{4} \lvert x_{i-k} - x_{i+k} \rvert$	yes
15	$\sum_{i=5}^{N-5} \sum_{k=0}^{5} \lvert x_{i-k} - x_{i+k} \rvert$	yes
16	$\sum_{i=6}^{N-6} \sum_{k=0}^{6} \lvert x_{i-k} - x_{i+k} \rvert$	yes
17	$\sum_{i=7}^{N-7} \sum_{k=0}^{7} \lvert x_{i-k} - x_{i+k} \rvert$	yes
18	$\sum_{i=10}^{N-10} \sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert$	yes
19	$\sum_{i=15}^{N-15} \sum_{k=0}^{15} \lvert x_{i-k} - x_{i+k} \rvert$	yes
20	$\sum_{i=20}^{N-20} \sum_{k=0}^{20} \lvert x_{i-k} - x_{i+k} \rvert$	yes
21	$\sum_{i=25}^{N-25} \sum_{k=0}^{25} \lvert x_{i-k} - x_{i+k} \rvert$	yes
22	$\sum_{i=10}^{N-10}((1 + \sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert)/(1 + \sum_{k=-10}^{9} \lvert x_{i-k} - x_{i+k} \rvert))$	no
23	$\sum_{i=10}^{N-10} \sqrt{(\sum_{k=0}^{10} \lvert x_{i-k} - x_{i+k} \rvert)^2/(1 + \sum_{k=-10}^{9} \lvert x_{i-k} - x_{i+k} \rvert)}$	no

Table 6.1: The trace functionals T used in the experiments. N is the total number of points along the tracing line and x_i is the ith sample along the line. The entry under column "Divide" gives the answer to the question "Shall we divide the result of that functional by the length of the tracing line or not?" Table continued on the next page.

Number	Functional	Divide
24	$\sum_{i=0}^{N-2} \lvert x_i - 2x_{i+1} + x_{i+2} \rvert$	yes
25	$\sum_{i=0}^{N-3} \lvert x_i - 3x_{i+1} + 3x_{i+2} - x_{i+3} \rvert$	yes
26	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert$	yes
27	$\sum_{i=0}^{N-5} \lvert x_i - 5x_{i+1} + 10x_{i+2} - 10x_{i+3} + 5x_{i+4} - x_{i+5} \rvert$	yes
28	$\sum_{i=0}^{N-2} \lvert x_i - 2x_{i+1} + x_{i+2} \rvert x_{i+1}$	yes
29	$\sum_{i=0}^{N-3} \lvert x_i - 3x_{i+1} + 3x_{i+2} - x_{i+3} \rvert x_{i+1}$	yes
30	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert x_{i+2}$	yes
31	$\sum_{i=0}^{N-5} \lvert x_i - 5x_{i+1} + 10x_{i+2} - 10x_{i+3} + 5x_{i+4} - x_{i+5} \rvert x_{i+2}$	yes

Table 6.2: The trace functionals T used in the experiments. N is the total number of points along the tracing line and x_i is the ith sample along the line. The entry under column "Divide" gives the answer to the question "Shall we divide the result of that functional by the length of the tracing line or not?"

Number	Functional	Divide
1	$Max_{i=0}^{N} x_i$	no
2	$Min_{i=0}^{N} x_i$	no
3	$\sqrt{\sum_{i=0}^{N} x_i^2}$	yes
4	$\frac{\sum_{i=0}^{N} i x_i}{\sum_{i=0}^{N} x_i}$	no
5	$\sum_{i=0}^{N} i x_i$	yes
6	$\frac{1}{N} \sum_{i=0}^{N} (x_i - \hat{x})^2$	no
7	c so that: $\sum_{i=0}^{c} x_i = \sum_{i=c}^{N} x_i$	no
8	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$	yes
9	c so that: $\sum_{i=0}^{c} \lvert x_{i+1} - x_i \rvert = \sum_{i=c}^{N-1} \lvert x_{i+1} - x_i \rvert$	no
10	$\sum_{i=0}^{N-4} \lvert x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4} \rvert$	yes

Table 6.3: The diametric functionals P used in the experiments. The entry under column "Divide" gives the answer to the question "Shall we divide the result of that functional by the length of the column of the Trace transform or not?"

Number	Functional	Divide
1	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert^2$	yes
2	$\sum_{i=0}^{N-1} \lvert x_{i+1} - x_i \rvert$	yes
3	$\sqrt{\sum_{i=0}^{N} x_i^2}$	yes
4	$\sum_{i=0}^{N} x_i$	yes
5	$Max_{i=0}^{N} x_i$	no
6	$Max_{i=0}^{N} x_i$- $Min_{i=0}^{N} x_i$	no
7	i so that $x_i = Min_{i=0}^{N} x_i$	no
8	i so that $x_i = Max_{i=0}^{N} x_i$	no
9	i so that $x_i = Min_{i=0}^{N} x_i$ without first harmonic	no
10	i so that $x_i = Max_{i=0}^{N} x_i$ without first harmonic	no
11	Amplitude of the first harmonic	no
12	Phase of the first harmonic	no
13	Amplitude of the second harmonic	no
14	Phase of the second harmonic	no
15	Amplitude of the third harmonic	no
16	Phase of the third harmonic	no
17	Amplitude of the fourth harmonic	no
18	Phase of the fourth harmonic	no

Table 6.4: The circus functionals Φ used in the experiments. The entry under column "Divide" gives the answer to the question "Shall we divide the result of that functional by the length of the circus function or not?"

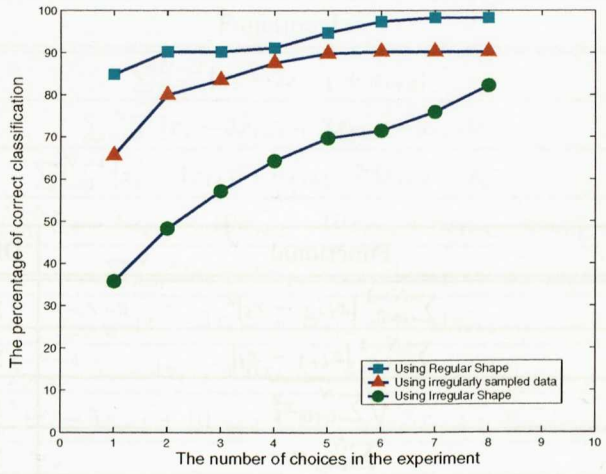


Figure 6.3: This figure shows that by increasing the number of choices, the percentage of correct classification increases as well. After selecting the first 8 choices the classification accuracy is 82%, as opposed to 90% for irregularly sampled data and 98% for regularly shaped and regularly sampled data.

6.2 An application to medical image characterisation

6.2.1 Introduction

Having the ability to recognise textures combined with the ability to isolate textured regions, will result to a method which allows one to classify objects in complex images. A kind of such complex images are some medical images. We are using 97 pathology images showing breast cancer and want to classify them in three classes, according to the severity of the condition. The management of the cancer patient depends on a point system used. The pathologist examines various images of the cancerous tissue and according to their appearance assigns points to the patient. One of the characteristics the pathologist looks for is the appearance of the cells. The cancerous cells tend to be diversified in shape and tend to form dense clusters. The phenomenon is called pleomorphism and from the image processing point of view it manifests itself as a change in texture. The trouble is that this change in texture does not occupy a whole

image, but only segments in it. So, before we attempt to characterise the degree of pleomorphism, we must first isolate the texture regions of the image. This is not a problem of texture segmentation in the usual sense of the word, since we are not trying to segment the different textures present in the image. All we are interested in here is to isolate the regions of fat (which appear uniformly coloured in the image) from the cell-filled regions. Once the cell-filled regions have been identified, then we may apply one of the methods we studied to characterise the texture and classify the image in one of the 3 classes pathologists use: those which for pleomorphism are assigned 1 point (mildest form of cancer), those which are assigned 2 points (average severity of the condition) and those which are assigned 3 points (most advanced form of cancer). So, as a first step we need to use an algorithm which separates the textured from the non-textured regions irrespective of the type of texture of the textured regions. Figure 6.4 shows one of these 97 medical images which belongs to class 1, (1 point). All images in our database have been assigned points by expert pathologists. So, the scheme we

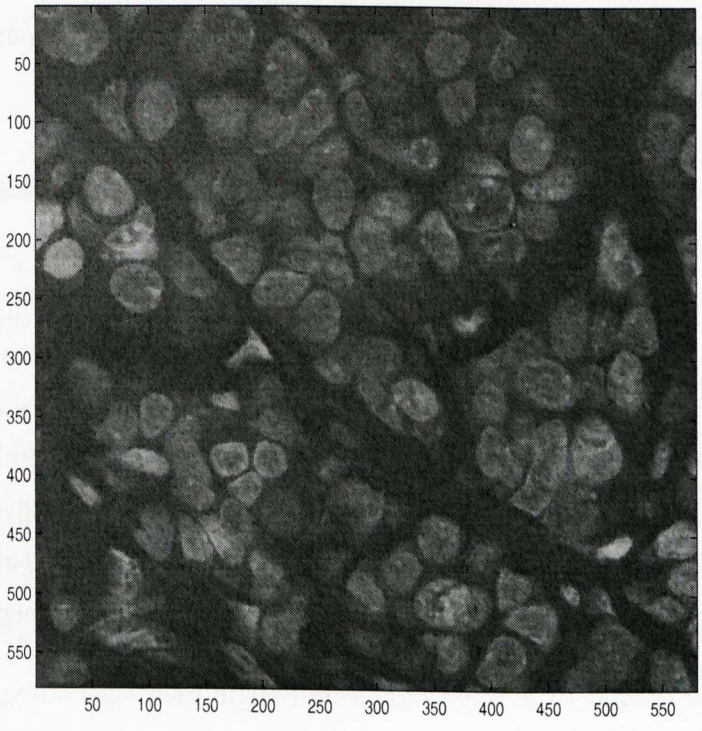


Figure 6.4: This figure shows one of the 97 medical images which are used in our experiments. This image belongs to class 1.

propose to use is:

- Identify in each medical image the largest textured region.
- Extract a training set of patterns for each category of image.
- Classify test textures using nearest neighbour and a vote counting system.

Next we present the simple method we use to identify the most significant textured region in each image. Then we present our classification experiments, and finally our conclusions.

6.2.2 Segmenting the images

We start by observing that textured regions are characterised by high density of edge pixels (edgels). By repeatedly then dilating the edgels, we expect the textured regions to form sooner or later large connected islands of pixels. So, to segment the images we perform the following steps:

- Using the Sobel operator we find the edgels of the image. We flag them as black pixels.
- Using mathematical morphology we dilate the edgels with a structuring element of size 3×3.
- After every dilation we perform connected component analysis of the black pixels and identify the connected component with the largest number of pixels. If the number of its pixels is above a certain threshold T_1, we go to the next step. Otherwise, we keep dilating the black pixels, until a major connected component made up from dilated edgels emerges.
- When we stop the dilation of the black components, we perform connected component analysis of the white components. Any white connected component which is fully surrounded by black pixels (i.e. does not touch the border of the image) and it has fewer pixels than a threshold T_2, is filled in with black pixels.

-1	0	1
-2	0	2
-1	0	1

(a)

-1	-2	-1
0	0	0
1	2	1

(b)

Figure 6.5: (a) Sobel mask for enhancing the vertical edges. (b) Sobel mask for enhancing the horizontal edges.

Next we describe each step in detail and show the output of each step.
We use the Sobel masks shown in figure 6.5 to estimate the horizontal and the vertical components of the gradient vector of each pixel. Let us call them δI_x and δI_y respectively.

The magnitude of the gradient vector is then:

$$M = \sqrt{\delta I_x^2 + \delta I_y^2}; \tag{6.1}$$

Then, we threshold the values of M by using a threshold T_0 and mark with 0 all pixels with values $M \geq T_0$ and with 255 (white) all pixels with values $M \leq T_0$. Threshold T_0 is chosen by the Otsu method [64]. As we do not perform non-maxima suppression, the edge strings we produce are thick, but this is not a problem. In fact it is desirable for the purpose we require the edgels. Figure 6.6 shows the histogram of the M values for the image of figure 6.4 and it marks with an arrow the position of the threshold identified by the Otsu method. Figure 6.7 shows the interclass difference versus threshold value, used by the Otsu method to choose the threshold. Figure 6.8 shows the edge map produced for the image of figure 6.4.

Figure 6.9 shows the edge map dilated once, and figure 6.10 shows the final edge map obtained after 5 dilations, using as value of threshold $T_1 = 33640$ which represents 10% of the image pixels. Figure 6.11 presents the filled in most significant black connected component using threshold $T_2 = 7436$ which was chosen to represent 10% of the number of pixels identified in the largest black connected component. Finally, figure 6.12

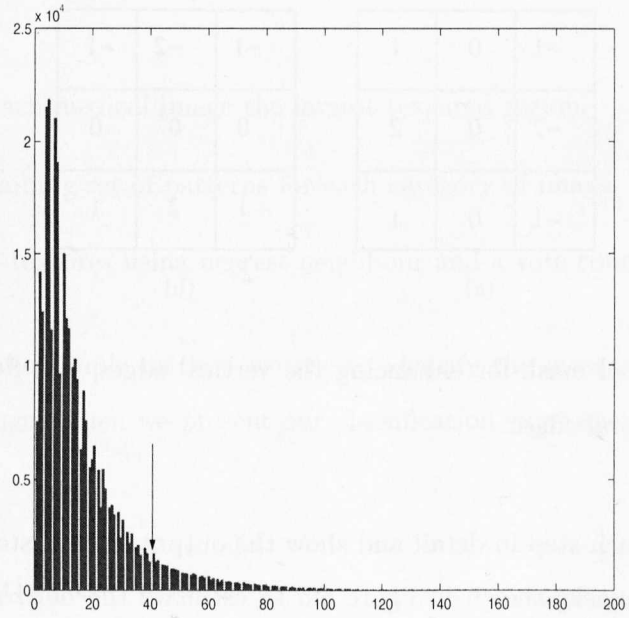


Figure 6.6: This figure shows the histogram of the M values for the image of figure 6.4. The arrow marks the position of the threshold used.

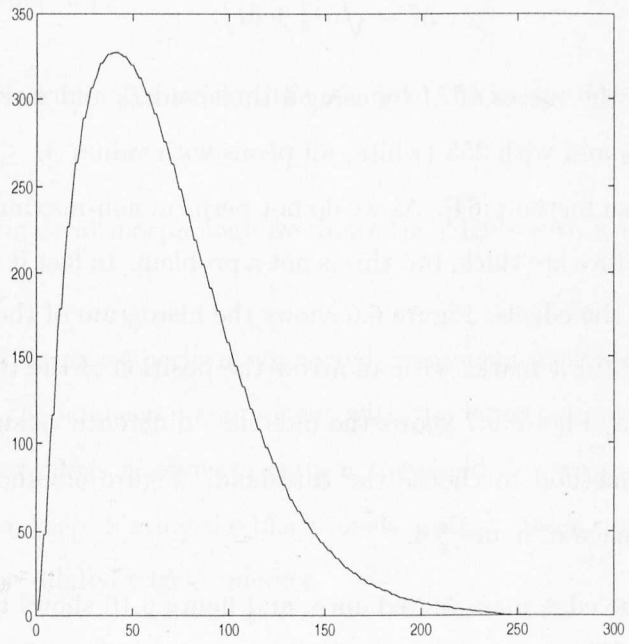


Figure 6.7: The interclass difference versus threshold value. According to Otsu's method we choose the threshold that maximises the interclass difference.

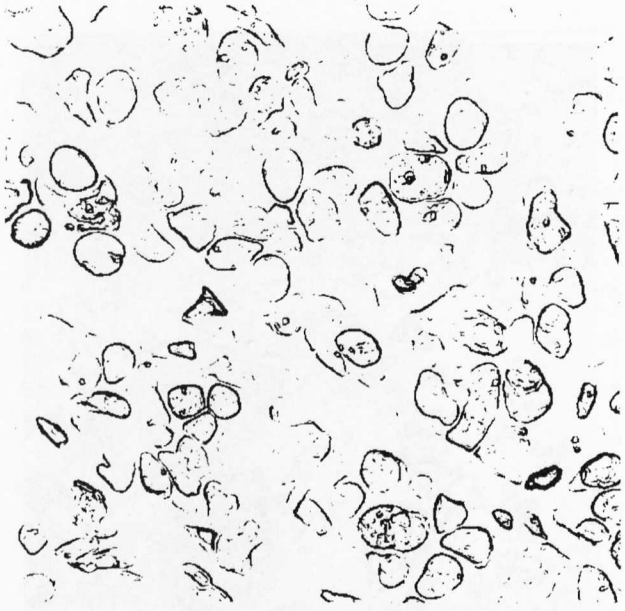


Figure 6.8: This figure shows the edge map produced for the image of figure 6.4.

presents the identified textured region superimposed on the original image.

Figures 6.13 to 6.16 show other original medical images with the border of their largest segmented textured regions, and the largest segmented textured regions separately. Two options are considered: use of 4 or 8 connectivity when performing connected component analysis. All segments for the 97 medical images were obtained fully automatically, namely using the Otsu method for the edge strength threshold and threshold T_1 and T_2 being 10% of the total number of pixels on the image and 10% of the total number of pixels in the largest connected component respectively. In the next section we use the Trace transform on these segments treating them as irregularly shaped textured regions.

6.2.3 Experiments

After identifying the largest area of cells in each pathology image, we use the Trace transform to extract features. Here, we have three classes. We use 10 images from each class for training. The test set includes 27 images from class 1, 30 images from class 2 and 10 images from class 3. Also these experiments were performed for two cases:

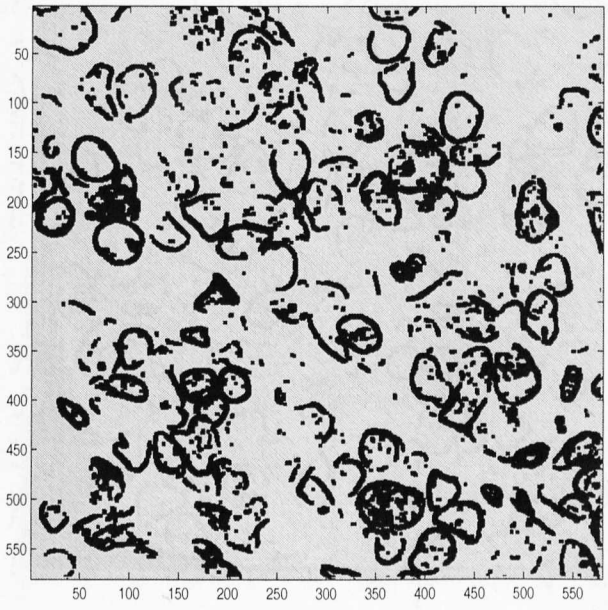


Figure 6.9: This figure shows the edge map of figure 6.4 dilated once.

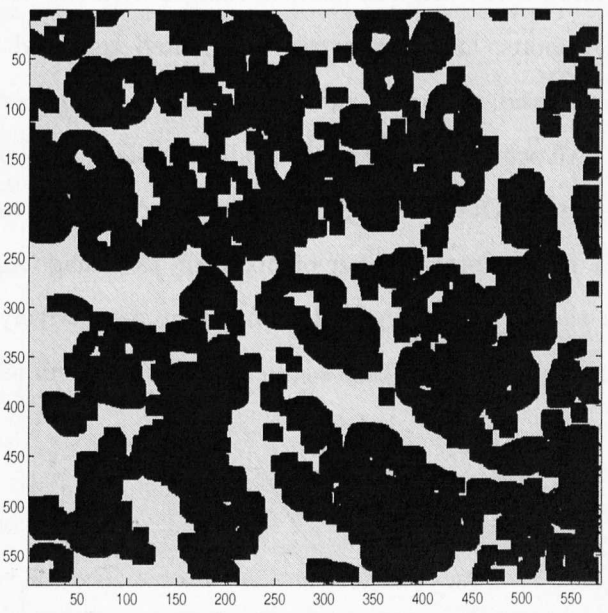


Figure 6.10: This figure shows the final edge map obtained after 5 dilations, using as value of threshold $T_1 = 33640$ which represents 10% of the image pixels.

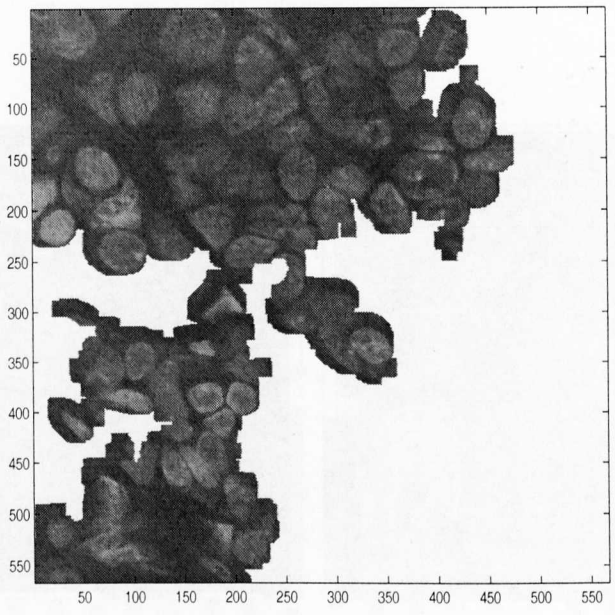


Figure 6.11: The largest segment of textured region which has high enough number of pixels to allow us characterise its texture.

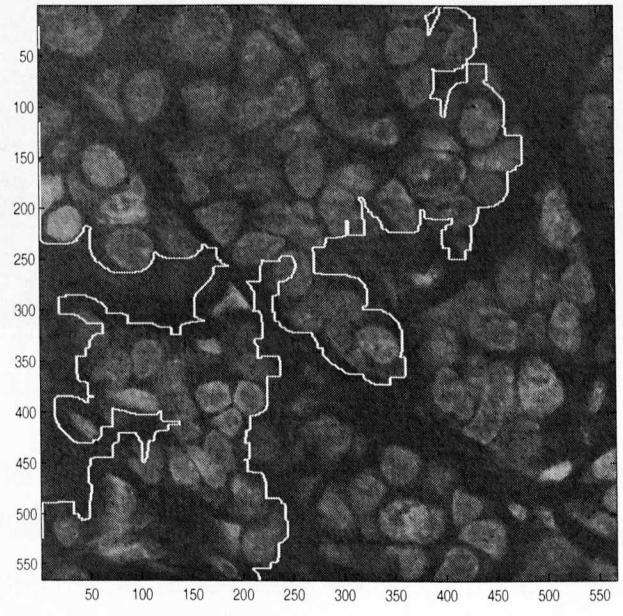


Figure 6.12: This figure presents the identified textured region superimposed on the original image.

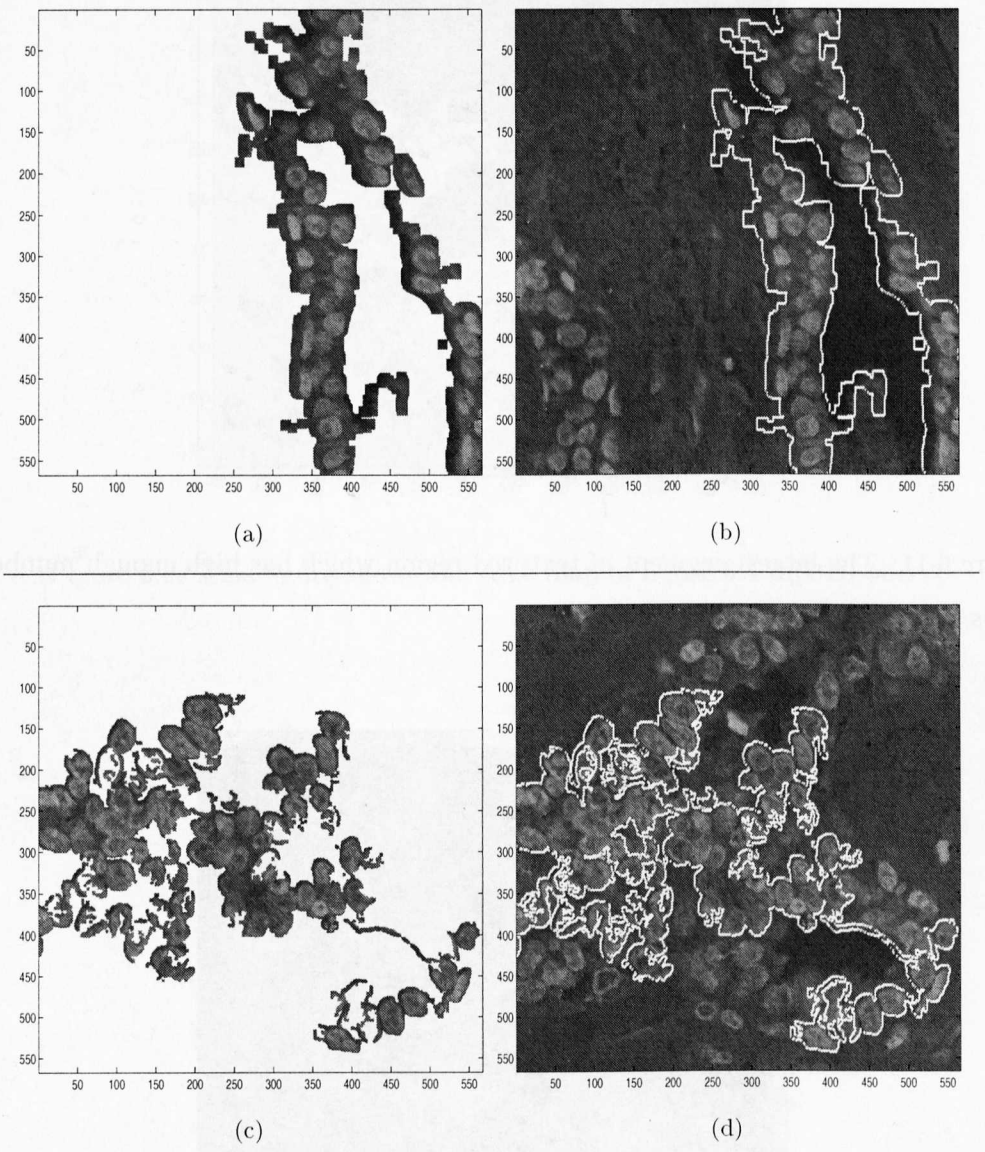


Figure 6.13: The largest segmented textured regions for images 4 and 5 on the left, and the borders of the segmented regions superimposed on the original images, on the right. These masks were constructed using 8-connectivity in the connected component analysis of the algorithm.

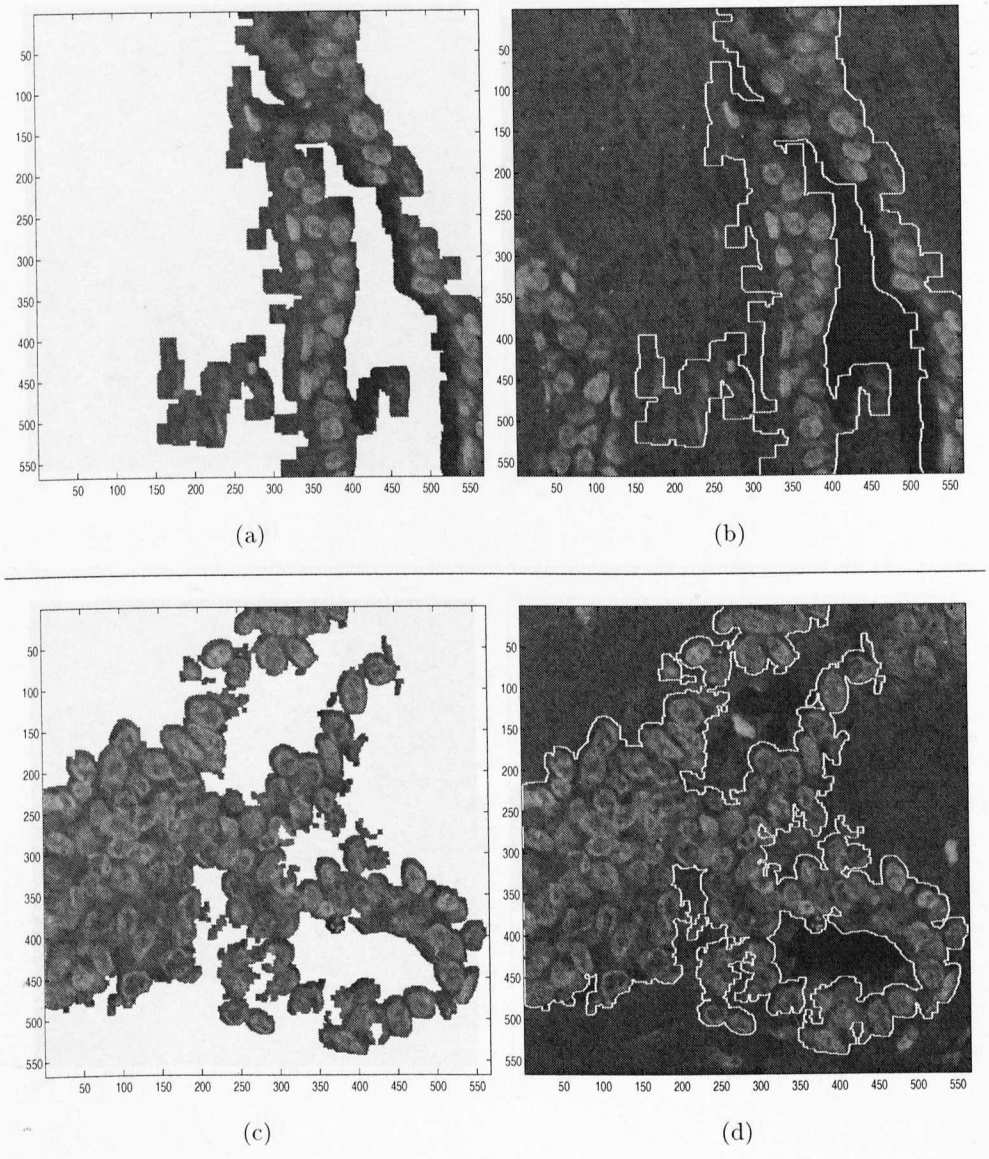


Figure 6.14: The largest segmented textured regions for images 4 and 5 on the left, and the borders of the segmented regions superimposed on the original images, on the right. These masks were constructed using 4-connectivity in the connected component analysis of the algorithm.

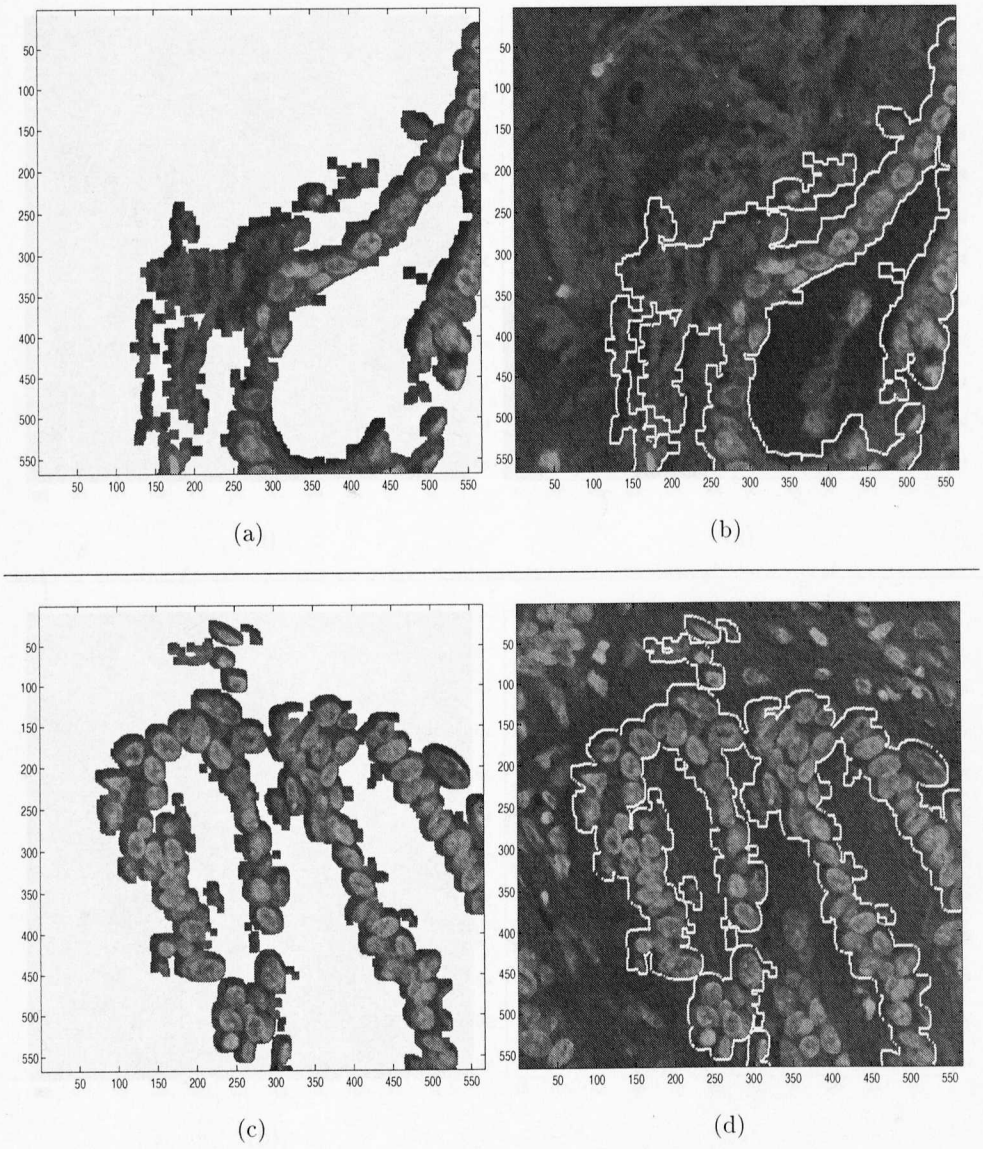


Figure 6.15: The largest segmented textured regions for images 6 and 7 on the left, and the borders of the segmented regions superimposed on the original images, on the right. These masks were constructed using 8-connectivity in the connected component analysis of the algorithm.

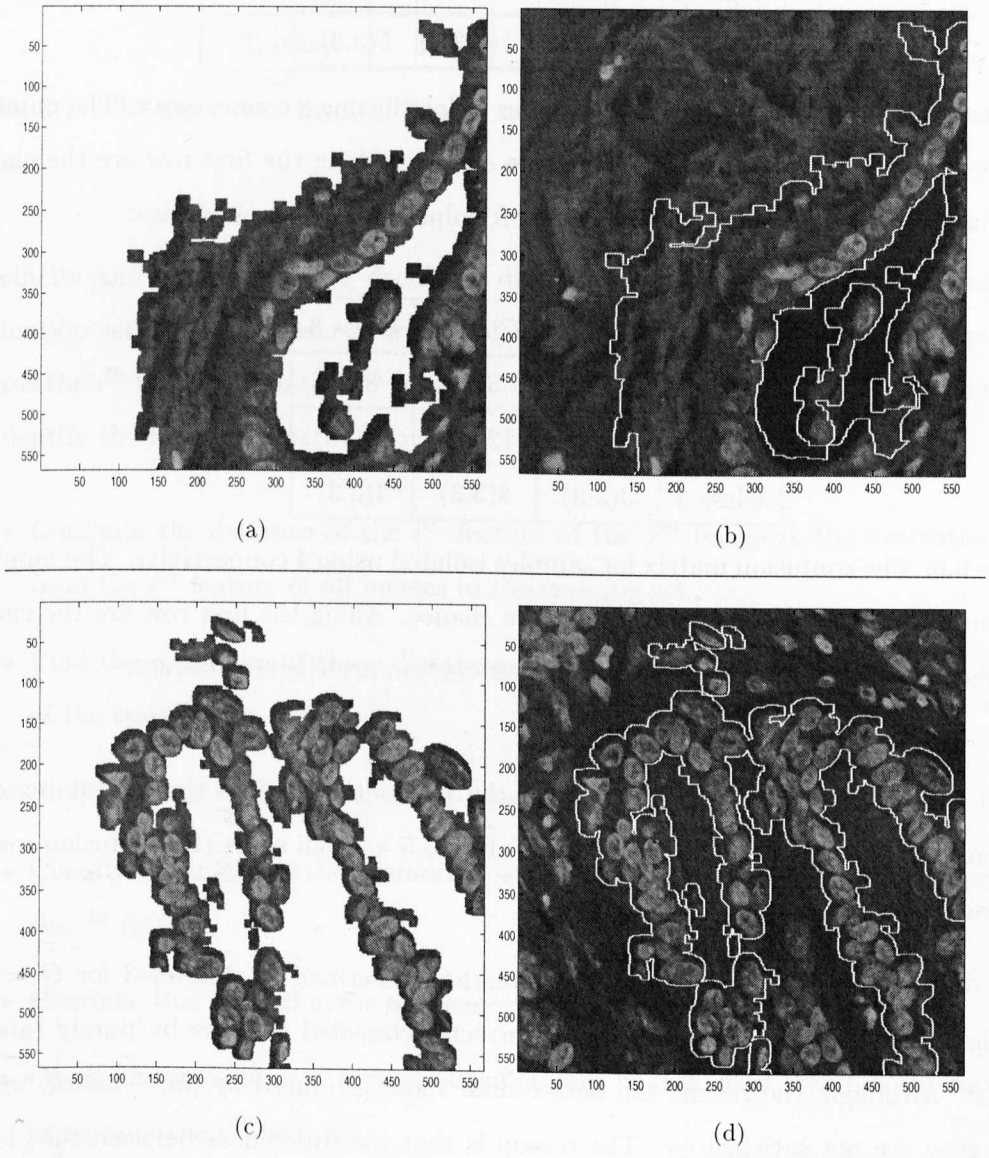


Figure 6.16: The largest segmented textured regions for images 6 and 7 on the left, and the borders of the segmented regions superimposed on the original images, on the right. These masks were constructed using 4-connectivity in the connected component analysis of the algorithm.

	Class 1	Class 2	Class 3
Class 1	11(9)	6(9)	10(9)
Class 2	13(10)	6(10)	11(10)
Class 3	4(3.3)	1(3.3)	5(3.3)

Table 6.5: The confusion matrix for samples isolated using 8 connectivity. The numbers inside brackets are those expected by pure chance. Along the first row are the classes produced by the algorithm while on the left column are the true classes.

	Class 1	Class 2	Class 3
Class 1	13(9)	6(9)	8(9)
Class 2	5(10)	12(10)	13(10)
Class 3	3(3.3)	3(3.3)	4(3.3)

Table 6.6: The confusion matrix for samples isolated using 4 connectivity. The numbers inside brackets are those expected by pure chance. Along the first row are the classes produced by the algorithm while on the left column are the true classes.

when we use 4-connectivity to construct the masks that isolate the largest textured region and when we use 8-connectivity. Tables 6.5 and 6.6 show the confusion matrix for each case.

The results were obtained by using the method described in chapter 3 for $Q = 2.5$. Inside brackets we give the number of correctly classified samples by purely random choice. Although, the results are better than those produced by pure chance, we see that they are not satisfactory. The reason is that the differences between the classes here are very subtle and we have only 3 classes. Next we shall see whether we may improve the accuracy of classification by using a more careful feature selection method.

6.2.4 A more sophisticated feature selection method

Trace transform allows us to have thousands of features to characterise a texture. We shall try now to choose from among all those features the ones which show maximum

	Class 1	Class 2	Class 3
Training	10	10	6
Evaluation	10	10	6
Testing	17	20	8

Table 6.7: The number of medical images in each class and each set are shown in this table.

specificity and sensitivity. We decide to divide our image in 3 sets: a training set, evaluation set and a test set. Our protocol is shown in table 6.7. Suppose that $f^s_{i,j}$ means the i^{th} feature from the j^{th} image in the s^{th} set. We use the following algorithm to identify the best features.

- Compute the distance of the i^{th} feature of the j^{th} image in the evaluation set, from the i^{th} feature of all images in the training set.
- Find the minimum of those distances. Suppose it corresponds to the k^{th} image of the training set.
- Find the class of that image. (k^{th} image)
- Classify the image in the evaluation set(j^{th} image), to that class with respect to the i^{th} feature.
- Continue this algorithm for all images in the evaluation set.
- Compute the number of correct classifications in each class, which is the result of using only the i^{th} feature.
- Continue this algorithm for all features.
- Sort the features regarding to the number of correct classifications in all three classes counted together.
- Use the n number of the best features to classify the evaluation set. This number is like threshold and can be obtained by seeing the results of various value of it(figure 6.17).

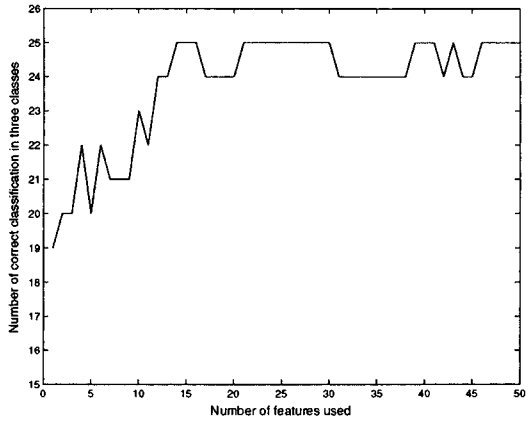


Figure 6.17: We can find the n, number of features with the best performance, by looking at this figure.

	Class 1	Class 2	Class 3
Class 1	10(3.3)	0(3.3)	0(3.3)
Class 2	0(3.3)	10(3.3)	1(3.3)
Class 3	0(2)	0(2)	5(2)

Table 6.8: The confusion matrix for the evaluation set. The numbers inside the brackets are those expected by pure chance. Along the first row are the classes produced by the algorithm while on the left column are the true classes.

- Use these n best features to classify the test set.

It is important to say that, because of the special shape and content of these pathology images, and because of the distribution of one feature along the images in each class not being normal distribution, Bhattacharya distance [82], Minkowski distance [40] and Mahalanobis distance are not suitable similarity measures between images. We use here the absolute value of the difference of two corresponding features. The result for the evaluation set is shown in table 6.8. The number of features we used is 25 and it has been obtained from figure 6.17. In this figure we show that we obtained correct classification for several numbers of the best features.

To find the confusion matrix for the case of pure chance, we work as follows: the total number of test images is 46. If we classify them randomly, we shall put one third of 46 in each class. But when we put 15.3 of images in class 1, how many of these images are really in class 1, and how many are in class 2 and how many are in class 3? We can find it by using this equation:

$$N_{c_i,c_j} = \frac{N_t}{3} \times \frac{N_{c_i}}{N_t} \tag{6.2}$$

where, N_{c_i,c_j} means the number of images classified in class j, and really is in class i. N_t is the total number of images in a set(evaluation or other set), N_{c_i} is the number of images really in class i. So for $N_{1,2}$ in the evaluation set, we have:

$$N_{1,2} = \frac{26}{3} \times \frac{10}{26} = 3.3 \tag{6.3}$$

As we can see, we have very good confusion matrix and high accuracy by using 25 of the first best features obtained by the above mentioned algorithm. What are these features and what is the result of using them for classifying the test set? Table 6.9 shows the best features and 6.10 shows the results of using them to classify test set. Although we have excellent correct classification of the evaluation set, the test set results are not acceptable. If we look at tables 6.11 to 6.13, which show for each of the test images how many features placed it in class 1, how many features classified it in class 2 and how many features classified it in class 3, we can see easily that most of features favour class 1. This means that images of class 3, or class 2, have a lot of characteristics of the images of class 1 and that is why we have more images classified in class 1.

6.3 Conclusions

We used 112 irregularly shaped texture samples and then the Trace transform method to recognise these textures. The results are obviously worse than those obtained when using full images on rectangular grids, however, the correct answer is within the 8 first choices 86% of the time. One should realise that this experiment is a very severe test of the method as some of the masks used are very small as one can see from figure 6.1 and several of the textures in the database are macro textures, which means that using

Feature	Trace Func.	Diam Func.	Circ Func.
1	15	5	17
2	13	7	18
3	12	7	18
4	3	3	4
5	18	9	4
6	18	8	15
7	16	7	18
8	14	9	2
9	14	9	1
10	13	2	4
11	9	7	5
12	9	3	2
13	8	2	4
14	7	2	3
15	5	10	5
16	3	3	3
17	3	2	4
18	31	9	5
19	31	3	4
20	30	3	6
21	29	3	6
22	24	5	2
23	17	8	2
24	17	2	4
25	15	10	13

Table 6.9: The best features among all 5580 features computed, which classify the images more accurately.

	Class 1	Class 2	Class 3
Class 1	13(5.7)	15(5.7)	6(5.7)
Class 2	4(6.6)	5(6.6)	2(6.6)
Class 3	0(2.6)	0(2.6)	0(2.6)

Table 6.10: The confusion matrix for the test set. The numbers inside the brackets are those expected by pure chance. Along the first row are the classes produced by the algorithm while on the left column are the true classes.

Image	Votes 1	Votes 2	Votes 3	Final
21	18	4	3	1
22	17	4	4	1
23	16	8	1	1
24	12	13	0	2
25	11	8	6	1
26	13	10	2	1
27	17	5	3	1
28	15	7	3	1
29	10	9	6	1
30	14	7	4	1
31	9	13	3	2
32	7	14	4	2
33	11	10	4	1
34	12	7	6	1
35	16	7	2	1
36	6	14	5	2
37	10	9	6	1

Table 6.11: For each of the test images, we can see how many features placed it in class 1, how many features classified it in class 2 and how many features classified it in class 3. All these images really belong to class 1.

Image	Votes 1	Votes 2	Votes 3	Final
58	12	10	3	1
59	13	11	1	1
60	10	11	4	2
61	13	11	1	1
62	14	8	3	1
63	12	12	1	2
64	20	3	2	1
65	14	7	4	1
66	17	5	3	1
67	13	6	6	1
68	9	10	6	2
69	13	6	6	1
70	16	4	5	1
71	10	11	4	2
72	11	5	9	1
73	10	9	6	1
74	11	13	1	2
75	15	8	2	1
76	13	9	3	1
77	12	10	3	1

Table 6.12: For each of the test images, we can see how many features placed it in class 1, how many features classified it in class 2 and how many features classified it in class 3. All these images really belong to class 2.

Image	Votes 1	Votes 2	Votes 3	Final
90	2	15	8	2
91	16	4	5	1
92	11	11	3	2
93	17	7	1	1
94	10	9	6	1
95	19	5	1	1
96	16	5	4	1
97	16	7	2	1

Table 6.13: For each of the test images, we can see how many features placed it in class 1, how many features classified it in class 2 and how many features classified it in class 3. All these images really belong to class 3.

such a mask on them destroys the texture entirely.

The fact that the resultant curves in figure 6.3 converge as the rank of accepted choices increases, shows that the Trace transform method degrades gracefully i.e. when the method fails to recognise the correct textures in the first position of choice, it does not place it too far away from it.

Then we tried to classify some pathology images by segmenting them and classifying these segments. The segmentation is good enough as we can see in some examples in figure 6.13 and 6.15. However, the classification results were very disappointing. In particular, the selected features showed very bad generalisation ability. One may argue that the features were chosen to over-fit the data (since they produced no error in the evaluation set), and that if we had chosen features that allowed some error in the evaluation set, better results would have been obtained from the test set as well. However, in medical application one can not allow error at all, otherwise the automatic system is not acceptable either by the clinicians or the patient. It was necessary to require 0% classification error. One may conclude from this experiments that the problem of these particular pathology images is not one of texture classification, but rather of object recognition. The classes we were trying to identify were defined in terms

of changed shape of individual cells. It is possible that human experts classify these images by concentrating on assessing the shapes of individual cells rather than assessing the overall appearance of the cell-filled regions as a whole. We believe, therefore, that a totally different approach is needed for the particular medical image processing problem, an approach that is not based on texture classification. One should apply a sophisticated technique to extract individual cells. Then each cell may be characterised by using the Trace transform, treating it as an individual object. This approach has not been followed.

Chapter 7

Conclusions and future work

7.1 Particular conclusions

- We can see that by increasing the number of features the results improve. This proves our first idea that using thousands of features may help us classify textures more accurately. These features do not need to make sense to the human conscious perception, and therefore their number can be very large. The relevance of these features to the task we wish to solve can be assessed in a training phase, and then these features can be combined with their appropriate weights to form a similarity measure between two images.

- The proposed method tested with all textures in the Brodatz album was shown to be much more powerful than the commonly used method based on co-occurrence matrix features, when we use them for perceptual grouping.

- The Trace transform method does not have to be trained with representations of all textures we wish to identify. As we saw in the experiments performed with 30 training textures different from those in the test set, even texture classes that were not represented in the training set used to decide the relative importance of the features, could be classified correctly. The results were only slightly worse than the results obtained with using all 112 textures for training.

- The use of perceptually meaningful features on their own did not produce good

results. The supplement of co-occurrence matrix features with perceptually meaningful features did not change the result, which was already very good when the full co-occurrence matrix was used. So, one does not need perceptually meaningful features (i.e. features which have linguistic names) in order to classify textures. Simply, one needs *many* features. If then one wishes to classify textures in a perceptually meaningful way, i.e. in a way that imitates the human ranking of textures in terms of similarity, then one may select from these many features those which rank the textures as the humans do. We demonstrated that for this task features computed from the Trace transform are the most appropriate (see figures 4.8 and 4.9).

- Identifying which features were the most useful in the perceptual grouping allows us to reverse engineer to some extend the human vision system and single out some features which may be used in the *subconscious* level to analyse textures. These are features which do not have linguistic terms to describe them. It is interesting to note that from the trace functionals, the functional which contributed to the construction of the most useful features is a simple differentiator, and from the diametric functionals, the functional which contributed to the construction of the most useful feature is taking the minimum. From the circus functionals the most useful functional is functional 1 which again is based on a simple differentiator.

- We demonstrated that by using the Trace transform method we may recognise textures from irregularly sampled images. The Hough transform was used as an interface that allowed us to identify tracing lines in the image and normalised convolution allowed us to deal with the irregularly placed samples along the tracing lines in order to compute the trace functionals.

- It is clear that when we increase the number of points in the Gaussian mask, and the number of lines used to trace the image, our results become better. The reason is that, by increasing the points in the mask, and the number of tracing lines, we make this method more and more similar to the method when applied to regularly sampled data.

- Overall the best results were produced by the Gaussian sampling pattern. This

is because this pattern had its points more uniformly spread all over the image than either the log polar or the retinal pattern. However, none of the sampling patterns was what it was supposed to be. This is because in our experiments we are limited by the bandwidth of the already digitised images. The sampling patterns are supposed to be used to sample an analogue scene.

- In addition, the human retina has 4,500,000 sampling points, as opposed to a few thousands we used here. So, given that we used less than 20% of the pixels in the original regularly sampled images, and we chose them at random positions, something that goes against the fundamental property of texture being a spatial property, the produced results are remarkably good in comparison with the results obtained by using regular grids.

- Using the trace transform after normalising each functional by dividing the result by its length, is appropriate for working with for irregularly shaped textured regions.

7.2 Overall conclusions and future work

- The Trace transform method is useful method in texture classification, especially for the case that it is acceptable to have a few answers instead of insisting on only the first answer being the correct one.

- In huge databases, we may find the Trace transform too slow. However, if we do not need to have features invariant to shifting, like for example in a face recognition problem with registered faces, we can use only one or a few number of points to create a trace matrix, i.e. we may use tracing lines all of which pass through the same point in the image. This way we may save the time.

- One of the advantage of the Trace transform in comparison with other methods is that, we may define new functionals in the Trace transform to produce new features. In the other methods, we discussed in this thesis, if we increase the number of features by increasing the number of grey levels (e.g. CM256), we do

not involve features that capture different aspects of the image, but only more features of the same nature as those we already have.

- The logic of the Trace transform, can be extended for 3D textures as well. This is because by using all possible lines crossing a point, we may have the same results as if we rotate that texture with any rotation. Invariance to shifting and scaling can be guaranteed in the same way. Also, we may use one point, or a few points to save CPU time if we are not interested in invariance to shifting.

- For future work, we may use circles (for square textures) or ellipses (for rectangular textures) instead of tracing lines to compute functionals. In other words, instead of computing image characteristics along tracing lines, we may compute them along any other type of curve we choose to use.

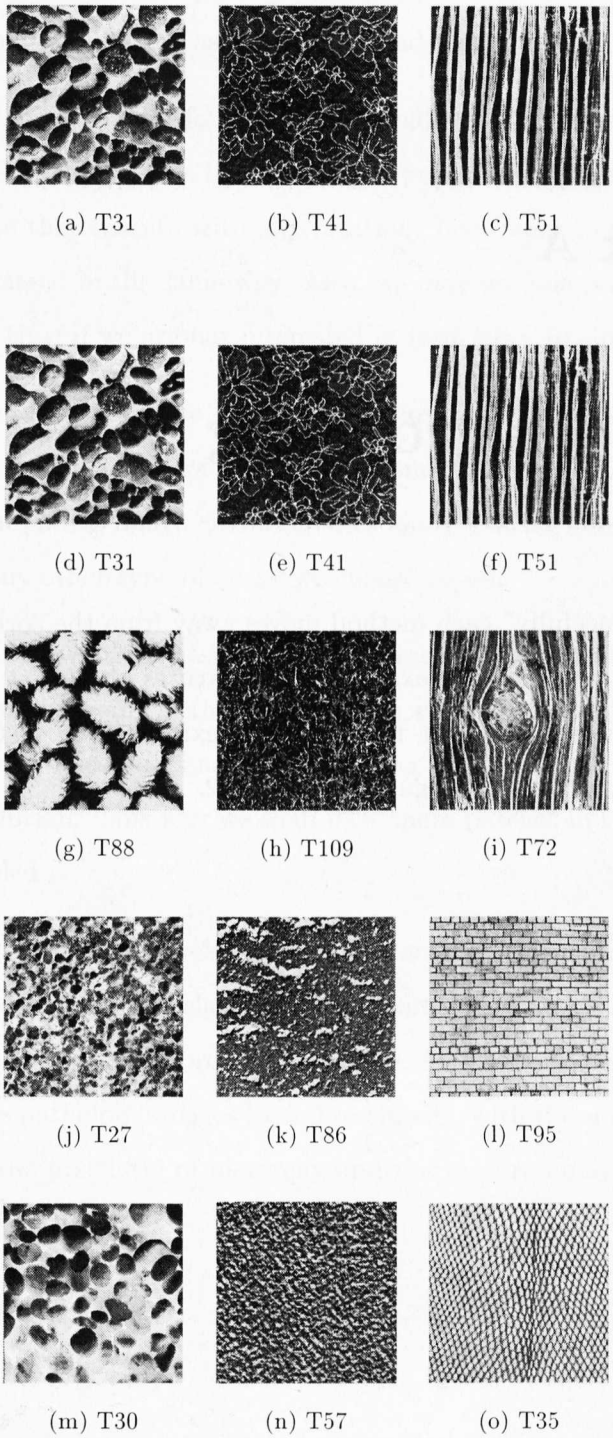
- One way to improve the classification accuracy of irregularly sampled data is to use more than one point in the image where we place the sampling mask. After all, our eyes often scan an image by foveating at several places in it in order to achieve recognition. This way we shall have more patches in the image which are densely sampled.

- A further improvement in the results may be achieved by using a hierarchical system where textures are first classified into broad classes and at a second stage they are reclassified inside each broad class. This, for example might have helped in the case of the pathology images I tried to classify, with the segmented cells being extracted at the first level of hierarchy and the texture analysis then performed at each individual cell.

Appendix A

Perceptual grouping

To indicate how "gracefully" each method moves away from the correct textures when classifying textures, we show an example of 3 textures chosen at random from the database and we present the four most similar textures each method produced, in decreasing order of similarity, in figures A.1 to A.12.



(a) T31 (b) T41 (c) T51

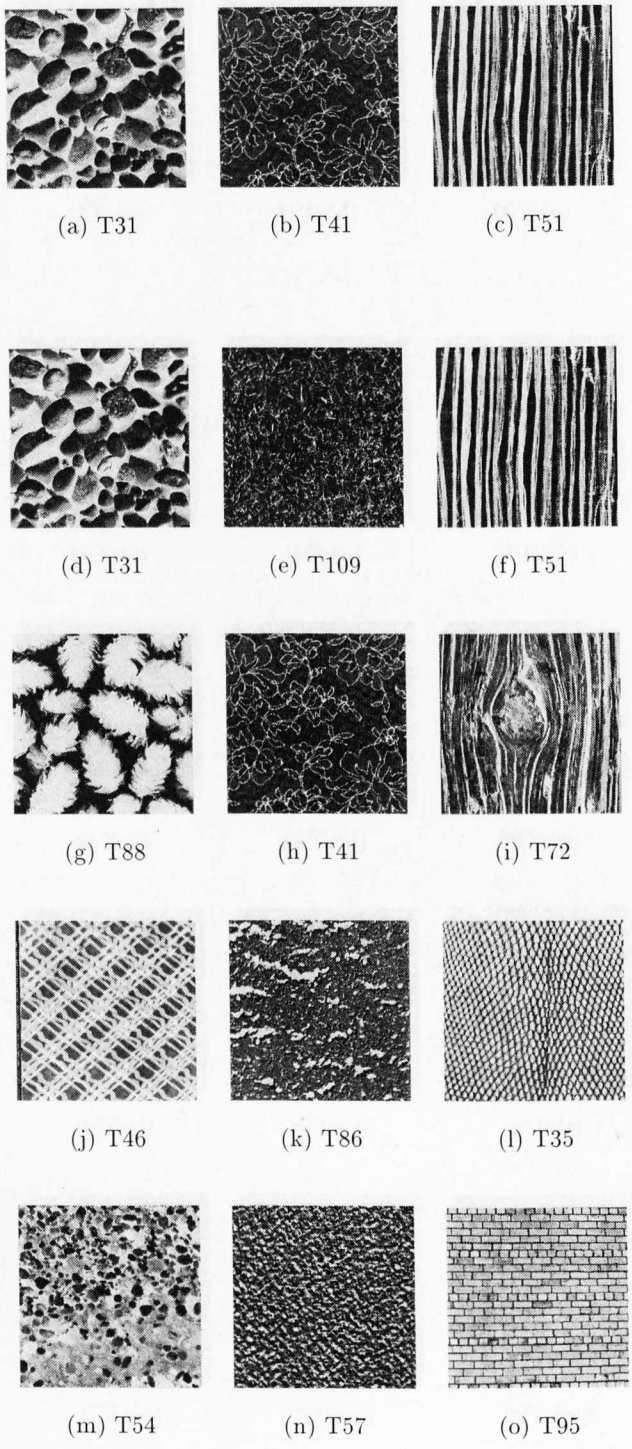
(d) T31 (e) T41 (f) T51

(g) T88 (h) T109 (i) T72

(j) T27 (k) T86 (l) T95

(m) T30 (n) T57 (o) T35

Figure A.1: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the TT method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

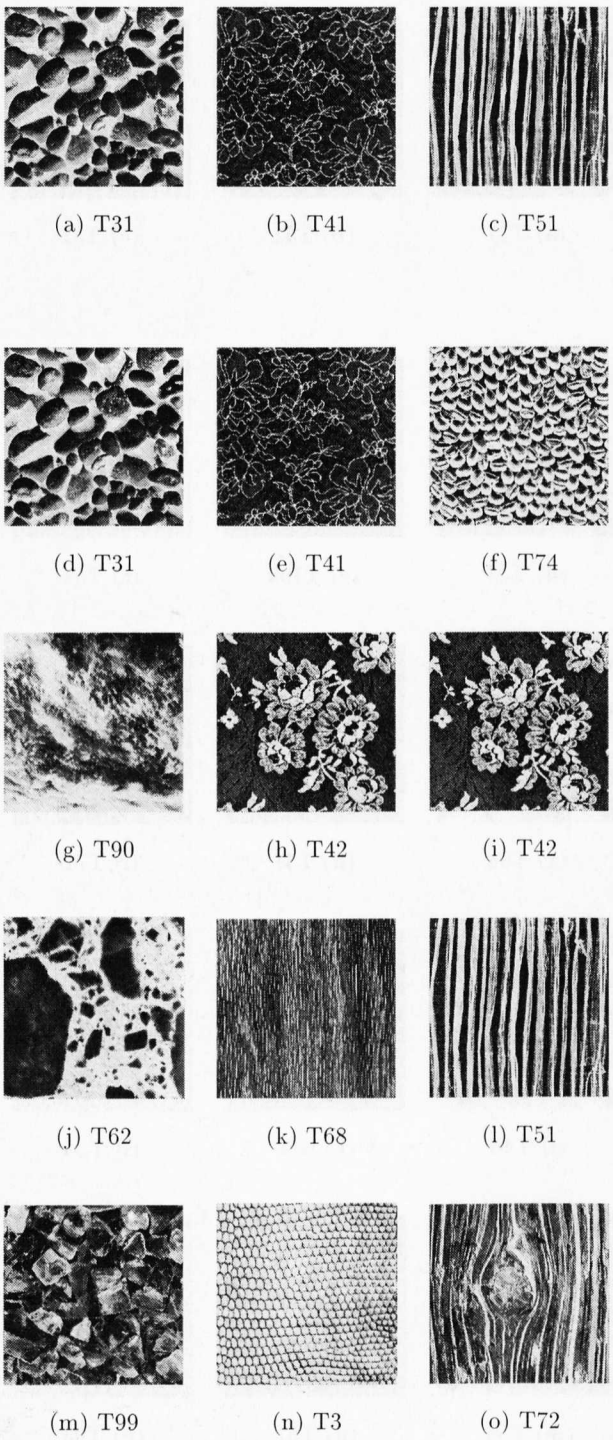
(d) T31 (e) T109 (f) T51

(g) T88 (h) T41 (i) T72

(j) T46 (k) T86 (l) T35

(m) T54 (n) T57 (o) T95

Figure A.2: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the BTT method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

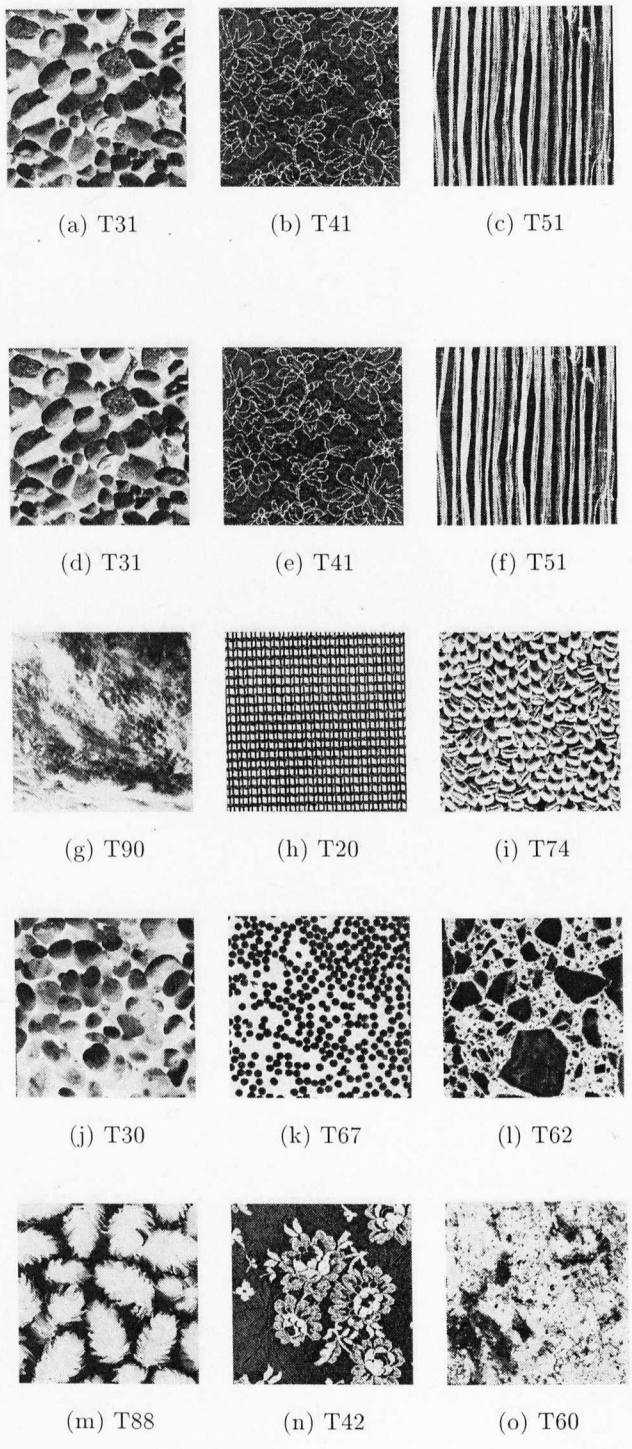
(d) T31 (e) T41 (f) T74

(g) T90 (h) T42 (i) T42

(j) T62 (k) T68 (l) T51

(m) T99 (n) T3 (o) T72

Figure A.3: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the FCM method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

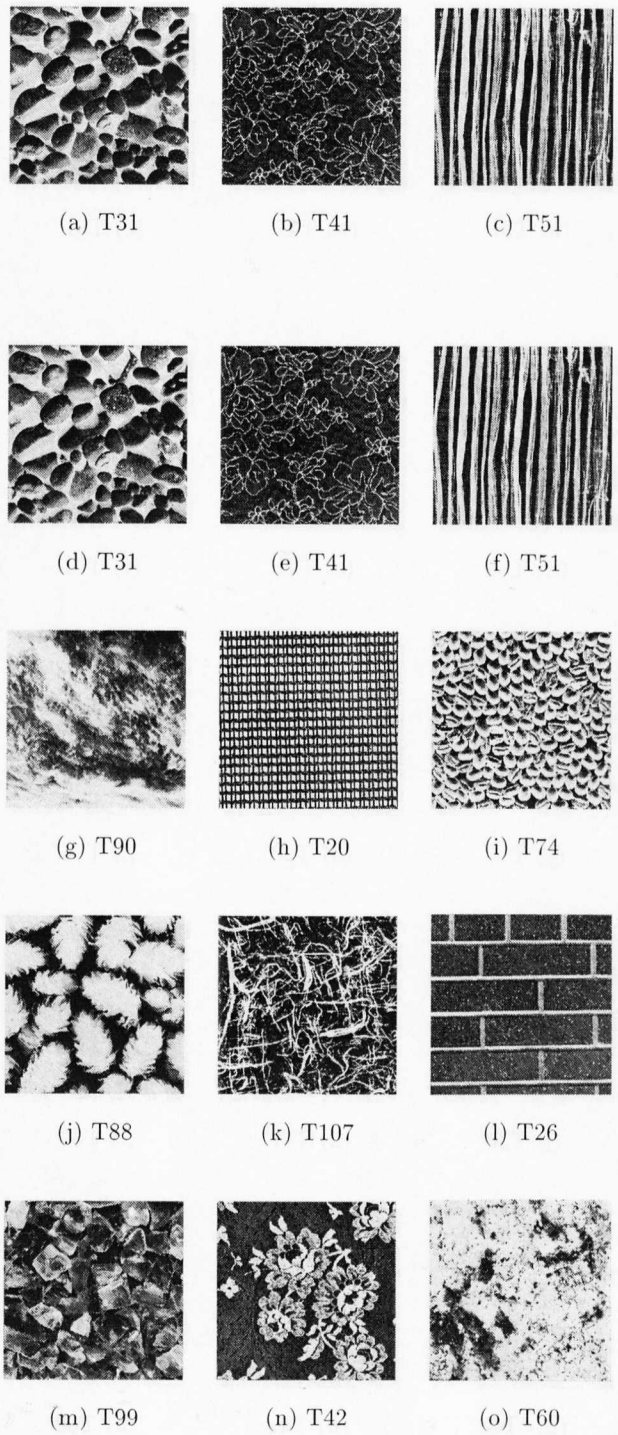
(d) T31 (e) T41 (f) T51

(g) T90 (h) T20 (i) T74

(j) T30 (k) T67 (l) T62

(m) T88 (n) T42 (o) T60

Figure A.4: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the CM16 method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

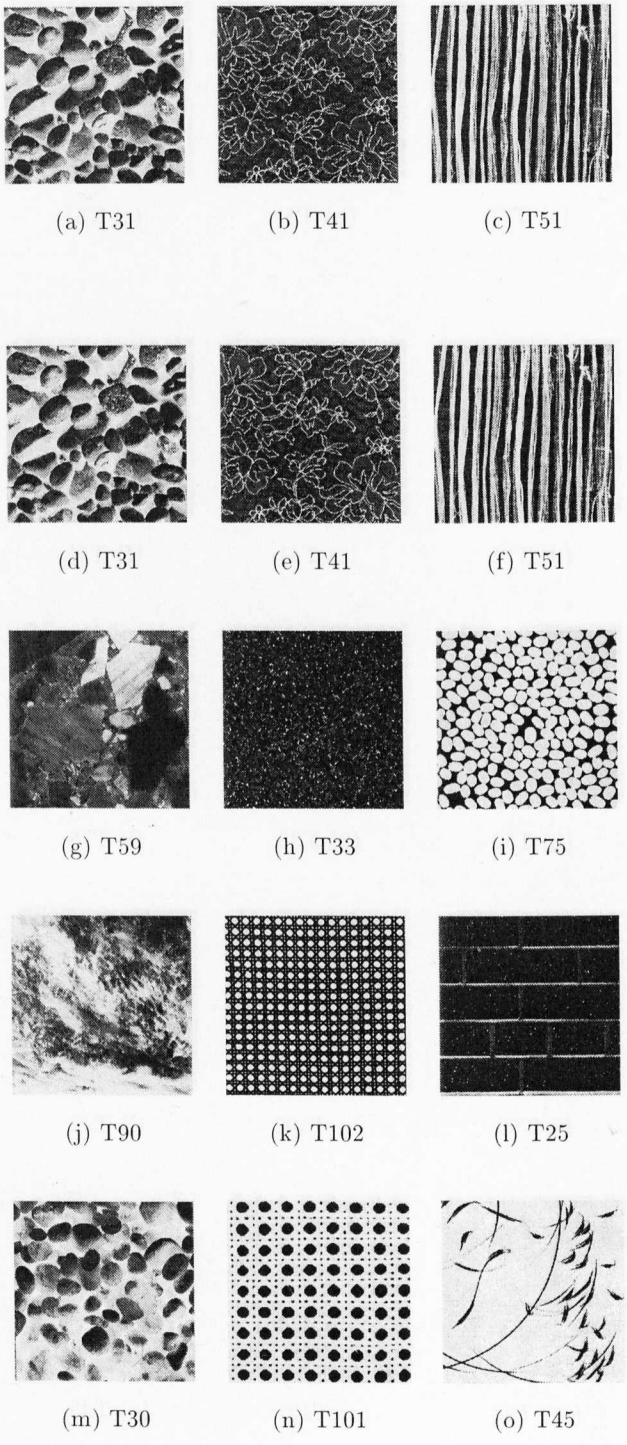
(d) T31 (e) T41 (f) T51

(g) T90 (h) T20 (i) T74

(j) T88 (k) T107 (l) T26

(m) T99 (n) T42 (o) T60

Figure A.5: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the CM16+F method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

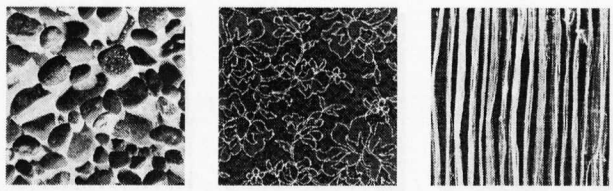
(d) T31 (e) T41 (f) T51

(g) T59 (h) T33 (i) T75

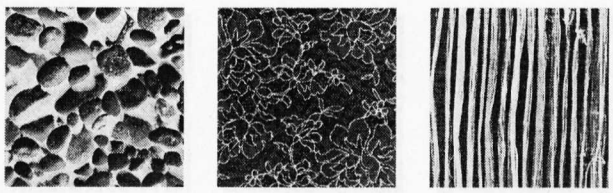
(j) T90 (k) T102 (l) T25

(m) T30 (n) T101 (o) T45

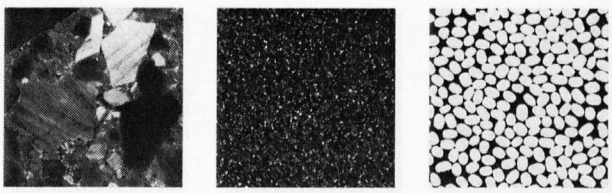
Figure A.6: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the CM256 method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



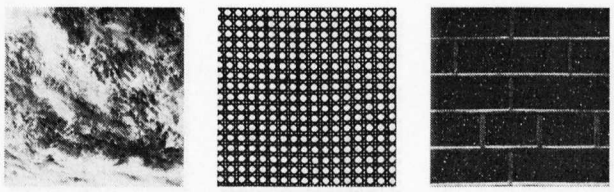
(a) T31 (b) T41 (c) T51



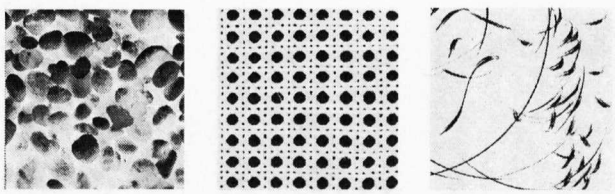
(d) T31 (e) T41 (f) T51



(g) T59 (h) T33 (i) T75

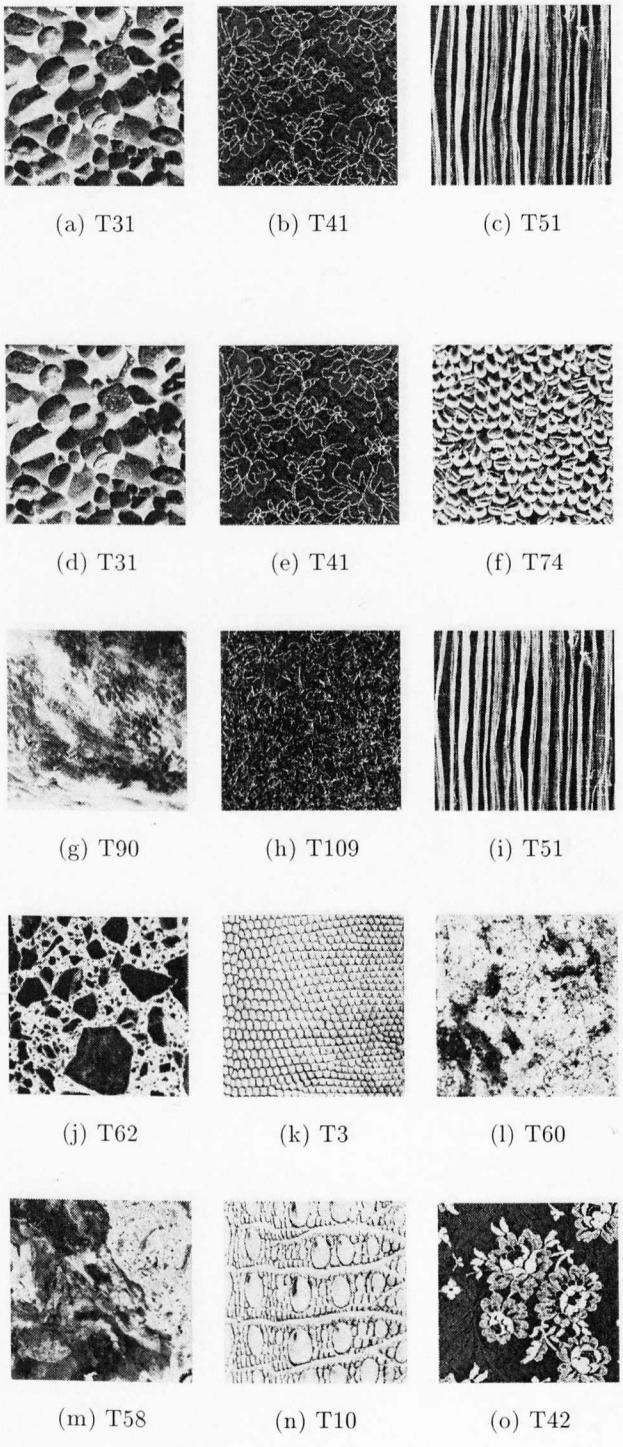


(j) T90 (k) T102 (l) T25



(m) T30 (n) T101 (o) T45

Figure A.7: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the CM256+F method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

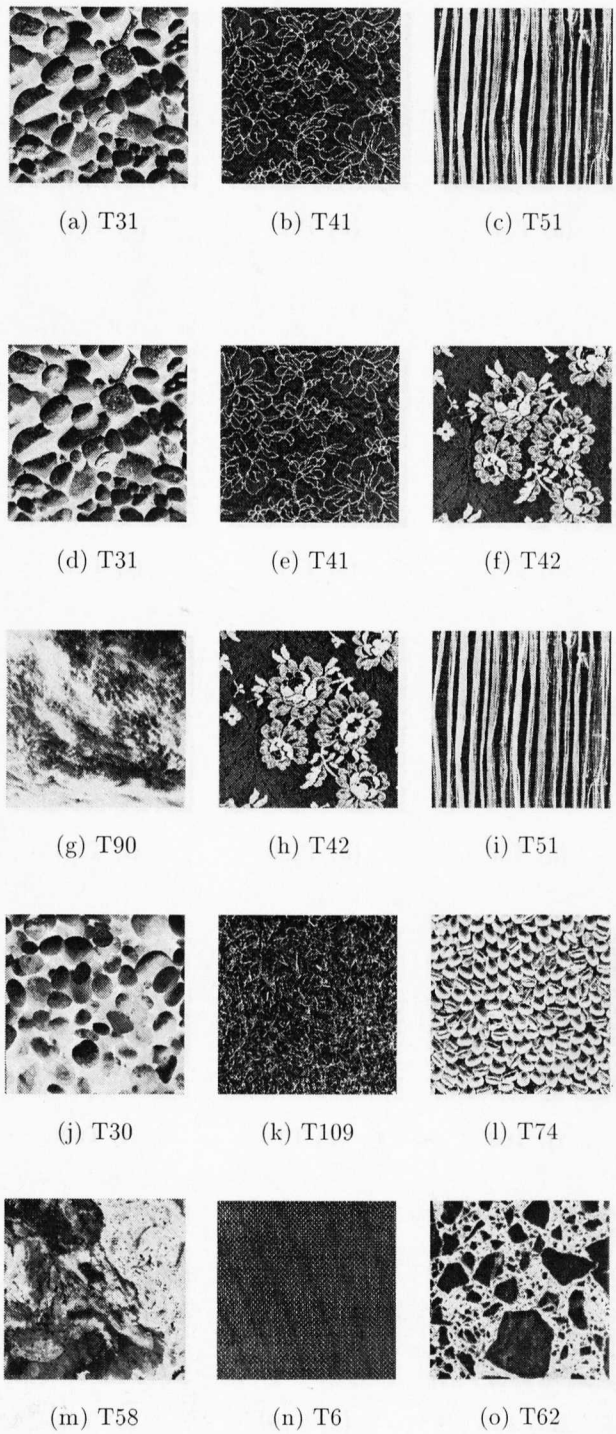
(d) T31 (e) T41 (f) T74

(g) T90 (h) T109 (i) T51

(j) T62 (k) T3 (l) T60

(m) T58 (n) T10 (o) T42

Figure A.8: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the FSDH method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

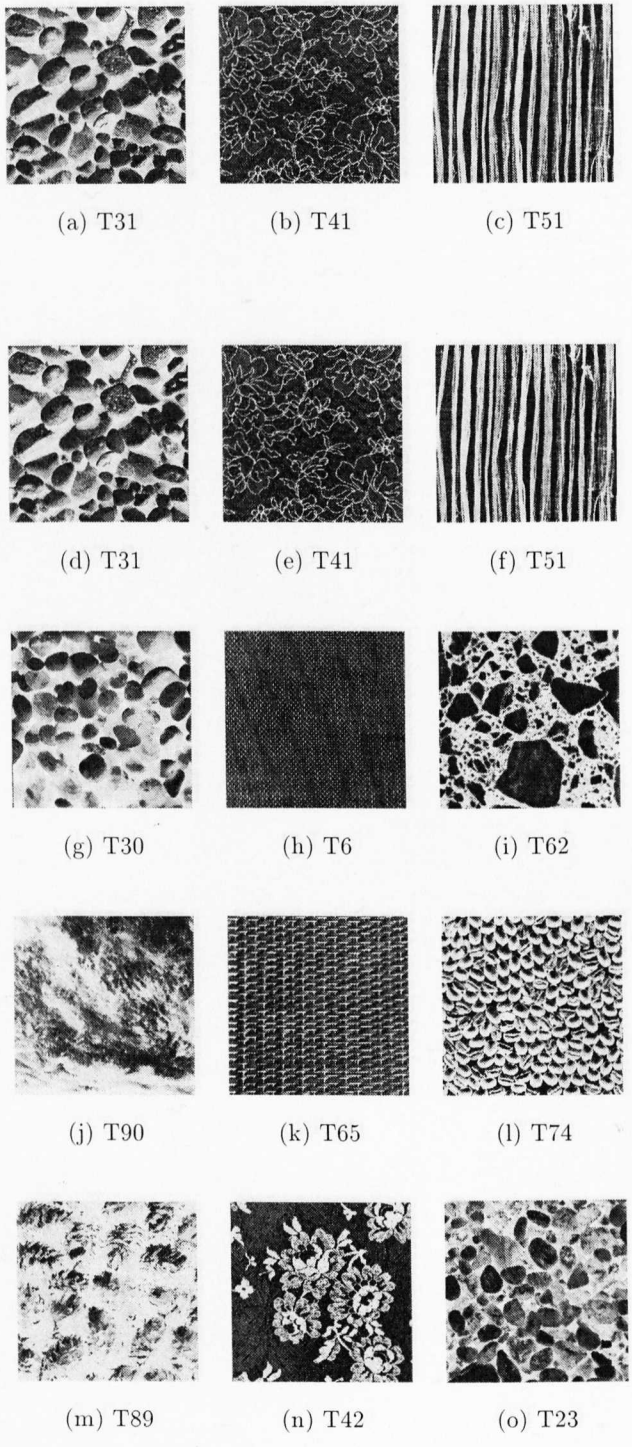
(d) T31 (e) T41 (f) T42

(g) T90 (h) T42 (i) T51

(j) T30 (k) T109 (l) T74

(m) T58 (n) T6 (o) T62

Figure A.9: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the SDH16 method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

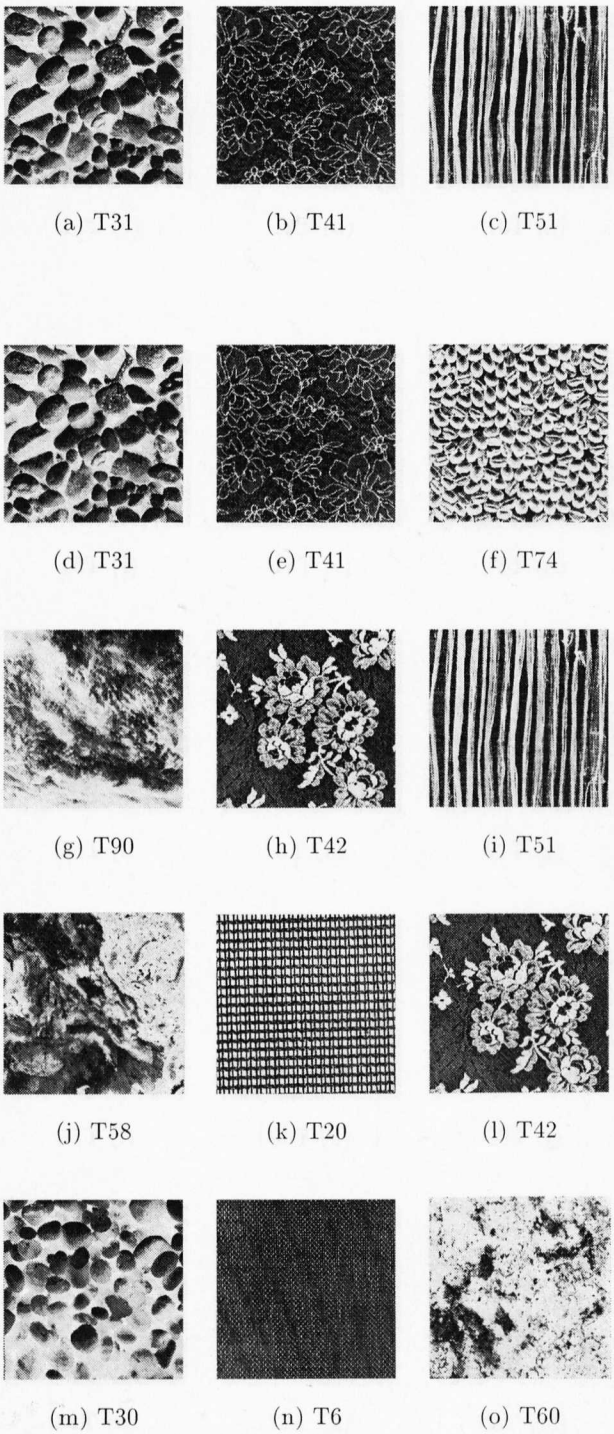
(d) T31 (e) T41 (f) T51

(g) T30 (h) T6 (i) T62

(j) T90 (k) T65 (l) T74

(m) T89 (n) T42 (o) T23

Figure A.10: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the SDH16+F method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

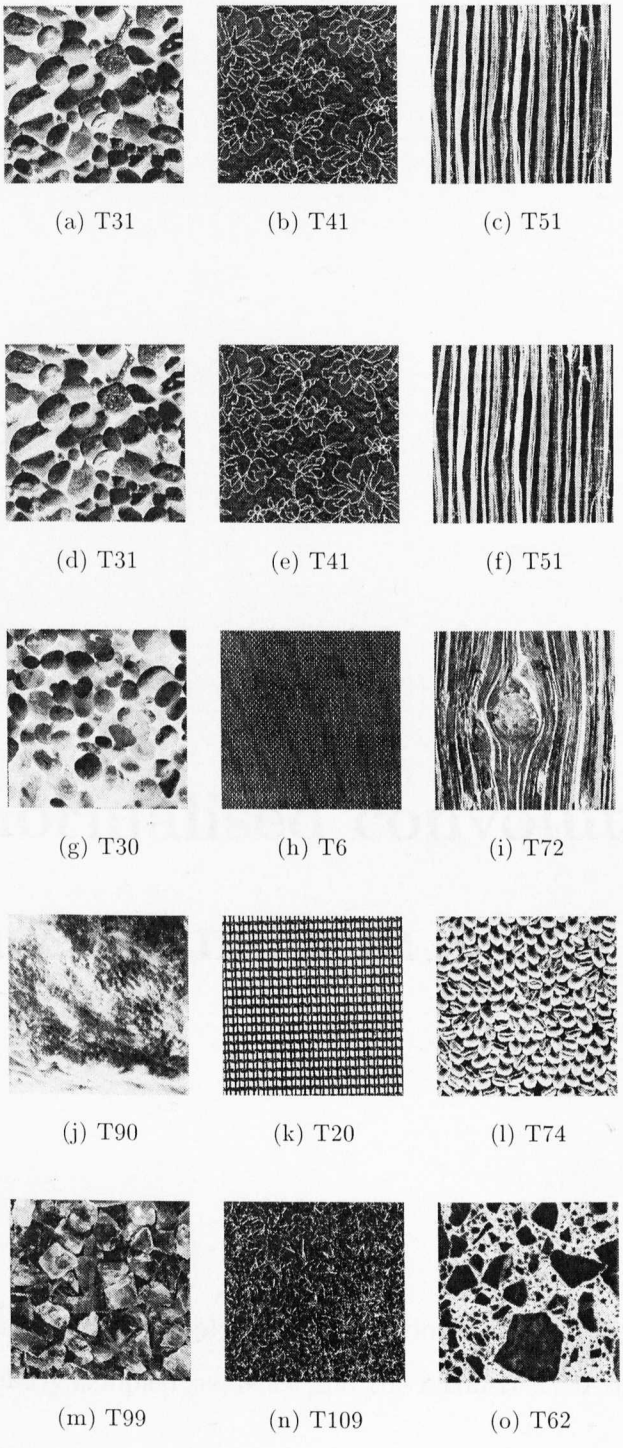
(d) T31 (e) T41 (f) T74

(g) T90 (h) T42 (i) T51

(j) T58 (k) T20 (l) T42

(m) T30 (n) T6 (o) T60

Figure A.11: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the SDH256 method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.



(a) T31 (b) T41 (c) T51

(d) T31 (e) T41 (f) T51

(g) T30 (h) T6 (i) T72

(j) T90 (k) T20 (l) T74

(m) T99 (n) T109 (o) T62

Figure A.12: At the top row we show three examples of the texture database. Underneath each texture we show the textures picked up by the SDH256+F method in the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} position of similarity respectively.

Appendix B

Using normalised convolution in the Trace transform

Here we compute the result of applying each functional of the Trace transform on the original, the irregularly sampled sequence and the reconstructed signal by normalised convolution.

We used all functionals for 50 lines taken from the original set of tracing lines. The results are shown in figure B.1 to figure B.59. We can see that the result of the reconstructed signal is much closer to the original than the result of the irregularly sampled signal in all functionals which have not used the max or min operator.

B.1 Trace functionals

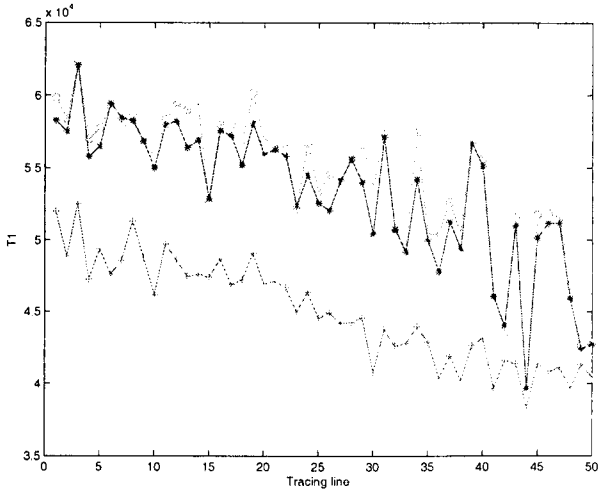


Figure B.1: The values of trace functional T_1 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

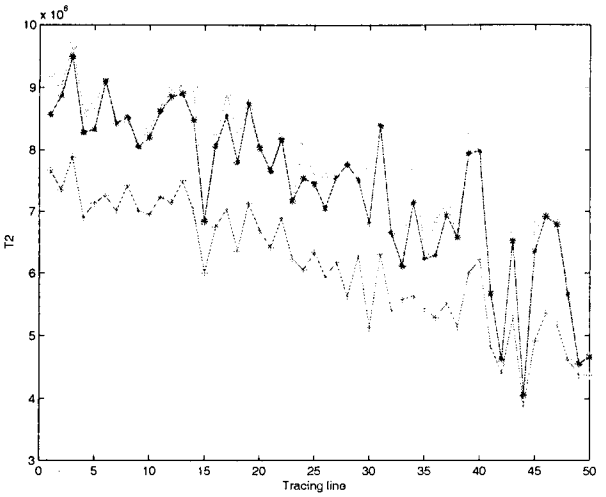


Figure B.2: The values of trace functional T_2 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

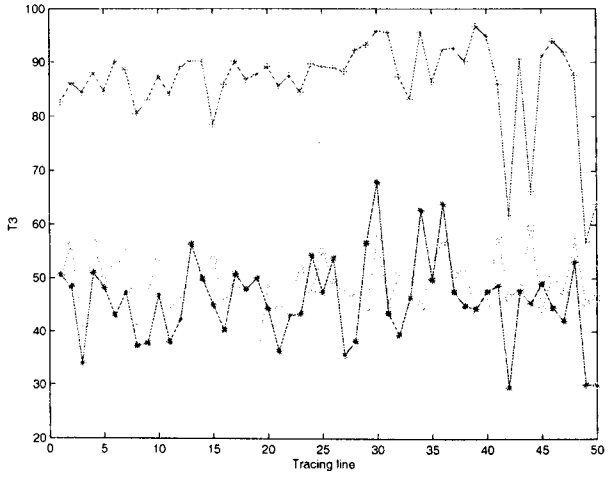


Figure B.3: The values of trace functional T_3 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

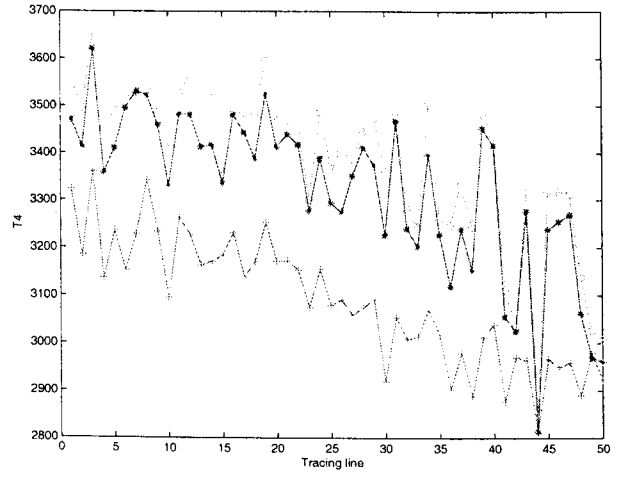


Figure B.4: The values of trace functional T_4 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

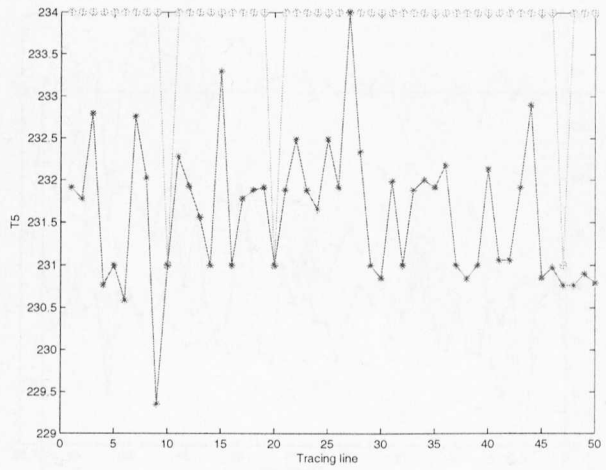


Figure B.5: The values of trace functional T_5 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

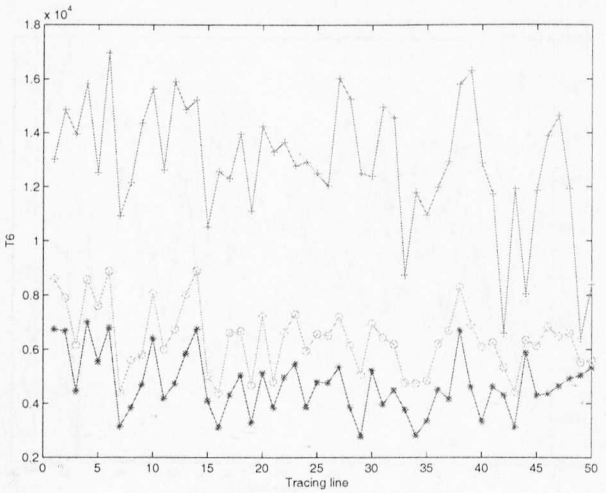


Figure B.6: The values of trace functional T_6 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

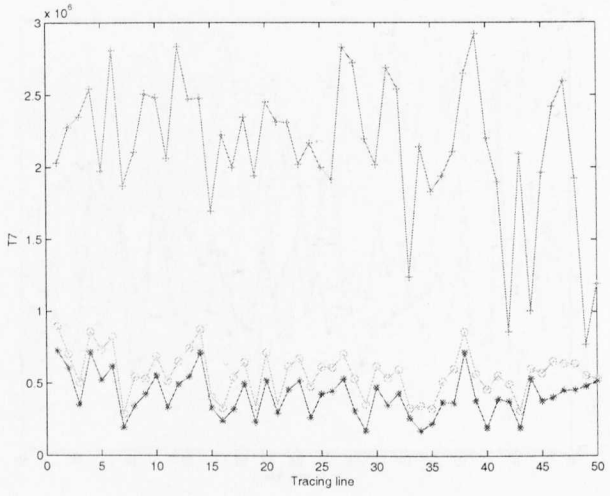


Figure B.7: The values of trace functional T_7 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

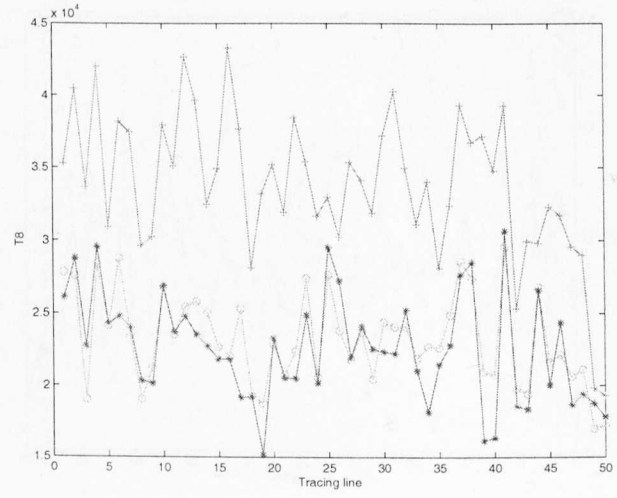


Figure B.8: The values of trace functional T_8 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

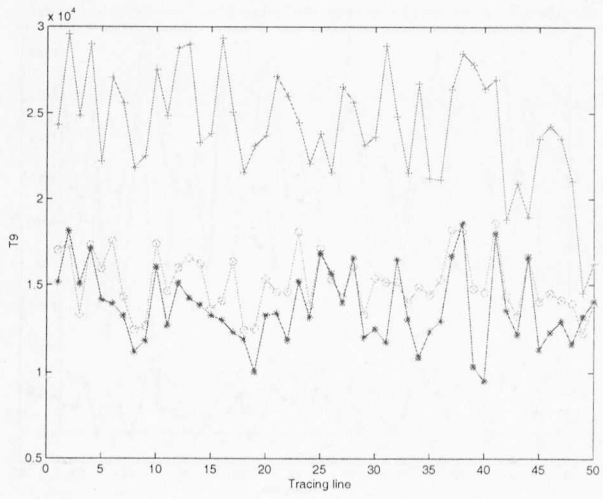


Figure B.9: The values of trace functional T_9 from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

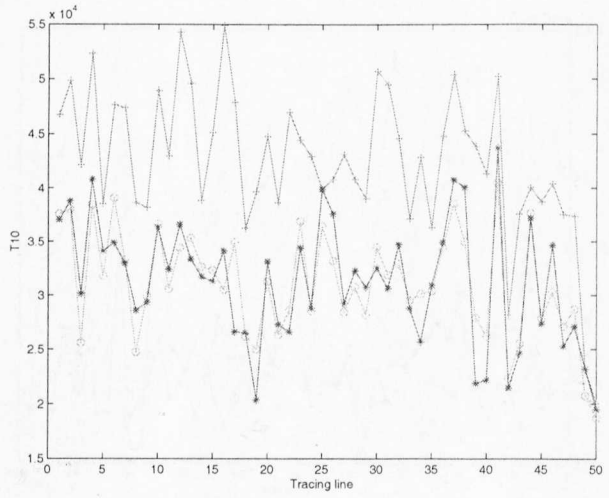


Figure B.10: The values of trace functional T_{10} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

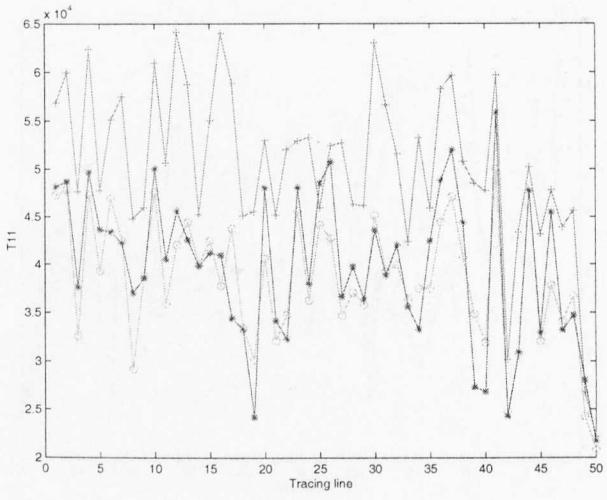


Figure B.11: The values of trace functional T_{11} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

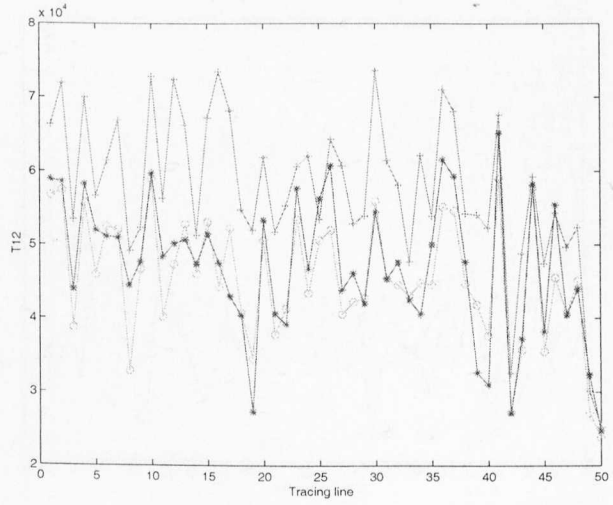


Figure B.12: The values of trace functional T_{12} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

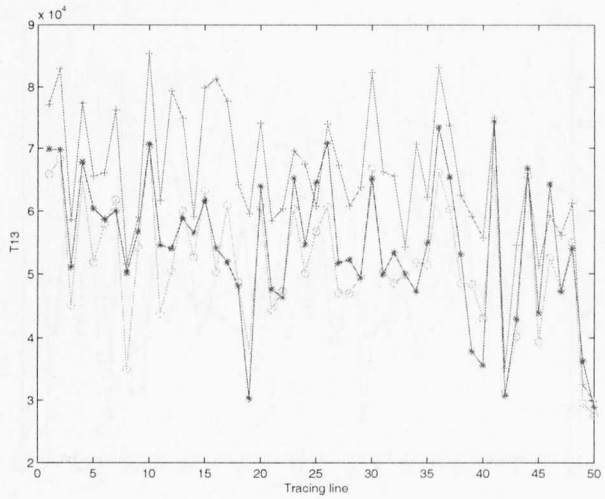


Figure B.13: The values of trace functional T_{13} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

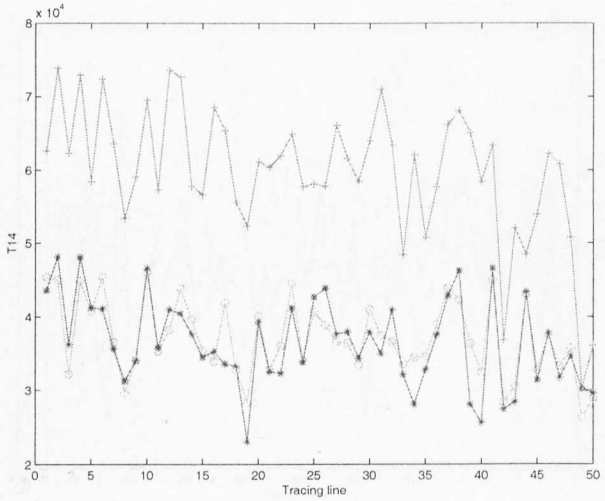


Figure B.14: The values of trace functional T_{14} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

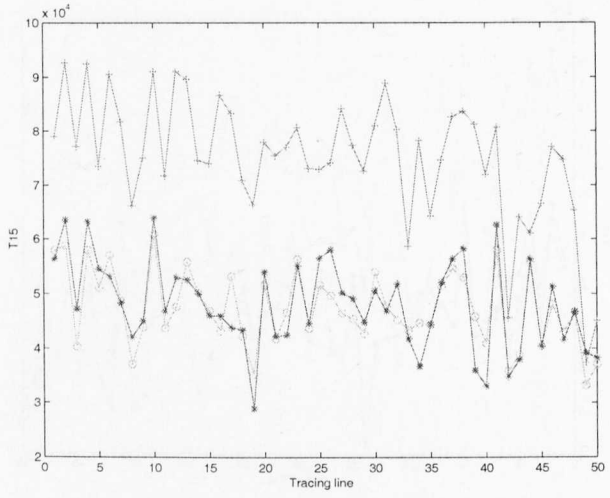


Figure B.15: The values of trace functional T_{15} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

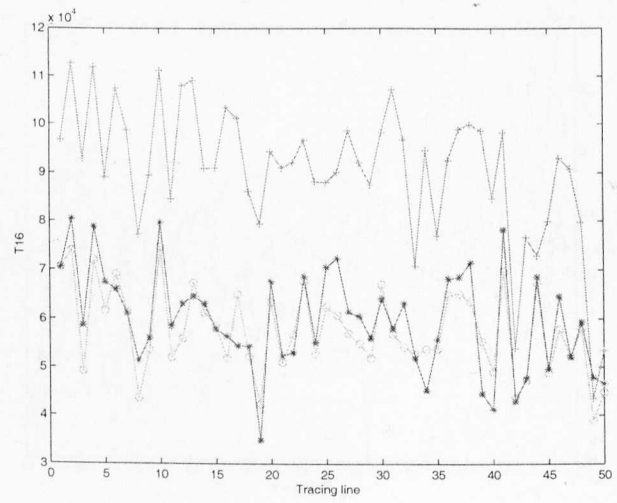


Figure B.16: The values of trace functional T_{16} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

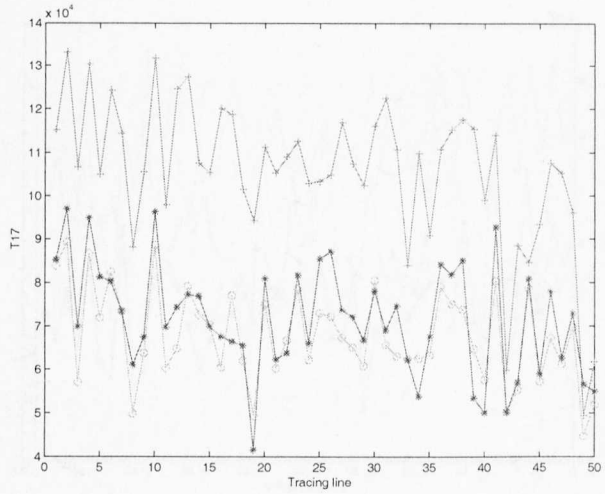


Figure B.17: The values of trace functional T_{17} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

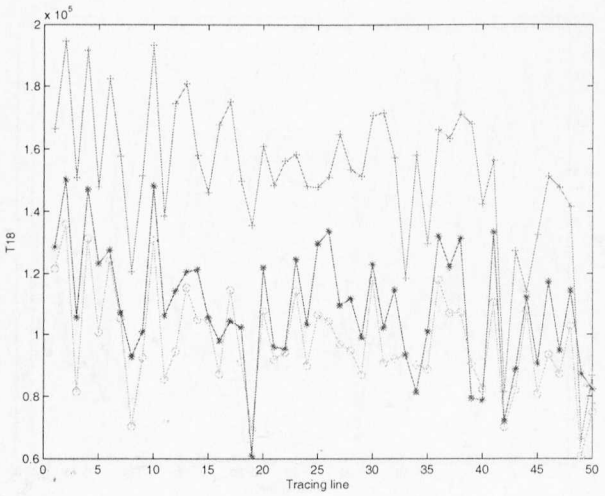


Figure B.18: The values of trace functional T_{18} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

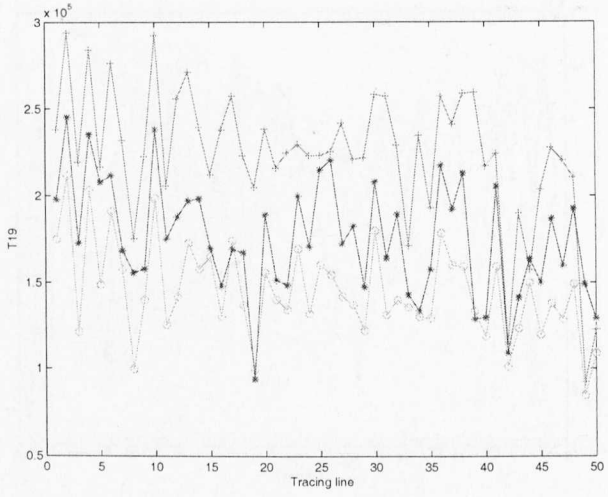


Figure B.19: The values of trace functional T_{19} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

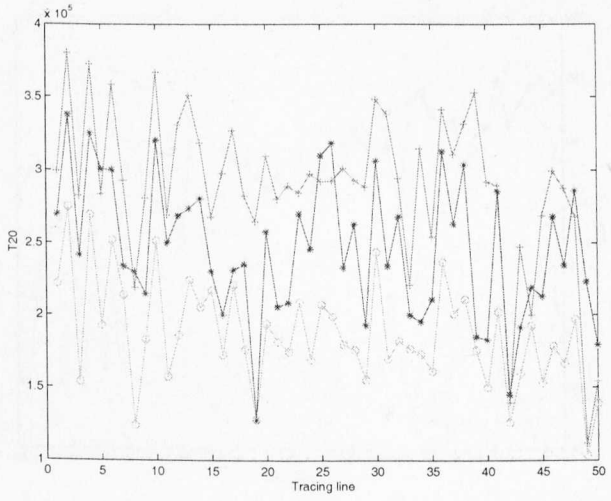


Figure B.20: The values of trace functional T_{20} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

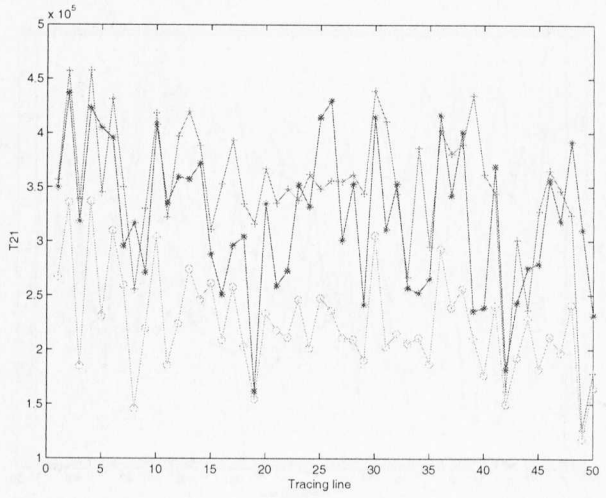


Figure B.21: The values of trace functional T_{21} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

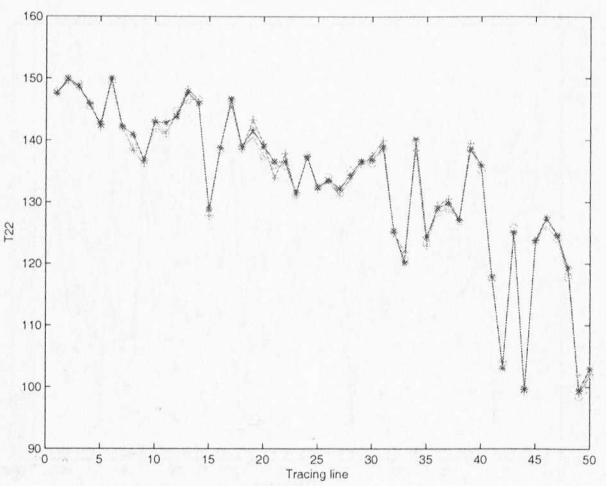


Figure B.22: The values of trace functional T_{22} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

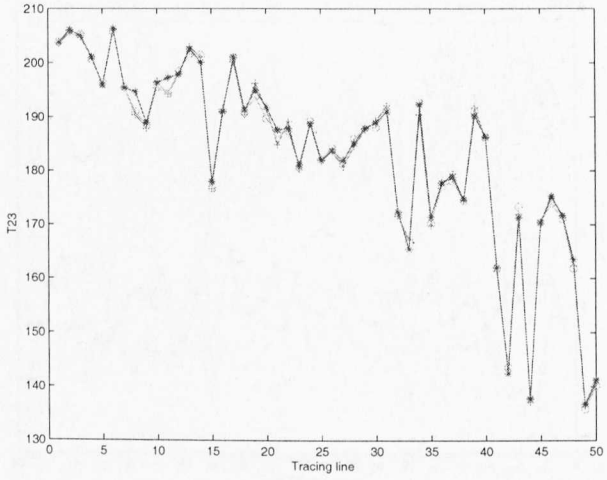


Figure B.23: The values of trace functional T_{23} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

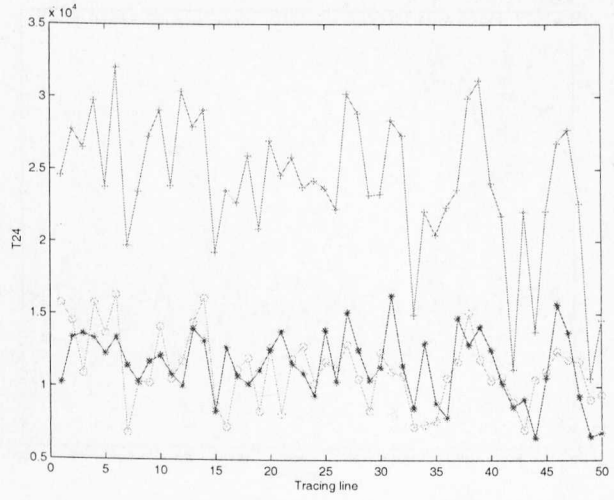


Figure B.24: The values of trace functional T_{24} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

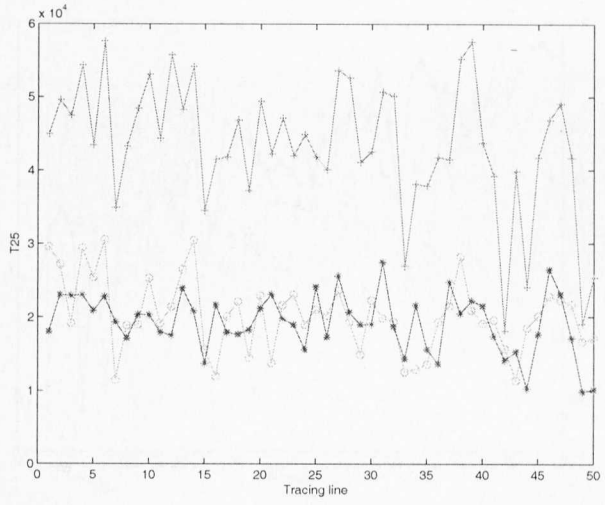


Figure B.25: The values of trace functional T_{25} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

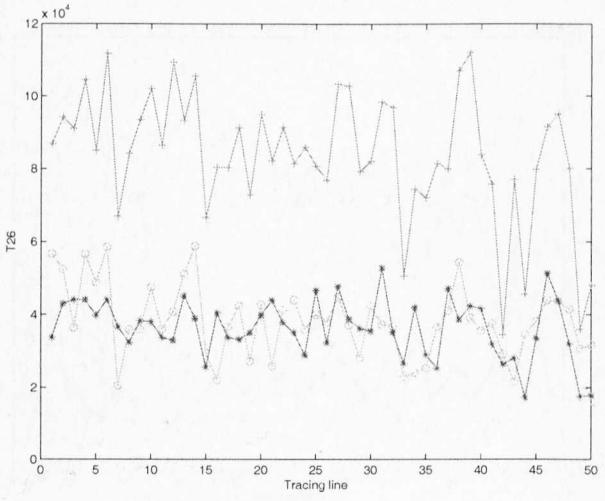


Figure B.26: The values of trace functional T_{26} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

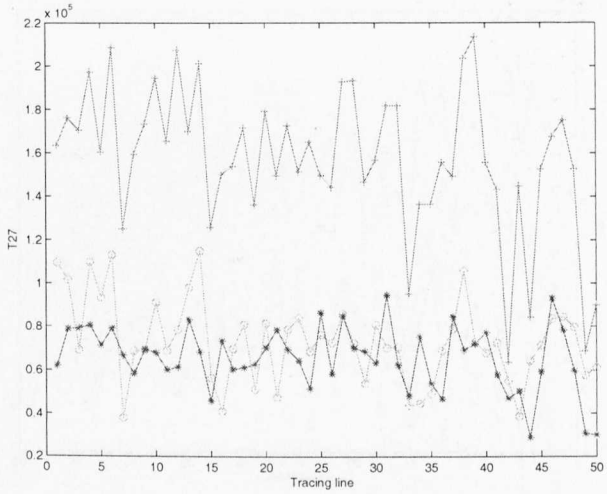


Figure B.27: The values of trace functional T_{27} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

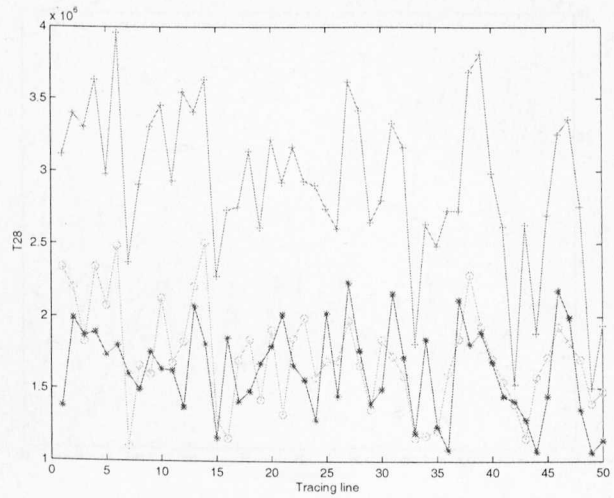


Figure B.28: The values of trace functional T_{28} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

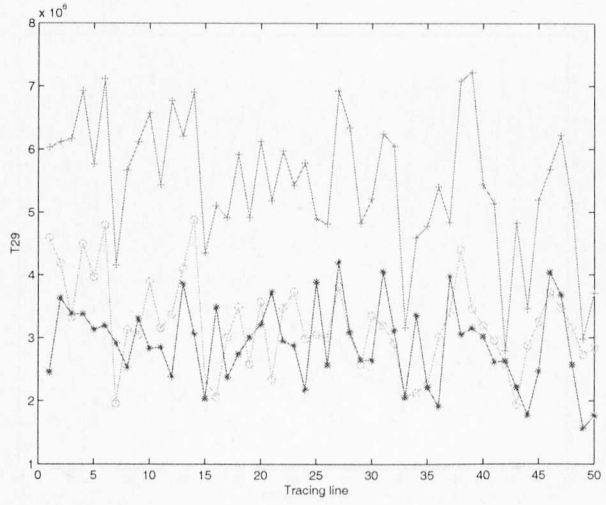


Figure B.29: The values of trace functional T_{29} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

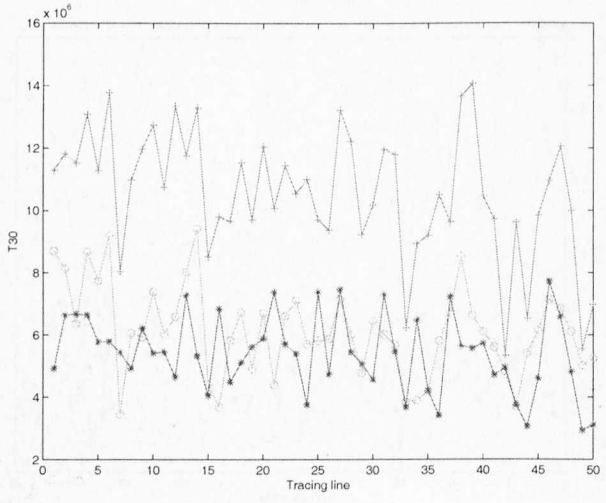


Figure B.30: The values of trace functional T_{30} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

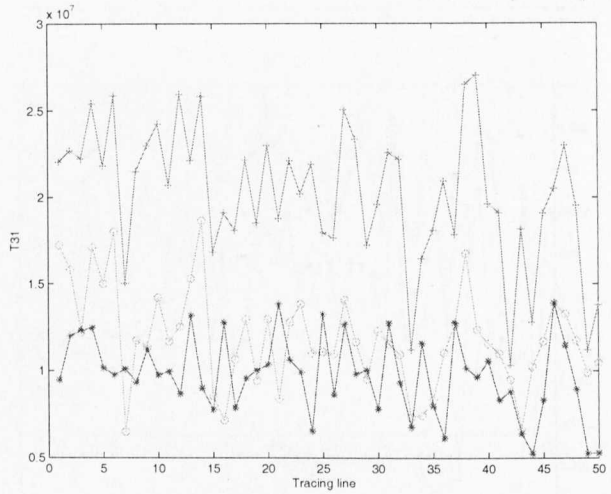


Figure B.31: The values of trace functional T_{31} from table 3.1: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

B.2 Diametric functionals

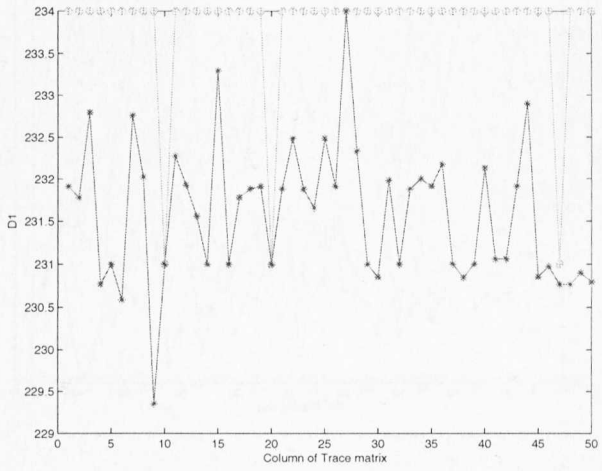


Figure B.32: The values of diametric functional D_1 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

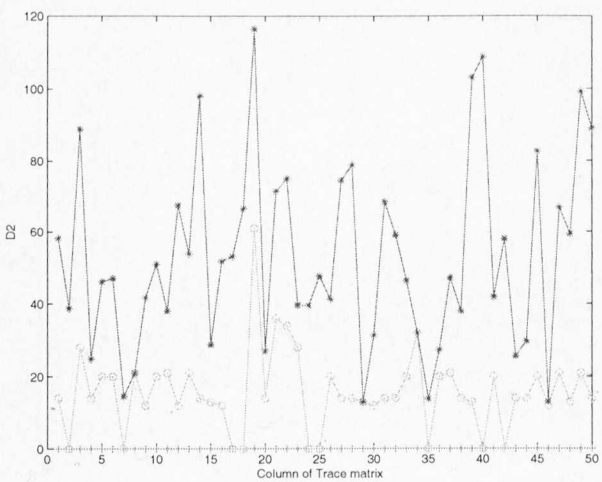


Figure B.33: The values of diametric functional D_2 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

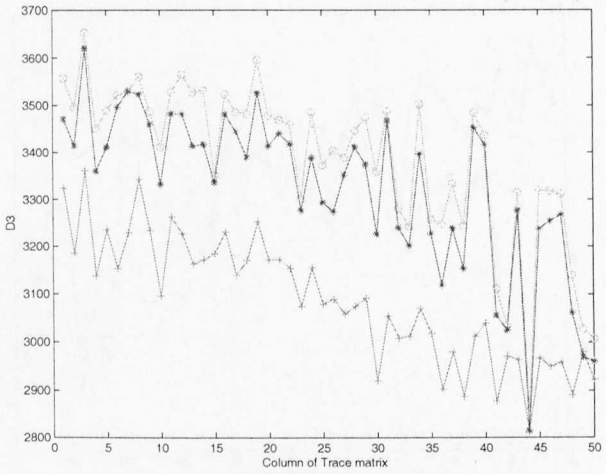


Figure B.34: The values of diametric functional D_3 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

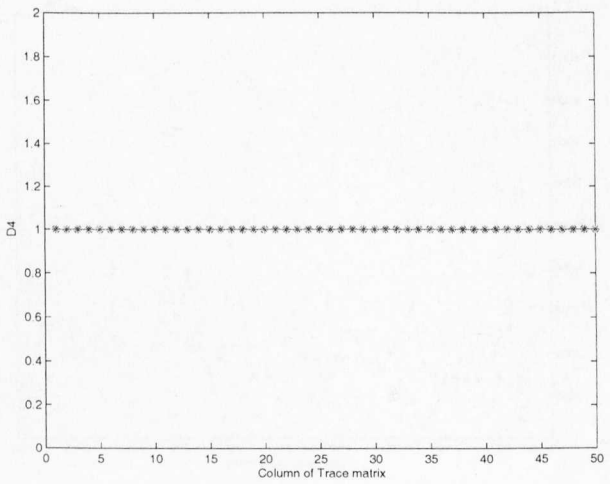


Figure B.35: The values of diametric functional D_4 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

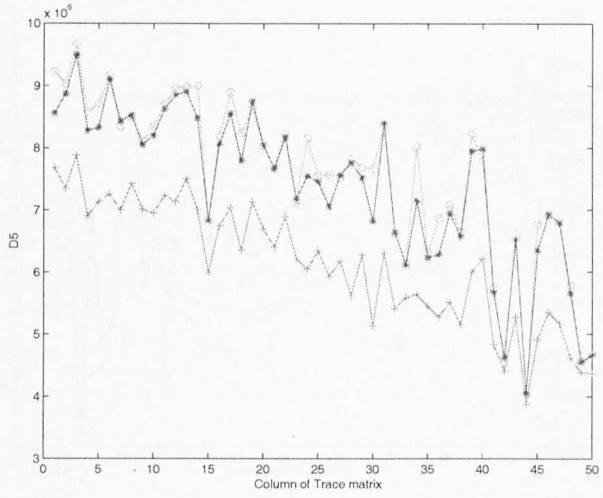


Figure B.36: The values of diametric functional D_5 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

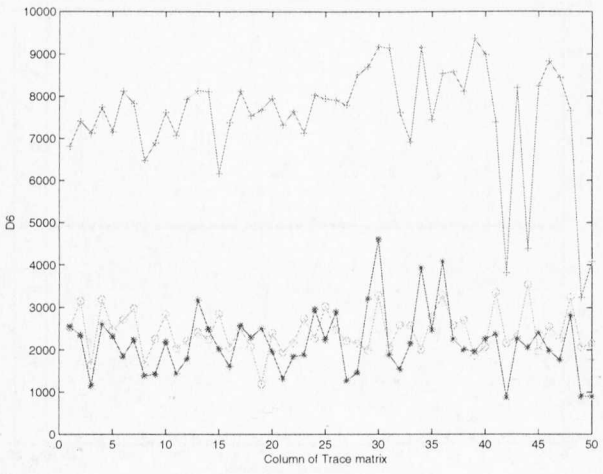


Figure B.37: The values of diametric functional D_6 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

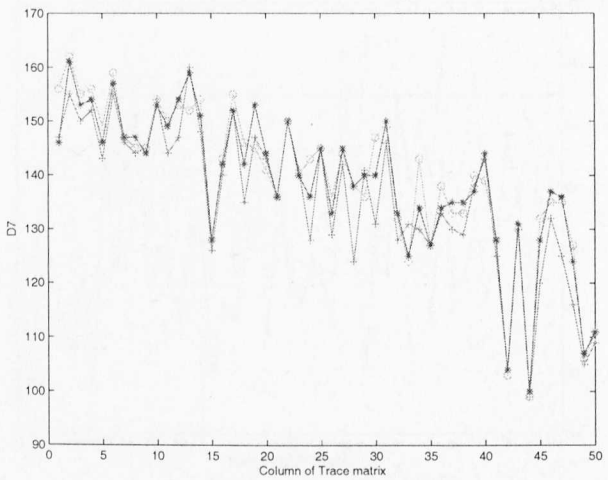


Figure B.38: The values of diametric functional D_7 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

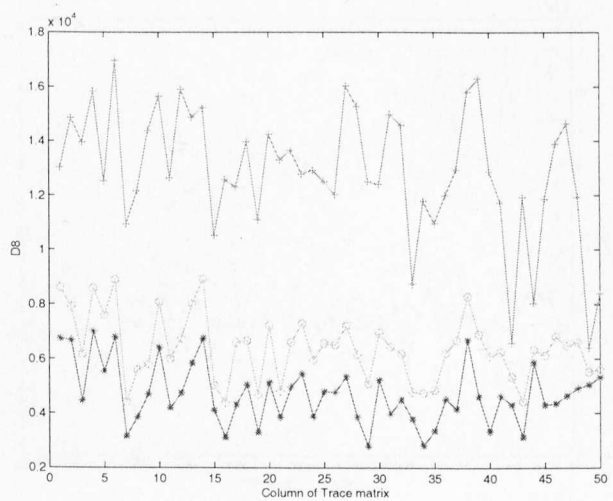


Figure B.39: The values of diametric functional D_8 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

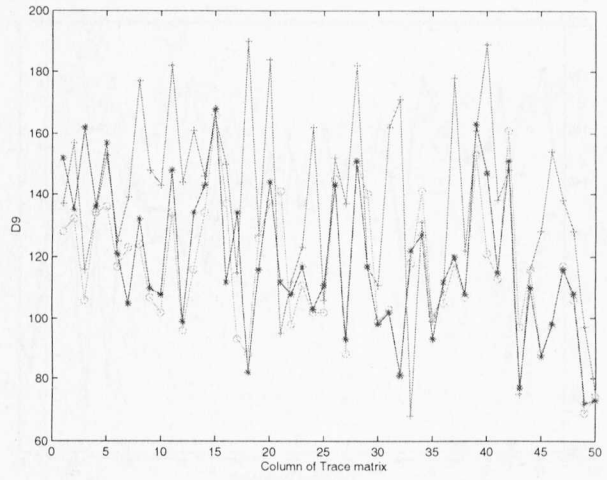


Figure B.40: The values of diametric functional D_9 from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

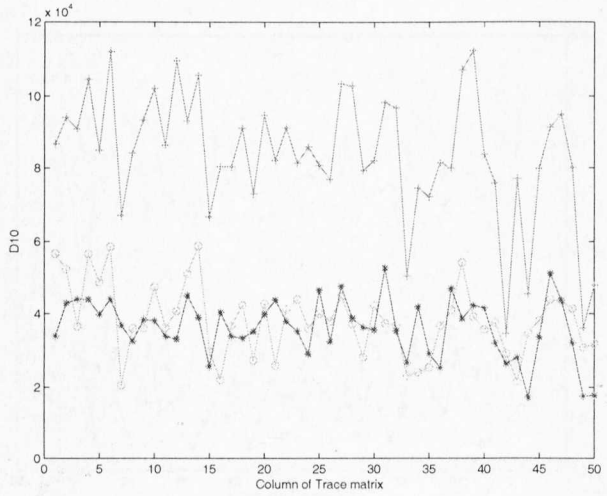


Figure B.41: The values of diametric functional D_{10} from table 3.3: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

B.3 Circus functionals

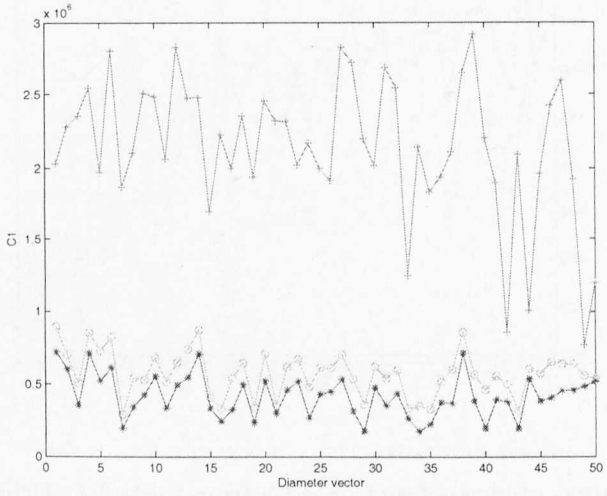


Figure B.42: The values of circus functional C_1 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

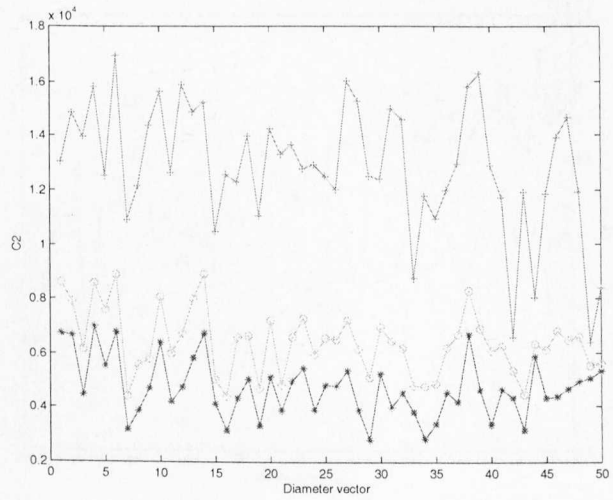


Figure B.43: The values of circus functional C_2 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

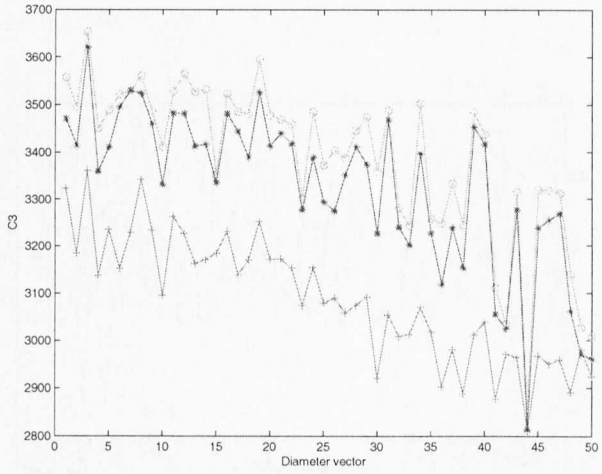


Figure B.44: The values of circus functional C_3 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

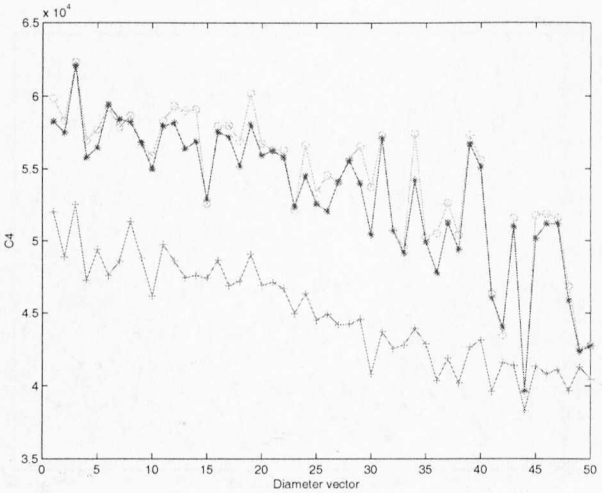


Figure B.45: The values of circus functional C_4 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

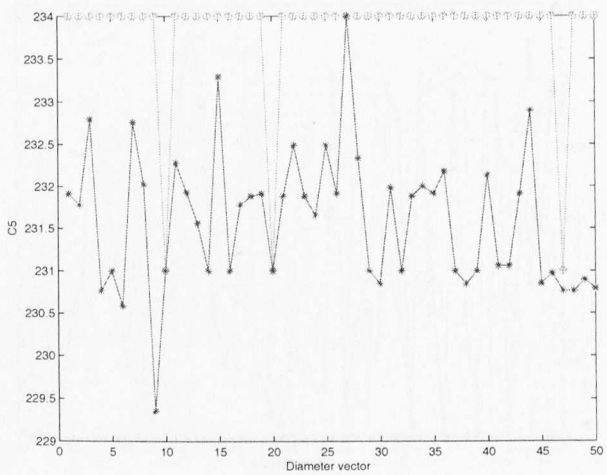


Figure B.46: The values of circus functional C_5 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

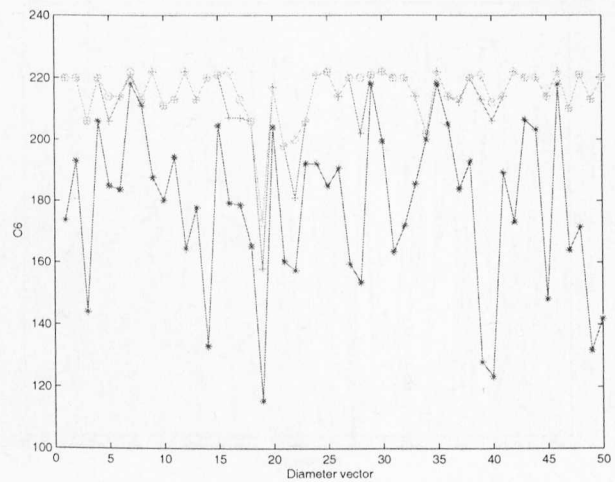


Figure B.47: The values of circus functional C_6 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

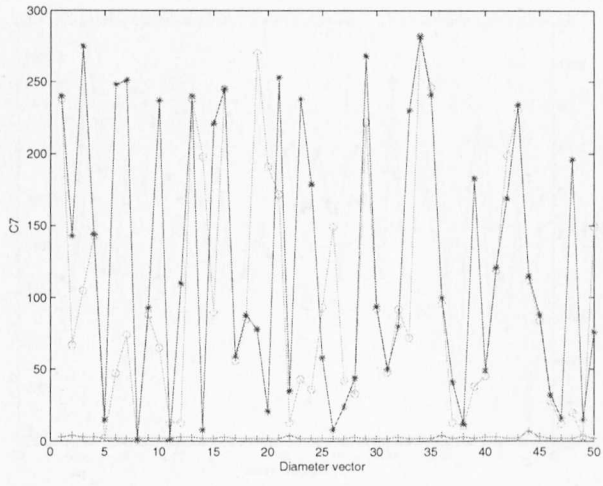


Figure B.48: The values of circus functional C_7 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

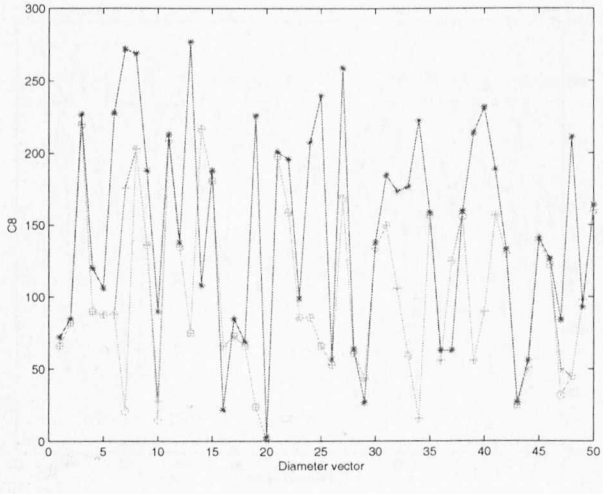


Figure B.49: The values of circus functional C_8 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

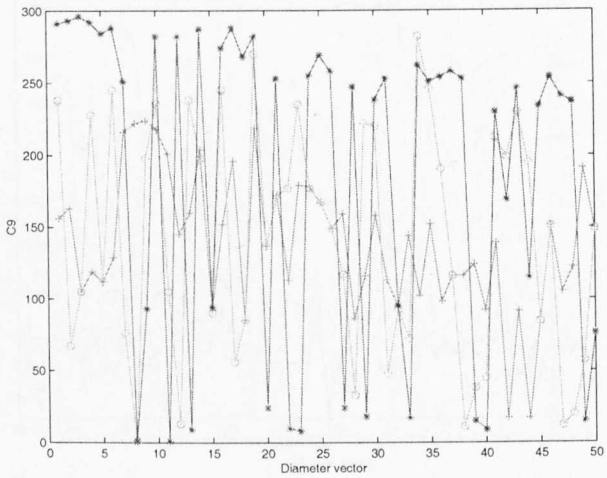


Figure B.50: The values of circus functional C_9 from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

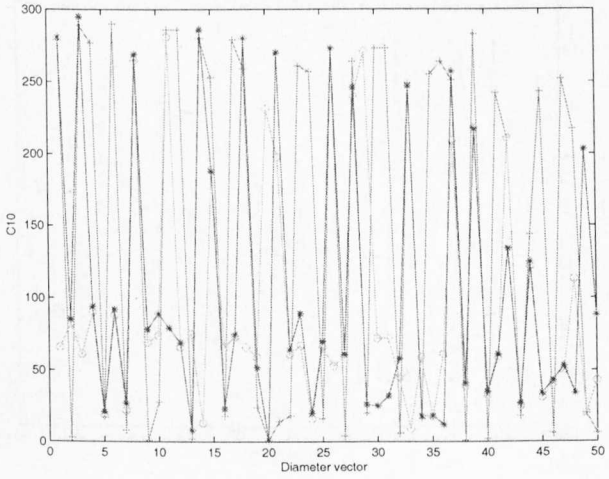


Figure B.51: The values of circus functional C_{10} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

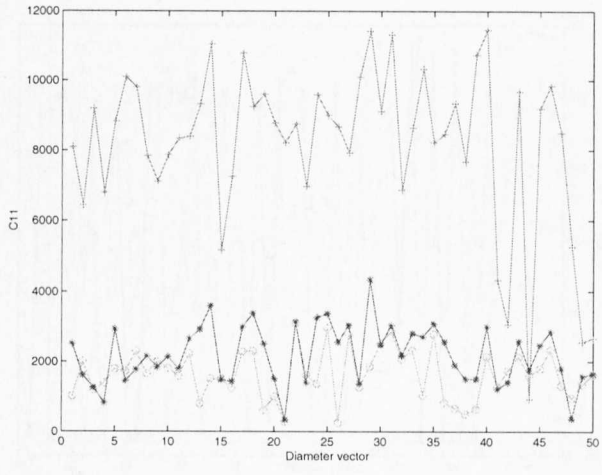


Figure B.52: The values of circus functional C_{11} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

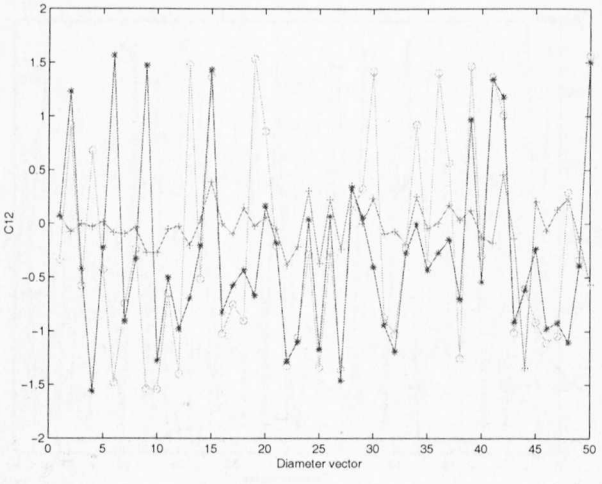


Figure B.53: The values of circus functional C_{12} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

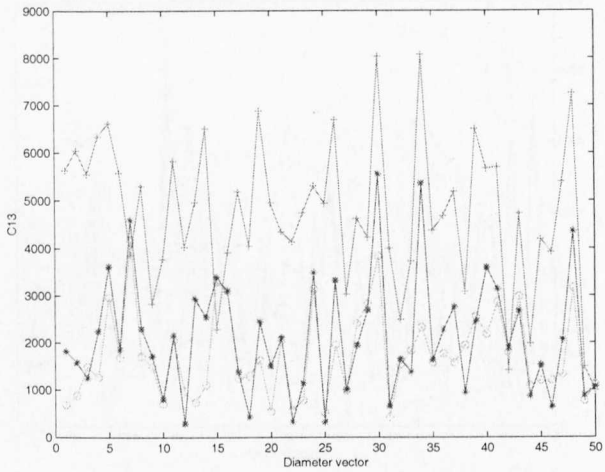


Figure B.54: The values of circus functional C_{13} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

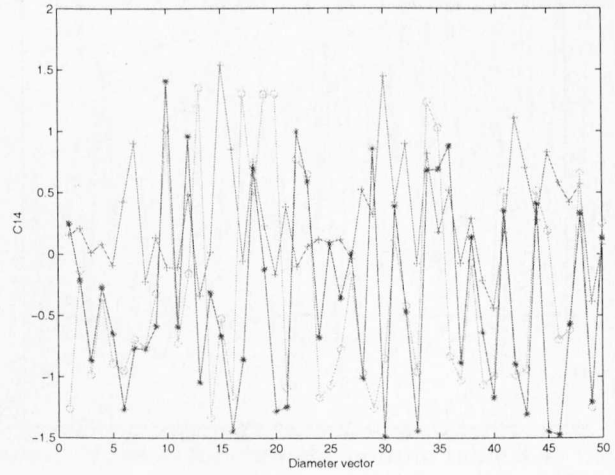


Figure B.55: The values of circus functional C_{14} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

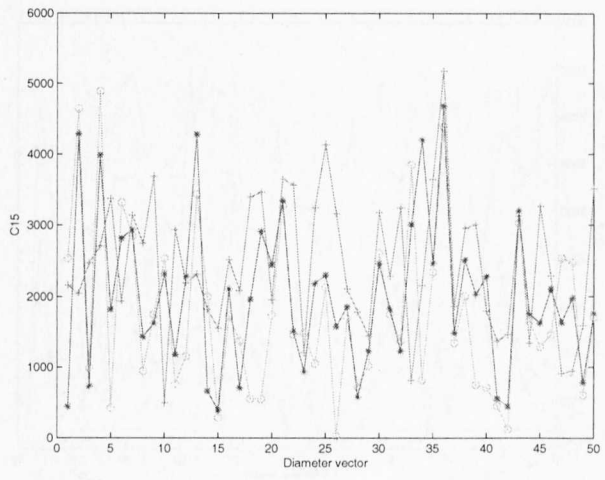


Figure B.56: The values of circus functional C_{15} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

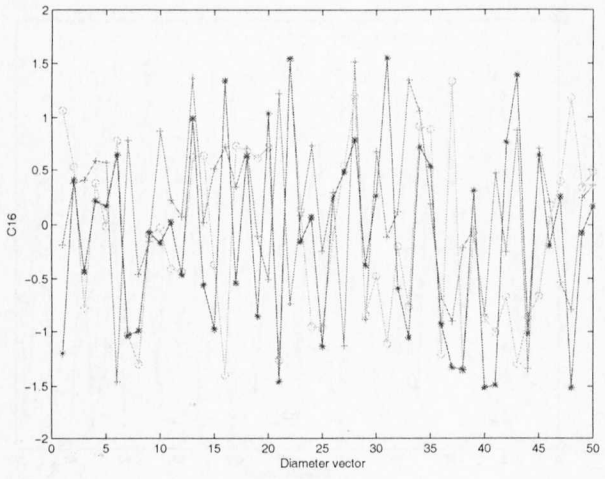


Figure B.57: The values of circus functional C_{16} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

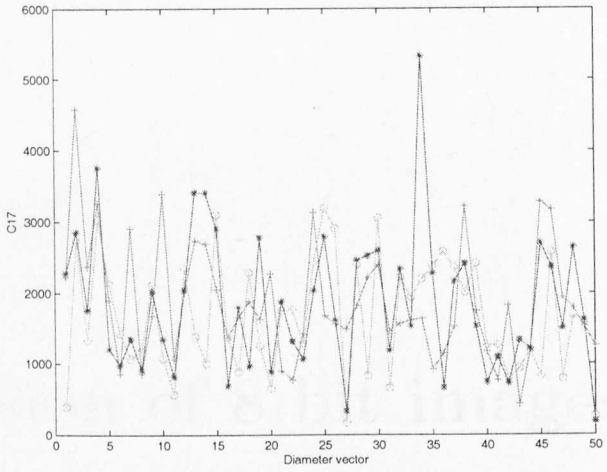


Figure B.58: The values of circus functional C_{17} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

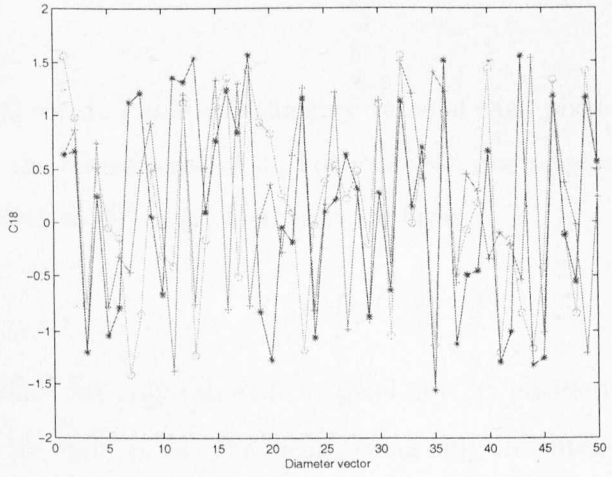


Figure B.59: The values of circus functional C_{18} from table 3.4: Using the full regularly sampled sequence of points ("o"), a sub-sampled version of it ("+") and a reconstruction of the sub-sampled version by normalised convolution ("*").

Appendix C

Conversion of 8 bit images to 4 bit

In some cases we want to convert 8 bit images to 4 bit, in order to have a smaller co-occurrence matrix. There is three method to do this:

- Simple division

  In this method we simply divide the grey value of each pixel of the image, by 16 and then use the integer part of it. For example, one pixel with grey value 201, will be converted to $\frac{201}{16} = 25$. Generally we have:

  $$g'_{i,j} = fix(\frac{g_{i,j}}{16}) \tag{C.1}$$

  where $g'_{i,j}$ is the 4 bit grey value of the pixel in (i, j) position and $g_{i,j}$ is the 8 bit grey value of the same pixel. fix means using only the integer part of the result.

- Using histogram normalisation

  In this method we use the formula:

  $$g'_{i,j} = fix\left(\frac{g_{i,j} - \min(g)}{\max(g) - \min(g)}\right) \times 15 \tag{C.2}$$

  For example, if the maximum grey value is 201 and the minimum grey value is 10, so the pixel with grey value 100, will have $fix(\frac{100-10}{201-10}) \times 15 = 7$.

- Using histogram equalisation

  Here we first find the total number of pixels, which in our case is $320 \times 320 = 102400$. Then we try to distribute these points in 16 equal groups, starting from the pixels with low grey value and end up with the pixels with the highest grey value. This way we have histograms with equal number of points in all bins.

The best result of texture classification for each one of these three methods using CM16, CM16+F, SDH16 and SDH16+F for texture classification are shown in table C.1.

Method	1	2	3
CM16	101	76	83
CM16+f	101	76	83
SDH16	103	94	85
SDH16+F	102	94	83

Table C.1: The best result for each of the three methods of grey level coarsening for four methods of texture classification. All numbers are out of 112.

As we can see the results of using the first method are better than for the other methods in all four methods of texture classification. The reason is:

- By using the non linear conversion of grey values (third method), we change the similarity of textures, because these four methods are very sensitive to changes in the textures.

Bibliography

[1] N. Abbadeni, D. Ziou, and W. Shengrui. Autocovariance-based perceptual textural features corresponding to human visual perception. *Proceedings of the 15th International Conference on Pattern Recognition*, 3:901–904, Sept 2000.

[2] L. Alparone, F. Argenti, and G. Benelli. Fast calculation of co-occurrence matrix parameters for image segmentation. *Electronics Letters*, 26:23–24, Jan 1990.

[3] K. Andersson and H. Knutsson. Continuous normalized convolution. *Proceedings of IEEE International Conference on Multimedia and Expo*, 1:725–728, Aug 2002.

[4] F. Argentini, L. Alparone, and G. Benelli. Fast algorithms for texture analysis using co-occurrence matrices. *IEE Proceedings, Part F: Radar and Signal Processing*, 136(7):443–448, 1990.

[5] P. Brodatz. *Texture: A photographic album for artists and designers.* New York: Dover, 1966.

[6] K.l. Chan and K. McCarty. Aspects of the statistical texture analysis of medical ultrasound images. *IEE Colloquium on Ultrasound Instrumentation*, pages 3/1–3/3, May 1990.

[7] Y.Q. Chen, M.S. Nixon, and D.W. Thomas. Texture classification using statistical geometrical features. *Proceedings of IEEE International Conference on Image Processing*, 3:446–450, Nov 1994.

[8] D.A. Clausi. Comparison and fusion of co-occurrence, gabor and mrf texture features for classification of sar sea-ice imagery. *Atmosphere-Ocean*, 39(3):183–194, 2001.

[9] D.A. Clausi and Y. Bing. Comparing cooccurrence probabilities and markov random fields for texture analysis of sar sea ice imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 42(1):215–228, Jan 2004.

[10] D.A. Clausi and Z. Yongping. Rapid determination of co-occurrence texture features. *IEEE 2001 International on Geoscience and Remote Sensing Symposium*, 4:1880–1882, July 2001.

[11] D.A. Clausi and Y. Zhao. Rapid extraction of image texture by co-occurrence using a hybrid data structure. *Computers and Geosciences*, 28(6):763–774, July 2002.

[12] F.S. Cohen, Z. Fan, and M.A. Patel. Classification of rotated and scaled textured images using Gaussian Markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):192–202, Feb 1991.

[13] C. Curcio, K. Sloan, R. Kalina, and A. Hendrickson. Human photoreceptor topography. *Journal of Comparative Neurology*, 292:497–523, 1990.

[14] T. Dacheng, L. Xuelong, Y. Yuan, Y. Nenghai, L. Zhengkai, and T. Xiao-ou;. A set of novel textural features based on 3d co-occurrence matrix for content-based image retrieval. *Proceedings of the Fifth International Conference on Information Fusion*, 2002.

[15] R.W. Ditchburn, D.H. Fendler, and S. Mayne. Vision with controlled movements of the retinal image. *Journal of Physiology*, 145:98, 1959.

[16] D.S. Early and D.G. Long. Image reconstruction and enhanced resolution imaging from irregular samples. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):291–302, Feb 2001.

[17] N.G. Fedotov and L.A. Shulga. Enhancing intellectual power of recognition systems based on new pattern recognition theory. *Artificial Intelligence Systems*, 1:192–197, Sept 2002.

[18] M.M. Galloway. Texture classification using gray level run length. *Computer Graphics and Image processing*, 4:172–179, 1975.

[19] M.D. Ganis, C.L. Wilson, and J.L. Blue. Neural network-based systems for handprint ocr applications. *IEEE Transactions on Image Processing*, 7(8):1097–1112, 1998.

[20] P. Garcia and M. Petrou. The use of boolean model for texture analysis of grey images. *Proceedings of 14th International Conference on Pattern Recognition*, 1:811–813, Aug 1998.

[21] P. Garcia-Sevilla and M. Petrou. Analysis of irregularly shaped texture region. *Computer Vision and Image Understanding*, 84:62–76, August 2001.

[22] D.D. Giusto and S. Fioravanti. Estimation of qth-order fractal dimensions. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1994.

[23] J.F. Haddon and J.F. Boyce. Co-occurrence matrices for image analysis. *Electronics and Communication Engineering Journal*, 5(2):71–83, April 1993.

[24] G.M. Haley and B.S. Manjunath. Rotation-invariant texture classification using a complete space-frequency model. *IEEE Transactions on Image Processing*, 8(2):255–269, 1999.

[25] R.M. Haralick. Statistical and structural approaches to texture. *Proceeding IEEE*, 67(5):786–804, May 1979.

[26] R.M. Haralick. *Handbook of Pattern Recognition and Image Processing, T.Y. Young and K.S. Fu(eds)*, chapter Statistical image texture analysis, pages 247–279. Academic Press, 1986.

[27] C.G. Healey and J.T. Enns. Building perceptual textures to visualize multidimensional datasets. *Proceedings of Visualization*, pages 111–118, Oct 1998.

[28] A. Hoogs, R. Collins, and R. Kaucic. Classification of 3d macro texture using perceptual observables. *Proceedings of the 16th International Conference on Pattern Recognition*, 4:113–117, 2002.

[29] http://www.ux.his.no/~tranden/brodatz.html. *Brodatz Album.*

[30] F. Hussain, N. Kehtarnavaz, F. Kallel, and J. Ophir. Texture analysis to characterize strain distribution in elastograms. *Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems*, pages 124–129, June 1999.

[31] Q. Iqbal and J.K. Aggarwall. Applying perceptual grouping to content-based image retrieval: building images. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:42–48, June 1999.

[32] A. Jain and L. Hong. On-line fingerprint verification. *Proceedings of the 13th International Conference on Pattern Recognition*, 3:596–600, Aug 1996.

[33] B. Johansson, H. Knutsson, and G. Granlund. Detecting rotational symmetries using normalized convolution. *Proceedings of the 15th International Conference on Pattern Recognition*, 3:496–500, Sept 2000.

[34] A. Kadyrov and M. Petrou. The trace transform as a tool to invariant feature construction. *Proceedings of Pattern Recognition*, 2:1037–1039, Aug 1998.

[35] A. Kadyrov and M. Petrou. Object descriptors invariant to affine distortions. *British Machine Vision Conference*, 2:391–400, Sept 2001.

[36] A. Kadyrov and M. Petrou. The trace transform and its applications. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 23:811–828, August 2001.

[37] A. Kadyrov and M. Petrou. Affine parameter estimation from the trace transform. *Proceedings of Pattern Recognition*, 2:798–801, Aug 2002.

[38] A. Kadyrov, M. Petrou, and J. Park. Korean character recognition with the trace transform. In *Proceedings of the International Conference on Integration of Multimedia Contents*, pages 7–12, Chosun University, Gwangju, South Korea, November 2001. ICIM 2001.

[39] E.Y. Kim, S.H. Park, and H.J. Kim. A genetic algorithm-based segmentation of markov random field modeled images. *IEEE Signal Processing Letters*, 7:301–303, November 2000.

[40] J. Kittler, R. Ghaderi, T. Windeatt, and J. Matas. Face verification using error correcting output codes. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:750–760, Dec 2001.

[41] H. Knutsson and C.F. Westin. Normalized and differential convolution. *IEEE Proceedings CVPR on Computer Society Conference*, pages 515–523, June 1993.

[42] H. Knutsson, C.F. Westin, and C.J. Westelius. Filtering of uncertain irregularly sampled multidimensional data. *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, 2:1301–1309, Nov 1993.

[43] M. Kokare, B.N. Chatterji, and Biswas. Cosine-modulated wavelet based texture features for content-based image retrieval. *Pattern Recognition Letters*, 25(4):391–398, March 2004.

[44] V. Kovalev and M. Petrou. Multidimensional co-occurrence matrices for object recognition and matchin. *Graphical Models and Image Processing*, 58(3):187–197, 1996.

[45] K. Laws. *Textured Image Segmentation.* PhD thesis, University of Southern California, Electronic Engineering, 1980.

[46] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:319–324, June 2003.

[47] J.G. Leu. On indexing the periodicity of image textures. *Image and Vision Computing*, 19(13):987–1000, Nov 2001.

[48] A. Lewis, R.Garcia, and L. Zhaoping. Understanding cone distributions from eye movement. Technical report, Dept. of Psychology, University College London., 2003.

[49] S. Li, J.T. Kwok, H. Zhu, and Y. Wang. Texture classification using the support vector machines. *Pattern Recognition*, 36(12):2883–2893, December 2003.

[50] H. Long and W.K. Leow. A hybrid model for invariant and perceptual texture mapping. *Proceedings of the 16th International Conference on Pattern Recognition*, 1:135–138, Aug 2002.

[51] A. Low. *Introductory Computer Vision and Image Processing*. Mcgraw Hill Book Company, 1991.

[52] R. Manthalkar, P.K. Biswas, and B.N. Chatterji. Rotation invariant texture classification using even symmetric gabor filters. *Pattern Recognition Letters*, 24(12):2061–2068, August 2003.

[53] L. Middleton. The co-occurrence matrix in square and hexagonal lattices. *7th International Conference on Control, Automation, Robotics and Vision*, 1:90–95, Dec 2002.

[54] T.K. Moon and W.C. Stirling. *Mathematical methods and algorithms for signal processing*, chapter Introduction and foundations. Prentice hall, 1999.

[55] T. Mouroutis, S.J. Roberts, A.A. Bharath, and G. Alusi. Compact hough transform and a maximum likelihood approach to cell nuclei detection. *Proceedings of International Conference on Digital Signal Processing*, 2:869–872, July 1997.

[56] L.S. Ng, M.S. Nixon, and J.N. Carter. Texture classification using combined feature sets. *1998 IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 103–108, April 1998.

[57] P.P. Ohanian and R.C. Dubes. Performance evaluation for four class of texture features. *Pattern Recognition*, 25(8):819–833, 1992.

[58] C. Orrite, A. Roy, and Alcolea. Surface segmentation based on perceptual grouping. *Proceedings of the International Conference on Image Analysis and Processing*, pages 328–333, Sept 1999.

[59] S.E. Palmer. The psychology of perceptual organization: A transformational approach. *Human and Machine Vision (J. Beck, B Hope and A. Rosenfeld, Eds.)*, pages 269–339, 1983.

[60] A. Papoulis. *Probability, random variables, and stochastic processes*, chapter Asymptotic theorems, page 49. McGRAW-HILL international editions, 1991.

[61] J.S. Payne, L. Heppelwhite, and T.J. Stonham. Perceptually based metrics for the evaluation of textural image retrieval methods. *IEEE International Conference on Multimedia Computing and Systems*, 2:793–797, June 1999.

[62] J.S. Payne, L. Hepplewhite, and T.J. Stonham. Applying perceptually-based metrics to textural image retrieval methods. *Proceedings of SPIE on Human Vision and Electronic Imaging*, 3959:423–433, Jan 2000.

[63] J.S. Payne and T.J. Stonbam. Can texture and image content retrieval methods match human perception? *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 154–157, May 2001.

[64] M. Petrou and P. Bosdogianni. *Image processing:texture*. John Wiley, 1999.

[65] M. Petrou and A. Kadyrov. Features invariant to affine distortions from the trace transform. *Proceedings of International Conference on Image Processing*, 3:852–855, Oct 2001.

[66] M. Petrou and A. Kadyrov. Affine invariant features from the trace transform. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 26(1):30–44, 2004.

[67] R.W. Picard and T. Kabir. Finding similar patterns in large image databases. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 5:161–164, April 1993.

[68] R.W. Picard, T. Kabir, and F. Liu. Real-time recognition with the entire brodatz texture database. *Proceedings on Computer Vision and Pattern Recognition*, pages 638–639, June 1993.

[69] C.M. Pun and M.C. Lee. Rotation-invariant texture classification using a two-stage wavelet packet feature approach. *IEE Proceedings of Vision, Image and Signal Processing*, 148:422–428, Dec 2001.

[70] C.M. Pun and M.C. Lee. Log-polar wavelet energy signatures for rotation and scale invariant texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):590–603, May 2003.

[71] T. Randen and J.H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transaction on Pattern Analysis and Machine Inteligence*, 21(4):291–310, 1999.

[72] T. Reed and Du Buf. Review of recent texture segmentation feature extraction techniques. *Graphics Image Processing (CVGIP): Image Understanding Computer Vision*, 57(3):359–372, May 1993.

[73] Y. Rubnerand and C. Tomasi. Texture metrics. *IEEE International Conference on Systems, Man, and Cybernetics*, 5:4601–4607, Oct 1998.

[74] R.J. Schalkoff. *Digital Image Processing and Computer Vision*, pages 295–300. John Wiley and Sons Inc.

[75] N. Sebe and M.S. Lew. Wavelet based texture classification. *Pattern Recognition* 15^{th} *Internatinal Conference*, 3:947–950, 2000.

[76] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysing, and Machine Vision*, chapter 14, page 646. PWS(An Imprint of Brooks/Cole Publishing Company), 2 edition, 1998.

[77] S. Srisuk, M. Petrou, W. Kurutach, and A. Kadyrov. Face authentication using the trace transform. *Conference on Computer Vision and Pattern Recognition*, 1:305–312, June 2003.

[78] T.F. Syeda-Mahmood. Recognizing similarity through a constrained non-rigid transform. *Proceedings of the 13th International Conference on Pattern Recognition*, 1:617–621, Aug 1996.

[79] S. Tabbone and L. Wendling. Technical symbols recognition using the two-dimensional radon transform. *Proceedings of 16th International Conference on Pattern Recognition*, 3:200–203, Aug 2002.

[80] T.N. Tan. Rotation invariant texture features and their use in automatic script identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):751–756, July 1998.

[81] A. Teuner, O. Pichler, J.E. Santos Conde, and B.J. Hosticka. Orientation and scale-invariant recognition of textures in multi-object scenes. *Proceedings of the International Conference on Image Processing*, 3:174–177, Oct 1997.

[82] C.W. Therrien. *Decision estimation and classification, An introduction to pattern recognition and related topics*, chapter 9, page 144. John Wiely and Sons, 1989.

[83] F. Tomita and S. Tsuji. *Computer Analysis of Visual Textures.* Kluwer Academic Publisher, 1990.

[84] M. Unser. Sum and difference histograms for texture classification. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 8:118–125, 1986.

[85] H. Vafaie and N.G. Bourbakis. Tree grammar scheme for generation and recognition of simple texture paths in pictures. *Third International Symposium on Intelligent Conference on Image Processing.*, pages 201–206, 1988.

[86] C. Vazquez, J. Konrad, and E. Dubois. Wavelet-based reconstruction of irregularly-sampled images: application to stereo imaging. *Proceedings of 2000 International Conference on Image Processing*, 2:319–322, Sept 2000.

[87] D. Voth. Face recognition technology. *IEEE*, 18(3):4–7, May-June 2003.

[88] C.F. Westin, K. Nordberg, and H. Knutsson. On the equivalence of normalized convolution and normalized differential convolution. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 5:457–460, April 1994.

[89] A.L. Yarbus. *Eye Movement and Vision.* Plenum Pres, 1967.

[90] T.Y. Young and K.S. Fu. *Handbook of Pattern Recognition and Image Processing.* Academic Press, 1986.

[91] K.C. Yow and R. Cipolla. A probabilistic framework for perceptual grouping of features for human face detection. *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 16–21, Oct 1996.

[92] F. Zhou, J.F. Feng, and Q.Y. Shi. Texture feature based on local fourier transform. *Proceedings of International Conference on Image Processing*, 2:610–613, Oct 2001.