# TEXTURE IDENTIFICATION USING ARTIFICIAL NEURAL NETWORKS AND 2D-AUTOREGRESSIVE MODEL

HE XU

# TEXTURE IDENTIFICATION USING ARTIFICIAL NEURAL NETWORKS and 2D-AUTOREGRESSIVE MODEL

BY

© He Xu

A thesis submitted to the School of Graduate

Studies in partial fulfillment of the

requirements for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

December 1993

St. John's Newfoundland

ISBN 0-315-91588-9

# Abstract

As an important aspect of image analysis, texture identification has been pursued by many researchers. Among techniques developed, the approach of modeling texture images through a 2-D Autoregressive (AR) Model is of special interest. The major problem with the modeling methods is the estimation of parameters due to the intensive amount of computation involved. From a parallel computing perspective, parameter estimation can be implemented by learning procedure of a neural network, and texture classification can be mapped into a neural computation. A multilayer network is proposed which consists of three subnets, namely the input subnet (ISN), the analysis subnet (ASN) and the classification subnet (CSN). The network obtains the classification capability through an adaptive learning procedure. In the processing phase, images proceed through the network without the preprocessing and feature extraction required by many other techniques.

An integrated texture segmentation technique is proposed to segment textured images. The technique is implemented by comparing local region properties, which are represented by a 2-D AR model, in a hierarchical manner. It is able to grow all regions in a textured image simultaneously starting from initially decided internal regions until smooth boundaries are formed between all adjacent regions. The performances of the classification and segmentation techniques are shown by experiments on natural textured images.

# Acknowledgments

I wish to express my thanks to my supervisor Dr. Siwei Lu for his insightful guidance, constructive suggestion and enthusiasm. Without his contribution, it would be impossible to give this thesis its current quality.

I would like to thank the systems support staff for providing all the help and assistance during my research.

I am also very grateful to the administrative staff who have helped in one way or another in the preparation of this thesis.

In addition, I would like to acknowledge the financial support received from the Department of Computer Science and the School of Graduate Studies.

Special thanks are due to Todd Wareham for his valuable comments and suggestions.

I would also like to thank my fellow graduate students, and in particular to Zhengqi Lu, Zhimin Shi, Helmut Roth, Sun Yongmei, Chen Hao and Guiming Chen for their help and encouragement during the preparation of the thesis. Finally my thanks also go to Patricia Murphy and Elaine Boone for their kind help and assistance.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 The Texture Analysis Problem

Texture is the term used to describe the organized area phenomena which exist on the surfaces of objects. Textures can be easily observed from natural images, such as images of grass lands, leaves of trees, a patch of sandy beach and many other outdoor scenes. Further examples are the textures that can be found from satellite mutispectral images, photomicrographs of ores and minerals, electron micrographs and microscopic images in biological or medical studies.

A universal texture definition, however, is difficult to give because of the diversity of natural and artificial textures. Generally, texture can be considered as "a structure composed of a large number of more or less ordered similar elements or patterns without one of these drawing special attention" [1]. In a uniform texture image, gray values of pixels exhibit some kind of homogeneity. It is believed that the variation of gray values in one subregion of a texture image will show a similar pattern with those of other subregions of the image.

Texture image classification and segmentation are the two main objectives in texture analysis. Texture image classification identifies an input texture sample as one of a set of possible texture classes, while the aim of texture segmentation is to divide an image with different textures into subregions that have uniform textures. For a texture analysis process, the first and the most important step is to extract texture features which can generalize the textural characteristics of the original images. The requirement for this step is to choose the textural features to be as compact as possible and as discriminating as possible. At the meanwhile, as a texture analysis process tends to consume long computational time, an efficient feature extraction is very critical for a method to be applied in real application.

Structural and the statistical features have been used for texture feature representation. Structural approaches appear to be appropriate for periodic textures with a low noise level and are not considered to be useful in real application. Most statistical features are based on tonal properties or pattern properties. They usually strong in measuring textures from one aspect, but deficient in describing others. Other statistical methods represent textures by stochastic models. These methods build more complete statistical models for textured images, however, are often limited in application due to the computational burden in model parameter estimation.

In this thesis, the approaches of texture classification and texture segmentation are explored with 2-D Autoregressive (AR) model representation and neural network implementation. Special efforts are made in seeking a natural combination between 2-D Autoregressive (AR) model and neural network concepts to perform a texture analysis task efficiently. The main idea is motivated by two important characteristics of neural computation, namely the highly parallel execution and the adaptive learning ability. With the neural computation, the process of establishing a stochastic model for a given texture maps to a neural adaptive learning process. Compared to models estimated by least square error (LSE) technique, which is often used for stochastic model parameter estimation, models built by the neural computation have proved to be more adaptive to gray level variation and more tolerant to the possible noise contained in natural images.

A texture segmentation algorithm, which can be considered as an application of the designed neural network, is also proposed. Current texture segmentation algorithms can be classified to four categories, including algorithms based on estimation theory, clustering, edge detection, and region extraction. The segmentation algorithm proposed in this thesis is an integrated region extraction technique which combines the strengths of region splitting and merging and the region growing techniques for texture segmentation. The algorithm is superior to a region merging and splitting algorithm in its parallel nature. The regions produced do not depended on the sequence in which regions are merged. Some drawbacks of a region growing tèchnique, such as, seed part is decided in a supervised manner and a seed part only contains one pixel, are also overcome. To handle the situation in which the types of textures appearing in an image under investigation are not known beforehand, a type determining mechanism is designed to be used before the segmentation procedure to decide the textures included in the image.

## 1.2 Structure of the System

The proposed neural network consists of three subnets, namely the input subnet (ISN), the analysis subnet (ASN), and the classification subnet (CSN) (Figure 1.1). Each subnet consists of more than one layer of nodes. Every two adjacent subnets have forward connections, in which the output of each subnet serves as input to the next subnet in the network.

The input subnet accepts a texture pattern as input and distributes the normalized input pattern to the analysis subnet. The analysis subnet consists of a set of channels, each of which models a particular texture class by a 2-D AR model and produces an error value that measures the difference between an input texture pattern and the pattern generated by that channel using the AR model. The analysis subnet computes a set of total error values. The classification subnet decides, using a competition mechanism, to which texture class the input pattern belongs.

An overview of the proposed segmentation algorithm is given as a flowchart in

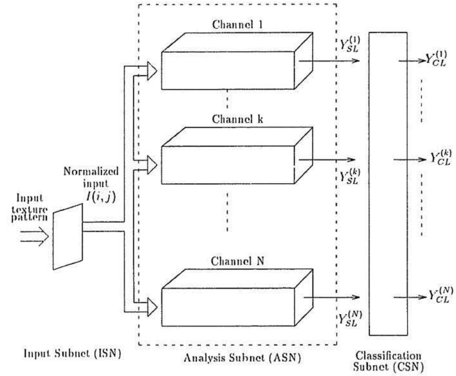


Figure 1.1: Overall structure of proposed neural classifier.

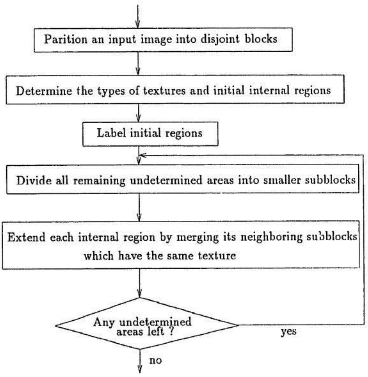

Parition an input image into disjoint blocks
Determine the types of textures and initial internal regions
Label initial regions
Divide all remaining undetermined areas into smaller subblocks
Extend each internal region by merging its neighboring subblocks which have the same texture
Any undetermined areas left ?
yes
no

Figure 1.2: Overview of the segmentation procedure.

Figure 1.2.

## 1.3 Organization of This Thesis

This thesis is organized into eight chapters. Chapter Two briefly surveys existing texture classification and segmentation algorithms. Chapter Three reviews typical neural networks and their application in image analysis and processing. Chapter Four introduces how the 2-D AR model can represent a two-dimensional random field defined on a textured image. The method of using neural computation to perform 2-D AR model parameter estimation is also addressed in this chapter. With the neural computation introduced in Chapter Four, the Chapter five describes the overall multilayer network structure and the computations for neurons at each level during the application phase of the network. In order to deal with images in the orientations different from training samples, Chapter Six provides two modified versions of the neural network to recognize textured images in arbitrary orientation. As an application of the neural network in Chapter Five, a segmentation algorithm, which is an integrated region extraction technique, is proposed in Chapter Seven. The performance of the segmentation algorithm is also discussed. The conclusion and discussion are given in Chapter Eight.

# Chapter 2

# Survey Of Texture Analysis Techniques

## 2.1 Introduction

This chapter gives a brief review of existing texture features and texture classification techniques. Texture segmentation techniques, including methods based on estimation theory, edge detection, clustering, edge extraction and region extraction, are also discussed.

## 2.2 Texture Features

Pure structural viewpoint assumes that a texture is described by some primitives following by a placement rule. In this view, a texture is considered to be generated by primitives which occur repeatedly according to the placement rule. Such methods are greatly limited because the types of regular textures that can be described by structural features are seldom encountered in the real world. Hence, structural features have been of little interest to most researchers.

As a result, the major focus in texture analysis over the last twenty years has been on statistical approaches. From the statistical point of view, texture is regarded as a sample of a probability distribution on the image space. Texture is either defined by statistic features or a stochastic model which is characterized by a set of parameters.

### 2.2.1 Commonly Used Statistical Features

In the late 1970's and early 1980's, most statistical approaches were based on the extraction of statistical features such as spatial frequency and probability densities of various local or co-occurrence properties. Some of those features have been successfully used in specific applications such as classification of geological terrain images, satellite images of clouds, and some segmentation experiments.

In the following paragraphs, seven types of commonly adopted statistical features, together with their advantages and limitations, are briefly discussed.

#### 2.2.1.1 Features from Digital Transform Techniques

In the digital transform techniques, an image is reexpressed into a new coordinate system, such as the sin-cosine basis set in the Fourier transform or the Walsh function basis set in the Hadamard transform. The coefficient values in the transformed image relate to spatial frequency. Since fine textures are rich in high frequencies and coarse textures are rich in low frequencies, the information from the transformed image in the spatial space can be used to identify textures. Several types of transforms, including Fourier, Hadamand and Slant transforms, have been used in this manner to do texture analysis. It is reported that no big difference has been found among the different transform methods [2].

The power spectrum can also be adopted to measure textures. In power spectrum method, three features are commonly used: (1) annular-ring geometry, which gives a total contribution of light energy of one frequency component, independent of direction; (2) wedge sampling geometry, which gives the total energy of one direction independent of frequency; and (3) parallel-slit sampling geometry, which measures the energy transmitted through a slit of specified length and width when the Fourier plane is rotated to a specified angle.

#### 2.2.1.2 Features from the Autocorrelation Function

The autocorrelation function can be used to describe textures because it indicates the sizes of primitives. In a textured image, the autocorrelation function will drop off and rise again periodically. If the primitives of the image are relatively large, its autocorrelation will drop off slowly with the distance. If the primitives are small, the autocorrelation will drop off quickly with distance. The spatial information can therefore be characterized by the correlation coefficient.

This approach is related to the Fourier transform technique in that the autocorrelation function and the power spectral density function are the Fourier transforms of each other. Experiments have also shown that both methods have similar performance [8].

#### 2.2.1.3 Features from Spatial Gray-Tone Dependence (Co-occurrence)

The spatial gray level dependence method is based on the estimation of the joint condition probability density function, $P(i,j|d,\theta)$. Each $P(i,j|d,\theta)$ denotes the probability of concurrence of a pair of gray levels $(i,j)$ at distance $d$ and angle $\theta$. The estimated values can be written in a matrix form, which is called the concurrence matrix. As such matrices of gray level spatial dependence frequencies depend on both the angular relationship and the distance between gray values in an image, they can capture both the direction and the size information. A variety of measures can be employed to extract useful textural information from those matrices, such as energy, contrast, correlation, entropy and local homogeneity [3].

### 2.2.1.4 Features from Generalized Gray-Tone Spatial Dependence

This method is similar to the co-occurrence method in that both describe textures by estimating the joint probability distribution. The co-occurrence method calculates the probability of two pixels with given grey values and distance, while this method describes the local texture information by calculating the joint probability distribution of a neighbor with eight directions.

The generalized Grey Tone Spatial Dependence considers an arrangement of gray values of a pixel neighborhood as a primitive. For example, a $3 \times 3$ neighborhood with 4 gray levels for each gray value will have $4^9$ different primitives. Histogram statistics which indicates the frequency of the occurrence of all the primitives in an image can then reveal the texture information. The problem with this technique is the heavy computation due to the high dimensionality for the probability distribution.

Recently, a texture spectrum approach was proposed which is a variation of Grey Tone Spatial Dependence [28]. In this approach an eight element neighborhood $E_1, E_2, E_3 ...... E_8$ is defined as

$$E_i = \begin{cases} 0 & \text{if } V_i < V_0 \\ 1 & \text{if } V_i = V_0 \qquad i = 1, ..., 8. \\ 2 & \text{if } V_i > V_0 \end{cases}$$

where $V_i$ is the grey value of a neighborhood element and $V_0$ is the central element. In this way, the number of primitives can be reduced but some of the detail information is lost as well.

#### 2.2.1.5 Features from Texture Edgeness

Textures can be described by the number of edges within per unit area [39] because coarse textures have a small number of edges per unit area and fine textures have a large number of edges per unit area. By using this approach, a gradient image can be first obtained by using Robert's gradient measure or other gradient extraction techniques. The average value of the gradient in the image can then be calculated to characterize a texture.

#### 2.2.1.6 Features from Run Length

A grey level run is a set of consecutive pixels with the same grey level value. Given the direction of the run, a matrix can be constructed in which each element $P_{(i,j)}$ indicates the number of runs with length j for grey value i. For a coarse texture, relative long runs would occur relatively often. In contrast, short runs would occur more frequently for fine textures. Several features can be defined, i.e., long runs emphasis, gray distribution, run length distribution and run percentage [8]. This method is very sensitive to noise [2].

#### 2.2.1.7 Features from Filter Masks

A set of masks can be used to characterize textures [9]. The approach consists of two steps. In the first step, a zero sum mask is used to convolve a whole input image. The convolution masks are designed to be sensitive to structures such as edges and spots. In the second step, a measure called texture energy is evaluated at each pixel in the convolved image over a large window by summing the values in the window. For a particular texture, the texture energy is conjectured to be within a certain range, so different texture energies can distinguish different textures.

### 2.2.2 A Comparison of Statistical Features

Many of the features described above have been used in special applications. A wide class of images have been tested by using the gray level co-occurrence method and the gray level tone features [4, 5, 19, 8]. It was concluded that features based on the gray level co-occurrence and the gray level tone perform much better than features from Fourier transforms [1, 8]. Therefore it is believed that texture patterns are more appropriately modeled in the space domain rather than the frequency domain since both gray level co-occurrence and gray level tone characterize the spatial relationships of gray levels in an image. One drawback for the co-occurrence method is that it consumes a large amount of computation because many matrices have to be computed. Another limitation is the lack of any theory to guide for choosing a particular set of features. For gray level tone features, the computation time can be extremely long because of the histogram technique adopted.

Features from texture edgeness are poorer than that of the concurrence method but better than that of the frequency method [1, 8].

The gray level running length method is very sensitive to noise, so it is not considered for gray value images. It is, however, suitable for binary images [2].

From the computational efficiency point of view, features from filtered masks can be regard as low cost features. The limitation is that a single mask is normally ideal in reflecting only one aspect of texture properties, e.g., edge detection or spot detection. Since natural images are often complex, it is hard to distinguish all textures by using one mask.

### 2.2.3 Describing Textures by Stochastic Models

Another statistical approach is to model a texture as a stochastic random field. In this approach, a textured image is viewed as a realization of a stochastic process which can be specified by a set of parameters. These parameters can then be used to identify textures in classification and segmentation. Since 1980, many researchers have been interested in using stochastic models to represent textures. The random models most commonly used are the Markov Random Field (MRF) [10, 11, 12], the Gibbs Random Field (GRF) [22, 24, 23] and the 2-D Autoregressive (AR) [14, 15, 16, 30] models.

An early study of stochastic approaches for texture identification used MRFs [10]. These are multidimensional generalizations of Markov Chains, which are defined in terms of conditional probabilities based on spatial neighborhoods. There are different orders of neighborhoods. Each neighborhood corresponds to a clique class (a clique is a graph whose vertex set is composed of vertices such that each one is a neighbor of all others). A set of parameters associated with the cliques of a given neighborhood configuration determine a MRF. These parameters constitute a feature space and thus can be used for texture classification and segmentation. The major disadvantage with these models is that the estimation of these parameters is very difficult.

Using an AR model, a textured image can be synthesized as a linear combination of the neighboring values with random noise values. The coefficients of these linear combinations could be considered as a set of features which explicitly express the spatial relation of each pixel with its neighbors. The estimation of the AR model is less difficult than that of MRF or GRF [14, 15, 16, 30].

In any stochastic model texture analysis approaches, three problems to be considered are: (1) choosing the model and the order of the model (or the appropriate neighborhood); (2) estimating the parameters of the model; and (3) choosing the appropriate classification or segmentation techniques. The first two problems are closely related with each other. A higher order neighborhood certainly increases the accuracy of the chosen model, but at the same time it also makes the stochastic estimation of the parameters extremely difficult. Different techniques are used to make such computations possible.

Due to the complexity of these models, choosing an appropriate neighborhood becomes very difficult. Some possible procedures, such as Akaike's information criterion, are used to decide a neighborhood [26]. A very heavy computation is involved in those procedures. Usually, a fixed size neighborhood is empirically determined.

For model parameter estimation, an early used technique is the coding method introduced by Besag in [20]. The coding method is basically a maximum likelihood estimation that yields parameter estimates which maximize the conditional joint distribution. By using this approach, an image is divided as two subsets, such that points of the two subsets are evenly intersected on the image. A maximum likelihood estimation from one subset can be obtained by the condition that the other subset is fixed; estimation of the other subset can then be achieved in the same way. The result is given by combining the two procedures properly. This method results in low efficiency because only a subset of the data is used.

An alternative parameter estimation was proposed for GRF [22] which consists of the histogram technique and a standard linear least square estimation. This approach expresses the sum of the potential functions by a set of components corresponding to all cliques contained in the neighborhood, and then using a histogram to estimate the joint distribution and obtain the parameters. The computation is easier than that of the coding method.

Least square estimation (LSE) and maximum likelihood estimation (MLE) are commonly used methods for parameter estimation. The results obtained by LSE and by MLE are nearly the same, however, the former is computationally easier.

Hence least square estimation is more frequently used in parameter estimation.

## 2.3 Classification Techniques

In most cases, a non-parametric classifier is used in texture classification since the pattern class distributions are not known. The nearest neighbor (NN) approach is frequently adopted. This decision rule assigns a pattern to the nearest class among all possible neighbors, which are computed from training samples. Euclidean distance is commonly used as the distance metric.

There have been attempts to design orientation independent classification algorithms by averaging features over different directions [18, 19]. For example, it has been suggested that features defined on gray level co-occurrence matrix can be averaged over four matrices computed from $0^o$, $45^o$, $90^o$ and $135^o$. However, the performance of such features has not been tested in experiment which uses differently oriented samples of textures. A stochastic texture model called the circular symmetric autoregressive mode was proposed in [18], which utilizes a circular neighborhood. The elements of this circular neighborhood can be thought of as symmetrical points located on a unit radius circle. Three rotation invariant features can in turn be extracted from this circular neighborhood.

## 2.4 Segmentation Techniques

Segmentation is the partitioning of an image into regions that are homogeneous with respect to some characteristics. Texture segmentation techniques can be categorized into four classes: (1) estimation theory based segmentation; (2) edge based segmentation; (3) clustering based segmentation; and (4) region based segmentation.

### 2.4.1 Based on Estimation Theory

Three segmentation techniques using Maximum *A Posteriori* (MAP) estimation have been developed, namely dynamic programming, stochastic relaxation (simulated annealing) and deterministic relaxation.

When using MAP as the estimation criterion, the object is to have an estimation rule which yields $x$ so that maximizes a posteriori distribution $P(X = x|Y = y)$ for a given $y$. The difficulty in determining $x$ is due to the fact that the maximization is to be done over $M^{N_1 N_2}$ possible configurations in an M-valued image of size $N_1 \times N_2$. To make the computation possible, approximations are adopted in each technique from different viewpoints.

#### 2.4.1.1 Dynamic Programming

MAP estimation techniques are used to develop suboptimal but computationally tractable segmentation algorithms to segment binary textured images [21]. Such algorithms use two stages of dynamic programming. In the first step, a generalized dynamic programming algorithm is applied to each row of the image, yielding a set of candidate segmentations for each row. In the second step, a final segmentation is formed from the partial results calculated in the first stage.

Dynamic programming can be extended to multilevel images by adopting Gibbs Distribution (GD) [22]. A hierarchical GD model is composed of two levels. At the higher level, a GD model is used to divide image pixels into regions with similar features. That is, if a pixel is of a certain type, its neighboring pixels should also have a high probability of being the same type. At the lower level, the features are modeled by another set of GD. Based on this hierarchical model, the MAP estimates for the processed region are carried out recursively. Because of the enormous computational complexity, the recursive algorithm is arranged as follows: first, a D-row strip is processed and an estimation for the strip is obtain by dynamic programming. Only keep the estimation for the first row and discard the rest. Then the strip consisting of rows from 2 to $D+1$ is processed in the same way, and so on, until the estimation for the whole image is obtained.

However this method is computationally tractable only for small grey value scales. For images more than four grey levels, hierarchical segmentation is constructed as a sequence of binary segmentation processes, i.e., an image is first segmented into two region types, each region is then segmented into two subregions, and so on. This algorithm is reported to be successful on images containing

2 or 3 texture types, where each texture type has 2 or 4 grey levels.

### 2.4.1.2 Stochastic Relaxation

A hierarchical stochastic model based on the Gibbs Distribution for texture image segmentation was presented in [24]. Stochastic relaxation and simulated annealing were used for computing the MAP estimation for segmentation.

A Gibbs Distribution has the following representation

$$\pi_{(w)} = \frac{e^{\frac{-U(w)}{T}}}{z}$$

where T stands for a computational analogue for "Temperature". High temperatures indicate a loose coupling between neighboring pixels and a chaotic appearance. At low temperature the coupling is tighter and the image appears more regular. Simulated annealing is a process that slowly decreases the temperature $T$ and forces the system into a low energy state.

This algorithm can converge to a global maximum. However, convergence to a global minimum is extremely slow and hence is not practically implementable. In practice, it either converges to a local maximum or terminates before it converges at all.

### 2.4.1.3 Deterministic Relaxation

Another kind of relaxation, named deterministic relaxation, was introduced in [25]. Let $y = \{y_{ij}\}$ be the observed image and $x$ be the segmentation of image $y$. The objective of segmentation is to maximize $P(X = x|Y = y)$ with respect to $x$ for a given $y$. In the deterministic relaxation problem, the scheme to maximize $P(X = x|Y = y)$ is: select a pixel $(i, j)$ and assume all neighborhood pixels $(k, l) \neq (i, j)$ are to be fixed at their optimum values; then update $x_{ij}$ by the value that maximizes $P(X = x|Y = y)$. Then move to other pixels and update each one sequentially. After many iterations the algorithm converges to the maximum of $P(X = x|Y = y)$. This deterministic relaxation is faster in convergence but not necessarily converges to a global one. In order to avoid to a local maximum, the method of varying the neighborhood during each iteration of the relaxation was used to make the convergence closer to the global maximum.

In a comparison described in [25], the results produced by deterministic relaxation, stochastic relaxation, and dynamic programming algorithms showed to be comparable. The deterministic relaxation with varying neighborhoods was demonstrated to be faster than stochastic relaxation and appears to be less possible to converge to local maximum than the deterministic relaxation is. From the experiments presented in [21], [22], [24], and [25], segmentation techniques based on estimation theories are mostly applied to images with only 2 to 4 gray levels and containing two or three textures. Besides, they require a large number of passes over the image and are extremely time consuming.

### 2.4.2 Based on Edge Detection

In object identification, edges can be extracted by identifying abrupt gray level changes in an image. For a textured image, edges between texture regions can be detected by identifying the changes in texture feature values. In extending conventional edge detection methods to perform texture segmentation, the texture features are computed by using overlapping or non-overlapping windows over the image, and the texture features extracted from each window are then transformed into real values. An edge detection operator (such as the Roberts operator ) is then applied to the transformed new image to obtain a gradient image. Finally, a predefined threshold is used to decide texture edges.

Several edge based segmentation techniques exist [37, 36, 38]. The main difference between those edge based segmentation techniques is the choice of the texture features.

Some drawbacks with this approach are: (1) at the place where the change of feature images between two regions is not abrupt enough, a gap occurs and the edge might be lost. Hence closed edges cannot be produced and linking procedure is needed to connect the incomplete edges into closed boundaries; and (2) the threshold value which is used to decide edges on gradient images is hard to be defined because of the complex boundaries between various textures.

### 2.4.3 Based on Clustering

In a textured image, each class of homogeneous regions is assumed to form a distinct cluster in the space of features selected. A clustering technique extracts texture features for each pixel in an image, and a clustering method is then used to group the points in the feature space into clusters [34, 35].

Some limitations of this technique are: (1) large amounts of computation time are required because features have to be extracted for each pixel in an image; (2) such clustering produces noisy boundaries because the neighboring information is not considered.

### 2.4.4 Based on Region Extraction

Region extraction is performed by comparing local region properties extracted over an appropriate window. This type of method includes the splitting and merging approach and the region growing approach.

A fundamental problem in a region based approach is the choice of an appropriate window size over which the local properties are extracted. If a small window is used, the properties are not reliable and it is difficult to accurately detect texture boundaries. If a textured image is measured over a large window, it is hard to find uniform texture regions.

A splitting and merging technique using a pyramid structure was described in [40]. In this approach, a block is determined as a uniform region if its contained blocks are considered to be similar in texture properties. Otherwise, the block is split and the process is repeated for each of its contained blocks. The segmentation is completed when no undecided block is left on the image. The boundaries produced by this method tend to be a sawtooth shape because of the difficulty of deciding the region properties when blocks are very small.

In the region growing technique, regions are extended from some starting points until the boundaries of regions are reached. A region growing technique was introduced in [42], in which each region in an image is grown starting from a selected position by using a modified random walk technique. In this walk, a move is made from a pixel to its neighbor if the similarity between the two pixels satisfies a predefined threshold. For each pixel in the image, a count is used to record how many times that pixel is visited. Regions are then extracted by thresholding the visit-count array. Some limitations of this approach are: (1) long computation time is required since features have to be computed for each pixel in the image; (2) large memory is required because a counter has to be used for each pixel during the random process; (3) some interior points in the regions are mislabeled because of the random nature of pixel visiting.

Compared to the clustering and the edge detection methods, the results produced by region based methods tend to be more stable and less noisy. One difficulty of region extraction methods is that there is no simple way to properly incorporate the local feature information. Another common limitation is that a great deal of computation time is required because the procedures are performed in an iterative manner.

The principles of primary texture classification and segmentation techniques, their strengths and weaknesses, as well as the performances have been reviewed in this chapter. A general review of current neural networks will be given in the next chapter.

# Chapter 3

# Neural Networks

## 3.1 Introduction

A review of important neural networks is presented in this chapter. In the past decade, there have been increasing interest in artificial neural networks, due to the learning capability and massive parallelism. A neural network is specified by a net topology, node characteristics, and a training rule which specifies how node-connection weights should be adapted during the learning process. Information is encoded in a neural network in a distributed fashion, and the ability to give correct output for each input is learned by training on selected samples according to the training rule.

Neural network systems have received extensive attention in the area of pattern recognition in recent years and have been applied to many areas, such as character recognition, image analysis, natural language processing, and speech recognition. The focus of this chapter is on those networks which have found their application in image processing and image analysis.

## 3.2 Back Propagation Neural Network

The back-propagation algorithm provides a mechanism for the development of multilayer networks composed of units with sigmoidal activation functions [54]. It assigns a strong discrimination power to multilayer networks. Theoretically, multilayer networks can be considered as nonlinear maps with the elements of the weight matrices as parameters. A typical multilayer network includes an input layer, an output layer and one or more hidden layers. The weights between two layers are adjusted to minimize a suitable function of the error between the output of the network and a desired output of the network. A discontinuous mapping such as a nearest neighbor rule is used at the last stage to map the input set into points in the range space corresponding to output classes. In recent years, back propagation networks have repeatedly shown their high discrimination ability. These networks are capable of recognizing more complex patterns than those that can be recognized by the classical image analysis operations.

Multilayer networks trained by the back-propagation algorithm have proved to be extremely successful at solving pattern recognition problems [47, 49]. In image analysis, back-propagation has been applied to target recognition, character recognition, image compression, and image coding [50, 51, 52, 53].

## 3.3 Competitive Neural Network

A competitive neural network consists of laterally interconnected nodes. The interaction in a competitive neural network includes positive interaction from a node to itself and negative interaction between a node to other nodes. Typical competitive networks are the Maximum network and Hopfield networks [55].

In a discrete Hopfield Network, all nodes in the net are connected to all others. A number of state vectors can be stored in such a net. If a distorted pattern is presented to the net, the net has the ability to recall the original pattern [64, 65]. The continuous Hopfield network is well suited to class optimization problems which can be described in terms of a energy function [65]. In this type of application, a problem is represented as a set of constraints, and energy function is designed according to those constraints. A weighted matrix is then constructed according to this energy function, and a differential equation that describes the dynamics of each neuron in the network is derived from this weighted matrix. The best solution can be derived by using the differential equation to minimize the configuration energy until the network converges to the lowest energy configuration.

Hopfield networks provide an alternative approach to pattern recognition, and have been used in object recognition [67], image segmentation [66] and the correspondence problem in stereo vision [68]. Limitations of the Hopfield networks include: (1) in discrete Hopfield net, the number of patterns that can be stored and recalled is severely limited by the number of nodes required; and (2) for continuous Hopfield networks, it is often very difficult to define a weight matrix to well represent constrains of an application [55].

## 3.4 Kohonen Self-Organizing Network

Kohonen's self-organizing network [60] consists of a single layer of neurons. Those neurons are highly interconnected within the layer as well as to the outside world. The Kohonen network can be considered a mapping in which points in N-dimensional pattern space are mapped into a smaller number of points in an output space. This mapping is achieved in a self-organized manner. The network has been used in object segmentation [69].

## 3.5 Adaptive Resonance Theory

Adaptive resonance theory (ART) [59, 47] is an extension of competitive learning schemes in which the learning of new information does not destroy old information. A feedback mechanism exists between the competitive layer and the input layer of a network. This feedback mechanism automatically switches between the stable and the plastic modes. If the network has learned previously to recognize an input vector, then a resonant state will be achieved quickly when that input vectors is presented. If no match is found, the network will enter a resonant state in which the new pattern will be stored for the first time.

Two architectures named ART1 and ART2 have been developed for binary input vectors and for gray-scale patterns respectively [48].

## 3.6 Other Networks

Several multilayer networks have been built for special purposes. Neocognitron is one such network which was designed for character recognition [57, 58]. The model is a hierarchical network consisting of many layers of cells, and variable connections between the cells in adjoining layers. At low stages, simple features such as horizontal and vertical short line segments are extracted and at high stages more complicated features are obtained based on the result of early stages. By supervised learning or unsupervised learning, the network can obtain the ability to recognize input patterns according to the differences in their shapes.

Other multilayer networks designed for image analysis include a five stage network designed for invariant object recognition of binary images [61], mutilayer Gabor-based Networks for image segmentation [62], and a hierarchical neural network for edge enhancement [63].

Typically, multilayer networks have the following limitations: (1) learning is slow when complex decision regions are required; and (2) a very large number of nodes can be required by a complex recognition task. In this case, it may be impossible to reliably estimate weights from the training data.

## 3.7 Learning Methods

There are two types of learning procedures, namely supervised learning and unsupervised learning. In supervised learning, a network is provided with a number of training sets in which each set consists of an input pattern and a desired output. The neural network updates its internal weights according to both the input and output data so as to produce the correct output for each input pattern [54, 55, 56]. Unsupervised learning assumes no correct responses are provided to a learning procedure. A network updates its weight according to the input patterns under certain assumptions about the nature of the data. Unsupervised learning usually refers to the learning procedure in self-organizing nets, for instance, the Adaptive Resonance Theory (ART) [59] and the Kohonen feature map [60]. The limitation of unsupervised learning is that it cannot learn arbitrary functions.

Most multilayer networks adopt supervised training [54, 56, 57]. In supervised learning, a network is provided with input-output pattern pairs. For each pair, the network updates its internal weights to decrease the difference between the actual network's output and the desired output. The weights are iteratively adjusted until the network can produce the desired output in response to each input pattern. Usually, in order to respond correctly to a set of input patterns which require different outputs, all pairs of input patterns and outputs will be alternately presented many times to a network.

The supervised training of a multilayer network typically involves a forward propagation phase followed by a backward propagation phase. The forward propagation phase starts by providing an input pattern to the input of a network, propagating the input forward through the hidden layers, and obtaining an output at the output layer. In the backward propagation phase, the difference between the output produced by the network and the desired output is calculated. A weight changing rule is then applied backward through the layers to change the internal weights of the network according to the amount of the difference value.

The square of error is usually utilized to measure how close a network is to obtaining the desired output. This value is decreased as the learning procedure proceeds. A learning procedure eventually stops at the state of convergence when the value of the square of error is equal to or less than a predefined value.

A general survey of important neural networks, the learning algorithms, and the application of neural networks on pattern recognition and image analysis has been given in this chapter. The principle of using neural computation in texture discrimination, the structure of the proposed neural network, its learning delta rule, and the classification and segmentation procedure will be presented from the next chapter.

# Chapter 4

# The 2-D AR Model Representation And Its Parameter Estimation through Neural Computation

## 4.1 Introduction

In the previous chapter, 2-D AR model has been proposed in texture representation for the purpose of texture classification or segmentation [14, 15, 16, 30]. However, due to the difficulty and computation burden of parameter estimation, low efficiency is often produced and the accuracy of discrimination is also affected. In this chapter, a method of performing 2-D AR model parameter estimation through a neural computation is implemented. It demonstrates that, with the adaptive capability of the neural network, the 2-D AR model parameter estimation can be carried out in a natural way. The corresponding neural structure and a detail derivation of the neural computation formulas are also presented.

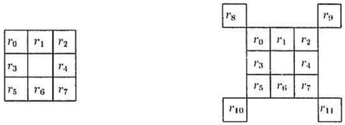
## 4.2 The 2-D AR Model Representation

Let $\{y(i,j)|i \in 1,...,M, j \in 1,...,N\}$ be the set of gray values of a given $M \times N$ image. As a 2-D random field, $y(i,j)$ can be described by the following equation:

$$y(i,j) = \sum_{r_q \in \phi} \theta_{r_q} y((i,j) \oplus r_q) + \beta\mu(i,j), \tag{4.1}$$

where $\phi$ denotes the associated neighborhood, $\theta_{r_q}$ is a set of parameters of the AR model which characterizes the dependence of a pixel on its neighbors. Each $\theta_{r_q}$ is related to the gray value of a neighboring pixel at position $((i,j) \oplus r_q)$, where "$\oplus$" is an operation which displaces position (i,j) to a position in the neighborhood of $(i,j)$ according to displacement $r_q$. $\mu(i,j)$ is an independent Gaussian random variable with zero mean and unit variance, and $\beta$ is the coefficient of $\mu(i,j)$.

The above equation can be interpreted as follows: in a textured image, the gray level $y(i,j)$ at location $(i,j)$ is related to the linear combination of the gray values of its neighboring pixels through the set of parameters $\theta_{r_q}$. In other words, the gray value of a pixel in a textured image can be represented by a combination



$\phi_1 = \{r_q\} \quad 0 \leq q \leq 7$
(a) $3 \times 3$ neighborhood

$\phi_2 = \{r_q\} \quad 0 \leq q \leq 11$
(b) neighborhood used in our algorithm

Figure 4.1: The neighborhood in 2-D AR model.

of gray values of its neighboring pixels which is specified by $\theta_{r_q}$. In this sense, the parameters $\theta_{r_q}$ can be used as a feature vector to distinguish different types of textures. The model parameters $\theta_{r_q}$ can be estimated from a given window. Compared to MRF models, a 2-D AR model has the advantage that the estimation of parameters is more direct. The least square estimation (LSE) and the maximum likelihood estimation (MLE) methods are commonly employed for the parameter estimation of AR model. In this thesis, neural computation is used to establish 2-D AR model for textures and to further identify textures by using the model.

One of the major difficulties of modeling textures by a stochastic model is selecting an appropriate neighborhood. Usually, a model with a large neighborhood fits textures better than a model with a small neighborhood. However, a small neighborhood can reduce the burden of computation. Details regarding the decision rules for choosing appropriate size and shape of a neighborhood can be found

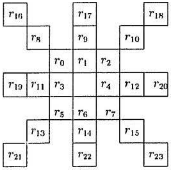


Figure 4.2: The neighborhood with 24 elements.

in [26, 27]. For computational simplicity, the $3 \times 3$ neighborhood (Figure 4.1(a)) is chosen by most techniques. In this thesis, a $3 \times 3$ neighborhood with four additional diagonal neighbors (Figure 4.1(b)) was chosen in order to represent more neighboring information within a reasonable amount of computation. For comparison, a 24 element neighborhood (Figure 4.2) was also utilized in an experiment. Results indicated that the 2-D AR model with this large neighborhood shows strong discrimination ability in the classification of some test texture samples. However the parameter estimation process for such a model is much more computationally expensive than a model with neighborhood shown in Figure 4.1(b).

## 4.3 Estimating AR Model Parameters by Neural Computation

The computation of a neural network used in pattern recognition usually consists of two phases: the learning phase and the recognition phase. In the learning process of a neural network, for each training sample, an input and a corresponding output are provided. The network first uses the input pattern together with its current weights (possibly incorrect) to produce an output, and this output is then compared with the desired output. If there is no difference, or the difference is within an allowable range, no training is necessary. Otherwise, the learning process is carried out to reduce the difference. A method of using an adaptive learning process to estimate the 2-D model parameters is proposed in this section.

### 4.3.1 Parameter Estimation by Using the Neural Network with Linear Activation Units

Assume that 2-D AR Model parameters $\theta_{r_q}$ for a given texture are initially assigned to random values. For position $(i,j)$, the difference between the gray value generated through parameter set $\theta_{r_q}$ and the real gray value $y(i,j)$ can be written as

$$y(i,j) - (\sum_{r_q \in \phi} \theta_{r_q} y((i,j) \oplus r_q) + \beta\mu(i,j)). \qquad (4.2)$$

The mean square error for a given image can then be defined by

$$\begin{aligned} E &= f(\{\theta_{r_q}\}) = \textstyle\sum_{i=1}^{M}\sum_{j=1}^{N}[y(i,j) - \hat{E}(\sum_{r_q\in\phi}\theta_{r_q}y((i,j)\oplus r_q) + \beta\mu(i,j))]^2 \\ &= \textstyle\sum_{i=1}^{M}\sum_{j=1}^{N}[y(i,j) - \sum_{r_q\in\phi}\theta_{r_q}y((i,j)\oplus r_q)]^2, \end{aligned} \tag{4.3}$$

where $\hat{E}$ computes the mean value [18]. Here, $E$ can be considered as the measure of the total error between a real image and its estimated image in which each gray value is obtained from the gray values of its neighbors and the parameter set $\theta_{r_q}$.

Equation 4.3 can be successfully represented as a network structure composed of linear activation units, in which each activation unit performs the function specified by

$$f_l(c(i,j)) = c(i,j) = y(i,j) - \sum_{r_q\in\phi}\theta_{r_q}y((i,j)\oplus r_q), \tag{4.4}$$

where $c(i,j)$ is the total input of a linear activation unit and $f_l(c(i,j))$ is the output of the unit (Figure 4.3).

The AR model parameters $\theta_{r_q}$ are considered as weights of input connections of the linear unit. The neighborhood of position $(i,j)$ in the original image is $\{((i,j)\oplus r_0),...,((i,j)\oplus r_N)\}$. Each input $y((i,j)\oplus r_q)$ is connected to the unit by a link with weight $\theta_{r_q}$. The central pixel $y(i,j)$ itself is connected to the unit by a fixed weight of -1. The total unit input $c(i,j)$ is the summation of all weighted inputs which is calculated by equation 4.4.

With this linear unit, a network structure corresponding to equation 4.3 can be designed as a two layer structure (Figure 4.4). The input texture sample of size

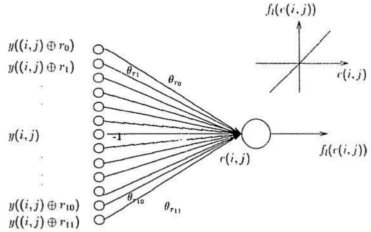


Figure 4.3: A linear activation unit.

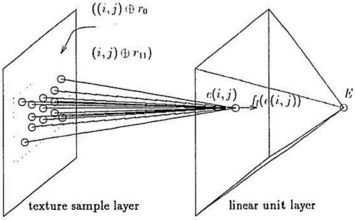


Figure 4.4: A two layer neural structure.

$M \times N$ is presented in the first layer. The second layer is a $M \times N$ two dimensional array of the linear units shown in Figure 4.3. All linear units in the second layer have the same set of weights for their input connections. The parameter estimation of a 2-D AR model can then be mapped to the adaptive learning process for this three layer neural network. In the adaptive learning process, the weight set is modified according to a delta rule to minimize the difference between an actual result and the desired output. When this learning process terminates, the weight set $\theta_{r_q}$ will specify the AR model for the sample presented to the neural network.

The weight sets $\theta_{r_q}$ constitute an error space where each single set of $\theta_{r_q}$ corresponds to one point in the error space. To find the weight set which minimizes the total system error $E$, the method of steepest slope is used to look for this minimum in the downward direction along the steepest slope in the error space. The gradient of the error space can be written as

$$\frac{\partial E}{\partial \theta_{r_q}} = 2\sum_{i=1}^{M}\sum_{i=1}^{N}\{[y(i,j) - \sum_{r_q \in \phi} \theta_{r_q} y((i,j) \oplus r_q)] y((i,j) \oplus r_q)\}, \tag{4.5}$$

which is the partial derivative of the total error $E$ with respect to each weight. Since this derivative should be proportional to the weight change, the rule for changing weight can be written as:

$$\theta_{r_q}(t+1) = \theta_{r_q}(t) + \Delta\theta_{r_q}(t), \qquad \Delta\theta_{r_q} \;\; = -\eta \tfrac{\partial E}{\partial \theta_{r_q}}, \tag{4.6}$$

where $t$ and $t+1$ refer to the current and previous weight modification cycles and $\eta$ is a constant called the learning rate. On each iteration, $\eta$ is applied to maintain the change of parameter set such that the minimal point at the error space can be reached in a comparatively small amount of computation. The value of $\eta$ has to be empirically determined. If $\eta$ is too large, the updating of the parameter set will never reach the minimum, while if $\eta$ is too small, the procedure takes too many iterations to converge.

Theoretically, a global minimal point should be found in the error space at the position where the gradient is zero. To reach this point by the learning process, $\theta_{r_q}$ are initially assigned to random values and then updated to be further "down" the error space by using the delta rule (equation 4.6). With an appropriately chosen value of $\eta$, the delta rule can find a set of weights approximately minimizing the error function.

### 4.3.2 Using Sigmoidal Activation Units

So far it has been shown that the network composed of linear activation units can be used to perform gradient descent in error space through a delta rule. This gradient descent is guaranteed to find the set of weights for a provided training sample so that the weight set corresponds to the minimum error when the training sample is presented to the network. However, it should be noticed that the set of weights produced by the learning process of the neural network with linear units is not necessarily the most accurate weight set for the given type of texture.

Let $\theta^o_{r_q}$ denote the weight set for a given type of texture. For a pixel of an arbitrary sample of this texture, the real gray value should be within a reasonable range of the value estimated by the weight set $\theta^o_{r_q}$. However, noise and distortion may occur in different places of any samples. For a pixel with a large random noise, the interrelation of the noise pixel with its neighbors would not share the same function with most of pixels in the image. In this case, the differences between the real gray value of the noise pixel and the gray values estimated through $\theta^o_{r_q}$ will be quite significant. A linear unit, by its nature, provides no mechanism to suppress the effect of such noise. Therefore, the accumulated effect caused by such noise will affect the weight training during the learning process. As a result, the weight adaptation in the learning process will depart to some extent from the best set of parameters of a given texture. A mechanism to suppress the effect of noise is therefore necessary to improve the accuracy of the weight adaptation.

A similar problem will also occur when a multilayer network, which has the linear unit layer as a part of configuration, is used to perform texture classification after learning. Since a test sample generally contains noise, suppression of those noise is needed to increase the classification accuracy of the neural network.

For these reasons, sigmoidal activation units are employed in the network. The characteristic of the units can be described by a sigmoidal activation function

$$f_s(e(i,j)) = \alpha\left(\frac{2}{1+e^{-\frac{e(i,j)}{\lambda}}} - 1\right). \tag{4.7}$$

where $\alpha$ is the maximum value of the output of the sigmoidal activation function. In this expression, the effect of $\lambda$ is to modify the shape of the sigmoidal function.

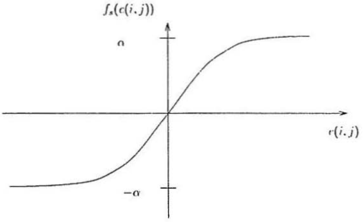


Figure 4.5: The sigmoidal activation function used.

A high value of $\lambda$ results in a more gently varying function than a low value of $\lambda$. The shape of the function is illustrated in Figure 4.5. The choice of parameters $\alpha$ and $\lambda$ in the function will be discussed later.

Correspondingly, the total error of the system with sigmoidal neural cell should be written as:

$$\begin{aligned} E = f(\{\theta_{r_q}\}) &= \sum_{i=1}^{M}\sum_{j=1}^{N}[f_s(e(i,j))]^2 \qquad (4.8) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N}[f_s(y(i,j) - \sum_{r_q\in\phi}\theta_{r_q}y((i,j)\oplus r_q))]^2. \end{aligned}$$

The partial derivative of the total error $E$ with respect to each weight can then be written as:

$$\frac{\partial E}{\partial\theta_{r_q}} = \sum_{i=1}^{M}\sum_{j=1}^{N}\frac{\partial[f_s(e(i,j))]^2}{\partial e(i,j)}\frac{\partial e(i,j)}{\partial\theta_{r_q}}. \qquad (4.9)$$

By equation 4.4, it is easy to see that

$$\frac{\partial e(i,j)}{\partial \theta_{r_q}} = -y((i,j) \oplus r_q), \tag{4.10}$$

and

$$\frac{\partial [f_s(e(i,j))]^2}{\partial e(i,j)} = 2 f_s(e(i,j)) \frac{\partial f_s(e(i,j))}{\partial e(i,j)}. \tag{4.11}$$

By the sigmoidal function 4.7, we have

$$\frac{\partial f_s(e(i,j))}{\partial e(i,j)} = -\frac{1}{2\alpha\theta_0}(f_s(e(i,j)) + \alpha)(f_s(e(i,j)) - \alpha), \tag{4.12}$$

giving

$$\frac{\partial [f_s(e(i,j))]^2}{\partial e(i,j)} = -\frac{f_s(e(i,j))}{\alpha\theta_0}(f_s(e(i,j)) + \alpha)(f_s(e(i,j)) - \alpha). \tag{4.13}$$

Substituting 4.10 and 4.13 in equation 4.9, we have

$$\frac{\partial E}{\partial \theta_{r_q}} = \sum_{i=1}^{M}\sum_{j=1}^{N}[\frac{f_s(e(i,j))}{\alpha\theta_0}(f_s(e(i,j)) + \alpha)(f_s(e(i,j)) - \alpha)y((i,j) \oplus r_q))]. \tag{4.14}$$

Correspondingly, the delta rule is written as

$$\theta_{r_q}(t+1) = \theta_{r_q}(t) + \triangle\theta_{r_q}(t),$$

$$\triangle\theta_{r_q} = -\eta \sum_{i=1}^{M}\sum_{j=1}^{N}[\frac{f_s(e(i,j))}{\alpha\theta_0}(f_s(e(i,j)) + \alpha)(f_s(e(i,j)) - \alpha)y((i,j) \oplus r_q)]. \tag{4.15}$$

In the application of this delta rule, a predefined threshold value $\epsilon$ is used as the termination condition. Each iteration involves two phases. In the first phase the total error $E$ and the partial derivatives of the total error with respect to each weight $\theta_{r_q}$ are computed. In the second phase, the appropriate weight changes are made according to equation 4.15. When all partial derivatives are reduced to be within the range defined by $\epsilon$, the process terminates.

### 4.3.3 The Selection of the Parameters of the Sigmoidal Activation Function

As stated previously, the sigmoidal activation function is used to suppress the noise effect so as to increase the accuracy of a weight set for a texture and to strengthen the stability of the classification process. The selection of the sigmoidal function parameters $\alpha$ and $\lambda$ is very critical for the sigmoidal activation units to make their expected contribution to the performance of the network.

#### 4.3.3.1 Selecting the value of $\alpha$

Parameter $\alpha$ gives the saturation value of the sigmoidal function. If $\alpha$ is too large, the output of a sigmoidal activation unit can never reach the saturation value and the noise effect will not be suppressed. When it is too small, the output of a sigmoidal activation unit is very easily to be brought to the saturation value. In this case, sigmoidal units with different weight sets will tend to give the same level of output values such that the discriminating ability of the network will be reduced and different textures will be mis-classified. In our application, the value of $\alpha$ has been empirically determined. As the total input of a sigmoidal unit is the difference between the real gray value of a pixel and its estimated gray value, the value of $\alpha$ is chosen to give the expected maximum difference value. When an input value of a sigmoidal unit is bigger than the maximum different value, the output is considered to be suppressed and maintained at the value of $\alpha$.

### 4.3.3.2 Selecting the value of $\lambda$

As mentioned above, when input to a sigmoidal unit input $e(i,j)$ is equal to or bigger then $\alpha$, the the unit output $f_s(e(i,j))$ yields the value of $\alpha$. However, for the sigmoidal activation function given by equation 4.7, a unit cannot actually reach its extreme value without infinitely large weights. Therefore, the values of $0.9\alpha$ and $0.1\alpha$ are typically used as target output values. Here, we assume when $c(i,j) = \alpha$ and $f_s(c(i,j)) = \frac{9}{10}\alpha$. By equation 4.7, we have

$$\frac{9}{10}\alpha = \alpha(\frac{2}{1+e^{\frac{\alpha}{\lambda}}} - 1) \tag{4.16}$$

which yields

$$\lambda = \frac{\alpha}{\log 19}. \tag{4.17}$$

Therefore the sigmoidal activation function in the network is given as

$$f_s(c(i,j)) = \alpha(\frac{2}{1+e^{-\frac{c(i,j)\log 19}{\alpha}}} - 1). \tag{4.18}$$

## 4.4 Using Total Error $E$ as the Measurement

The parameters of an AR model provide a good set of features for texture identification. However in real applications, an efficient computation is also strongly demanded. Many stochastic model algorithms directly use parameters of stochastic models as texture features so that the estimation process has to be performed for each textured image to be classified. In the proposed method the total error

$E$ is used to measure the difference between an input test sample and a trained pattern.

During the training process of a texture, the updating of the parameter set $\theta_{r_q}$ continues until the total error reaches its global minimum. When this optimum set $\theta_{r_q}$ is applied to images of the same type of texture, the total produced errors should be within a reasonable range of the global minimum. On the other hand, total errors should be fairly large when this parameter set $\theta_{r_q}$ is applied to images of any other textures. Hence, the differences in total errors can be used in the classification process to distinguish different types of textures.

# Chapter 5

# The Network Structure And Its Performance for Classification

## 5.1 Introduction

In the last chapter, the principle of 2-D model parameter estimation through a neural computation and the appropriate neural structure are given. This chapter describes the overall structure of the proposed neural network, which is composed of three subnets: the input subnet (ISN), the analysis subnet (ASN) and the classification subnet (CSN). For each of subnets in the network, the design principle, configuration and behavior are introduced. In addition, this chapter gives the training algorithm of the network, as well as the performance of this network applied in natural texture image classification.

## 5.2 An Outline of the Architecture

The proposed network consists of three subnets: the input subnet (ISN), the analysis subnet (ASN) and the classification subnet (CSN) as shown in Figure 1.1. As mentioned earlier, the input subnet accepts an input texture pattern and outputs the normalized input pattern to the analysis subnet. The analysis subnet consists of a set of channels, where each channel models a particular texture class. During a classification, each channel calculates a total error under the assumption that the input pattern can be classified into the texture type corresponding to this channel. This total error indicates the difference between the input pattern and the estimated pattern generated in the channel. All channels take the input pattern from the input subnet and process it separately. The outputs of all channels participate in the competition at the classification subnet which selects one with the smallest total error as the winner. The input image is then classified to the class whose corresponding channel gives the smallest total error among all channels.

## 5.3 The Input Subnet

During both the training and the testing processes, input images are supplied to the input subnet. The input subnet consists of a two-dimensional array of size $M \times M$, where $M \times M$ is the size of an input pattern $\{I(i,j) | i,j \in 1,...,M\}$. The output of the subnet is a normalized version of the original input pattern (Figure

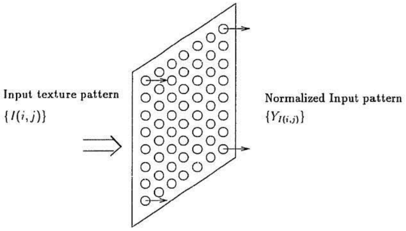


Figure 5.1: The input subnet

5.1).

Variations in lighting, lens, films and digitizers usually introduce monotonic transformations of the image gray values. In order to make input patterns independent of the intensity of the original image, normalization is performed to guarantee that images which are monotonic transformations of one another produce the same input to the analysis subnet.

There are different ways to do normalization for an input vector. Considering an input image of size $M \times M$ as a vector of $M \times M$ elements, one way to normalize is to divide each element of the input vector by the total length of the input vector. This method is utilized in this network because it shortens the length of an input vector to unit length without changing the direction of an input vector. In this case, the total length of the input vector is calculated by

$$\sqrt{\sum_{i=1}^{M}\sum_{i=1}^{M} I(i,j)^2}. \tag{5.1}$$

Therefore $Y_{I(i,j)}$, the output value of node $I(i,j)$ in the input subnet, is defined as

$$Y_{I(i,j)} = \frac{I(i,j)}{\sqrt{\sum_{i=1}^{M}\sum_{i=1}^{M} I^2(i,j)}}. \tag{5.2}$$

## 5.4 The Analysis Subnet

The analysis subnet consists of a set of channels which work in parallel. Each channel has the three layer structure described in Chapter Three. Here, the first layer, the second layer and the third layer are called as the buffer layer (BL), the estimation layer (EL) and the the summation layer (SL) respectively. The structure of one such channel is shown in Figure 5.2. Each node of the buffer layer receives its normalized input from the corresponding node of the input subnet. Each node in the estimation layer is connected to the nodes of its corresponding window in the buffer layer. The summation layer in a channel has only one node which calculates the sum of the output values of all nodes in the estimation layer. All channels have the same input from the input subnet but work independently. The results of all channels, which form a one-dimensional array of size $N$, are input into the classification subnet.

As shown in Figure 5.2, the node at position $(i,j)$ of the estimation layer has its input connections linked to a small neighborhood $\phi$ centered at position $(i,j)$ in

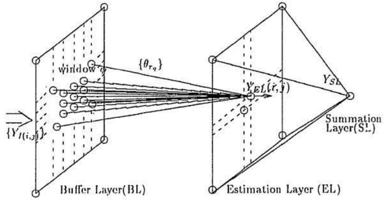


Figure 5.2: The structure of a channel in the analysis subnet.

the buffer layer, where $\phi = \{r_q\}$, $q \in \{1, ..., Q\}$ and $Q$ is the number of components of the neighborhood. All nodes in the estimation layer of a channel share the same set of input connection weights $\theta^k_{r_q}$, where $\theta^k_{r_q}$ is the weight of the connection from node $((i,j) \oplus r_q)$ in the buffer layer to node $(i,j)$ in the estimation layer and superscript $k$ denotes channel $k$.

The output value of node $(i,j)$ in the estimation layer is calculated by

$$Y^{(k)}_{EL}(i,j) = f_s(Y^{(k)}_I(i,j) - \sum_{r_q \in \phi} \theta^{(k)}_{r_q} Y^{(k)}_I((i,j) \oplus r_q)), \tag{5.3}$$

where $f_s$ is the sigmoidal activation function described in equation 4.7.

Let $Y^{(k)}_{SL}$ be the output value of the node at the summation layer of channel $k$. $Y^{(k)}_{SL}$ is the summation of outputs of all nodes in the estimation layer. The equation

$$Y^{(k)}_{SL} = \sum_{i=1}^{M} \sum_{j=1}^{M} Y^{(k)}_{EL}(i,j)^2 \tag{5.4}$$

specifies the total error between the input pattern and the pattern generated by channel $k$.

## 5.5 The Classification Subnet

The output of the summation layer $\{Y_{SL}^{(k)} | k \in 1, ..., N\}$ is the input to the classification subnet. The classification subnet selects the input with the smallest value as the winner in competition. This subnet is composed of two layers: the inverse layer (IL) and the competitive layer (CL). Each of the two layers is a one-dimensional array of $N$ nodes. The competitive layer is a MAXNET which selects the node with the largest input value as the winner in competition. The inverse layer is inserted ahead of the competitive layer to transform the input with the smallest value into the largest value.

### 5.5.1 The Inverse Layer

The purpose of the inverse layer is to invert the input values of the competition subnet so that the smallest input value turns out to be the largest value after passing through this layer.

The output of the summation layer is a one-dimensional array: $\{Y_{SL}^{(k)} | k \in 1, ..., N\}$. The summation of all values of this one dimension array is written as

$$S = Y_{SL}^{(1)} + ... + Y_{SL}^{(k-1)} + Y_{SL}^{(k)} + Y_{SL}^{(k+1)} + ... + Y_{SL}^{N}. \tag{5.5}$$

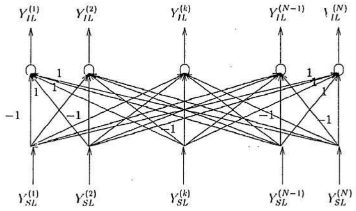


Figure 5.3: The inverse layer.

Now we write another array in which the $k$th entry is the result of subtracting value $Y_{SL}^{(k)}$ from the summation $S$

$$Y_s = \{(S - Y_{SL}^{(1)}), ..., (S - Y_{SL}^{(k)}), ..., (S - Y_{SL}^{N})\}. \tag{5.6}$$

If $Y_{SL}^{(k)}$ is the smallest output of the analysis subnet, then value $(s - Y_{SL}^{(k)})$ must be the biggest value in array $Y_s$.

The structure of the inverse layer is as follows: all node are fully connected to the output of the summation layer. Let $Y_{IL}^{(k)}$ denote node $k$ in this layer. For each node $Y_{IL}^{(k)}$, the weights of connections to all nodes in the summation layer are fixed at 1, except for the connection between nodes $Y_{IL}^{(k)}$ and $Y_{SL}^{(k)}$ which is fixed to -1 (Figure 5.3). Given such connections, the output value of each node in the layer is equal to

$$\begin{aligned} Y_{IL}^{(k)} &= Y_{SL}^{(1)} + \ldots + Y_{SL}^{(k-1)} - Y_{SL}^{(k)} + Y_{SL}^{(k+1)} + \ldots + Y_{SL}^{N} \\ &= \sum_{k=1}^{N} Y_{SL}^{(k)} - 2 \times Y_{SL}^{(k)}. \end{aligned} \tag{5.7}$$

## 5.5.2 The Competition Layer

Nodes in the competition layer are connected to the nodes in the inverse layer. In this competition layer, an iterative procedure finally produces one winner node with positive output and drives all remaining nodes to zero output.

Figure 5.4 shows the structure of the competition layer where the activation values are denoted as $\{Y_{CL}^{(k)} | k \in 1, ..., N\}$. The initial activation values of nodes $Y_{CL}^{(k)}$ are the input values from nodes $Y_{IL}^{(k)}$. The strength of the inhibitory connection sent from a node to every other nodes in this layer is proportional to the current activation value of the node. When a node receives inhibition from other nodes, its current activation value is reduced.

Each node also has an excitatory connection to itself. The strength of the excitatory connection is proportional to the previous activation value of that node.

Let $w_{jk}$ be the connection weight from node $j$ to node $k$ in this layer,

$$w_{jk} = \begin{cases} +1 & \text{for } j = k, \\ -\varepsilon & \text{for } j \neq k,\ \varepsilon < \frac{1}{N} \qquad k, j = 1, 2, , ..., N. \end{cases}$$

The output value of node $Y_{CL}^{(k)}$ at discrete time $(t + 1)$ is computed as

$$Y_{CL}{}_{(t+1)}^{(k)} = f_t(Y_{CL}{}_{(t)}^{(k)} - \varepsilon \sum_{j \neq k} Y_{CL}{}_{(t)}^{(j)}). \tag{5.8}$$

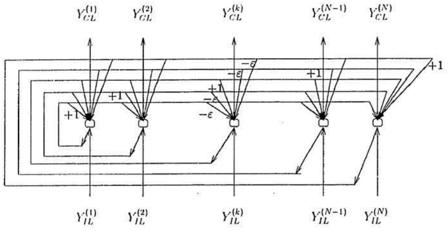


Figure 5.4: The competition layer.

where

$$f_t(\mu) = \begin{cases} \propto \mu & \mu > 0 \\ 0 & \mu < 0 \end{cases}$$

During processing, the activation value of each node is updated in each step according to equation 5.8. The first term in the right hand of the equation is the previous activation value of unit $k$. The second term denotes the sum of inhibition from all other nodes. In each step, the node which initially has the highest value gets the greatest excitation and the lowest inhibition. Let $\mu$ represent the total input (the excitatory connection subtracted by the inhibitory connection) of a node. If $\mu$ is positive, $f_t(\mu)$ is positive and the output of the node is positive. All nodes whose $\mu$ values are negative will be forced to zero.

The process of activation value updating is iteratively carried out until all nodes except one have output value zero. The remaining node with positive output is the node which has the smallest subnet input to the classification subnet. The original input image should be classified to the class corresponding to this node.

One problem occurs in the competition layer when two nodes have the largest initial activation value. In this case, the two nodes with the largest value will be driven to zero simultaneously because they receive the same amount of excitation from themselves and inhibition from each other, and no node will be declared to be a winner node. However, this is not likely to happen because it is impossible for two channels to produce the same amount of total error. Small difference always exists between output values produced by different channels.

## 5.6 The Learning Procedure

From the architecture and the computational procedure described above, it is clear that proper sets of weights $\theta_{r_q}^{(k)}$ are critical for ensuring accurate classification. The set of weights for a class of textured images should characterize, with the highest degree of possibility, the most frequently appearing gray value distribution in images of that class. In Chapter Four, we have already presented the principle of how a weight set can be trained to fit a texture pattern as a 2-D AR model. This section describes the training procedure used to enable the multilayer network to classify textures under investigation.

Supervised learning is employed in the learning procedure of this network. In order to increase the probability of correct classification, a group of training samples are needed for each type of texture for the network to be sufficiently adapted to the variance between samples of that type of texture. For each of subnets in the network, the design principle, configuration and behavior are introduced.

Suppose the number of samples for a type of texture is $M$. Two methods can be used in supervised training. In the first method, each sample is provided to the network respectively and the average of the resulting $M$ sets of weights resulting from the training process is taken as the weight set for the corresponding texture. This method is simple and quick, but it does not ensure the accuracy of the resulting weight set.

In the second method, the $M$ samples of the texture are provided to the network alternately in the training process. Training process stops when the network is convergent on all $M$ samples of the texture. This process starts by initially providing one sample to the network. After the total error value for that sample is calculated, the weights are modified according to the delta rule. In the next weight updating cycle, instead of using the same input sample, another texture sample is provided to the input layer of the network. All samples of the texture are alternately used in this manner until the total error value of each sample is reduced below the predefined threshold value. This method has a drawback in that the speed of convergence is slower compared to first training method. In order to make the learning process more efficient, two stages of training are adopted. In the first stage, one sample is presented to the input layer and the weight set is updated according to the delta rule. This process terminates when the weight change for each weight is reduced below a predefined value $T_1$. In the second stage, all $M$ samples are used alternately to refine the weight set formed in the first stage. As the weight set has already been initially trained on one sample of the texture, convergence is fast in the second stage for the set of training samples which contain only small variations from the sample presented at the first stage.

Only weights in the analysis subnet need to be built during the training procedure; all other internal connections in both the input and classification subnets can be set up before the training. For a classifier which can distinguish $N$ types of textures, $N$ channels are needed, each of which is trained individually using the patterns from the corresponding texture.

The training algorithm could be organized as: for each texture

<u>The first stage:</u>

Step 1: Choose a sample from training sample of the texture. Decide a channel index $k$ for this texture.

Step 2: Initialize weight set $\theta_{r_q}^{(k)}$ to small random values.

Step 3: Provide the selected sample to the network.

Step 4: For each element of set $\theta_{r_q}^{(k)}$

1. Compute the derivative $\frac{\partial E}{\partial \theta_{r_q}^{(k)}}$ according to equation 4.14.

2. Update $\theta_{r_q}^{(k)}$ by equation 4.15.

Step 5: Repeat step 4 until the derivative $\frac{\partial E}{\partial \theta_{r_q}^{(k)}}$ for each $\theta_{r_q}^{(k)}$ satisfies

$$\frac{\partial E}{\partial \theta_{r_q}^{(k)}} \leq T_1. \tag{5.9}$$

<u>The second stage:</u>

Step 6: Select a group of training samples of the texture.

Step 7:

1. (a) Provide a sample to the network.

   (b) Update weight set $\{\theta_{r_q}^{(k)}\}$ according to equation 4.15.

   (c) Determine if the terminating condition

   $$\frac{\partial E}{\partial \theta_{r_q}^{(k)}} \leq T_2. \tag{5.10}$$

   is satisfied for each $\theta_{r_q}^{(k)}$.

2. Check if all samples have been presented to the network and if the convergence condition is satisfied in the presentation of each sample. If so, go to the Step 8. If not, back to Step 7.

Step 8: Repeat Steps 1 through 7 until all textures are learned.

## 5.7 The Network in Texture Classification

The performance of the proposed neural network was examined on the classification problem of thirty-eight natural textures. In comparing with other algorithms, textures used for the training and testing of the system were digitized from the Brodatz album [70], which is commonly used by researchers in texture identification. These thirty-eight textures are shown in Appendix Figure A.1. Pictures are digitized on sizes from a $300 \times 300$ to $350 \times 350$ pixel grid with a gray level resolution of 256 levels. Fourteen samples, each with a $64 \times 64$ pixel grid, were arbitrarily selected from the digitized image of each texture, in which four of them were randomly selected to be used in the training stage and the remaining ten saved as test samples.

### 5.7.1 Training

Several parameters have to be decided before the training stage. The parameter $\alpha$, which is the extreme value of the sigmoidal activation function, was experimentally chosen as $\alpha = 0.03$. Correspondingly, according to equation 4.17, $\lambda$ was calculated as

$$\lambda = \frac{\alpha}{\log 19} = \frac{0.03}{\log 19} = 0.01018870. \tag{5.11}$$

The experiment shows that when $\alpha$ is greater than 0.03, the sigmoidal activation function does not suppress the noise effectively and tends to behave like a linear activation function. A value smaller than 0.03 will improve the performance in some cases; however in other cases, it can result in a wrong classification by mixing up two textures. The reason for this is that some of the discrimination information is suppressed by the low value of $\alpha$.

These two parameters $\alpha$ and $\lambda$ are independent of the texture images under examination and were used throughout both the training and the testing processes. It should be noted that the two values were decided on the basis of the size of the input subnet, which is also the size of all training and testing samples. Since the data provided to the analysis subnet is normalized, if the size of an input pattern is changed, the average value of a normalized sample in the buffer layer of the analysis subnet is also changed. The value of $\alpha$ should then be modified accordingly.

Values of $T_1$ and $T_2$, which are used in the first and the second stages of the training process to decide convergence, also need to be predefined. In the experiment, we selected $T_1 = 0.001$ and $T_2 = 0.00025$, which means that in the first stage, the initial training is declared to be convergent when the derivatives $\frac{\partial E}{\partial \theta_{r_q}^{(k)}}$ for each $\theta_{r_q}^{(k)}$ are less than or equal to 0.001, and in the second stage, the training converges when $\frac{\partial E}{\partial \theta_{r_q}^{(k)}}$ with respect to each $\theta_{r_q}^{(k)}$ for all presented samples are less than or equal to 0.00025.

For some images, e.g., those containing large primitives, the convergence condition has to be somehow different from that for fine textures. When trying to simultaneously fit a weight set to several training samples of a texture with large primitives, it is difficult for the weight set to be as precise as that for a fine texture. In those cases, the training process often becomes stable in the state in which most but not all values of $\theta_{r_q}$ are less than the predefined $T_2$. There are two ways to handle this situation: (1) scaling down the original texture images to give fine resolution pictures, or (2) increasing the size of the neighborhood to fit large primitives. In our experiments, a generally satisfactory classification can still be achieved in the current window size for these reasonably large primitive textures in their original resolution. Therefore, in practice, we set a maximum iteration number $M$. During the weight updating, if the convergence condition defined using $T_2$ has not been satisfied when the maximum iteration number $M$ is reached, the current weight is considered to be the best for the texture examined and the training procedure stops.

The learning rate was defined as $\eta = 0.3$ during the weight modification. Table 5.1 shows the resulting weight sets of six natural textures from the Brodatz album, which are Loose Burlap (D103), Handmade Paper (D109), Plastic Bubbles (D111), Woolen Cloth (D19), Beach Sand (D28), Grass Lawn (D9) respectively.

For different textures, a different number of iterations is required to reach convergence during the learning phase. For most of the textures in the experiment, learning terminated after a reasonable amount of computation. In Figure 5.5, the percentage of the number of convergent textures over the total number of textures is shown as a function of the number of iterations. For fine textures, such as Woolen

| | $(a)$ | $(b)$ | $(c)$ | $(d)$ | $(e)$ | $(f)$ |
|---|---|---|---|---|---|---|
| $r_0$ | −0.1928389 | −0.0199287 | −0.1029087 | 0.1025216 | −0.1351613 | −0.0055586 |
| $r_1$ | 0.4674090 | 0.3457456 | 0.2974779 | 0.2206792 | 0.3639432 | 0.3526068 |
| $r_2$ | −0.2539033 | −0.0111343 | −0.0529170 | 0.0518353 | −0.056798 | 0.0062077 |
| $r_3$ | 0.4799422 | 0.1702416 | 0.3829967 | 0.1540470 | 0.3529322 | 0.1161405 |
| $r_4$ | 0.4788981 | 0.1721074 | 0.3839767 | 0.1526799 | 0.3572567 | 0.1158203 |
| $r_5$ | −0.2563388 | −0.0115873 | −0.0548410 | 0.0496973 | −0.0532309 | 0.0045432 |
| $r_6$ | 0.4732172 | 0.3423978 | 0.3089632 | 0.2224699 | 0.3596855 | 0.3560610 |
| $r_7$ | −0.1971821 | −0.0201352 | −0.1081048 | 0.1003185 | −0.1358091 | −0.0032193 |
| $r_8$ | −0.0085986 | 0.0084499 | −0.0063427 | −0.0108847 | −0.0062826 | 0.0056830 |
| $r_9$ | 0.0075197 | 0.0134750 | −0.0191099 | −0.02138411 | −0.0212406 | 0.0151671 |
| $r_{10}$ | 0.006158 | 0.0145237 | −0.0174884 | −0.0205692 | −0.0217451 | 0.0107174 |
| $r_{11}$ | −0.0067542 | 0.0035249 | −0.0069352 | −0.0033976 | −0.0043165 | 0.0096820 |

* (a) Loose Burlap (D103), (b)Handmade Paper (D109), (c) Plastic Bubbles (D111), (d) Woolen Cloth (D19), (e)Beach Sand (D28), (f) Grass Lawn (D9).

Table 5.1: Weight sets established in the training process.

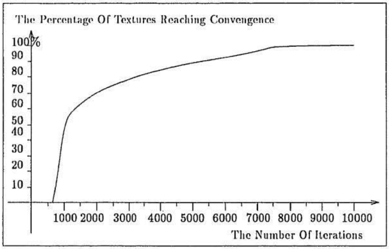


Figure 5.5: The number of iterations needed to reach convergence.

Cloth (D19) and Water (D37), the convergence was around in 650 iterations on average, which is much less than that required by coarse textures, such as Plastic Pellets (D66) and Coffee Beans (D75), which often required in excess of 3000 iterations to satisfy the convergence condition.

The computational burden of the learning stage is not a drawback of this approach because the learning procedure is applied only once for each type of texture and the learning result is not affected by adding new types of textures to be classified by the network. Moreover, in the classification phase, images proceed through the network without the preprocessing and feature extraction required by many other techniques. Hence, overall computation time has been considerably reduced.

### 5.7.2 Classification Testing

A classification testing of thirty-eight textures was conducted after the training. For each texture under investigation, a total number of ten test samples were provided to the network. The performance of the network is quite encouraging. As shown in Table 5.2, for twenty-three out of the thirty-eight textures, all samples were properly classified. Satisfactory results were also obtained for seven other textures, in which eight or nine samples out of ten samples were correctly classified for each texture class. For the remaining eight textures, five to seven samples were

Table 5.2: Result of classification.

Input Textures

Assigned Textures

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | 7 | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2 | |
| 3 | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | |
| 6 | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | 5 | | | | | | | | | 2 | | | | | | | | | | 2 | | | | | 1 | | | | | | |
| 9 | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | 9 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | 6 | | | | | | | | | | | | 2 | | 2 | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | | | | | 9 | | | | | | | | | 1 | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | 6 | | | | | | 2 | | 1 | | | |
| 28 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 10 | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | 7 | | 2 | | | | | | |
| 31 | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | 9 | | | | | | | |
| 32 | | | | | | | 2 | | | | | | | | | | | | | | | | | | | | | | | | | 8 | | | | | | |
| 33 | | | | | | | | | | | | | | | | | 2 | | | | | | | | | | | | | | | | 8 | | | | | |
| 34 | | | | | | | | | | | | | | | | | | | | | | | | | | | 5 | | | | | | | 5 | | | | |
| 35 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 3 | | | | 7 | | | |
| 36 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 4 | | | | | | | | 6 | | |
| 37 | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 9 | |
| 38 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 10 |

Tabel 5.2: (Continue)

(38 textures in total, 10 test samples for each texture).

| | | |
|---|---|---|
| 1. Pressed cork (D4); | 2. Grass lawn (D9); | 3. Woven aluminum wire (D14) ; |
| 4. Straw (D15); | 5. Herringbone weave (D17); | 6. Woolen cloth (D19); |
| 7. French canvas (D20); | 8. French canvas (D21); | 9. Pressed calf leather (D24); |
| 10. Beach sand (D28); | 11. Beach sand (D29); | 12. Pressed cork (D33); |
| 13. Netting (D34); | 14. Water (D37); | 15. Straw screening (D49); |
| 16. Raffia with threads (D51); | 17. Oriental cloth (D52); | 18. Oriental cloth (D53); |
| 19. Straw matting (D55); | 20. Straw matting (D56); | 21. Handmade paper (D57); |
| 22. Oriental rattan (D64); | 23. Oriental rattan (D65); | 24. Plastic pellets (D66); |
| 25. Wood grain (D68); | 26. Coffee beans (D74); | 27. Coffee beans (D75); |
| 28. Grass fiber cloth (D76); | 29. Cotton canvas (D77); | 30. Grass fiber cloth (D79); |
| 31. Oriental straw cloth (D81); | 32. Raffia looped to pile (D84); | 33. Sea fan (D87); |
| 34. Brick wall (D94); | 35. Loose burlap (D104); | 36. Cheesecloth (D105); |
| 37. Handmade paper (D109); | 38. Plastic bubbles (D111). | |

* The index beginning as Capital letter D is used in the Brodatz album.

properly classified. Reasons for the mis-classification are: (1) the serious distortion in some test samples as well as the learning texture samples; and (2) the resolution of some texture images. For instance, as the original pictures for Raffia Woven With Cotton Threads (D51) and Coffee Bean (D75) in the Brodatz album were digitized to around the size of $300 \times 300$ images, primitives tend to be quite large. In those cases, the classification accuracy will be increased after scaling down the digitized images used in the testing.

It should be noted that the number of textures in the test is much bigger than that of many experiments reported in the literature. They often use less than ten textures in the demonstration of their particular methods. The large number of textures in this test certainly increases the difficulty of recognition.

In order to compare the performance, the least square estimation (LSE) was

Table 5.3: The comparison between LSE technique and network approaches.

* The data inside bracket is the result of using LSE.

| Input Textures | Assigned Textures 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 (4) | | 2 (4) | | 1 (1) | | | | | 0 (1) |
| 2 | | 7 (6) | | | | 3 (4) | | | | |
| 3 | 1 (2) | | 9 (8) | | | | | | | |
| 4 | | | | 10 (7) | | | 0 (1) | 0 (1) | 0 (1) | |

| | | |
|---|---|---|
| 1. Grass lawn (D9); | 2. Loose burlap (D104); | 3. Handmade paper (D109); |
| 4. Plastic bubbles (D111); | 5. Woolen cloth (D19); | 6. Oriental straw cloth (D81); |
| 7. Beach sand (D28); | 8. Plastic pellets (D66); | 9. Coffee beans (D74); |
| 10. Handmade paper (D110); | | |

implemented in the estimation of 2-D AR model parameters for several textures. The same set of training samples and testing samples were used. The results showed that this network does give a better classification performance. For example, using parameters from the LSE method, three out of ten testing samples of Plastic Bubbles (D111) were mis-classified as textures Beach Sand (D28), Plastic Pellets (D66) or Coffee Beans (D74). In contrast, these same three samples were correctly classified to D111 by using the network method, resulting in a 30% classification accuracy increase. A 30% classification increase was also achieved for texture Grass Lawn (D9).

# Chapter 6

# Rotation Invariant Classification

## 6.1 Introduction

The neural network described in the last chapter requires that the testing samples of a texture provided to the network have the same orientation as that of the training samples. However, in some application, it might be difficult to maintain the orientation of the test samples to be the same as that of the training samples. A texture classification architecture which is insensitive to rotation is desirable in those cases.

Attempts at building a texture rotation invariant classifier to classify testing samples with arbitrary rotation angles are described in this chapter. Two types of a rotation classifier are proposed in which the neighborhoods of the 2-D AR model and the structure of the neural network are modified accordingly. In the first type, AR models are established on eight directions and testing samples with arbitrary rotation can be approximately modeled by one of the eight models. A multi-circular AR model is used in the second type to extract the rotation invariant features such that samples of one texture in any rotation can be characterized by one feature set. Experiments of identifying natural texture images which are taken from real objects are then conducted for testing the two types of rotation invariant classifiers.

## 6.2 Model Based on Multi-Directions

For a given texture, when a testing sample is rotated, the relation between a center pixel and its neighboring pixels is also changed and the 2-D AR model established from the training samples observed in a fixed orientation no longer fits. In the multi-direction approach, 2-D AR models based on eight directions are used. Testing samples with arbitrary rotation can then be approximately modeled by one of the eight models which has the smallest angle difference to the testing sample.

As shown in Figure 6.1, a neighborhood with 24 elements is used and eight parameter sets of the model are estimated to fit texture samples corresponding to eight directions. The eight parameter sets can be divided as two groups, one for the parameter sets on directions $90^o$, $0^o$, $-90^o$ and $180^o$, and the other one for the parameter sets on directions $45^o$, $-45^o$, $-135^o$ and $135^o$. For each of the

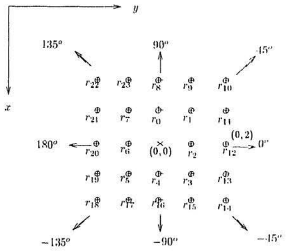


Figure 6.1: Neighborhood in the rotation invariant classifier based on multi-directions

two groups, elements in one parameter set correspond to the same set of positions on images as elements of another parameter set, but are denoted by different notations. Therefore, one parameter set can be obtained from any of the other parameter sets in the same group. For example, assume $\theta_{r_q}^{\alpha}$ denote parameter $r_q$ of the parameter set on direction $\alpha$ ($\alpha \in \{90^o, 45^o, 0^o, -45^o, -90^o, -135^o, 180^o, 135^o\}$). The parameter set on direction $0^o$ can then be obtained from the parameter set on direction $90^o$ by assigning $\theta r_0^{90^o}$ to $\theta r_6^{0^o}$, $\theta r_1^{90^o}$ to $\theta r_7^{0^o}$, ......, $\theta r_{23}^{90^o}$ to $\theta r_{19}^{45^o}$. As a result, the training process for a type of texture only needs to be performed twice and the computation time of the training phase is not greatly increased by increasing number of models in the rotation invariant classifier.

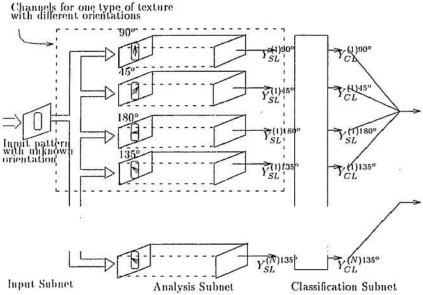


Figure 6.2: The modified network structure for model based on multi-directions.

The multilayer network structure is modified as shown in Figure 6.2. For each type of texture, eight channels are needed, each corresponding to one direction. All outputs from the eight channels of one texture participate in the competition in the classification subnet. An input sample will be classified to the type of texture such that one of its eight channels is the winner in the competition.

## 6.3 Model Based on the Extraction of the Rotation Invariant Features

A circular symmetric AR model was suggested in [18] for extracting the rotation invariant feature of a textured image. The neighborhood in the circular model is composed of eight pixels which are evenly located on a unit radius circle. The summation of eight neighboring pixels is considered as a single variable in the random field, and the intensity of a center pixel is related to the summation through a parameter, which is taken as a rotation invariant feature.

In this section, the circular symmetric AR model is extended to create a multi-circular AR model. The circular neighborhood consists of 36 elements distributed on three circles with radii of one unit, two units and three units (Figure 6.3). The numbers of elements on the three circles are 8, 12 and 16 respectively. The average of the intensity values of pixels on each of those circles is computed and considered

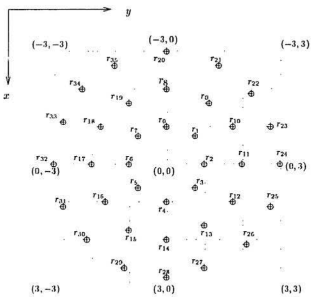


Figure 6.3: Neighborhood for multi-circular AR model.

as one variable, resulting in the following AR model equation

$$\begin{aligned} y(i,j) &= \theta_0 \times \frac{1}{8} \sum_{r_q \in \phi_0} y((i,j) \oplus r_q) + \theta_1 \times \frac{1}{12} \sum_{r_q \in \phi_1} y((i,j) \oplus r_q) \\ &\quad + \theta_2 \times \frac{1}{16} \sum_{r_q \in \phi_2} y((i,j) \oplus r_q) + \beta\mu(i,j), \end{aligned} \tag{6.1}$$

where $\phi_0 \in r_0, ..., r_7$, $\phi_1 \in r_8, ..., r_{19}$ and $\phi_2 \in r_{20}, ..., r_{35}$. When a texture sample is rotated, the sum of gray values of pixels on a circle around $y(i,j)$ remain the same. Therefore, the sum of gray values of pixels on each circle in the neighborhood can be considered as a rotation invariant variable. The three parameters specifying the circular AR model can be used as the rotation invariant features such that sample on arbitrary rotation can be modeled.

Note that the coordinates of most elements in this neighborhood are not integers and thus the gray values of those elements cannot be directly obtained from a digitized image. Interpolation should be performed to estimate the values of those elements from their neighboring digitized pixel values. The interpolated value of a pixel can be obtained via a weighted combination of gray values of its four nearest neighboring pixels, in which the coefficient of the value of each neighboring pixel should be inversely proportional to the distance between the neighboring pixel and the pixel to be estimated. Therefore, for a pixel $z_0(x_0, y_0)$, its gray value can be estimated by a combination of values of its four neighboring pixel $z_1(x_1, y_1)$, $z_2(x_2, y_2)$, $z_3(x_3, y_3)$ and $z_4(x_4, y_4)$ by equation:

$$z_0(x_0, y_0) = \frac{\frac{1}{d_1} * z_1(x_1, y_1) + \frac{1}{d_2} * z_2(x_2, y_2) + \frac{1}{d_3} * z_3(x_3, y_3) + \frac{1}{d_4} * z_4(x_4, y_4)}{\sum_{i=1}^{4} \frac{1}{d_i}}. \quad (6.2)$$

where

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}.$$

According to the coordinates of interpolated pixels and their nearest digitized pixels, the coefficients which specify the contribution of digitized pixels can be calculated using equation 6.2. Since the summation of all pixel values on a circle is of interest, the contributions of each digitized pixel on all interpolated and non-interpolated elements are summed to obtain a coefficient for the pixel. Table 6.1 shows those coefficients. Since these coefficients are symmetric with respect to the center pixel, only the coefficients for half of the pixels in the neighborhood are listed. The summation of intensity values corresponding to each circle is calculated by accumulating the intensity value of each associated pixel weighted by its coefficient.

## 6.4 The Rotation Invariant Experiment

The abilities of the two types of rotation invariant classifier in classifying texture samples with arbitrary rotation angle were tested in experiments. The textured images were digitized from natural objects by using a camera. The natural objects include the cover of a paper box, three patterns of woolen sweaters and four

| Circle 1 | | Circle 2 | | circle 3 | |
|---|---|---|---|---|---|
| Coordinate | Cofficient | Coordinate | Cofficient | Coordinate | Cofficient |
| (0, 0) | 0.6636 | (-1, 1) | 0.3890 | (-1, 2) | 0.1530 |
| (0, 1) | 1.4334 | (0, 2) | 1.2834 | (-2, 1) | 0.1530 |
| (-1, 0) | 1.4334 | (-2, 0) | 1.2834 | (-2, 2) | 0.6559 |
| (-1, 1) | 0.4006 | (-1, 2) | 0.5300 | (0, 3) | 1.2920 |
| | | (-2, 1) | 0.5300 | (-3, 0) | 1.2920 |
| | | (-2, 2) | 0.2834 | (-1, 3) | 0.5656 |
| | | | | (-3, 1) | 0.5656 |
| | | | | (-2, 3) | 0.2728 |
| | | | | (-3, 2) | 0.2728 |
| | | | | (-3, 3) | 0.0904 |

Table 6.1: Coefficients for pixels in a multi-circular model.

patterns of clothes. In order to test rotation invariance, each object was rotated with relative angles of rotation $90^o, 60^o, 30^o$, as well as two other arbitrary relative angles, resulting in five images for each texture. One $64 \times 64$ window of $90^o$ orientation for each texture is shown in Appendix Figure A.2. A $64 \times 64$ window of texture Paper Box Cover for the five orientations is shown in Appendix Figure A.3.

Five textures, including Sweater (1), Cloth (1), Cloth (2), Cloth (3) and Paper Box Cover, were used to test the rotation invariant classification abilities of the multi-direction model. Training samples were arbitrarily chosen from the image of relative orientation of $90^o$ and $45^o$ for each type of texture. The training parameters were selected as $\eta = 0.2$ and $T_2 = 0.0005$. After the weight sets were produced for the texture in orientation of $90^o$ and $45^o$, the weight sets for models at the other six orientations were then obtained by reassigning parameters of the model on $90^o$ or

Assigned Texture

Input Texture
↓

| | Cloth(1) | Cloth(2) | Cloth(3) | Sweater(1) | Box Cover |
|---|---|---|---|---|---|
| Cloth(1) | 11 | 5 | | 9 | |
| Cloth(2) | | 10 | 8 | | 7 |
| Cloth(3) | | 3 | 14 | 8 | |
| Sweater(1) | | 3 | | 22 | |
| Box Cover | | | | | 25 |

Table 6.2: Testing results for multi-direction model.

$45^{o}$ to different notations corresponding to each orientation. In this way, weights for all channels were obtained.

For each texture, five testing samples were selected from each of the images with different relative orientation including the one used in the training phase. This resulted in a total of twenty-five testing samples per texture class. The testing result is presented in Table 6.2.

The performance of the multi-circular model was examined on a set of eight textures, including the textures used in the testing of the multi-directional model and three other textures Sweater (2), Sweater (3), and Cloth (4). Similarly, samples chosen from the images in the orientation of $90^{o}$ were used in the training phase. Since only one model parameter set is needed for each texture in the multi-circular model, the weight set extracted for each texture was directly used as the weight for channel of the texture. The parameters of the training phase are chosen as $\eta = 3$ and $T_2 = 0.000001$. The results are given at Table 6.3.

Input Texture ↓ Assigned Texture

| | Cloth_1 | Cloth_2 | Cloth_3 | Cloth_4 | Sweater_1 | Sweater_2 | Sweater_3 | Box Cover |
|---|---|---|---|---|---|---|---|---|
| Cloth_1 | 16 | | 6 | 3 | | | | |
| Cloth_2 | | 20 | | 5 | | | | |
| Cloth_3 | | | 17 | | | 3 | 5 | |
| Cloth_4 | | | | 21 | | 4 | | |
| Sweater_1 | | | | | 25 | | | |
| Sweater_2 | | | | | | 25 | | |
| Sweater_3 | | | | 4 | 2 | 5 | 14 | |
| Box Cover | | 4 | | | | | | 21 |

Table 6.3: Testing results for multi-circular model.

The experiments show that both multi-direction model and multi-circular model can be used to classify texture samples which have different rotations from the original training samples. From the test results, the performance of the multi-circular model is seems better than that of the multi-direction model, which yields a lower classification accurate rate for some of the textures used in the experiments. The reason should be due to the ability of the multi-circular model in catching the rotation invariant features of textures.

# Chapter 7

# A Texture Segmentation Technique

## 7.1 Introduction

The structure and performance of the proposed neural network for texture classification has been presented in Chapter Five. The modified versions of the network for invariant texture classification are described in Chapter Six. As we have mentioned in the introduction chapter of this thesis, texture segmentation is the other important task of texture discrimination by which an input image of more than one uniform region is partitioned into several homogeneous subimages. The segmentation algorithm described in this chapter is an integrated region extraction technique, which can be considered as an application of the system given in Chapter Five. Please note that, because, by common understanding, regions of the same texture but in different orientation are generally considered as different regions in an image, the rotation invariant classification networks introduced in Chapter Six are not used in this segmentation algorithm.

Region extraction methods for texture segmentation are generally categorized into regions splitting and merging techniques and region growing techniques [40, 42]. In a region splitting and merging technique, an image is divided into blocks and each of the blocks is described in terms of certain region properties. Adjacent blocks which have similar region properties are merged and blocks which have large property differences from their neighboring adjacent blocks are split. One disadvantage of the region splitting and merging technique is its sequential nature. The regions produced sometime depend on the order in which regions are merged together. In a region growing technique, regions are expanded from some starting seed points until the boundaries of texture regions are reached. A pixel (or a group of pixels) can be included into an adjacent determined neighboring region if their region properties are sufficiently similar. The major drawback of a region growing technique is that the starting seed points are often selected in a supervised manner. A common problem for region extraction techniques in general is that the local information, which has to be used heavily in the segmentation, is difficult to extract and incorporate into a segmentation procedure.

The region extraction technique presented in this chapter is an integrated texture segmentation technique which combines the strengths of region splitting and merging techniques and the region growing techniques. This segmentation technique is implemented by comparing local region properties, which are represented by the 2-D AR model, in a hierarchical manner. The details of the segmentation procedure, its performance in natural texture segmentation, as well as the performance evaluation are given in the following sections. For the situation that texture classes in an image can not be known beforehand, a type determining algorithm is proposed to decide the texture types in an image.

## 7.2 A Few Concerns in the Designing of Segmentation Technique

The objective of texture segmentation is to segment an image which consists of different textures into uniform texture regions. In addition, precise and clear boundaries of texture regions are desired. When designing a texture segmentation algorithm, three fundamental issues which should be considered are: (1) the texture property representation; (2) the frame size over which the local texture characteristics are observed; and (3) the segmentation procedure.

The proposed segmentation algorithm uses the 2-D AR model to represent the texture properties. As we have mentioned, for a region extraction technique, the computation time is usually expensive since the algorithms have to be performed in an iterative manner. In the case of using 2-D AR model for texture representation, the efficiency of a segmentation procedure strongly depends on the efficiency of texture feature extraction in each observed window. If the parameter estimation of 2-D AR model has to be done for each window in all iterations, the computation time needed will be large. To make the algorithm practical, in our approach, instead of estimating parameters for each observed window, the total error mathematically described by equation 3.4 is used as a measurement to compare texture properties of different regions.

The selection of an appropriate window size over which texture properties are observed usually depends on the coarseness of the textures of interest. It affects the performance of a segmentation procedure in a certain manner. On the one hand, the size should be large enough so that an enclosed region can exhibit texture properties of the surroundings, while on the other hand, the window size should be small enough to allow the accurate determination of boundaries. Our integrated approach uses a fixed size frame in combination with varying size blocks to ensure both accurate texture feature extraction and boundary determination. The fixed size frame is chosen to be big enough to allow the common texture pictures to exhibit their properties. All texture properties are extracted using this frame throughout the segmentation procedure. On the other hand, blocks with varying size are used as the basic elements for segmentation to form accurate boundaries. For a block smaller than a frame, texture properties are obtained through the fixed size frame in which the undetermined block is placed as the center.

A segmentation procedure which combines the region splitting and merging and region growing techniques operates as follows. Initially, an underlying image is partitioned into blocks with equal size as a splitting and merging technique. Then, for each texture region, an internal area is determined which is considered as the seed part for region growing. There are two differences between the integrated technique and a region growing technique in the determination of these seed parts. Firstly, in a region growing technique, seed parts are often decided in a supervised manner, while in the integrated approach the seed parts are decided in an unsupervised manner. Secondly, a seed part often only contains one pixel in a region growing technique. For the integrated approach, a seed part may contain a large internal area (one or more frames) so that the computational burden for the following region growing procedure is greatly reduced. During the region growing procedure of the integrated approach, adjacent blocks of those determined internal regions are examined in each iteration. A block is merged into its adjacent internal region if its texture property is the same as that of the internal region. Otherwise, the block is partitioned into smaller blocks and then further examined in the next iteration.

One further concern is related to the texture types which exist in an image. In general cases, we assume that some prior knowledge about the types of textures which may appear in images of a particular application is available, such that samples for each possible texture can be provided to the neural network and trained through the learning process for use in the segmentation procedure. In the case that the types of textures appearing in an image to be segmented are not known beforehand, a texture type determining mechanism is proposed at the end of this chapter to decide the texture classes in an image.

## 7.3 The Integrated Region Extraction Procedure

The integrated procedure consists of two stages, namely the initial stage and the refining stage. In the initial stage, large internal areas of texture regions are segmented and the types of textures and the number of texture regions in the examined image are also determined. In the refining stage, the internal regions resulting from the initial stage are extended using blocks with decreasing size in a hierarchical manner until accurate boundaries are formed between texture regions.

In the resulting image, all pixels in each homogeneous region are expected to be labeled by the same label. Clear boundaries will then be shown by the difference in labels of two adjacent regions. Note that, regions with the same texture in different positions of an image are labeled by an identical label.

### 7.3.1 The Initial Stage

Let $G = \{g(m, n) | m \in 1, ..., M, n \in 1, ..., N\}$ represent an image of size $M \times N$. The image is partitioned into disjoint blocks of the frame size $S_f \times S_f$. Let $\{R^{(0)}(p, q) | 0 < p < \frac{M}{S_f}, 0 < q < \frac{N}{S_f}\}$ denote the set of disjoint blocks. The identification procedure is applied to all those blocks. Since the texture types of all blocks inside a homogeneous texture region should be identical, a block in the internal part of a texture region should have the same texture type as all of its neighboring blocks. While a block located at the boundary part of a texture region should have a different texture type than those of its the neighboring blocks belonging to other texture regions. In the initial stage, a block is considered to be part of the internal area of a texture region when it has the same texture class as its four neighboring blocks in the horizontal and vertical directions; otherwise, it is considered as an undetermined block that will be divided and further examined at the refining stage. The algorithm in the initial stage is given as Figure 7.1.

Figure 7.2 shows the initial segmentation result for an image with two texture regions $A$ and $B$. The two areas marked by dashed lines are internal regions decided in the initial stage. For example, block $A_4$ is assigned to be a part of internal area of texture $A$ since the block itself, as well as all of its neighboring blocks $A_1$, $A_5$, and $A_6$, are also identified as texture $A$. Blocks located beside or across the texture boundary parts will be considered as undetermined blocks. For example, though block $A_3$ is identified as texture type $A$, as the major portion of

```
Scan all blocks from left to right and top to bottom:

if     $R^{(0)}(p,q)$           $\in$   $A$   and
       $R^{(0)}(p \pm 1, q))$   $\in$   $A$   and
       $R^{(0)}(p, q \pm 1))$   $\in$   $A$
then   label block $R^{(0)}(p,q)$ to region A
else   $R^{(0)}(p,q)$ stays undetermined
```

Figure 7.1: The algorithm of the initial stage.

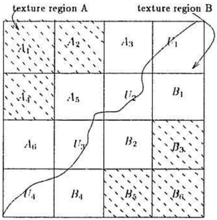


Figure 7.2: An example of initial stage.

this block belongs to texture $A$, this block cannot be segmented to the internal part of region $A$ because it has a different type than that of the two neighboring blocks $U_2$ and $U_3$, whose major parts are in region $B$. In this case, a mis-segmentation would occur if block $A_3$ was to be segmented to the internal area of region $A$ since a small portion of this block belongs to region $B$.

During the initial stage, the number of the regions and the texture types included in the image are also computed. To improve the accuracy, the texture types of the blocks considered in the subsequent refining procedure are constrained to the set of texture types identified in the initial stage. Therefore, in the multilayer network, the output of channels corresponding to those textures which are not included in the texture types of the image will be suppressed in the refining stage so that the competition is only conducted among the outputs of channels of the textures decided in the initial stage.

## 7.3.2 The Refining Stage

In the first iteration of the refining stage, undetermined blocks of frame size $S_f \times S_f$ are divided into four smaller blocks of size $\frac{S_f}{2} \times \frac{S_f}{2}$. Each of the internal regions produced by the initial stage is then extended by merging adjacent blocks of size $\frac{S_f}{2} \times \frac{S_f}{2}$ from the boundaries of internal regions toward boundaries of the texture regions. When no more changes occur to blocks of size $\frac{S_f}{2} \times \frac{S_f}{2}$, all undetermined blocks will be divided to smaller blocks of size $\frac{S_f}{4} \times \frac{S_f}{4}$. This procedure is recursively performed until no more undetermined blocks are left in the image or the block size of $1 \times 1$ is reached. In each iteration, two issues to be considered are: (1) determining the neighboring blocks of internal region; and (2) determining the texture type for a neighboring block of an internal region.

Let $R^{(k)}(p,q)$ denote a block at the stage when the block size is reduced to size $\frac{S_f}{2^k} \times \frac{S_f}{2^k}$ in the refining procedure, where $0 \leq p \leq \frac{2^k \times M}{S_f}$ and $0 \leq q \leq \frac{2^k \times N}{S_f}$. This block is considered to be adjacent to an internal region $A$ if any three of its neighboring adjacent blocks have already been determined to belong to the internal region.

To determine the texture type of block $R^{(k)}(p,q)$, a frame containing this block as its center is examined. If the identification process shows that block $R^{(k)}(p,q)$ and its neighboring internal region have identical texture, the block can possibly be included to the texture region. However, in order to ensure segmentation accuracy, this result has to be justified by examining other neighboring blocks of block $R^{(k)}(p,q)$. For each of the undetermined neighboring blocks, a frame is located in the same manner as that of block $R^{(k)}(p,q)$ to examine the texture type of the undermined block. When the neighboring blocks other than the three neighboring blocks included in the internal region also show the same texture as the internal region, block $R^{(k)}(p,q)$ can be merged into the internal region. Otherwise, the refining procedure continues to investigate whether the block can be segmented into other adjacent internal texture regions, if there are any.

It should be noted that the examination of the texture types of neighboring blocks is very critical in forming smooth region boundaries on the resulting images. In most cases, a block across the boundary of two texture regions contains different portions of two textures. The texture type of the block is likely to be decided as the texture type of the major portion when the block is examined. If the block is included into the internal region corresponding to the major portion, the small portion of the other region will be mis-segmented. As a consequence, a boundary with a sawtooth shape will be produced, as is often observed in images derived using existing algorithms, instead of a smooth one. This problem is avoided effectively in the segmentation procedure described here.

Let $F(x, y)$ denote a frame starting at position $(x, y)$ on image $G$. The algorithm of the refining stage is shown as Figure 7.3.

An example of the refining stage is given as Figure 7.4. The center block in this figure is denoted as $a_0$. The neighboring blocks $a_1$, $a_2$, $a_4$ have already been determined as texture $A$. By using a frame positioned at the center of $a_0$, the texture type of $a_0$ is identified as type $A$ which is the same as its neighboring internal region. To verify this result, the neighbor blocks $a_3$, $a_5$, $a_6$, $a_7$ and $a_8$ are examined. In this case, the texture type of $a_3$ and $a_6$ have already been decided as type $A$. The remaining blank blocks $a_5$, $a_7$ and $a_8$ are then also identified as type $A$ through the network computation. Therefore block $a_0$ is merged to the internal area of texture $A$. However, when the same procedure is applied to block $a_8$, $a_8$

```
$k = 1$;
do {
    do {
            scan all blocks from left to right and top to bottom:
            {
                if ($R^{(k)}(p,q)$ is undetermined)
                then {
                        if (three adjacent neighboring blocks of $R^{(k)}(p,q) \in A$)
                        then {
                                locate a frame $F(x,y)$ so that block $R^{(k)}(p,q)$ is
                                                        at the center of $F(x,y)$
                                                where
                                                        $x = S_f \times (\frac{p}{2^k} + \frac{1}{2^{(k-1)}} - 1)$
                                                        $y = S_f \times (\frac{q}{2^k} + \frac{1}{2^{(k-1)}} - 1)$;
                                if ($F(x,y) \in A$)
                                then {
                                        locate frames for all undetermined neighboring
                                                                        blocks of $R^{(k)}(p,q)$;
                                        if (all neighboring frames $\in A$)
                                        then label $R^{(k)}(p,q)$ to region $A$;
                                        else $R^{(k)}(p,q)$ stays undetermined;
                                }
                                else $R^{(k)}(p,q)$ stays undetermined;
                        }
                }
            }
    } while (no block is newly labeled in this iteration)
    $k = k + 1$;
} while ($\frac{S_f}{2^k}$ >= 1 and there are undetermined blocks)
```

Figure 7.3: The algorithm of the refining stage.

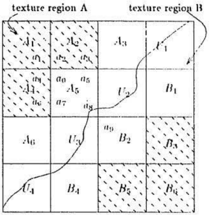


Figure 7.4: The example of refining stage

stays undetermined since its neighboring block $a_9$ cannot be considered to have the same texture as region $A$. Thus $a_8$ will be divided into four smaller blocks which will be examined individually in the next iteration.

To examine the performance, images from the Brodatz album [70] were used to form collaged images since they are frequently used by researchers for the comparison of algorithms. Two considerations in the creating of these collaged images are: (1) visually similar textured regions were applied to demonstrate the robust performance of the algorithm; and (2) boundaries between two textured regions include straight lines in different directions and circles. Hence, the segmentation results from the images can be used to show the performance on all possible boundary sceneries. Fifteen textures from the album were used, including Pressed cork

(D 4), Herringbone weave (D 16) and Herringbone weave (D 17), French canves (D 20), Pressed calf leather (D 24), Beach sand and pebbles (D 27), Beach sand (D 28) and Beach sand (D 29), Water (D 37), Straw matting (D 56), Handmade paper (D 57), Wood grain (D 68), Cotton canvas (D 77), Oriental straw cloth (D 81) and Raffia looped to a high pile (D 84).

The frame size was selected as $32 \times 32$. A frame smaller than this size was considered as not big enough to exhibit texture characteristics. Therefore, the block size in the initial stage was $32 \times 32$ and it could be reduced to $16 \times 16$, $8 \times 8$, ... , and $1 \times 1$ in the refining procedure. Figure 7.5 shows the segmentation result for the image containing two similar textures Pressed cork (D 4) and Cotton canvas(D77) (Figure 7.5(a)). The situation after the initial segmentation stage is shown in Figure. 7.5(b), in which two internal regions are formed and an undetermined area is left on the image. Figure. 7.5(c) - (e) are the intermediate results when the decreasing block size is used in the refining procedure; note that in Figure 7.5(c), which is the result after the first iteration of the refining stage, the major areas on the image have been determined. Figure 7.5(f) is the final segmentation result, in which two textures are successfully discriminated and a smooth boundary is formed. For comparison between the resulted boundary and the expected boundary, a bright circle as an idea boundary for the textured image is superimposed on the Figure 7.5(b)–(f).

Another example in which the original image contains three textures, Pressed cork (D4), Beach sand (D28) and Water (D37), is given as Figure 7.6. The original image is in Figure 7.6(a). Its final segmentation is shown in Figure 7.6(f), in which the improvement of the boundary part is clearly shown. The remaining experimental results are listed in the Appendix Figure A.4.

The experiments show that the detected texture regions have a good agreement with the the actual ones. Nearly no mis-segmented pixels appear in the interior parts of texture regions. Smooth boundaries are formed not only for regions with straight line boundaries but also for regions of circular boundaries, which are generally believed to be difficult to segment. Some parts of images (such as narrow corners) cannot be segmented accurately since their areas are not large enough to contain a frame.

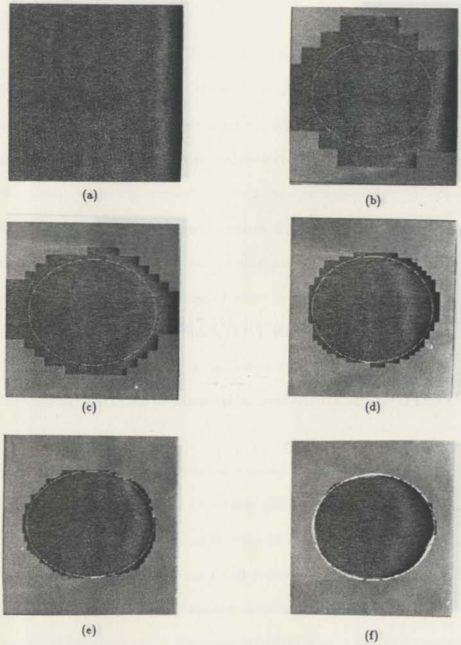


Figure 7.5: A segmentation result. (a) the original image; (b) the result of the initial stage; (c)-(e) the intermediate results of the refining stage; (f) the final result.

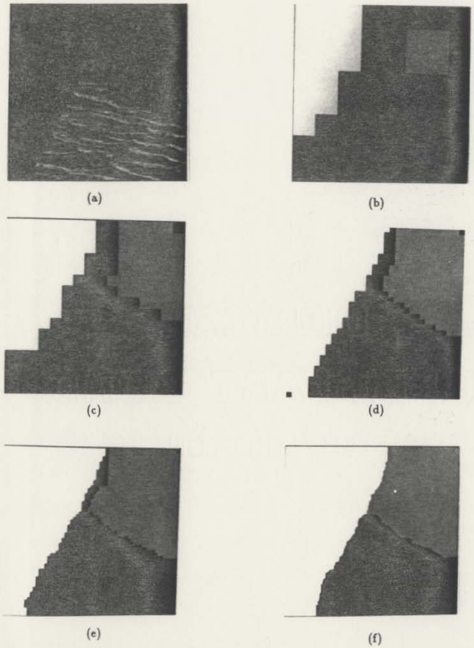


Figure 7.6: A segmentation result. (a) the original image; (b) the result of the initial stage; (c)–(e) the intermediate results of the refining stage; (f) the final result.

### 7.3.3 Result Evaluation

A segmentation method is usually evaluated according to both the amount of mislabeled subregions in interior parts of texture regions and the accuracy in locating region boundaries. However, it is difficult to judge and compare among various segmentation techniques in part because of the lack of commonly appropriate quantitative measures, and the different ways or sources in construction and selection of the testing images. A simple and common criterion applied in segmentation evaluation is the percentage of mis-segmented pixels, which is calculated by dividing the number of all mis-segmented pixels in an image by the total number of pixels. However, we should notice that this measure cannot accurately reflect the boundary accuracy. Besides, it also varies with the percentage of boundary area in a testing image because the mis-segmented pixels mostly occur around the boundary parts.

A set of quantitative error measures, namely the mean boundary error, the maximum error over the length of a boundary and the root-mean-square boundary error, suggested in [46] are more appropriate in evaluating the segmentation of boundary areas. As these three quantitative error measures are linearly related with each other, only the mean boundary error is used in this thesis to evaluate the quality of boundary parts. The mean boundary error is defined by averaging the line-by-line difference between the boundary in the original test image and the boundary in the segmentation image [46].

| Testing Image | Size | No. of Regions | Boundary | Mis-segmented (%) | Mean Boundary Error(pixels) |
|---|---|---|---|---|---|
| $D_{17\_24\_81\_16}$ | $256 \times 256$ | 4 | Straight line | 4.69 | 5.13 |
| $D_{56\_37\_4\_68}$ | $256 \times 256$ | 4 | Straight line | 5.92 | 8.73 |
| $D_{17\_24}$ | $300 \times 287$ | 2 | Circle | 2.12 | 2.57 |
| $D_{24\_4}$ | $300 \times 278$ | 2 | Circle | 1.98 | 2.26 |
| $D_{28\_24}$ | $319 \times 292$ | 2 | Circle | 4.57 | 3.83 |
| $D_{37\_4}$ | $300 \times 281$ | 2 | Circle | 3.48 | 3.32 |
| $D_{57\_29}$ | $350 \times 305$ | 2 | Circle | 2.86 | 3.22 |
| $D_{77\_24}$ | $340 \times 275$ | 2 | Circle | 1.39 | 2.13 |
| $D_{77\_4}$ | $340 \times 275$ | 2 | Circle | 1.25 | 2.28 |
| $D_{84\_24}$ | $350 \times 289$ | 2 | Circle | 1.30 | 1.90 |
| $D_{65\_20}$ | $300 \times 278$ | 2 | Circle | 4.39 | 3.31 |
| $D_{84\_4}$ | $350 \times 289$ | 2 | Circle | 1.65 | 2.62 |

Table 7.1: The quantitative evaluation of the segmentation results.

Table 7.1 shows the evaluation results of using both criterion on twelve resulting images in the segmentation experiments. The first two images were selected because they contain four texture regions. The remaining ten images consist of regions with circle boundaries. For the percentage of mis-segmented pixels, a range from 1.30% to 5.92% of mis-segmentation rates were obtained. Promising results were also shown by the calculation of the mean boundary error, in which significant small errors were yielded in those testing images of two regions with circular boundaries.

The results given in Table 7.1 are comparable with those presented in [46], in which the performance of texture segmentation using different texture features were compared. Eighteen testing images, each of which was composed by simply connecting two texture images of square shape, were used in that paper to evaluate the segmentation performance. Regarding the mean boundary error, a range of 2 to 8 pixel errors was reported in testing using the Haralick (co-occurrence matrix) and Laws methods, which were considered to be good results among all testing results given in that paper. In Table 7.1, the mean boundary errors for the two images of four regions are comparatively larger (5.13 pixel and 8.73 pixel) than that of other images in the table, since more sophisticated situations (four types of texture joining in one area) appear on both images. However, these two results should still be considered as satisfactory results when comparing with those reported in [46].

## 7.4 A Type Determining Mechanism

The segmentation procedure introduced above assumes the types of textures which may appear in an image to be segmented belong to a fixed set of textures and that samples of textures have already been provided to the network for training before the network is used in the segmentation. However, in some circumstances, the types of textures that appear in an image may not be known beforehand. A type determining mechanism must be used beforehand to decide the texture classes in such images. For this procedure, the user needs to provide the number of textures (N) in an image.

We first define an average difference value, called the range difference, which measures the variance in relative total error values of a neighborhood. Assume $\theta_{r_q}$ is the AR model parameter set of a center block in an image and $E_0$ is the corresponding total error value of the block. Let $E_1, ..., E_8$ denote the total values of its eight neighboring blocks which are obtained by applying $\theta_{r_q}$ to each of these blocks respectively. The range difference can be defined as:

$$D_A = \frac{|E_0 - E_1| + |E_0 - E_2| + ... + |E_0 - E_8|}{8}. \tag{7.1}$$

When the center block in the original image is an interior part of a texture region, the total error values of adjacent blocks are on one level, so the value of the range difference is small. When the center block is on a texture boundary, the value of the range difference will be large.

A procedure can then be designed to decide the types of textures. The first step is to label disjoint blocks of frame size in an image to be segmented according to the range difference value. For each block in the underlying image, if all of its examined neighboring blocks are not labeled, the block is presented to the network, resulting in a set of parameters. A range difference value is then calculated. If the range difference value is less than a threshold value $T_D$, a new label is assigned to the block. If not, this block stays unlabeled. On the other hand, if one of the neighbor blocks has already been labeled, a range difference value is computed by applying the parameter set of the corresponding labeled neighboring region to the center block and all its neighboring blocks. If this value is less than $T_D$, the block is assigned the same label with the labeled neighboring block.

The second step is to combine the adjacent regions into one region. After the second step, a number of disconnected labeled regions are obtained. Assume that the number of disconnected regions is $M$. On the principle that the distance between any two parameter sets in a cluster is less than the distance between two sets in two cluster, these $M$ sets of parameters can be clustered to $N$ categories (N is the number of textures in the image provided by user). In each category, the 2-D AR model can be obtained by averaging the parameter sets included. In this way, the AR models for textures in an image can be obtained.

The type determining mechanism was tested on four images which contain two, three, or four types of textures respectively. For each image, parameter sets were first estimated using the type determining mechanism and consequently, the integrated segmentation procedure was applied to perform the segmentation. The threshold value $T_D$ was selected as 0.04. These four images and their results are shown at Appendix Figure A.5.

# Chapter 8

# Conclusions and Future Research

## 8.1 Summary of Contributions

In this thesis, approaches of texture classification and segmentation which use the 2-D AR model for texture property representation are studied. A multilayer neural network has been developed to perform texture feature extraction for texture classification and segmentation. The network consists of three subnets, i.e., the input subnet, the analysis subnet and the classification subnet. The input subnet receives an input and distributes the normalized input pattern to the analysis net. The analysis subnet consists of a set of channels, each of which models a type of texture by its weighted connections and producing a total error which measures the difference between an input texture and the texture modeled by the channel. The classification subnet is a competition mechanism which decides the texture class to which the input pattern belongs.

The learning procedure of the network maps the process of establishing an AR model for a textured image into a neural computation so that a weight set produced by the neural computation is the 2-D AR model parameter set of the input texture. A generalized delta rule is designed by which the weight set is computed according to the gradient of the error space using an iterative process. The use of the sigmoidal activation function successfully suppresses the contribution of the random noise in a uniform texture region and increases the accuracy of a weight set resulting from the learning process. It also improves the system stability during the classification and the segmentation phases.

For the situation in which a testing image has a different orientation from the training samples, two types of rotation invariant classifier have been proposed. In the first type, models established for the same texture related to eight directions are considered. A testing sample with unknown rotation can be approximately modeled by the one of the eight models which has the smallest angle difference to the testing sample. For the second type, a circular neighborhood of thirty-six pixels is expanded from the circular autoregressive model proposed in [18]. The AR model using this circular neighborhood is able to extract the rotation invariant features of a texture so that a testing sample with arbitrary orientation can be classified.

For textured image segmentation, an integrated region extraction technique has been designed. This technique is implemented by comparing local region properties, which are computed by use of the neural network, in a hierarchical manner. This technique grows all regions in a textured image simultaneously starting from internal regions to the boundaries of textures using blocks of decreasing size. The internal regions, as well as the number and the type of the textures included in an image are decided in the initial stage of the procedure. In the refining stage, the properties of neighboring blocks are used to determine the local texture property of a underlying block. To handle the situation in which the types of textures appearing in an image under investigation are not known beforehand, a type determining mechanism is designed to be used before the segmentation procedure to decide the textures included in the image.

A series of experiments were conducted to test the performances of the proposed texture classification and segmentation techniques on natural textured images. A classification problem of thirty-eight natural textures provided by the the Brodatz album was considered to demonstrate the ability of the artificial neural network in classification. Highly satisfactory results were achieved for most of the textures participating in the testing. The abilities of the two types of rotation invariant classifiers in classifying test samples of unknown rotation were tested on images of six natural objects. Samples on chosen directions were provided to the networks for training, and then testing samples of arbitrary directions were presented to be recognized. For the multi-circular model, 60% to 100% correct classification rates were reached. The result from the multi-direction model is poorer than that of the multi-circular model, however for most of the textures, 80% to 90% classification rates were still obtained. The texture segmentation algorithm was tested on a number of images composed of natural textures from Brodatz Album. Based on the performance evaluation, it can be concluded that the segmentation procedure performs very well. The type determining mechanism was also tested on several images which contain two to four types of textures respectively. When the number of types of textures is provided, the type determining mechanism can successfully locate the internal parts of texture regions and subsequently provide a set of AR model parameters corresponding to all textures in the image. Generally, satisfactory segmentation is shown in the resulting images from using the type determining mechanism followed by the segmentation procedure, though some of the boundaries are not sufficiently smooth due to the less accurate location of some internal regions when a unique threshold value is required to apply to all underlying images.

There are several advantages to the neural network texture classifier proposed in this thesis. First of all, the adaptive learning process accurately estimates an AR model for a texture by using the sigmoidal activation function and the generalized delta rule. This gives the system a strong stability in producing good texture features which have small differences to variances inside a texture and large differences to variances between different textures. Second of all, the network is easy to extend because of its modular structure in which all channels work independently. Only one more channel and its corresponding training procedure are needed to enable the system to distinguish a new type of texture; moreover, such an extension does not affect existing channels. Finally, a high computation rate is provided by the network in response to the input testing samples presented to the input layer. Unlike traditional classifiers, which tend to process competing hypotheses sequentially, the neural network classifier tests competing hypotheses in parallel.

Regarding the segmentation procedure, as a frame is input and forwarded directly through the layers in the network, the computation time for texture feature extraction on each examined frame in every iteration of the segmentation procedure is dramatically reduced. Other important advantages of the segmentation algorithm are that the number and the locations of the regions, as well as the major internal areas of large regions can be efficiently determined in the initial stage. Unlike some other texture segmentation techniques such as clustering or edge detection methods, the number or location of regions and the approximate segmentation may not be produced until the whole segmentation procedure is completed. Using this algorithm, an approximate segmentation is typically reached after one or two iterations in the refining stage.

An additional important advantage of the segmentation procedure is that the neighboring texture properties are used to determine texture type for each block and ensure an accurate segmentation. This results in two improvements: (1) the generation of smooth boundaries; and (2) the elimination of mis-labeled interior pixels. Due to variances in a texture, it frequently happens that some interior points in a region are mis-labeled using segmentation techniques. An additional smoothing step should be taken to eliminate such mis-labeled points in those techniques. However, very few mis-labeled interior points occur in our results for this segmentation procedure. Taking a block instead of a pixel as a basic unit for segmentation seems to be another reason for the strong ability to overcome a small area of nonuniformity in a texture region.

## 8.2 Future Research

### 8.2.1 Improving the Learning Speed

The speed of learning tends to be slow when a presented image contains large primitives. A way to improve the speed is to add a momentum term to the weight modification equation 4.15. This momentum term is proportional to the amount of the weight change in the last cycle of the weight adaptation. This term indicates that the change in a weight in the current step should be related to the change in the last step. The coefficient of the momentum term can be varied according to the total error change during the weight modification to speed up the learning process. As the total energy of the system is decreased, the value of the coefficient of the momentum term is increased adaptively.

### 8.2.2 Improving the Accuracy of the Segmentation on Narrow Parts of Textures

An accurate segmentation cannot be achieved for narrow parts of textures such as small corners since these areas are not large enough to contain a frame. One possible way of improving the performance in this case is to duplicate a block under examination to form a bigger area and then apply the segmentation procedure to the formed area to decide the texture type for the block. The existence of the narrow parts on texture images can then be detected by the neighboring situation of the block.

### 8.2.3 The Automatic Determination of the Number of Textures

In the current algorithm, when no prior knowledge about the types of different textures in a underlying image is available, the type determination mechanism is performed based on the number of the textures supplied by the user. Determining the number of texture categories in an image is a very difficult problem and no single method has provided a satisfactory solution so far. Related methods are developed by comparing the results when different number of textures are supposed and selecting the one which shows the best result. A frequently used method is the modified Hubert (MH) index [43], which is a measure of correlation between the matrix of pattern distances and the matrix of class distance when mapping back the patterns into cluster solution. The true number of texture categories is estimated by identifying a significant knee in the curve of MH with different numbers of textures.

One possible method that can be used for the segmentation technique proposed in this thesis is to compare the value of the total variance when different numbers of textures are assumed. Given a texture number $M$, a corresponding number of $M$ sets of AR parameters can be obtained by using the type determination mechanism. By subsequently applying the parameter sets back to the image, the value of total variance can be calculated by accumulating the total error values when classifying each disjoint frame of the original image based on estimated parameters sets. When the given texture number $M$ is less than the real number of textures, the value of the total variance is large. As $M$ is increased from the minimum of two textures to the real number of textures, the total variance should be decreased. After the number of textures is equal to the real number of textures, the total variance should be maintained at a constant level. Therefore the true number of textures can be determined by searching for the significant knee of the curve of the total variance.

### 8.2.4 Incorporating Scale Changes

Another desirable property for a texture classifier is the invariance to changes in scale of an image from the zooming of a camera. Effective and computationally reasonable methods of scale changes invariant classification have not been seen in the literature because of the difficulty of extracting the common features of scaled textures. The choice of the feature, the size of the neighborhood and the window all have to be adapted as the scales of images are changed. Therefore in practice, instead of developing a scale invariant classifier which could accept arbitrary scales of testing images, methods based on multiscale are developed in [44, 45]. For the network in this thesis, expanding the current neural network to distinguishing textured images of multiscale can be another direction of future research. One implementable method is training each type of texture using testing samples on different scales respectively. More sophisticated features which incorporate scale changes should also be studied in future research.

# Bibliography

[1] L. V. Gool, P. Dewaele and A. Oosterlinck, "Texture Analysis Anno 1983", *Computer Vision, Graphics, and Image Processing*, 29, 336-357, 1985.

[2] R. M. Haralick, "Statistical and Structural Approaches to Texture", *Proceedings of the IEEE*, 67(5), 786-804, May 1979.

[3] R. M. Haralick, K. Shanmugam and I. Dinstein, "Texture Features for Image Classification", *IEEE Trans. Syst. Man. and Cybern.*, 3(6), 610–621, 1973.

[4] J. S. Read and S. N. Jayaramamurthy, "Auto Generation of Texture Feature Detectors", *IEEE Trans. Comput.*, 20(7), 803-812, 1972.

[5] D. Terzopoulos and S. W. Zucker, "Detection of Osteogenesis Imperfecta by Automated Texture Analysis", *Computer Vision Graphics Image Processing*, 20, 229-243, 1982.

[6] L. S. Davis, S. A. Johns and J. K. Aggarwal, "Texture Analysis Using Generalized Co-occurrence Matrices", *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(3), 251-259, 1979.

[7] S. Y. Lu and S. Fu, "Stochastic Tree Grammar Inference for Texture Synthsis and Discrimination", *Computer Graphics and Image Processing*, 9, 234-245, 1979.

[8] J. S. Weszka, C. R. Dyer and A. Rosenfeld, "A Comparative Study of Texture Measures for Terrail Classication", *IEEE. Trans. Syst. Man. and Cybern.*, 6(4), 269-285, 1976.

[9] K. I. Laws, "Textured Image Segmentation", *Ph.D. Thesis*, University of Southern California, Jan. 1980.

[10] L. N. Kanal, "Markov Mesh Models", *Computer Graphics and Image Processing*, 12, 4, 371-375, 1980.

[11] M. Hassner and J. Sklansky, "The Use of Markov Random Fields as Models of Texture", *Computer Graphics and Image Processing*, 12(4), 357–370, 1980.

[12] G. R. Cross and A. K. Jain, "Markov Random Field Texture Models", *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(1), 25–39, 1983.

[13] J. T. Tou, D. B. Kao and Y. S. Chang, "Pictorial Texture Analysis and Synthesis", *Third Int. Joint Conf. on Pattern Recognition*, 590a–590p, Aug. 1976.

[14] B. Kartikeyan and A. Sarkar, "An Identification Approach for 2-D Autoregressive Models in Describing Textures", *Computer Vision, Graphics, and Image Processing*, 53(2), 121–131, 1991.

[15] R. Chellappa and R. L. Kashyap, "Texture Synthesis Using 2-D Non-Causal Autoregressive Models", *IEEE Trans. Acoust. Speech Signal Process*, 33, 959-963, 1985.

[16] P. Souza, "Texture Recognition via Autoregression", *Pattern Recognition*, 15(6), 471-475, 1982.

[17] R. L. Kashyap, R. Chellappa and A. Khotanzad, "Texture Classification Using Features Derived from Random Field Models", *Pattern Recognition Letter*, 1, 43–50, 1982.

[18] R. L. Kashyap and A. Khotanzao, "A Model-Based Method for Rotation Invariant Texture Classification", *IEEE Trans. on Pattern Anal. Mach. Intell.*, 8(4), 472–481, 1986.

[19] L. S. Davis, S. Johns and J. K. Aggrawal, "Texture Analysis Using Generalized Co-Occurrence Matrices", *IEEE Trans. Pattern Anal. Mach. Intell.*, 1, 251-259, 1979.

[20] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems", *J. Royal Stat. Soc. Ser. B*, 36, 192–236, 1974.

[21] F. R. Hansen and H. Elliott, "Image Segmentation Using Simple Markov Field Model", *Computer Graphics and Image Processing*, 20, 101–132, 1982.

[22] H. Derin and H. Elliott, "Modeling and Segmentation of Noise and Textural Image Using Gibbs Random Fields", *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1), 39–55, 1987.

[23] H. Derin and W. S. Cole, "Segmentation of Textured Images Using Gibbs Random Fields", *Computer Vision, Graphics, and Image Processing*, 35, 72-98, 1986.

[24] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Image", *IEEE Trans. Pattern Anal. Mach. Intell.*, 6, 721–741, 1984.

[25] H. Derin and C-S Won, "A Parallel Image Segmentation Algorithm Using Relaxation with Varying Neighborhoods and Its Mapping to Array Processors", *Computer Vision, Graphic, and Image Processing*, 40, 74–78, 1987.

[26] R. L. Kashyap and R. Chellappa, "Estimation and Choice of Neighbors in Spatial-Interaction Models of Images", *IEEE Trans. on Information Theory*, 29(1), 60–72, 1983.

[27] R. L. Kashyap and R. Chellappa, "Decision Rules for Choice of Neighbors in Random Field Models", *Computer Vision, Graphic, and Image Processing*, 15, 301–318, 1981.

[28] L. Wang and D. C. He, "Texture Classificatoin Using Texture Spectrum", *Pattern Recognition*, 23(8), 905–910, 1990.

[29] A. L. Vickers and J. W. Modestino, "A Maximum Likelihood Approach to Texture Classification", *IEEE Trans. Pattern Anal. Mach. Intell.*, 4, 61–68, 1982.

[30] J. Mao and A. K. Jain, "Texture Classification and Segmentation Using Multiresolution Simultaneous Autoregressive Models", *Pattern Recognition*, 25(2), 173–188, 1992.

[31] R. Conners and C. Harlow, "A Theoretical Comparison of Texture Algorithms", *IEEE Trans. Pattern Anal. Mach. Intell.* 3, 25–39, 1980.

[32] S. W. Zucker, "Picture Segmentation by Texture Descrimination", *IEEE Trans. Comput.*, 24, 1288–1233, 1975.

[33] K. S. Fu and J. K. Mui, "A Survey on Image Segmentation", *Pattern Recognition*, 13, 3–16, 1981.

[34] A. Khotanzao and A. Bouarfa, "Image Segmentation by a Parallel, Non-Parametric Histogram Based Clustring Algorithm", *Pattern Recognition*, 23(9), 961–973, 1990.

[35] P. M. Narendra amd M. Goldberg, "A Non-Parametric Clustering Schema for LANDSAT", *Pattern Recognition*, 9(4), 207–215, 1977.

[36] X-H Yu, J. Yla-Jasski and B-Z Yuan, "A New Algorithm for Texture Segmentation Based on Edge Detection", *Pattern Recognition*, 24(11), 1105–1112, 1991.

[37] D. C. He and L. Wang, "Detecting Texture Edges from Images", *Pattern Recognition*, 25(6), 595–600, 1992.

[38] A. Khotanzao and J. Y. Chen, "Unsuperised Segmentation of Textured Images by Edge Detection in Multidimension Features", *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(4), 414–421, 1989.

[39] A. Rosenfeld and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis", *IEEE Trans. Comput.*, 20, 562–569, 1971.

[40] M. Pietikainen and A. Rosenfeld, "Image Segmentation by Texture Using Pyramid Node Linking", *IEEE Trans. Syst. Man and Cybern.*, 11(11), 822–825, 1981.

[41] P. C. Chen and T. Pavlidis, "Segmentation by Texture Using Correlation", *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(1), 61–68, 1982.

[42] T. R. Reed, H. Wechsler and M. Werman, "Texture Segmentation Using a Diffusion Region Growing Technique", *Pattern Recognition*, 23(9), 953–960, 1990.

[43] R. C. Dubes, "How Many Clusters Are Best? – An Experiment", *Pattern Recognition*, 20(6), 645–663, 1987.

[44] J. You and H. A. Cohen, "Classification and Segmentation of Rotated and Scaled Textured Images Using Texture 'Tuned' Masks", *Pattern Recognition*, 26(2), 245–258, 1993.

[45] L. I. Larkin and P. J. Burt, "Mutil-Resolution Texture Energy Measures", *IEEE Proc. of Conf. on Computer Vision and Pattern Recognition*, Washington D.C., pp. 194–200, June 19-23, 1983.

[46] J. M. H. Du Buf, M. Kardan and M. Spann, "Texture Feature Performance for Image Segmentation", *Pattern Recognition*, 23(3/4), 291–309, 1990.

[47] G. A. Carpenter, "Neural Network Models for Pattern Recognition and Associative Memory", *Neural Networks*, 2, 243–257, 1989.

[48] G. A. Cappenter and S. Crossberg, "ART2: Self-Origanization of Stable Category Recognition Codes for Analog Input Patterns", *Applied Optics*, 26(23), 4919–4930, Dec. 1987.

[49] G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics, and Image Processing*, 37, 54–115, 1987.

[50] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical System Using Neural Network", *IEEE Trans. on Neural Nework*, 1(1), 1990.

[51] D. J. Burr, "Experiments on Neural Net Recognition of Spoken and Written Text", *IEEE Trans. Acoust., Speech, Signal Process*, 36(7), 1162–1168, 1988.

[52] B. Widrow, R. G. Winer and R. A. Baxter, "Layered Neural Nets for Pattern Recognition," *IEEE Trans. Acoust., Speech, Signal Process*, 36(7), 1109–1118, 1988.

[53] R. H. Silverman and A. S. Noetzel, "Image Processing and Pattern Recognition in Ultrasonograms by Backpropagation", *Neural Network*, 3, 593–603, 1990.

[54] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representation by Error Propagation", *Parallel Distributed Processing*, The MIT Press Cambridge, Massachusetts, London, England, 318–362, 1988.

[55] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Company, Inc, 1989.

[56] S. I. Gallant, " Perceptron-Based Learning Algorithms", *IEEE Trans. on. Neural Networks*, 1(2), 179–191, 1990.

[57] K. Fukushima and S. Miyake, "Neocognition: A Neural Network Model for a Mechanism of Visual Pattern Recognition", *IEEE Trans. Syst. Man and Cybern.*, 13(5), 826–834, 1983.

[58] K. Fukushima, "Neocognition: A Hierarchical Neural Network Capable of Visual Pattern Recognition", *Neural Networks*, 1, 119–130, 1988.

[59] A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics, and Image Processing*, 37, 54–115, 1987.

[60] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, 43, 59–69, 1982.

[61] S. D. You and G. E. Ford, "Connectionist Model for Object Recognition", *SPIE Proc. Application of Artificial Neural Network*, Orlando, Florida, 200–207, Apr. 1992.

[62] J. M. Aguiler and J. L. Contreras-Vidal, "Image Segmentation Through Gabor-Based Neural Networks", *SPIE Proc. Application of Artificial Neural Network*, Orlando, Florida, 44–51, Apr. 1992.

[63] S. Lu and A. Szeto, "Improving Edge Measurement on Noisy Images by Hierarchical Neural Networks", *Pattern Recognition Letters*, 12, 155–164, 1991.

[64] J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", *Proc. Nat. Acad. Sci.*, 81, 3088–3092, May 1984.

[65] J. Hopfield and D. W. Tank, " 'Neural' Computation of Decisions in Optimization Problems", *Biol. Cybern.*, 52, 141–152, 1985.

[66] C. L. Huang "Parallel Image Segmentation Using Modified Hopfield Model", *Pattern Recognition Letters*, 13, 345–353, 1992.

[67] S. S. Yu and W. H. Tsai, "Relaxation by the Hopfield Network", *Pattern Recognition*, 25(2), 197–209, 1992.

[68] N. M. Nasrabadi and C. Y. Choo, "Hopfield Network for Stereo Vision Correspondence", *IEEE Trans. on Neural Networks*, 3(1), 5–13, 1992.

[69] A. Ghosh and S. K. Pal, "Neural Network, Self-Organization and Object Extraction", *Pattern Recognition Latters*, 13, 387–397, 1992.

[70] P. Brodatz, *Textures: A Photographic Album for Artisters and Designers*. New York, Dover 1966.

[71] S. W. Lu and H. Xu, "Classification Of Texture Images Using Artificial Neural Network", *International Conference on Signal Processing '93/Beijing*, Oct., 1993.

[72] S. W. Lu and H. Xu, "Textured Image Segmentation Using Autoregressive Model and Artificial Neural Network", *IEEE. SMC'93 Conference*, Le Toquet, France, Oct., 1993.

# Appendix

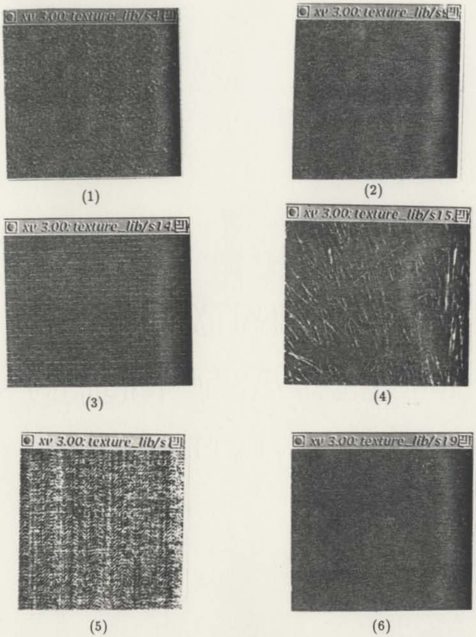
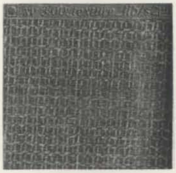
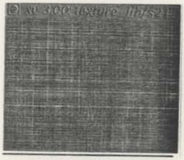
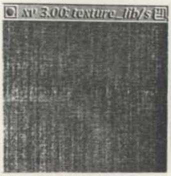


Figure A.1: Textures from the Brodatz album for classification experiments.
(1) Pressed cork (D4); (2) Grass lawn (D9);
(3) Woven aluminum wire (D14); (4) Straw (D15);
(5) Herringbone weave (D17); (6) Woolen cloth (D19);

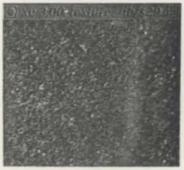

(7)


(8)



(9)

(10)


(11)

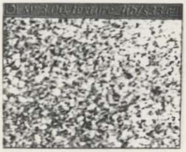
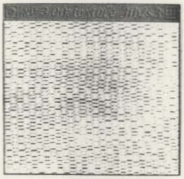
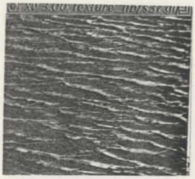

(12)

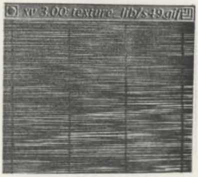
Figure A.1 (Continue):
(7) French canvas (D20);
(8) French canvs (D21);
(9) Herringbone weave (D16);
(10) Beach sand (D28);
(11) Beach sand (D29);
(12) Pressed cork (D33);



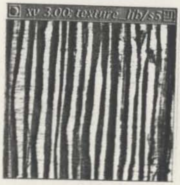
(13)



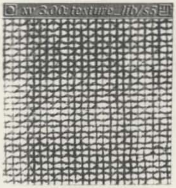
(14)



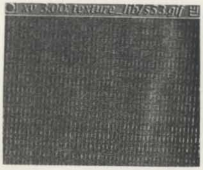
(15)



(16)



(17)

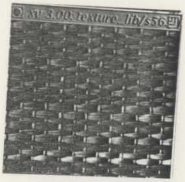


(18)

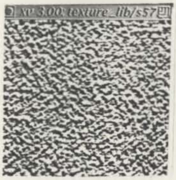
Figure A.1 (Continue):
(13) Netting (D34); (14) Water (D37);
(15) Straw screening (D49); (16) Raffia with threads (D51);
(17) Oriental cloth (D52); (18) Oriental cloth (D53);

xv 3.00: texture_lib/s

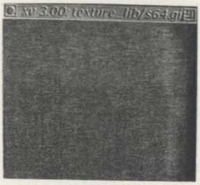
(19)



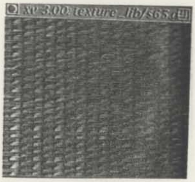
(20)



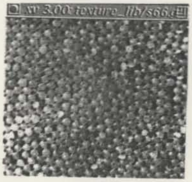
(21)



(22)



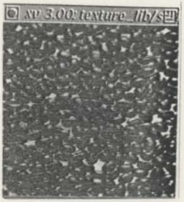
(23)



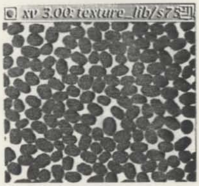
(24)

Figure A.1 (Continue):
(19) Oriental Straw cloth (D80); (20) Straw matting (D56);
(21) Handmade paper (D57); (22) Oriental rattan (D64);
(23) Oriental rattan (D65); (24) Plastic pellets (D66);

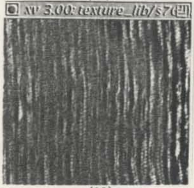
(25)



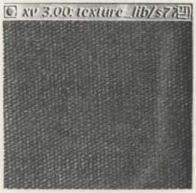
(26)



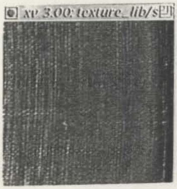
(27)



(28)



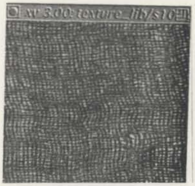
(29)



(30)

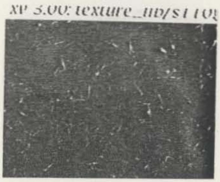
Figure A.1 (Continue):
(25) Wood grain (D68);
(26) Coffee beans (D74);
(27) Coffen beans (D75);
(28) Grass fiber cloth (D76);
(29) Cotton canvas (D77);
(30) Grass fiber cloth (D79);

(31)

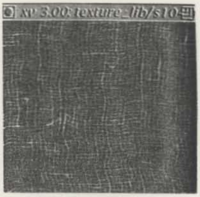


(32)

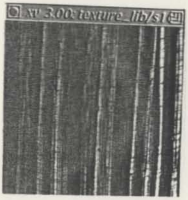
(33)



(34)

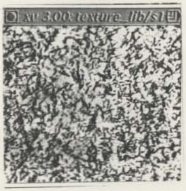


(35)

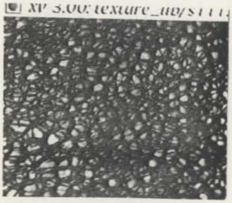


(36)

Figure A.1 (Continue):
(31) Oriental straw cloth (D81); (32) Loose burlap (D103);
(33) Sea fan (D87); (34) Handmade paper (D110)
(35) Loose burlap (D104); (36) Cheesecloth (D105);

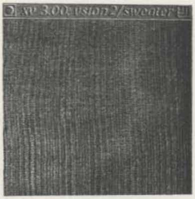


(37)

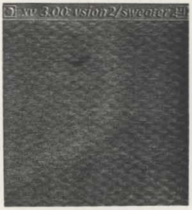


(38)

Figure A.1 (Continue):
(37) Handmade paper (d109); (38) Plastic bubbles (D111).

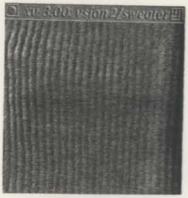


(1)

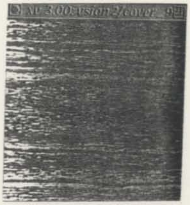


(2)

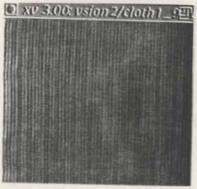
Figure A.2: Textures used in rotation invariant classification experiments.
(1) Sweater (1); (2) Sweater (2);



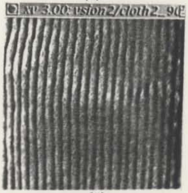
(3)



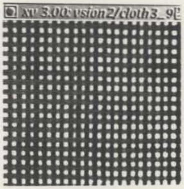
(4)



(5)



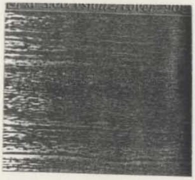
(6)



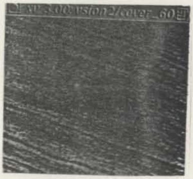
(7)

(8)

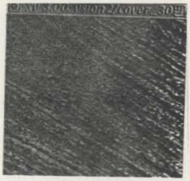
Figure A.2: (Continue)
(3) Sweater(3); (4) Paper Box Cover;
(5) Cloth (1); (6) Cloth (2);
(7) Cloth (3); (8) Cloth (4).



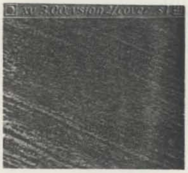
(1)



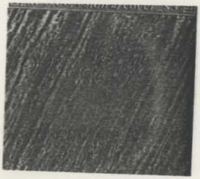
(2)



(3)



(4)



(5)

Figure A.3: Images of texture Paper Box Cover on five orientations.
(1) 90°; (2) 60°;
(3) 30°; (4) (5) two arbitrary angles.

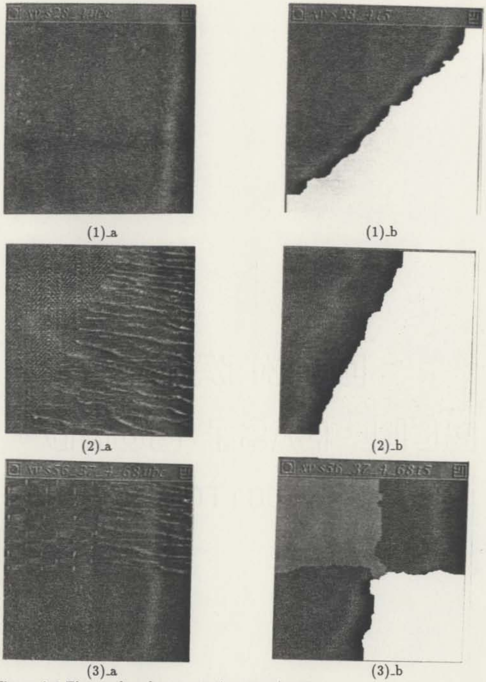
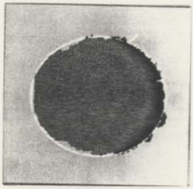
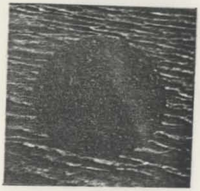


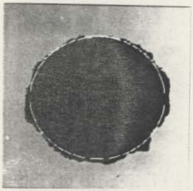
Figure A.4:The results of segmentation experiments.

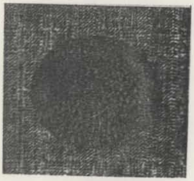
(1)_a image composed of Beach sand (D28) and Pressed cork (D4);
(1)_b segmentation result of (1)_a;
(2)_a image composed of Herringbone weave (D17) and water (D37);
(2)_b segmentation result of (2)_a;
(3)_a image composed of Straw matting (D56), Wood grain (D68), Water (D37) and Pressed cork (D4);
(3)_b segmentation result of (3)_a;

(4)_a


(4)_b


(5)_a


(5)_b


(6)_a

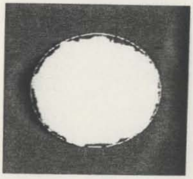
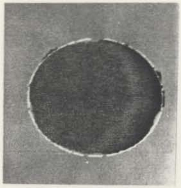

(6)_b

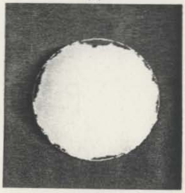
Figure A.4 (Continue)

(4)_a image composed of Pressed calf leather (D24) and Pressed cork (D4);
(4)_b segmentation result of (4)_a;
(5)_a image composed of Water (D37) and Pressed cork (D4);
(5)_b segmentation result of (5)_a;
(6)_a image composed Herringbone weave (D17) and Pressed calf leather (D24);
(6)_b segmentation result of (6)_a;

(7)_a


(7)_b

(8)_a


(8)_b

(9)_a

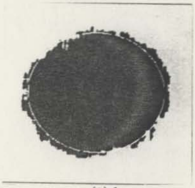
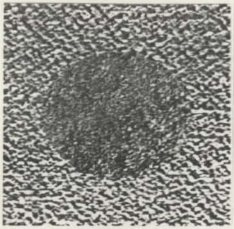
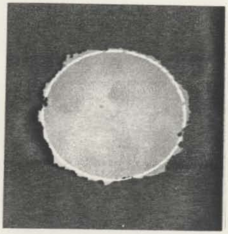

(9)_b

Figure A.4 (Continue)

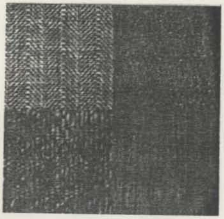
(7)_a image composed of Cotton canvs (D77) and Pressed cork (D4);
(7)_b segmentation result of (7)_a;
(8)_a image composed of Raffia looped to pile (D84) and Calf leather (D24);
(8)_b segmentation result of (8)_a;
(9)_a image composed of Beach sand (D28) and Calf leather (D24);
(9)_b segmentation result of (9)_a;



(10)_a



(10)_b

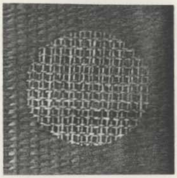


(11)_a

(11)_b

Figure A.4 (Continue)

(10)_a image composed of Handmade paper (D57) and Beach sand (D29);
(10)_b segmentation result of (10)_a;
(11)_a image composed of Herringbone weave (D17), Calf leath (D24), Oriental straw cloth (D81), and Herringbone weave (D16);
(11)_b segmentation result of (11)_a;


(12)_a

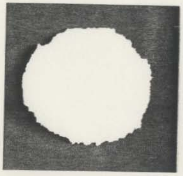
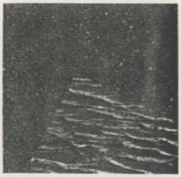

(12)_b

Figure A.4 (Continue)
(12)_a image of Oriental rattan (D65) and French canvas (D20);
(12)_b segmentation result of (12)_a.


(1)_a

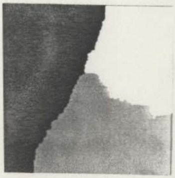
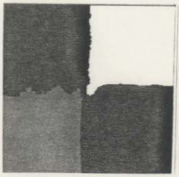

(1)_b

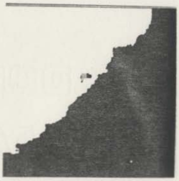
Figure A.5: Segmentation results when using type determining mechanism;
(1)_a image composed of Pressed cork (D4), Beach sand (D28) and Water (D37);
(1)_b segmentation result of (1)_a;

(2)_a

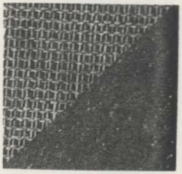


(2)_b

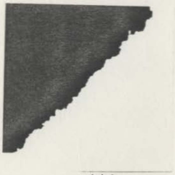
(3)_a



(3)_b



(4)_a



(4)_b

Figure A.5 (Continue)
(2)_a image composed of Herringbone weave (D17), Calf leather (D24), Oriental straw cloth (D81), and Herringbone weave (D16);
(2)_b segmentation result of (2)_a;
(3)_a image composed of Beach sand (D28) and Pressed cork (D4);
(3)_b segmentation result of (3)_a;
(4)_a image composed of French canvas (D20) and Beach sand (D28);
(4)_b segmentation result of (4)_a.