TEXTURE CLASSIFICATION AND RETRIEVAL USING RANDOM NEURAL NETWORK MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALPER TEKE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

DECEMBER 2003

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Volkan Atalay
Supervisor

Examining Committee Members

Prof. Dr. Uğur Halıcı

Assoc. Prof. Dr. Volkan Atalay

Assoc. Prof. Dr. Ali Doğru

Assoc. Prof. Dr. Sibel Tarı

Assoc. Prof. Dr. Göktürk Üçoluk

ABSTRACT

TEXTURE CLASSIFICATION AND RETRIEVAL USING RANDOM NEURAL NETWORK MODEL

Teke, Alper
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Volkan Atalay

December 2003, 45 pages

Texture is one of the most important characteristics used in computer vision and image processing applications. In this thesis, a new texture classification and retrieval method is proposed for texture analysis applications. The technique makes use of the random neural network model and it is supervised. The main aim is to represent textures with parameters which are the random neural network weights and classify and retrieve textures using this texture definition. The network has neurons that correspond to each image pixel, and the neurons are connected according to neighboring relationship between pixels. The method is tested on artificial images produced by using Brodatz album and texture blocks cut from remotely sensed imagery.

Keywords: Random Neural Network, Texture Classification, Texture Retrieval, Remote Sensing

ÖZ

RASTGELE SİNİR AĞLARI MODELİ KULLANARAK DOKU SINIFLAMA VE ERİŞİMİ

Teke, Alper
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Volkan Atalay

Aralık 2003, 45 sayfa

Doku, bilgisayar görüntüsü ve görüntü işleme uygulamalarında kullanılan en önemli karakteristiklerden biridir. Bu çalışmada, doku çözümlemesi uygulamaları için yeni bir doku sınıflama ve erişimi yöntemi ileri sürülmektedir. Bu yöntem, rastgele sinir ağları modelini kullanır. Ana hedef, dokuları, rastgele sinir ağının ağırlıkları olan parametreler olarak ifade etmek ve bu doku tanımını kullanarak doku sınıflama ve erişimidir. Ağ, görüntü elemanlarına karşılık gelen nöronlardan oluşur ve bu nöronlar görüntü elemanlarının komşuluk ilişkilerine göre birbirlerine bağlanır. Bu yöntem Brodatz albümü kullanılarak üretilen yapay doku görüntülerinde ve uzaktan algılama görüntülerinde denenmiştir.

Anahtar Kelimeler: Rastgele Sinir Ağları, Doku Sınıflama, Doku Erişimi, Uzaktan Algılama

ACKNOWLEDGMENTS

I am very grateful to my advisor Dr. Volkan Atalay for his ideas, invaluable contributions, and support during my studies. I would also like to thank to M. Ali Öziç and M. Akif Üçel for their encouragement and support. Thanks are also to Turan Yüksel for helping me on LaTeXand Mustafa Yaman for giving ideas about his previous work on the same topic. My best friends (you know who you are) also get equivalent credits for their motivational support. Inevitably, I would like to thank to my parents Ayşen and Ali Kemal, my sister Melike and my huge family. Without them, none of the parts of this work would be possible. I would like to dedicate this work to them.

TABLE OF CONTENTS

ABSTRACT . . . iii

ÖZ . . . iv

ACKNOWLEDGMENTS . . . v

TABLE OF CONTENTS . . . vi

LIST OF TABLES . . . viii

LIST OF FIGURES . . . ix

LIST OF ABBREVIATIONS . . . x

CHAPTER

1 INTRODUCTION . . . 1

- 1.1 Motivation . . . 2
- 1.2 Purpose . . . 3
- 1.3 Scope . . . 4
- 1.4 Organization . . . 5

2 BACKGROUND ON RANDOM NEURAL NETWORK MODEL . . . 6

- 2.1 Random Neural Network . . . 6
- 2.2 Learning in the Recurrent Random Neural Network . . . 10
- 2.3 Survey of previous RNN Applications . . . 12
  - 2.3.1 Texture Generation . . . 13
  - 2.3.2 Associative memory . . . 13
  - 2.3.3 Optimization . . . 13
  - 2.3.4 Magnetic resonance imaging (MRI) . . . 13
  - 2.3.5 Function approximation . . . 14
  - 2.3.6 Mine detection . . . 14
- 2.4 Texture Generation using RNN . . . 15

3 TEXTURE CLASSIFICATION AND RETRIEVAL 18
3.1 Introduction 18
3.2 RNN for Texture Classification and Retrieval 19
3.3 Texture Classification Algorithm 22
3.4 Texture Retrieval Algorithm 23
4 EXPERIMENTS AND RESULTS 25
4.1 Implementation and Test Platform 25
4.2 Test Images 25
4.3 Experimental Results 27
4.3.1 Texture Classification Experiments 27
4.3.2 Texture Retrieval Experiments 28
5 CONCLUSION AND FUTURE WORK 33
REFERENCES 36
APPENDIX
A TEST IMAGES 40

LIST OF TABLES

TABLE

3.1 Algorithm CLASSIFY-TEXTURE 22
3.2 Algorithm RETRIEVE-TEXTURE 23

4.1 Test Images used in Texture Classification 26
4.2 Test Images used in Texture Retrieval 26
4.3 Texture Classification Errors for Different Window Sizes n 27
4.4 Texture Classification Errors for Different Window Sizes n 28

LIST OF FIGURES

FIGURES

2.1 Representation of a neuron in the RNN. 7
2.2 RNN for Texture Generation. 15
2.3 Generated Textures. 16

4.1 Texture Classification Error. 29
4.2 Texture Retrieval Error. 29
4.3 Test image A.4 and classification result for 9×9 window. 30
4.4 Test image A.6 and classification result for 9×9 window. 31
4.5 Part of the test image A.9, classification result for 9×9 window and correct classification. 32

A.1 512×256 test image of two-texture classes (D1, D68) from Brodatz album. 40
A.2 512×256 test image of two-texture classes (D1, D4) from Brodatz album. 40
A.3 512×256 test image of two-texture classes (D15, D68) from Brodatz album. 41
A.4 512×256 test image of two-texture classes (D6, D112) from Brodatz album. 41
A.5 256×256 test image of 4 texture classes (D10, D68, D1, D49) from Brodatz album. 42
A.6 256×256 test image of 4 texture classes (D16, D58, D93, D25) from Brodatz album. 42
A.7 256×256 test image of 5 texture classes (D105, D91, D38, D6, D112) from Brodatz album. 43
A.8 256×256 test image of 5 texture classes (D3, D59, D61, D77, D93) from Brodatz album. 43
A.9 Single band, grayscale Image (IRS 1C Pan) of size (1600x2400) with spatial resolution 5m 44
A.10 Single band, grayscale Image (SPOT Pan) of size (4000x2600) with spatial resolution 10m 45

LIST OF ABBREVIATIONS

2-D	Two-dimensional
3-D	Three-dimensional
EMI	Electromagnetic interference
GNN	Gelenbe Random Neural Network
IRS	Indian Remote Sensing Satellite
n	Window Size
N	Number of neurons
Pan	Panchromatic
RNN	Random Neural Network
RNNSIM	Random Neural Network Simulator
SPOT	Satellite Pour l'Observation de la Terre
TSP	Traveling Salesman Problem
MR	Magnetic Resonance
MRF	Markov Random Fields
MRI	Magnetic Resonance Imaging
NP	Non-Polynomial
W	Weight Matrix

CHAPTER 1

INTRODUCTION

Texture can be defined as the variations of intensities forming certain patterns which are the result of physical surface properties such as surface roughness or reflectance differences [1]. In computer vision and image processing applications, “texture” is one of the most important characteristics for identifying, discriminating and synthesizing objects or regions in an image. Haralick [2] defines texture as one of the three fundamental pattern elements for interpreting images, whereas the other elements are the spectral and contextual features. According to Haralick, textural features contain information about the spatial distribution of the tonal variations creating patterns on the surface. Analysis of images by using textural properties has a very wide range of application areas such as medical imaging, remote sensing, industrial quality inspection and content based image retrieval. Texture classification and retrieval are the major topics in the texture analysis. For example, the textural properties of the images in medicine are characterized and used for efficient diagnosis of several type of defects in the organ tissues related to serious problems such as cancer [3, 4], while in remote sensing, the same type of measures are used for the recognition of earth objects and the classification of terrain types [2, 5, 6]. Another application area is the automated visual quality inspection in the production line of industrial goods where the texture on the surface characterizes the quality of the products such as the ones having painted metallic or lumber surfaces [7, 8]. Besides, textural features provide valuable tools for segmenting the images by its content with an application for indexing images in a large image database to query the images by specified types such as hair, grass, sky, cloth etc. [9, 10, 11].

In order to analyze images by texture, basically four approaches have been proposed [1]: statistical, geometric, model-based and signal-processing. Statistical approaches use several statistical measures to define textural properties of an image. Haralick [2] proposes the co-occurrence matrix by computing the relative frequencies of gray-level pixels at relative displacements. 14 features are suggested including correlation, entropy, contrast and angular second moment etc. Geometric approaches have limited practical capability. Since textures confronted in real life are neither totally deterministic nor stochastic, the tight assumptions on the image are the disadvantage of these methods. Jain and Tuceryan [12] use voronoi tessellation features for texture segmentation. In addition, Petland introduces a fractal-based texture analysis system using the fractal dimension as the texture description [13]. Signal-processing approaches use spatial and spatial frequency filters, filter banks and frequency transforms such as fourier transform, wavelet transforms and wavelet models [1, 14, 15]. Lastly, model based methods make an assumption of an image model to describe texture. Parameters of a model define the perceptual properties of the visual texture. Especially, random field models such as Gibbs and Markov random fields have been extensively used to model images [16, 17, 18, 19].

1.1 Motivation

Remote sensing is one of the major areas where texture plays an important role. In applications of remote sensing such as environmental monitoring of land resources, analyzing and planning of agricultural and/or urban areas, detecting temporal changes etc. remotely sensed imagery is very widely used for extracting useful information. Increasing availability of the images, increasing quality of spectral and spatial properties and sophisticated software systems make remote sensing images more attractive to spatial analysts. Especially, classification of pixels satisfying a homogeneity property in terms of a metric are widely used in the remote sensing software systems. Such functions generally make use of spectral properties of the images and classify the image into several spectral classes with or without supervision of the user. In addition to these procedures, non-spectral features such as texture may also be used to increase the efficiency of the segmentation. Especially in urban areas, textural information plays a more important role with road network and block building structure.

Haralick has described one of the earliest applications of the use of texture in the analysis of remote sensing images [2]. Since then, Jensen and Toll [20] continue testing and improvement of this co-occurrence matrix based texture feature extracting scheme for remote sensing images. However, the results generally show small improvement in classification accuracy especially until Shih and Schowengerdt [21] use a high frequency filter measure of texture for classifying geomorphological surfaces in a region. Meanwhile, different approaches are proposed by Nguyen and Quinqueton [22] using the measure of irregularity as the texture measure and Blom and Daily [23] using the variances of square windows of various sizes as measures of texture in analysis of rock types in radar images.

After the random neural network model (RNN) have introduced by Gelenbe [24, 25], it is used for solving several different problems [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. In their studies, Atalay, Gelenbe and Yalabik [31, 32] propose a texture generation algorithm using RNN model, and they obtain good results and concluded that RNN can be used to represent and analyze texture. Furthermore, there are many areas in remote sensing which textural information extracted from aerial imagery play an important role and lots of research are done recently on this topic. Therefore, these facts become supplemental sources of motivation for us to design and verify a texture classification and retrieval algorithm using random neural network model.

1.2 Purpose

The purpose of this thesis is to develop and examine a texture classication and retrieval method by random neural network model and verify the performance of the method in classifying images with multi-textures and retrieving specified texture within remote sensing images.

The random neural network model proposed by Gelenbe [24, 25] is a connectionist model, in which signals in the form of impulses which have unit amplitude travel among the neurons. Positive signals represent excitation, whereas negative signals represent inhibition to the receiving neuron. RNN used for texture representation consists of N neurons when $n \times n$ window is used and a neuron in the network corresponds to a pixel in the image. Every neuron is recurrently connected to its 8-neighbors. RNN is used as an autoassociative memory.

The texture classification algorithm proposed is a supervised algorithm. Input image to this algorithm is assumed to be composed of different texture classes. The aim in this algorithm is to label each pixel with one of these classes. For each texture class, we construct a recurrent RNN and train each RNN with sample images of the corresponding texture class. Once RNNs are trained, a pixel of the test image along with its neighbors is fed to every trained RNN and the label for the current pixel is determined by the RNN having the minimum error. One of the important parameters is to determine the number of neurons in the RNN. When the number of neurons are increased, texture can be represented more accurately, whereas the training time increased.

Our texture retrieval algorithm very much resembles the classification algorithm. In this case, we have only one (texture) class so that one recurrent network as autoassociative memory is sufficient. This network is trained with an input texture image which is to be retrieved. After training, a pixel of the test image along with its neighbors is fed to every trained RNN and if the error of the network is less than a certain threshold, the texture of this pixel is the same with input texture image, i.e. the pixel is retrieved.

1.3 Scope

The scope of the thesis is limited to the retrievel of predefined textures such as urban areas form single-band grayscale remote sensing images. IRS Pan images [37] of 5m resolution and SPOT images [38] of 10m resolution from several cities of Turkey are used for the segmentation. In addition, in order to accurately compute and compare the performance of the method, we use texture mosaics created from popular texture source; the Brodatz texture album [39] which are used for texture classification experiments. The retrieval performance of the satellite images is computed by comparing the results of data supplied by Yön Ltd., a professional remote sensing company producing land use clutters for cities of Turkey mostly for micro-cell planning purposes. The data is produced by both visual interpretation and an accompanying field work providing us enough accuracy for the aimed qualitative and quantitative performance analysis.

1.4 Organization

The organization of the thesis is as follows. In Chapter 2, the random neural network model is reviewed. The learning in RNN and survey of previous RNN applications are also presented. In Chapter 3, the texture classification and retrieval algorithms are explained. The experiments and results are given in Chapter 4. Finally, Chapter 5 concludes the thesis.

CHAPTER 2

BACKGROUND ON RANDOM NEURAL NETWORK MODEL

In this chapter, first, background information on random neural network is given. Then a learning algorithm for RNN is presented. Finally, some of the RNN applications are explained.

2.1 Random Neural Network

In the random neural network model by Gelenbe [24, 25] signals in the form of impulses which have unit amplitude travel among the neurons. Positive signals represent excitation, whereas negative signals represent inhibition to the receiving neuron. Thus, an excitatory impulse is interpreted as a "+1" signal, while an inhibitory impulse is interpreted as a "-1". Each neuron i has a state $k_i(t)$ which is its potential at time t represented by a non-negative integer.

When the potential of neuron i is positive, it is referred to as being 'excited' and it can transmit impulses (fire). The impulses will be sent out at a Poisson rate r_i with independent, identical exponentially distributed inter-impulse intervals. The impulses transmitted will arrive at neuron j as excitation signals with probability p^+_{ij} and as inhibitory signals with probability p^-_{ij}. A neuron's transmitted impulse may also leave the network with probability d_i, therefore, $d_i + \sum_{j=1}^{n}[p^+_{ij} + p^-_{ij}] = 1$. To make these probabilities easier to work with, let $w^+_{ij} = r_i p^+_{ij}$ and $w^-_{ij} = r_i p^-_{ij}$; then firing rate of neuron i, r_i, is $\sum_{j=1}^{n}[w^+_{ij} + w^-_{ij}]$. The w matrices can be viewed as being analogous

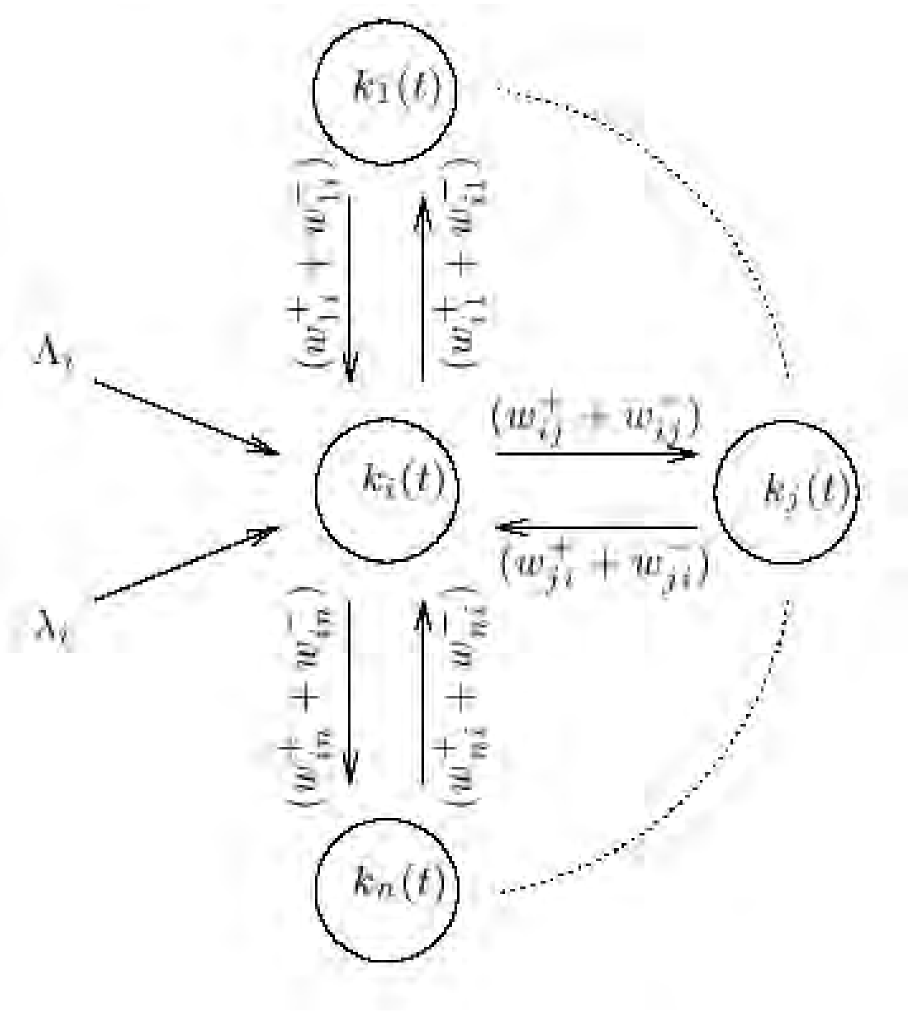


Figure 2.1: Representation of a neuron in the RNN.

to the synaptic weights in connectionist models. though they specifically represent rates of excitatory and inhibitory impulse emission. Since the w matrices are formed through a product of rates and probabilities, they are guaranteed to be non-negative.

Exogenous excitatory and inhibitory signals, meaning those arriving to the neuron from a source outside of the network, also arrive to neuron i at rates Λ_i and λ_i, respectively. These are analogous to the input received by the input neurons in a connectionist model; again however, they represent rates.

Figure 2.1 shows the representation of a neuron in the RNN using the model parameters that have been defined above. In this figure only the transitions to and from a single neuron i are considered in a recurrent fashion. All the other neurons can be interpreted as the replicates of neuron i.

At this point, it is necessary to consider the dynamics of the random neural network model by analyzing the possible state transitions. Within a time interval of Δt, several transitions can occur which change a neuron's state $k_i(t)$:

- The potential $k_i(t)$ of a neuron will decrease by one whenever it fires regardless of the type of the signal emitted (excitation or inhibition). Also, when an exogenous inhibitory signal arrives from outside the network to neuron i, its potential drops to $k_i(t)-1$ at time $t+\Delta t$. Moreover, neuron i might receive an inhibitory impulse from another neuron j whose effect will again be to decrement the value of k_i at time t by one.

- Arrival of an exogenous excitatory signal from outside, or an excitatory impulse from another neuron within the network will result in incrementing the neuron potential by one, yielding $k_i(t) + 1$

- Needless to say, the value of ith neuron's state remains unchanged when none of the events described above occur.

In the case when *self-inhibition* is allowed, the value of the neuron's state can drop by two units in a single time step, however this case will not be considered in the following expressions.

Also in this model, *self-excitation* is not of interest because in its presence, the potential of the neuron may increase without bound which would lead to instability. There are also some boundary conditions which prevent some of the transitions from occurring. First of all, a neuron can fire only when it has a positive potential as explained above. Second, when the neuron has a potential of zero, the arrival of new inhibitory signals does not decrease its value further. All of these constraints will be unified in a single expression when the state transitions are expressed in mathematical form.

Let $\mathbf{k}(t) = k_1(t), \ldots, k_n(t)$ be the vector of signal potentials at time t and $\mathbf{k} = k_1, \ldots, k_n$ be a particular value of the vector, and lets define the probability $p(\mathbf{k}, t) = Pr[\mathbf{k}(t) = \mathbf{k}]$. The behavior of the probability distribution of the network state can be derived through the following equations. Since $\mathbf{k}(t) : t \geq 0$ is a continuous time Markov chain, it satisfies an infinite system of *Chapman-Kolmogorov* equations.

$$
\begin{aligned}
&p(\mathbf{k}, t+\Delta t) = \\
&\qquad \textstyle\sum_i [p(k_i^+, t) r(i) d(i) \Delta t + p(k_i^-, t) \Lambda t \mathbf{1}[k_i(t) > 0] \\
&\qquad + p(k_i^+, t) \lambda(i) \Delta t + p(k_i, t)(1 - \Lambda(i)\Delta t) \\
&\qquad \times (1 - \lambda(i)\Delta t)(1 - r(i)\Delta t) \mathbf{1}[k_i(t) > 0] \\
&\qquad + \textstyle\sum \{p(k_{ij}^{+-}, t) r(i) p^+(i,j) \Delta t \mathbf{1}[k_j(t) > 0] \\
&\qquad + p(k_{ij}^{++}, t) r(i) p^-(i,j) \Delta t + p(k_i^+, t) r(i) p^-(i,j) \Delta t \mathbf{1}[k_j(t) = 0]\}] \\
&\qquad + o(\Delta t)
\end{aligned}
$$

where

$$
\mathbf{1}[x] = \begin{cases} 1 & if\ x\ is\ true \\ 0 & otherwise \end{cases}
$$

For steady state analysis, let $p(\mathbf{k})$ denote the stationary probability distribution which is equal to $\lim_{t\to\infty} Pr[k_i(t) > 0]$ if it exists. Thus, in steady state, stationary probability distribution, $p(\mathbf{k})$, must satisfy the global balance equations:

$$
\begin{aligned}
&p(\mathbf{k}) \textstyle\sum_i [\Lambda(i) + [\lambda(i) + r(i)] \mathbf{1}[k_i > 0]] = \\
&\qquad \textstyle\sum_i [p(k_i^+) r(i) d(i) + p(k_i^-) \Lambda(i) \mathbf{1}[k_i > 0] \\
&\qquad + p(k_i^+) \lambda(i) + \textstyle\sum_j \{p(k_{ij}^{+-}) r(i) p^+(i,j) \mathbf{1}[k_j > 0] \\
&\qquad + p(k_{ij}^{++}) r(i) p^-(i,j) + p(k_i^+) r(i) p^-(i,j) \mathbf{1}[k_j = 0]\}]
\end{aligned}
$$

The stationary probability distribution associated with the model is the value which will be taken to be the output of the network, and is given by:

$$
q_i = \lim_{t\to\infty} Pr[k_i(t) > 0],\ i = 1, \ldots, n \tag{2.1}
$$

which reduces the form

$$
q_i = \lambda^+(i) / [r(i) + \lambda^-(i)] \tag{2.2}
$$

where the $\lambda^+(i)$, $\lambda^-(i)$ for $i = 1, \ldots, n$ satisfy the system of nonlinear simultaneous equations

$$
\lambda^+(i) = \sum_j q_j w_{ji}^+ + \Lambda(i),\ \lambda^-(i) = \sum_j q_j w_{ji}^- + \lambda(i) \tag{2.3}
$$

To put Equation 2.2 into words, the steady state probability that the neuron i is excited is simply equal to the ratio of the sum of all the rates of arriving excitatory signals to the sum of the rates of arriving inhibitory signals together with the firing rate of that particular neuron.

2.2 Learning in the Recurrent Random Neural Network

The algorithm chooses the set of network parameters $\mathbf{W}$ in order to learn the training set of K input-output pairs $(\boldsymbol{\iota}, \mathbf{Y})$ where the set of successive inputs is denoted $\boldsymbol{\iota} = \{\iota_1, \ldots, \iota_K\}$, and $\iota_k = (\Lambda_k, \lambda_k)$ are pairs of excitation and inhibition signal flow rates entering each neuron from outside of the network:

$$\boldsymbol{\Lambda}_k = [\Lambda_k(1), \ldots, \Lambda_k(n)],\ \lambda_k = [\lambda_k(1), \ldots, \lambda_k(n)]$$

The successive desired outputs are the vectors $\mathbf{Y} = \{y_1, \ldots, y_K\}$, where each vector $y_k = (y_{1k}, \ldots, y_{nk})$, whose elements $y_{ik} \in [0, 1]$ correspond to the desired output values for each neuron. The network adjusts its parameters to produce the set of desired output vectors in a manner that minimizes a cost function E_k:

$$E_k = \frac{1}{2}\sum_{i=1}^{n} a_i(q_i - y_{ik})^2,\ a_i \geq 0$$

In this network, *all* neurons are generalized to be output neurons, therefore, if it is desired that a neuron j is to be removed from the network output, and therefore the error function, it suffices to set $a_j = 0$

Recall that the steady state output rate of all neurons in the network is given by Equations 2.2 and 2.3.

Both of the n by n weight matrices $\mathbf{W}_k^+ = \{w_k^+(i, j)\}$ and $\mathbf{W}_k^- = \{w_k^-(i, j)\}$ must be adjusted after each input is presented, by computing for each input $\iota_k = (\Lambda_k, \lambda_k)$, a new value $\mathbf{W}_k^+$ and $\mathbf{W}_k^-$ of the weight matrices. Since the weight matrices represent a rate times a probability, only solutions for which all values in the matrices are positive are valid.

Let $w(u, v)$ denote any weight term, which would be either $w(u, v) \equiv w-(u, v)$, or $w(u, v) \equiv w+(u, v)$. The weights will be updated using gradient descent method:

$$w^{new}(u, v) = w^{old}(u, v) - \eta \partial E / \partial w(u, v)$$

The partial derivative of the cost function can be computed and substituted to obtain the update difference equation:

$$w_k(u, v) = w_{k-1}(u, v) - \eta \sum_{i=1}^{n} a_i(q_{ik} - y_{ik})[\partial q_i / \partial w(u, v)]_k \tag{2.4}$$

where $\eta > 0$ is the learning parameter which is constant over each iteration of training, and

1. q_{ik} is calculated using the input ι_k and $w(u,v) = w_{k-1}(u,v)$, in Equations 2.2 and 2.3.

2. $[\partial q_i/\partial w(u,v)]_k$ is evaluated at the values $q_i = q_{ik}$, $w^+(u,v) = w^+_{k-1}(u,v)$ and $w^-(u,v) = w^-_{k-1}(u,v)$.

To compute $[\partial q_i/\partial w(u,v)]_k$ the following equation is derived from expressions 2.2 and 2.3:

$$\begin{aligned}\partial q_i/\partial w(u,v) = {} & \textstyle\sum_j \partial q_j/\partial w(u,v)[w^+(j,i) - w^-(j,i)]/(r(i) + \lambda^-(i)) \\ & -\mathbf{1}[u = i]q_i/(r(i) + \lambda^-(i)) \\ & +\mathbf{1}[w(u,v) \equiv w^+(u,i)]q_u/(r(i)/ + \lambda^-(i)) \\ & -\mathbf{1}[w(u,v) \equiv w^-(u,i)]q_u q_i/(r(i) + \lambda^-(i))\end{aligned}$$

Let $\mathbf{q} = (q_1, \ldots, q_n)$, and define the $n \times n$ matrix

$$\mathbf{W} = \{[w^+(i,j) - w^-(i,j)q_j]/(r(j) + \lambda^-(j))\} \quad i,j = 1, \ldots, n$$

The vector equations can now be written as:

$$\partial \mathbf{q}/\partial w^+(u,v) = \partial w^+(u,v)\mathbf{W} + \gamma^+(u,v)q_u$$

$$\partial \mathbf{q}/\partial w^-(u,v) = \partial w^-(u,v)\mathbf{W} + \gamma^-(u,v)q_u$$

where the elements of the n-vectors $\boldsymbol{\gamma}^+(u,v) = [\gamma^+_1(u,v), \ldots, \gamma^+_n(u,v)]$ and $\boldsymbol{\gamma}^-(u,v) = [\gamma^-_1(u,v), \ldots, \gamma^-_n(u,v)]$ are

$$\gamma^+_i(u,v) = \begin{cases} -1/(r(i) + \lambda^-(i)) & if\ u = i, v \neq i \\ +1/(r(i) + \lambda^-(i)) & if\ u \neq i, v = i \\ 0 & for\ all\ other\ values\ of\ (u,v) \end{cases}$$

$$\gamma^-_i(u,v) = \begin{cases} -(1 + q_i)/(r(i) + \lambda^-(i)) & if\ u = i, v = i \\ -1/(r(i) + \lambda^-(i)) & if\ u = i, v \neq i \\ -q_i/(r(i) + \lambda^-(i)) & if\ u \neq i, v = i \\ 0 & for\ all\ other\ values\ of\ (u,v) \end{cases}$$

Notice that

$$\begin{aligned}\partial \mathbf{q}/\partial w^+(u,v) &= \gamma^+(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \\ \partial \mathbf{q}/\partial w^-(u,v) &= \gamma^-(u,v)q_u[\mathbf{I} - \mathbf{W}]^{-1} \qquad (2.5)\end{aligned}$$

where $\mathbf{I}$ denotes the n by n identity matrix. Hence the main computational effort in this algorithm is to obtain $[\mathbf{I} - \mathbf{W}]^{-1}$. This is of time complexity $O(n^3)$, or $O(mn^2)$ if an m-step relaxation method is used.

From the above, the complete learning algorithm for the network can be given. First initialize the matrices $\mathbf{W}_0^+$ and $\mathbf{W}_0^-$ in some appropriate manner. This initiation can be made at random if no better method can be determined. Choose a value of η, and then for each successive value of k starting with $k = 1$ proceed as follows:

1. Set the input values to $\iota_k = (\Lambda_k, \lambda_k)$.

2. Solve the system of nonlinear equations given in 2.2 and 2.3 with these values, perhaps by using an iterative method such as Gauss-Seidel.

3. Solve the system of linear equations (2.5) with the results of (2).

4. Using Equation 2.4 and the results of (2) and (3), update the matrices $\mathbf{W}_k^+$ and $\mathbf{W}_k^-$. Since the "best" matrices (in terms of gradient descent of the quadratic cost function) which satisfy the *nonnegativity* constraint are sought in any step k of the algorithm, if the iteration yields a negative value of a term, there are two alternatives:

   (a) Set the term to zero, and stop the iteration for this term in this step k; in the next step, $k + 1$ iterate on this term with the same rule starting from its current zero value;

   (b) Go back to the previous value of the term and iterate with a smaller value of η.

This general scheme can be specialized to feed-forward networks by noting that the matrix $[\mathbf{I}-\mathbf{W}]$ will be triangular, yielding a computational complexity of $O(n^2)$, rather than $O(n^3)$, for each gradient iteration. Furthermore, in a feed-forward network, the equations given in 2.2 and 2.3 are simplified in that q_i is only dependent upon q_j for $j < i$. This reduces the computational effort required to solve 2.2 and 2.3.

2.3 Survey of previous RNN Applications

The RNN model has been proven to be successful in a variety of applications when used either in a feed-forward or a fully recurrent architecture. In most problems, RNN yields strong generalization capabilities, even when the training data set is relatively small compared to the actual testing data. The model also achieves fast learning due to its computational simplicity for weight updating process. In the following, the past work related to RNN will be described briefly.

2.3.1 Texture Generation

Generation of artificial textures is a useful function in image synthesis systems. Authors in [31, 32] describe the use of the RNN model to generate various textures having different characteristics. Since texture generation is closely related to our proposed work, it is presented in Section 2.4 in details.

2.3.2 Associative memory

In [26], the network's ability of acting as autoassociative memory is examined and a technique for reconstructing distorted patterns is developed, which is based on properties of the network. The performance of the resulting approach has been investigated through experiments, which yielded promising results. Also, in [27], the author shows how distributed associative memory can be used to compute membership functions for decision-making under uncertainty.

2.3.3 Optimization

The traveling salesman problem (TSP) is commonly considered as a benchmark case for heuristic methods among hard combinatorial optimization problems. It is shown in [28] that the dynamical RNN yields solutions to the TSP which are close to the optimal in a majority of instances tested. Yet another application is the vertex covering problem, which is designated as NP-complete. In [29, 30] authors compare the performances of the RNN, the conventional Greedy Algorithm, the Hopfield network, and simulated annealing, when applied to the same problem. Results reveal that the RNN heuristic is superior to the others in terms of overall optimization.

2.3.4 Magnetic resonance imaging (MRI)

Brain MR images contain massive information requiring lengthy and complex interpretation (as in the identification of significant portions of the image), quantitative evaluation (as in the determination of the size of certain significant regions), and sophisticated interpretation (as in determining any image portions which indicate signs of lesions or of disease). In [33], RNNs are used to extract precise morphometric information from MRI scans of the human brain. A method for classification of gray matter from MR images is proposed, and the classification performance is shown to be very similar to those that are known to be obtained by a human expert carrying out manual volumetric analysis of brain MR images.

2.3.5 Function approximation

In [34], approximation of arbitrary continuous functions on $[0,1]^S$ using the "Gelenbe" random neural network (GNN) is studied. It is shown that the clamped GNN and the bipolar GNN have the universal approximation property, using a constructive method which exhibits networks constructed from a polynomial approximation of the function to be approximated. There are no restrictions on the structure of the networks except for limiting them to being feed-forward. In [35], the design of GNN approximators with a bounded number of layers is discussed. It is shown that the feed-forward CGNN and BGNN with s hidden layers (total of $s+2$ layers) can uniformly approximate continuous functions of s variables.

2.3.6 Mine detection

In [36], authors introduce an RNN approach to mine detection which provides a robust non-parametric method, based on training the network using data from a previously calibrated portion of the minefield, or from a similar minefield. This approach is shown to be very effective for detecting mines and rejecting false alarms. Experimental evidence indicates that the neural network trained for mine detection and false alarm rejection on a small calibration site, can be effective on geographical locations which are distinct and far removed from the locations where training of the network takes place. It is also shown that the RNN trained for a specific EMI instrument can be effective when it produces decisions based on data from a different EMI sensor instrument.

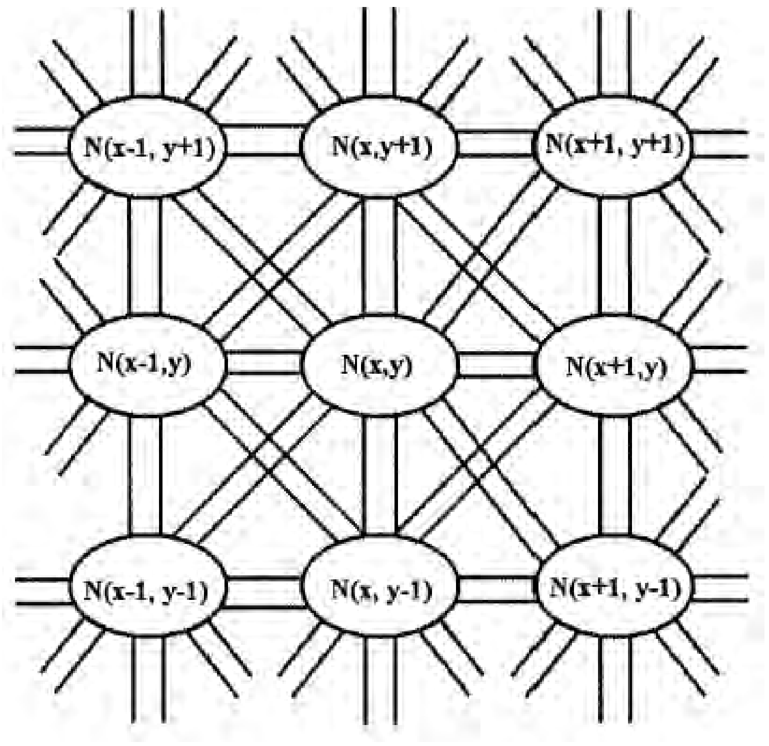


Figure 2.2: RNN for Texture Generation.

2.4 Texture Generation using RNN

The RNN model to generate textures associates a neuron for every pixel in the image. The topology of the network is chosen according to 8-connectedness among the pixels so that each neuron has connection with only its 8 neighbors (see Figure 2.2).

For simplicity, a pixel will be represented by a single index x rather than (i, j). x_0 indicates the right immediate pixel (at position $(i+1, j)$) of the pixel x. Other neighbors will be numbered according to the counter-clockwise order. For generation purposes

$$r(x) = r$$

assumed and all connection in the network are symmetric:

$$w(x, x_d) = w(x_d, x) = w$$

Here the generic term w is used to indicate both w^+ and w^-. But, they will be differentiated by sign of the weight value. From the symmetry assumption

$$w_0 = w_4,\ w_1 = w_5,\ w_2 = w_6,\ w_3 = w_7$$

And in this case

$$r(x) = r = \sum |w_d|$$

We also assumed that external positive signals arrive with the same constant rate to all neurons and there is no external negative signal coming to neurons.

$$\Lambda(x) = c, \ \lambda(x) = 0$$

Under all these assumptions, the Equation 2.2 can be rewritten as

$$q(x) = \frac{\sum_d w_d^+ q(x_d) + c}{\sum_d w_d^- q(x_d) + r} \tag{2.6}$$

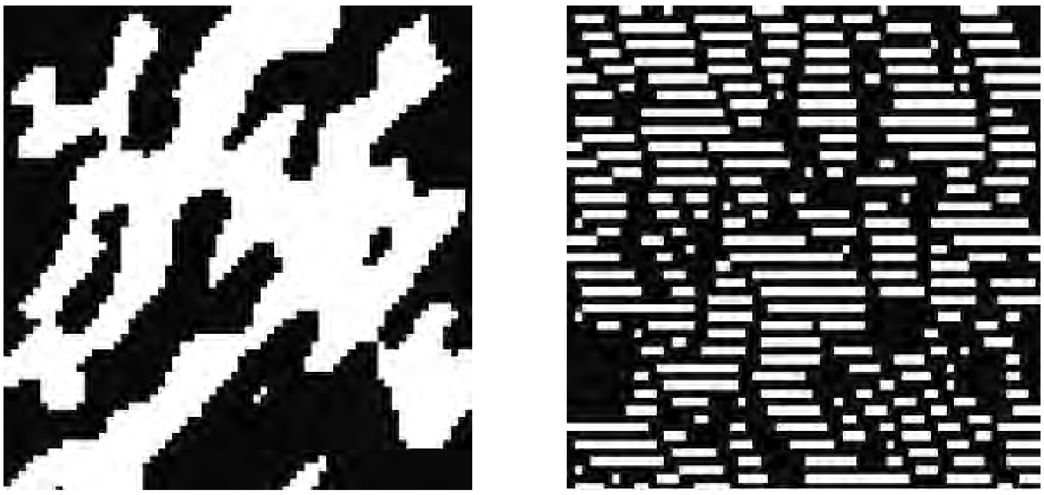


Figure 2.3: Generated Textures.

Then, texture generation algorithm for gray level is as follows. First choose c according to the desirable average gray level and determine all weights. Then

1. Generate at random value of 0 or 1 for each pixel x and assign it to variable $q_0(x)$ for each x.

2. Starting with $i = 0$ up to $i = N$ iterate on Equation 2.6 as:

$$q_{i+1}(x) = \frac{\sum_d w_d^+ q_i(x_d) + c}{\sum_d w_d^- q_i(x_d) + r}$$

3. Compute the average gray level G of the picture:

$$G = \frac{\sum_x q_N(x)}{Size_of_picture}$$

4. Assign the gray level of each pixel as follows:

$$f(x) = 1 \; if \; q_N(x) > G, \; f(x) = 0 \; if \; q_N(x) \leq G$$

The directional weights w_d are the most important parameters for the textures which will be generated. Two textures for different weights can be seen in Figure 2.3. First texture is generated with $w_0 = w_4 = w_2 = w_6 = w_1 = w_5 = 1$, all other weights are equal to 0, and second one with $w_0 = w_4 = -1$, $w_2 = w_6 = w_3 = w_7 = 1$, and all others equal to 0.

Numerical iterations of the field equations of the model, starting with a randomly generated gray-level image, are shown to produce textures having different desirable features such as granularity, inclination, and randomness. The experimental evaluation shows that the RNN provides good results, at a computational cost less than that of other approaches such as Markov random fields (MRF).

CHAPTER 3

TEXTURE CLASSIFICATION AND RETRIEVAL

3.1 Introduction

Texture classification and retrieval plays an important role in many tasks such as remote sensing, medical imaging, robot vision and query by content in large image databases. Various methods for texture feature extraction have been proposed during the last decades, however texture analysis problem still remains difficult and subject to intensive research.

In order to analyze images based on texture, several approaches have been proposed: statistical, geometric, signal processing and model-based. In statistical approaches several statistical measures to define textural properties of an image are used. In [40], analysis by cooccurence matrix is described. A cooccurrence matrix describes two dimensional (joint) probability density functions and it is obtained by computing the frequencies of gray-levels of pixels at particular relative displacements. Haralick suggests 14 features, including correlation, entropy, contrast and angular second moment etc. of the cooccurrence matrix. Geometric approaches are the least used features because of the limited practical capability. Since textures confronted in real life are neither totally deterministic nor stochastic, the tight assumptions on the image are the disadvantage of these methods. In [41], Voronoi tessellation features are used for texture segmentation. Signal processing approaches use spatial and spatial frequency filters and filter banks frequency transforms such as Fourier transform, wavelet transforms and wavelet models [42, 14]. Finally, model based methods make an assumption of an image model to describe texture. Parameters of a model define the perceptual properties of the visual texture. Especially, random field models such as Gibbs and Markov random fields have been extensively used to model images [43].

In this study, a novel texture analysis method for texture classification and retrieval is described. This method uses the random neural network (RNN) model which is proposed by Gelenbe [24].

3.2 RNN for Texture Classification and Retrieval

We describe a new texture classification and retrieval method based on the Random Neural Network model. In this method, for each texture class, a recurrent random neural network is constructed. This network consists of $n \times n$ neurons, and each neuron is connected recurrently to its immediate 8-neighbors. Figure 2.2 shows such a configuration for a 3×3 network. In this method, a neuron in the network corresponds to a pixel in the image. The network is used as an autoassociative memory, i.e. both input and output of the neurons are the same, which are the gray level values of the image for the corresponding pixels [26]. An autoassociative memory is a memory system which associates a particular information to itself and which is able to recognize and correctly recall this information from partial or corrupted version used as input. Each recurrent RNN is then trained with image samples from a single texture class using the learning algorithm presented in Section 2.2, and therefore, the weights of the network are found. The weights indeed correspond to the textural parameters for further analysis.

During training, the choice of initial weights is important, since it influences the convergence of the error minimization procedure. There exist three different kinds of initialization procedure:

- Random Initialization: The connection weight matrices W_0^+ and W_0^- are initialized by small positive random variables, which are uniformly distributed between 0 and V_{max} where V_{max} is a positive number close to 0.

- Hebbian Rule: Hebbian “reinforcement law” [44] is used for initialization. Hebbian rule for synaptic weight initialization is:

$$w_d^{+(0)} = \frac{1}{N}\sum_{x=1}^{N}[q(x)\ NOR\ q(x_d)],\ w_d^{-(0)} = \frac{1}{N}\sum_{x=1}^{N}[q(x)\ XOR\ q(x_d)]$$

where XOR and NOR are exclusive-or operator and its complement respectively.

- Quadratic Optimization: Third initialization procedure is based on the reformulation of state equations of network as a linear system and try to solve it as an quadratic optimization problem [45]. Equation 2.6 can be written in a linear form as follows.

$$[\hat{q}(x) - \hat{q}(x_0)]w_0^+ + \ldots + [\hat{q}(x) - \hat{q}(x_7)]w_7^+ +$$
$$\hat{q}(x)[1 + \hat{q}(x_0)]w_0^- + \ldots + \hat{q}(x)[1 + \hat{q}(x_7)]w_7^- = \hat{c}$$

Here $\hat{c}$ represents desired average gray level value and values of w_d^+ and w_d^- for $d = 0, \ldots, 7$ must be determined (16 unknowns). In a texture, there will be more than 16 pixels (also more than 16 neurons in the network) for which similar equalities are valid. This is an overdetermined system of linear equalities. For a fragment of image whose first pixel is i and the last one is j, we have

$$\boldsymbol{Aw} = \boldsymbol{c}$$

where

$$\boldsymbol{w} = (w_0^+, \ldots, w_7^+, w_0^-, \ldots, w_7^-)$$

$$\boldsymbol{A} = \begin{bmatrix} \hat{q}(i) - \hat{q}(i_0) & \ldots & \hat{q}(i) - \hat{q}(i_7) & \hat{q}(i)(1 + \hat{q}(i_0)) & \ldots & \hat{q}(i)(1 + \hat{q}(i_7)) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{q}(j) - \hat{q}(j_0) & \ldots & \hat{q}(j) - \hat{q}(j_7) & \hat{q}(j)(1 + \hat{q}(j_0)) & \ldots & \hat{q}(j)(1 + \hat{q}(j_7)) \end{bmatrix}$$

$$\boldsymbol{c} = (\hat{c}, \ldots, \hat{c})^T$$

In this case, we'll try to minimize the norm of the error $\boldsymbol{Aw} - \boldsymbol{c}$ by a choice of weights $\boldsymbol{w}$. Then finding the values for directional weights is transformed to an optimization problem with constraints.

$$\text{minimize } \tilde{\tilde{\boldsymbol{C}}} = (\boldsymbol{Aw} - \boldsymbol{c})^T(\boldsymbol{Aw} - \boldsymbol{c}) \text{ with } w_d \geq 0$$

This is a quadratic programming problem with inequality constraints. After simple transformations we get;

$$\tilde{\tilde{\boldsymbol{C}}} = \boldsymbol{w}^T \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{A}^T + \boldsymbol{c}^T \boldsymbol{c}$$

Since $\boldsymbol{c}^T \boldsymbol{c}$ does not have any effect on the minimization, the problem reduces to

$$\text{minimize } \tilde{\tilde{\boldsymbol{C}}} = 2\left(\frac{1}{2}\boldsymbol{w}^T \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{A}^T \boldsymbol{c}\right) \text{ with } w_d \geq 0$$

Let $\boldsymbol{H} = \boldsymbol{A}^T, \boldsymbol{b} = \boldsymbol{A}^T \boldsymbol{c}$, then the cost becomes

$$\tilde{\boldsymbol{C}} = 2\left(\frac{1}{2}\boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{b}^T\right)$$

After changing variables:

$$w_d^+ = b_d^+ - z_d^+,\ w_d^+ \geq 0 \Rightarrow (b_d^+ - z_d^+) \geq 0 \Rightarrow z_d^+ \leq b_d^+$$

and the same for w_d^-,

$$\text{minimize } \boldsymbol{C} = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{H} \boldsymbol{b} - \boldsymbol{z}^T \boldsymbol{H} \boldsymbol{b} + \boldsymbol{z}^T \boldsymbol{b} \text{ with } z_d^+ \leq b_d^+,\ z_d^- \leq b_d^-$$

Using the Langrangian, the solution is given by Kuhn-Tucker optimality conditions: in our case minimum is reached for

$$\frac{\partial \boldsymbol{C}}{\partial z_d} + \boldsymbol{y}^T = 0$$

$$y_d \geq 0,\ z_d \leq b_d,\ y_d(z_d - b_d) = 0 \text{ for } z_d = z_d^+ \text{ or } z_d^-,\ b_d = b_d^+ \text{ or } b_d^-$$

This gives

$$\boldsymbol{H}\boldsymbol{z} + \boldsymbol{y}^T = (\boldsymbol{H} - \boldsymbol{I})\boldsymbol{b}$$

which implies either $y_d = 0$ or $z_d = b_d$. The system will have many solutions. Among them the one that will minimize the cost will be chosen. After that, initial weights are found, and used for training.

We use Hebbian Rule for initialization procedure, because it is the best for error minimization and convergence according to the experiments proven to be [46].

3.3 Texture Classification Algorithm

Our texture classification algorithm is a supervised algorithm. Input image to this algorithm is assumed to be composed of different texture classes. The aim in this algorithm is to label each pixel with one of these classes. The supervised algorithm is made up of two parts: training and labeling. First, for each texture class, we construct a recurrent RNN and train each RNN with sample images of the corresponding texture class. Once RNNs are trained, a pixel of the test image along with its neighbors is fed to every trained RNN and the label for the current pixel is determined by the RNN having the minimum error.

In the training part, first, a recurrent network, consisting of $N = n \times n$ neurons, is constructed for each class. Then, using each $n \times n$ window in the training image, the network is trained using the learning algorithm given in Section 2.2. When training is finished, textural parameters, which are the weights of the network in our case, are found for each class.

Table 3.1: Algorithm CLASSIFY-TEXTURE

```
CLASSIFY-TEXTURE()
for each class $c$ in the input image
do construct a recurrent $RNN_c$
   initialize weights of $RNN_c$ (using Hebbian Rule in 3.2)
   for each $n \times n$ window $train\_data_i(c)$
   do $train\_network(train\_data_i(c))$ (using algorithm in Section 3.2)


for each class $c$ in the input image
do for $j \leftarrow 0$ to $N$
   do $distance_c(j) \leftarrow MAX\_FLOAT$
   for each $n \times n$ window $input\_image_i$ containing data in $image\_data$
   do $output\_image_i \leftarrow rnn\_output_c(input\_image_i)$
      $dist \leftarrow Euclidean\_Distance(input\_image_i, output\_image_i)$
      $distance_c(j) \leftarrow min(dist, distance_c(j))$ where pixel $j \in input\_image_i$


for each pixel $j$ in the input image
do $classification(j) \leftarrow \arg_c \min(distance_c(j))$
```

The classification phase is divided into two main parts. In the first part, the distances of each pixel to the every class are found. In order to do this, each $n \times n$ window in the input image is given to the network as input. As an autoassociative memory, the output of the network shows how close the input window to the texture that is used for training that network. Therefore, the Euclidean Distance between input and output of the network is used as a distance measure. A distance matrix is created with the same size of input image, and the calculated value for the window is assigned the pixels, which are in the input window. If a distance has already been assigned to a pixel, which may be the case when the pixel belongs to another window, the minimum of the distances is assigned. After all the distances are found, class assignment can be done. Every pixel in the image is assigned to closest class, that is the class with the minimum distance. Table 3.1 gives the algorithm for texture classification.

3.4 Texture Retrieval Algorithm

Our texture retrieval algorithm very much resembles the classification algorithm. But in calculating distances, this time non-overlapping windows are used, because experiments are performed using big remote sensing images and execution time is a main issue. In this case also, we have only one (texture) class so that one recurrent network

Table 3.2: Algorithm RETRIEVE-TEXTURE

```
RETRIEVE-TEXTURE()
 1   construct recurrent RNN for texture to be retrieved
 2   initialize weights of the network (using Hebbian Rule in 3.2)
 3  **for** each $n \times n$ window $train\_data_i(c)$ containing data in $train\_data(c)$
 4  **do** $train\_network(train\_data_i(c))$ (using algorithm in Section 3.2)
 5  **for** each non-overlapped $n \times n$ window $input\_image_i$ in $image\_data$
 6  **do** $output\_image_i \leftarrow rnn\_output(input\_image_i)$
 7      $dist \leftarrow Euclidean\_Distance(input\_image_i, output\_image_i)$
 8      $distance(j) \leftarrow dist$ where pixel $j \in input\_image_i$
 9
10  **for** each pixel $i$ in the input image
11  **do if** $distance(i) \leq threshold$
12          **then** $retrieved(i) \leftarrow 1$;
13          **else** $retrieved = 0$;
14
```

as autoassociative memory is sufficient. This network is trained with an input texture image which is to be retrieved.

Again, we can divide the algorithm into two parts; training and retrieval. The training part is exactly the same as in the classification. In the retrieval phase, first, the distance of each pixel to the texture to be retrieved is found. To do this, each non-overlapped $n \times n$ window in the input image is given as input into the constructed network. By using the learned weights and the input image window, we obtain an output image window. Euclidean Distance between input and output image windows is then calculated. A distance matrix is created with the same size of input image, and the calculated value for the window is assigned the pixels in the input window. After all the distances are found, retrieval can be done. Each pixel having a distance minimum than a certain threshold is labeled as belonging the same class with the texture to be retrived. The algorithm for texture retrieval can be seen in Table 3.2.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Implementation and Test Platform

The proposed method is implemented in MATLAB [47]. Random Neural Network Simulator (RNNSIM) for MATLAB [48, 49] is also used as a toolbox for some RNN routines.

4.2 Test Images

There are two types of experiments performed. First type of experiments is the classification of the images by texture using algorithm presented in Section 3.3. In the experiments, test images are taken from popular texture source; the Brodatz texture album [39]. Test images are texture mosaics created by cutting and pasting homogenous texture blocks from these sources. All textures are grayscale images with 8-bit representation per sample, i.e. 256 gray levels. Images are histogram equalized before processed, therefore they can not be discriminated for local gray mean level or local variance. Performances measuring the feature extraction quality of the method for all test images are computed by comparing the classified images with ground truth classifications pixel by pixel. Furthermore, classification error, which is “the most common measure of performance for a recognition system” [50] is used as the performance criterion.

In Table 4.1, the images used in texture classification listed with properties such as size, the texture source and the existing number of texture classes in the images. The

Table 4.1: Test Images used in Texture Classification

Name (Fig. No.)	Type	Width $\times$ Height	Number of classes
A.1	Brodatz	512×256	2
A.2	Brodatz	512×256	2
A.3	Brodatz	512×256	2
A.4	Brodatz	512×256	2
A.5	Brodatz	256×256	4
A.6	Brodatz	256×256	4
A.7	Brodatz	256×256	5
A.8	Brodatz	256×256	5

first 4 images, A.1 through A.4, include two different texture blocks of size 256×256. Next images are multi-texture images including more than two texture classes. A.5 and A.6 are created by four 128×128 texture blocks and A.7 and A.8 by five 128×128 texture blocks.

Second type of experiments is retrieval of texture using algorithm presented in Section 3.4, but using satellite imagery as the input images. The method can be used effectively for segmentation of urban/non-urban areas, forest/deforested, water/land etc. in large images. The performance of the method is evaluated by both qualitative analysis of the achieved segmentation, which is a visual evaluation by expert user and quantitative analysis by using ground truth segmentation of the satellite images.

Experiments for retrieval is performed using remote sensing images (one IRS [37] Panchromatic and one SPOT [38] Panchromatic image) from the cities of Turkey supplied by Yön Ltd. with the ground truth classifications available. Our main aim is to make a qualitative assessment of the performance visually but also the correct classifications of the data enable us to make an accurate quantitative comparison in addition to the visual verification.

Table 4.2: Test Images used in Texture Retrieval

Name	Type	Width $\times$ Height	Spatial Res. (m)	Num. of Bands
A.9	IRS 1C Pan	1600×2400	5	1
A.10	SPOT 3 Pan	4000×2600	10	1

Table 4.2 shows several properties of the images used in the retrieval experiments. The common property of these images with the ones in Table 4.1 is that they both are single band, eight bit grayscale images. The spatial resolution shows the width and

height of the area on the earth which a single pixel in the image represents. Therefore, this property gives the measure of the minimum size of the objects on earth which can be discriminated on the images.

All of the test images can be seen in Appendix A.

4.3 Experimental Results

4.3.1 Texture Classification Experiments

In the first type experiments, we use texture mosaics including two or more texture blocks in one image created from the textures in Brodatz texture album [39]. The images are in grayscale with samples quantized by 8-bits corresponding to 256 gray levels.

Since algorithm is supervised for each texture (class) in the input image, a training image is prepared and used in learning of the RNN for corresponding texture. In experiments carried out in this section, the test images are classified more than once with different window sizes, i.e. different number of neurons in RNNs.

Table 4.3: Texture Classification Errors for Different Window Sizes n

Name (Fig. No.)	$n = 5$	$n = 7$	$n = 9$	Minimum
A.1	4.5	1.6	1.2	1.2
A.2	17.3	14.7	10.2	10.2
A.3	18.9	16.9	17.6	16.9
A.4	2.6	1.8	1.3	1.3
A.5	27.4	27.3	18.4	18.4
A.6	13.0	9.6	7.3	7.3
A.7	23.9	19.2	15.3	15.3
A.8	25.7	11.1	10.2	10.2
Average	16.7	12.8	10.2	10.1

In Table 4.3 the experimental results are given for the texture classification using different window sizes n. Also, the window size vs. classification error for test images graph can be seen in Figure 4.1. Also, in Figure 4.3 and Figure 4.4, classification results of test images A.4 and A.6 for 9×9 window size are shown.

The experimental results show that, when window size is increased, classification error is decreased. Because, with bigger window size, the texture is represented with RNN more correctly. However, this time, learning stage is took more time. The average success of the algorithm is almost %90. This result is comparable with that of the previous algorithms in the literature.

4.3.2 Texture Retrieval Experiments

The experiments for texture retrieval are carried out for the satellite image whose properties are given in Table 4.2. In the experiment, texture to be retrieved is urban area in the image. Hence, a training image in urban-area is used in learning. The texture retrieval is performed for different window sizes.

Table 4.4: Texture Classification Errors for Different Window Sizes n

Name (Fig. No.)	$n = 5$	$n = 7$	$n = 9$	Minimum
A.9	24.4	23.3	21.8	21.8
A.10	23.6	23.0	21.2	21.2
Average	24.0	23.1	21.5	21.5

In Table 4.4 the experimental results are given for the texture retrieval using different window sizes n. Also, the window size vs. retrieval error for test images graph can be seen in Figure 4.2. Also, in Figure 4.4, part of the test image A.9, its classification using 9×9 window and ground truth classification supplied by Yön Ltd. are shown. The texture part, which is inside the red square in the first image of the figure, is used for training of the network.

As in texture classification, the experimental results show that, when window size is increased, classification error is decreased, as RNN can represent texture more correctly with bigger window size. The average success of the algorithm is almost %80. This result is acceptable when using remote sensing imagery.

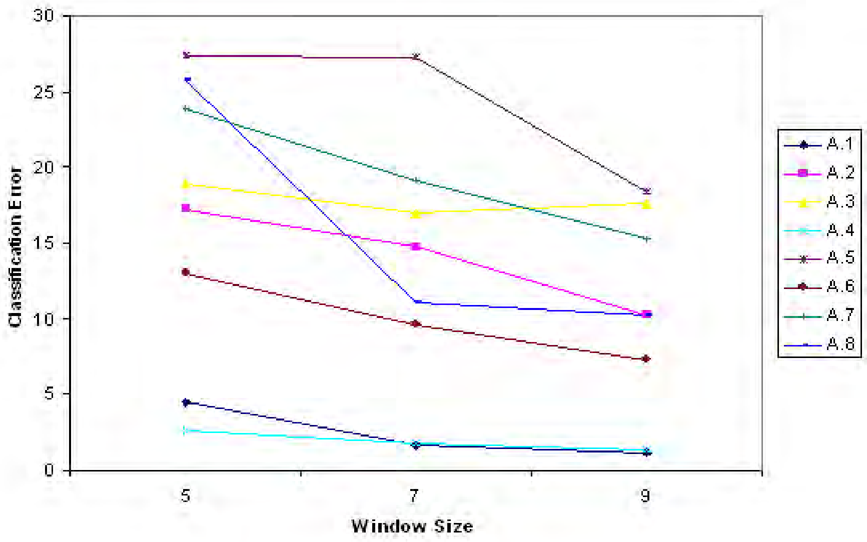


Figure 4.1: Texture Classification Error.

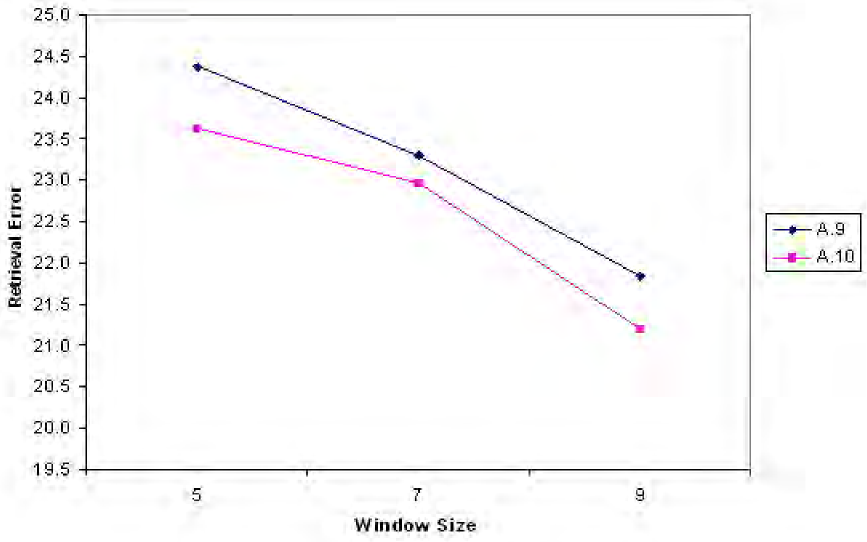


Figure 4.2: Texture Retrieval Error.

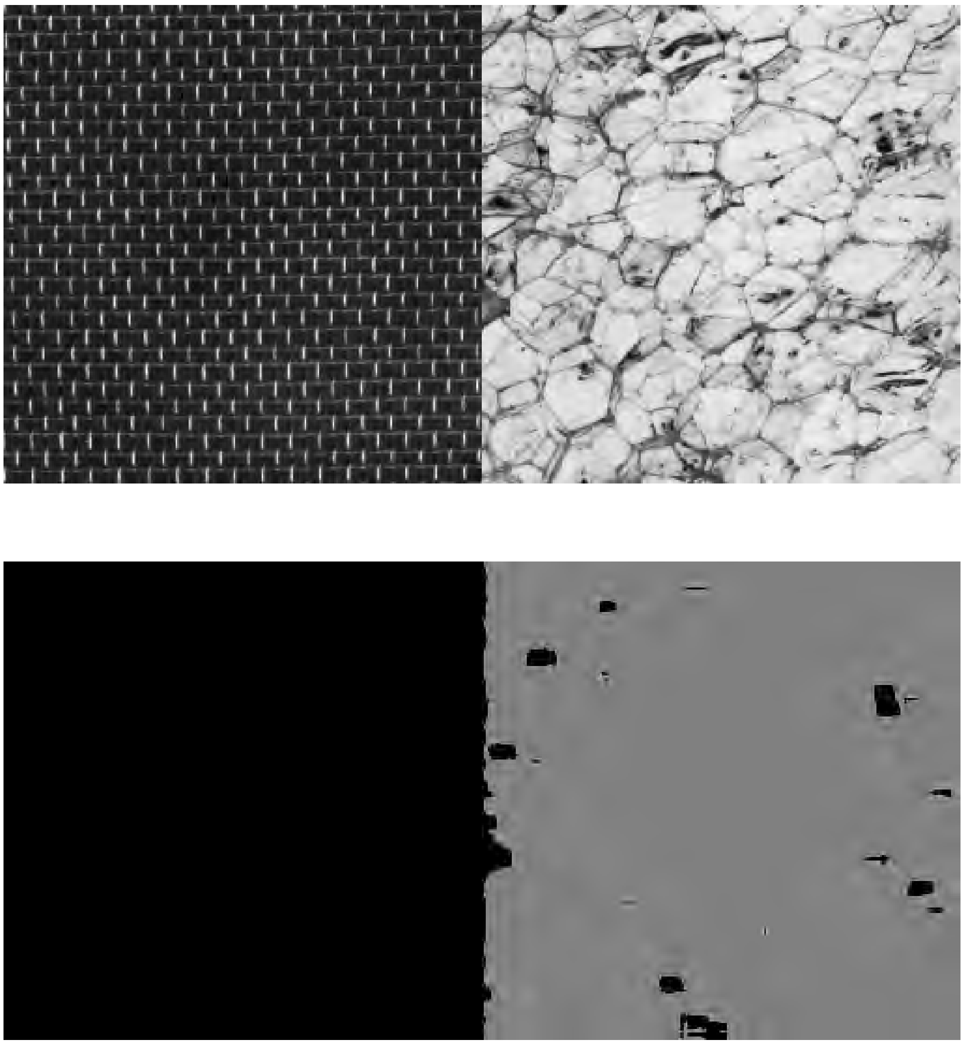
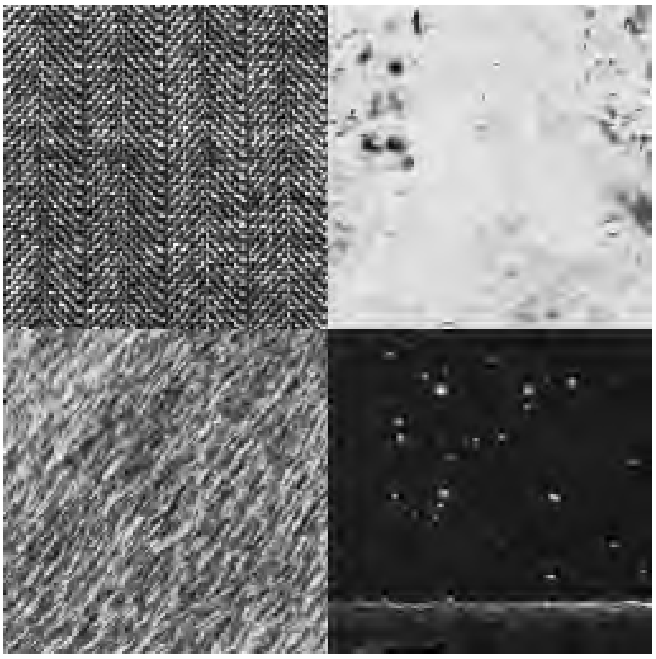


Figure 4.3: Test image A.4 and classification result for 9×9 window.



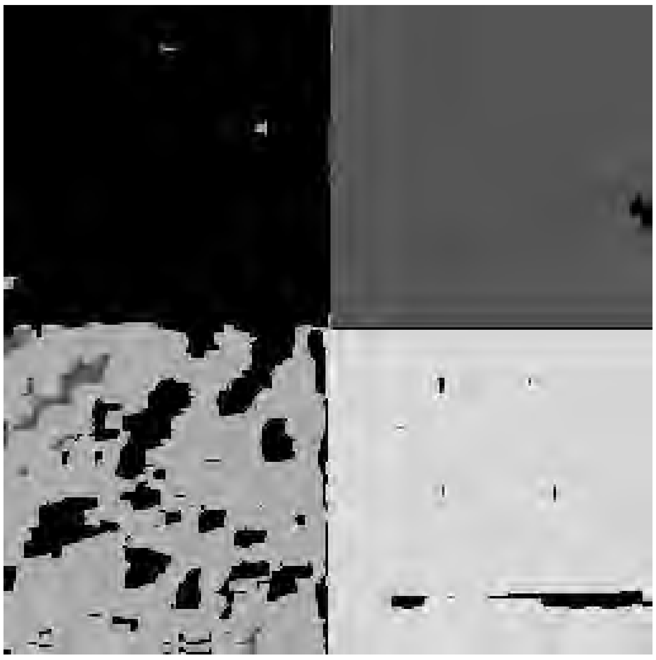


Figure 4.4: Test image A.6 and classification result for 9 × 9 window.

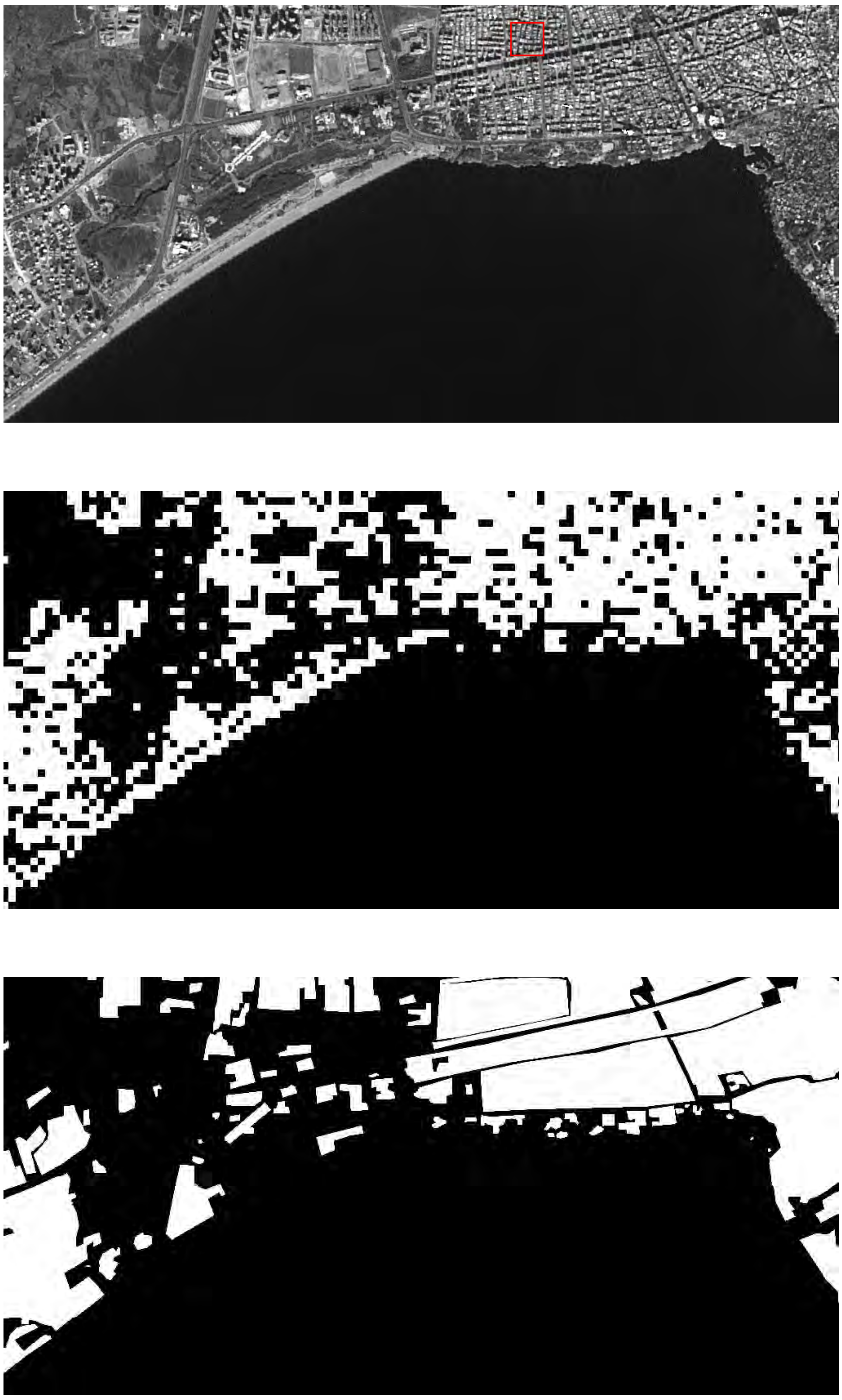


Figure 4.5: Part of the test image A.9, classification result for 9×9 window and correct classification.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Texture classification and retrieval algorithms using random neural network model are presented. The methods are tested and verified by developing two different applications. The first application includes a supervised texture classification implementation. Test images used in the experiments are created from texture source; Brodatz Album which is used in many similar projects of texture analysis and the classification error is used as the performance measure of the experiments. The following conclusions are drawn about the texture classification capability of our method from results of the first application experiments.

- Main parameter for texture classification which directly affects the performance of the classification is the number of neurons N (window size n). N should be large enough for constructing the image block which displays homogenously the properties of the texture to be described. On the other hand, larger N yields greater computational time which is another issue that is to be taken into account.

- Since the algorithm is supervised, the selected train image (texture) is very important. It must be good enough to represent texture so that the algorithm could classify input image correctly

- Results obtain through experiments show that classification error is comparable to that of previous works on classification which make use of other methods.

In the second application, texture retrieval scheme is designed specifically for large remote sensing images where computational time is a crucial issue. RNN is trained using non-overlapping image windows and a training image is chosen representing the properties of a texture class to be searched for homogenously. After training, blocks in the input image is fed into network and if the error is below certain threshold these block is retrieved. This is like a two-class classification where in the experiments an urban/nonurban classification is performed. Conclusions which are drawn from the results achieved in the experiments for this application are as follows.

- Since non-overlapping windows covering the whole image is processed, the computation time required for texture signature computation is dramatically reduced in this scheme.
- Retrieval of urban area class in remote sensing imagery achieved a significant performance of about 80 percent. The parameter which directly affects the retrieval is the threshold value.
- The algorithm described can be used as a top level procedure for a hierarchical classification system. This way, regions in the image which should not be considered for time consuming lower procedural are eliminated. Regions where detailed analysis needed (which is the small part of the whole image),such as pixel-by-pixel classification, another alternative method can be used.

The following subjects are suggested for further study, which can make use of the findings of this thesis:

- Proposed algorithms can be implemented in an imperative language. By doing this, execution time will be decreased. So experiments can be done using large window sizes and more complex textures.
- After classification or retrieval, some post-processing techniques can be applied such as clumping, i.e. combining contiguous group of pixels in one class. This will decrease the error in classification and retrieval results.
- Using texture retrieval in remote sensing images, not only urban/nonurban discrimination but also water/land, forest/openland and object/background discrimination can be performed.

- In this study, neurons are connected to each other according to 8-connectedness of pixels in the image. The connection using some lattice structure or layered architecture may improve the results for some texture types.

REFERENCES

[1] M. Tuceryan and A. Jain, "Texture analysis," in *Handbook of Pattern Recognition and Computer Vision* (C. H. Chen, L. F. Pau, and P. S. P. Wang, eds.), pp. 235–276, World Scientific Publishing, 1993.

[2] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 610–621, 1973.

[3] A. N. Esgiar, R. N. G. Naguib, B. S. Sharif, M. K. Bennett, and A.Murray, "Microscopic image analysis for quantitative measurement and feature identification of normal and cancerous colonic mucosa," *IEEE Transactions on Information Technology in Biomedicine*, vol. 2, pp. 197–203, 1998.

[4] C. M. Wu, Y. C. Chen, and K. S. Hsish, "Texture features for classification of ultrasonic liver image," *IEEE Transactions on Medical Imaging*, vol. 11, pp. 141–152, 1992.

[5] P. M. Mather, *Computer Processing of Remotely-Sensed Images: An Introduction.* John Wiley and Sons, Inc. New York, NY, 1987.

[6] J. R. Irons and G. W. Petersen, "Texture transforms of remotely sensed data," *Remote Sensing Environment*, vol. 11, pp. 359–370, 1981.

[7] A. L. Amet, A. Ertuzun, and A. Ercil, "An efficient method for texture defect detection: Subband domain co-occurrence matrices," *Image and Vision Computing*, vol. 18, pp. 543–553, 2000.

[8] A. Baykut, A.Atalay, A. Ercil, and M. Guler, "Real time defect inspection of textured surfaces," *Journal of Real Time Imaging*, vol. 6, pp. 17–27, 2000.

[9] A. Carkacioglu and F. Y. Vural, "Sasi: A new texture descriptor for image retrieval," *Conference On Image Processing, Thesselenoki, Greece*, 2002.

[10] Y. Xu, P. Duygulu, E. Saber, M. Tekalp, and F. Y. Vural, "Object based image retrieval based on multi-level segmentation," *IEEE International Conference on Acoustics, Speech, and Signal Processing, June 5-9, Istanbul, Turkey*, 2000.

[11] W. Ma and B. Manjunath, "Texture-based retrieval from for browsing and retrieval of image data," *IEEE Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, 1996.

[12] M. Tuceryan and A. K. Jain, "Texture segmentation using voronoi polygons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 211–216, 1990.

[13] A. Pentland, "Fractal-based description of natural scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 661–674, 1984.

[14] M. Yaman, "Texture discrimination and segmentation of remotely sensed imagery by using adaptive subband decomposition," Master's thesis, Middle East Technical University, Computer Engineering Department, 2003.

[15] C. Zhu and X. Yang, "Study of remote sensing image texture analysis and classification using wavelet," *International Journal of Remote Sensing*, vol. 13, pp. 3167–3187, 1998.

[16] G. Cross and A. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 25–39, 1983.

[17] M. Goktepe, V. Atalay, N. Yalabik, and C. Yalabik, "Unsupervised texture based image segmentation by simulated annealing with markov random field and potts models," in *Proceedings of the Fourteenth International Conference on Pattern Recognition (ICPR), Brisbane, Australia*, pp. 820–822, IEEE Computer Society Press, 1998.

[18] N. Y. M. Goktepe and V. Atalay, "Unsupervised segmentation of gray level markov model textures with hierarchical self organizing maps," in *Proceedings of the Thirteenth International Conference on Pattern Recognition (ICPR), Wien, Austria*, vol. 4, pp. 90–94, IEEE Computer Society Press, 1996.

[19] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using gibbs random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 39–55, 1987.

[20] J. R. Jensen and D. L. Toll, "Detecting residential land-use development at the urban fringe," *Photogrammetric Engineering and Remote Sensing*, vol. 48, pp. 629–643, 1982.

[21] E. H. H. Shih and R. A. Schowengerdt, "Classification of arid geomorphic surfaces using landsat spectral and textural features," *Photogrammetric Engineering and Remote Sensing*, vol. 49, pp. 337–347, 1983.

[22] P. T. Nguyen and J. Quinqueton, "Space filling curves and texture analysis," in *Proceedings Sixth International Conference on Pattern Recognition, Munich, Germany*, 1982.

[23] R. G. Blom and M. Daily, "Radar image processing for rock-type discrimination," *IEEE Transactions on Geoscience Electronics*, vol. 20, pp. 343–351, 1982.

[24] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502–511, 1989.

[25] E. Gelenbe, "Stability of the random neural network model," *Neural Computation*, vol. 2, no. 2, pp. 239–247, 1990.

[26] E. Gelenbe, A. Stafylopatis, and A. Likas, "An extended random network model with associative memory capabilities," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 307–312, June 1991.

[27] E. Gelenbe, "Distributed associative memory and the computation of membership functions," *Information Sciences*, vol. 57–58, pp. 171–180, 1991.

[28] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," *ELEKTRIK, Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 2, pp. 1–10, 1994.

[29] E. Gelenbe and F. Batty, "Minimum cost graph covering with the random neural network," in *Computer Science and Operations Research* (O. Balci, R. Sharda, and S. Zenios, eds.), pp. 139–147, Pergamon, NY, 1992.

[30] A. Ghanwani, "A qualitative comparison of neural network models applied to the vertex covering problem," *ELEKTRIK, Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 2, pp. 11–19, 1994.

[31] V. Atalay, E. Gelenbe, and N. Yalabık, "The random neural network model for texture generation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 6, no. 1, pp. 131–141, 1992.

[32] V. Atalay and E. Gelenbe, "Parallel algorithm for colour texture generation using the random neural network model," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 6, no. 2, pp. 437–446, 1992.

[33] E. Gelenbe, Y. Feng, and K. Krishnan, "Neural network methods for volumetric magnetic resonance imaging of the human brain," *Proceedings of the IEEE*, vol. 84, pp. 1488–1496, 1996.

[34] E. Gelenbe, Z. Mao, and Y. Li, "Function approximation with the random neural network," *IEEE Transactions on Neural Networks.*

[35] E. Gelenbe, Z. Mao, and Y. Li, "Function approximation by random neural networks with a bounded number of layers,"

[36] E. Gelenbe and T. Koçak, "Mine detection using local area information and the random neural network," *IEEE Transactions on PAMI.*

[37] Indian Space Research Organization, IRS, `http://www.isro.org/`.

[38] SPOT Image, SPOT, `http://www.spotimage.fr`.

[39] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* Dover, NY, 1966.

[40] R. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.

[41] A. K. Jain, *Fundamentals of Digital Image Processing.* Prentice-Hall, NJ, 1989.

[42] G. Strang and T. Nguyen, *Wavelets and Filter Banks.* WellesleyCambridge Press, Massachusetts, MA, 1997.

[43] G. Cross and A. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 25–39, 1983.

[44] D. Hebb, *The organization of behavior.* John Wiley and Sons, NY, 1949.

[45] V. Atalay, "Learning in random neural network by optimization," *The Thirteenth International Symposium on Computer and Information Sciences, Antalya, Turkey*, pp. 143–148, 1998.

[46] V. Atalay and E. Gelenbe, "Storage and reconstruction of image textures using random neural network,"

[47] The MathWorks Inc., MATLAB, `http://www.mathworks.com/`.

[48] H. M. Abdelbaki, RNNSIM, `http://www.cs.ucf.edu/~ahossam/rnnsimv2/`.

[49] H. M. Abdelbaki, "Random neural network simulator for use with matlab," technical report, University of Central Florida, September 1999.

[50] T. Randen and J. H. Husøy, "Filtering for texture classification: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

APPENDIX A

TEST IMAGES

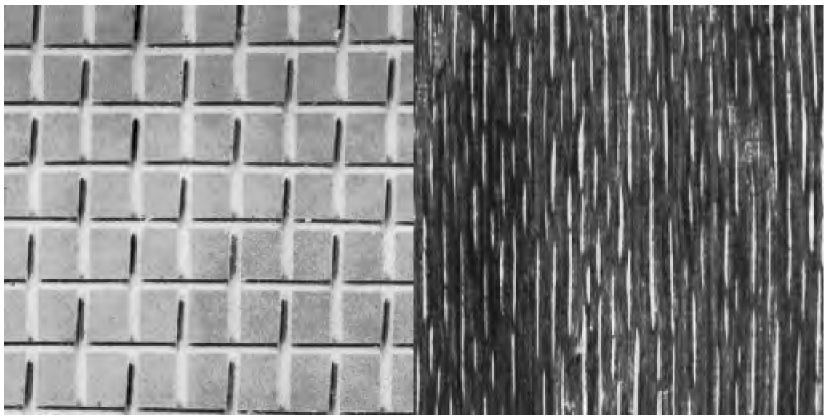


Figure A.1: 512×256 test image of two-texture classes (D1, D68) from Brodatz album.

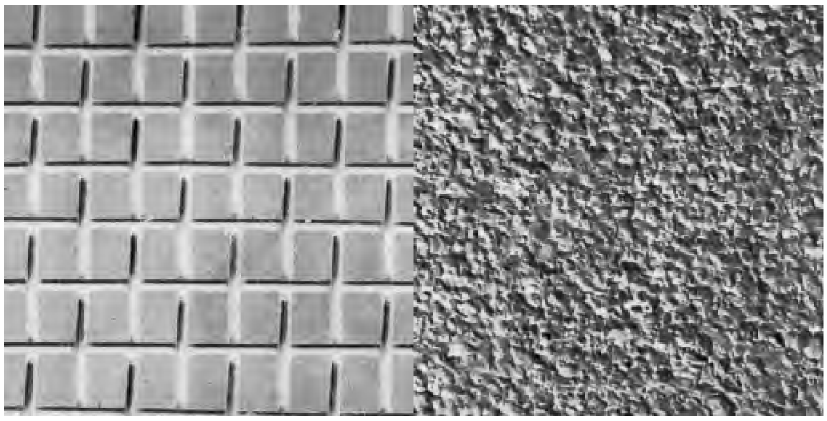


Figure A.2: 512×256 test image of two-texture classes (D1, D4) from Brodatz album.

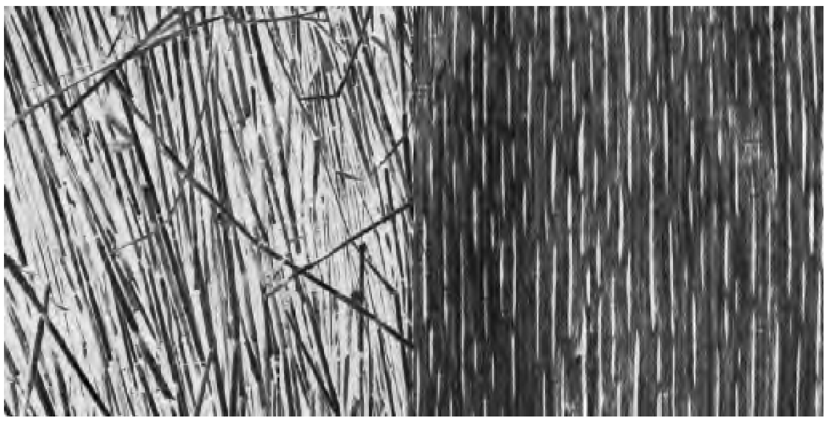


Figure A.3: 512×256 test image of two-texture classes (D15, D68) from Brodatz album.

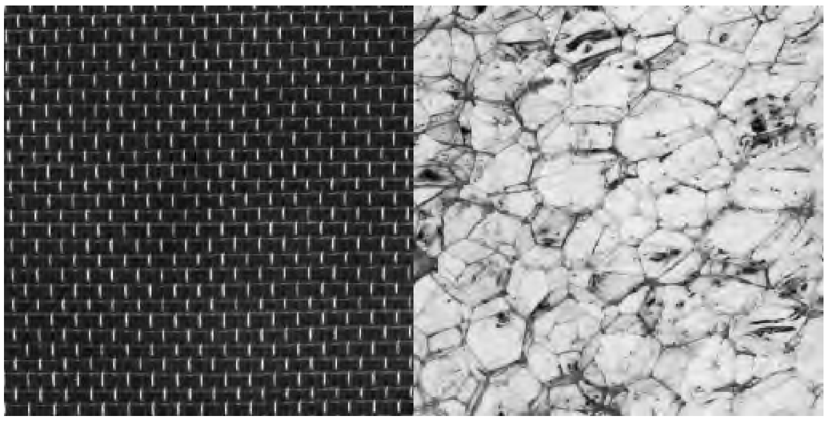


Figure A.4: 512×256 test image of two-texture classes (D6, D112) from Brodatz album.

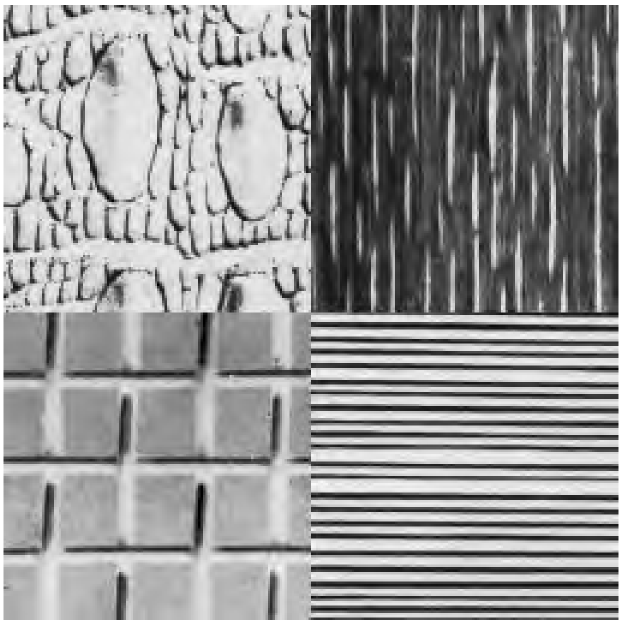


Figure A.5: 256 × 256 test image of 4 texture classes (D10, D68, D1, D49) from Brodatz album.

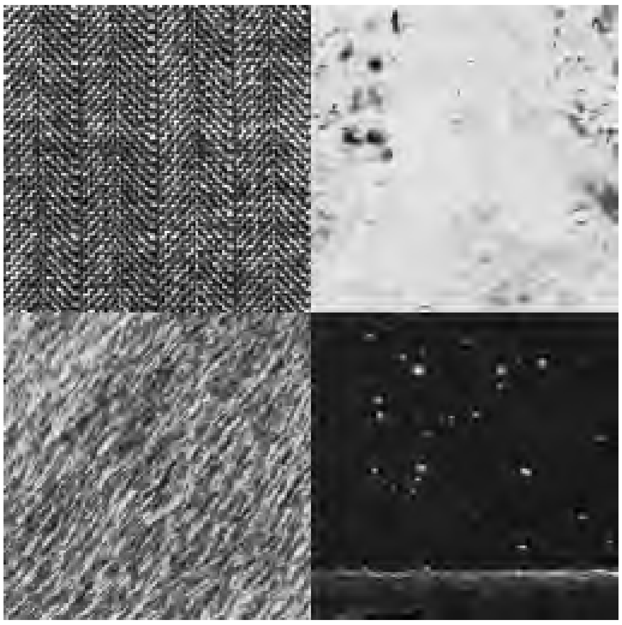


Figure A.6: 256 × 256 test image of 4 texture classes (D16, D58, D93, D25) from Brodatz album.

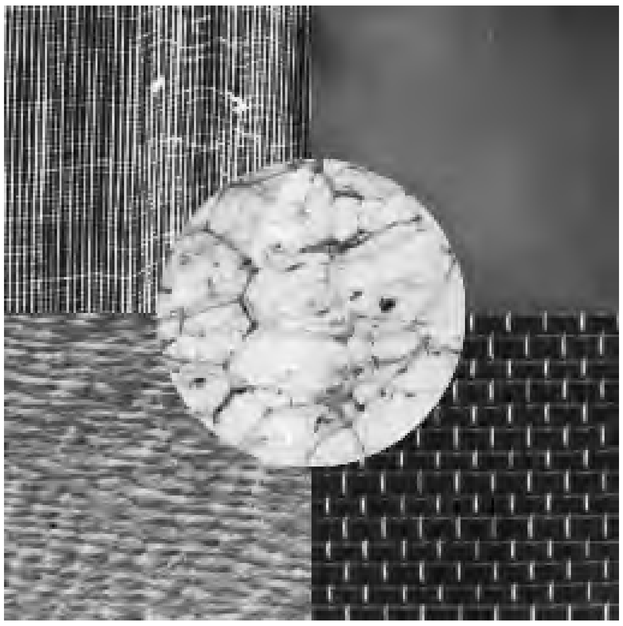


Figure A.7: 256×256 test image of 5 texture classes (D105, D91, D38, D6, D112) from Brodatz album.

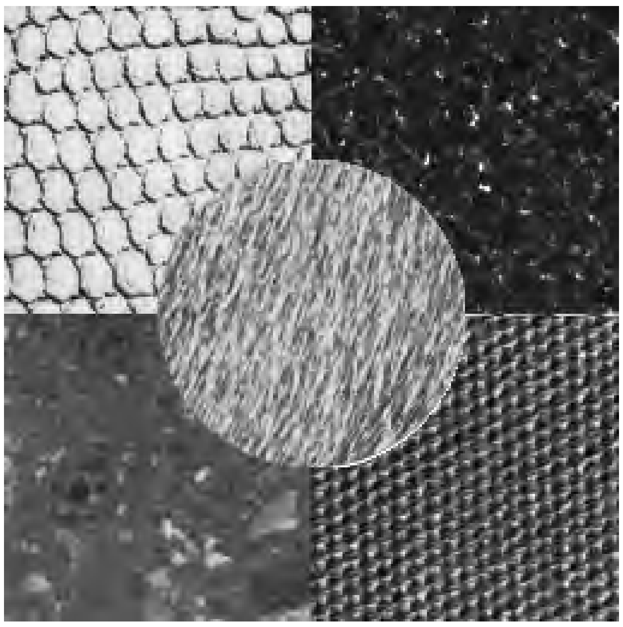


Figure A.8: 256×256 test image of 5 texture classes (D3, D59, D61, D77, D93) from Brodatz album.

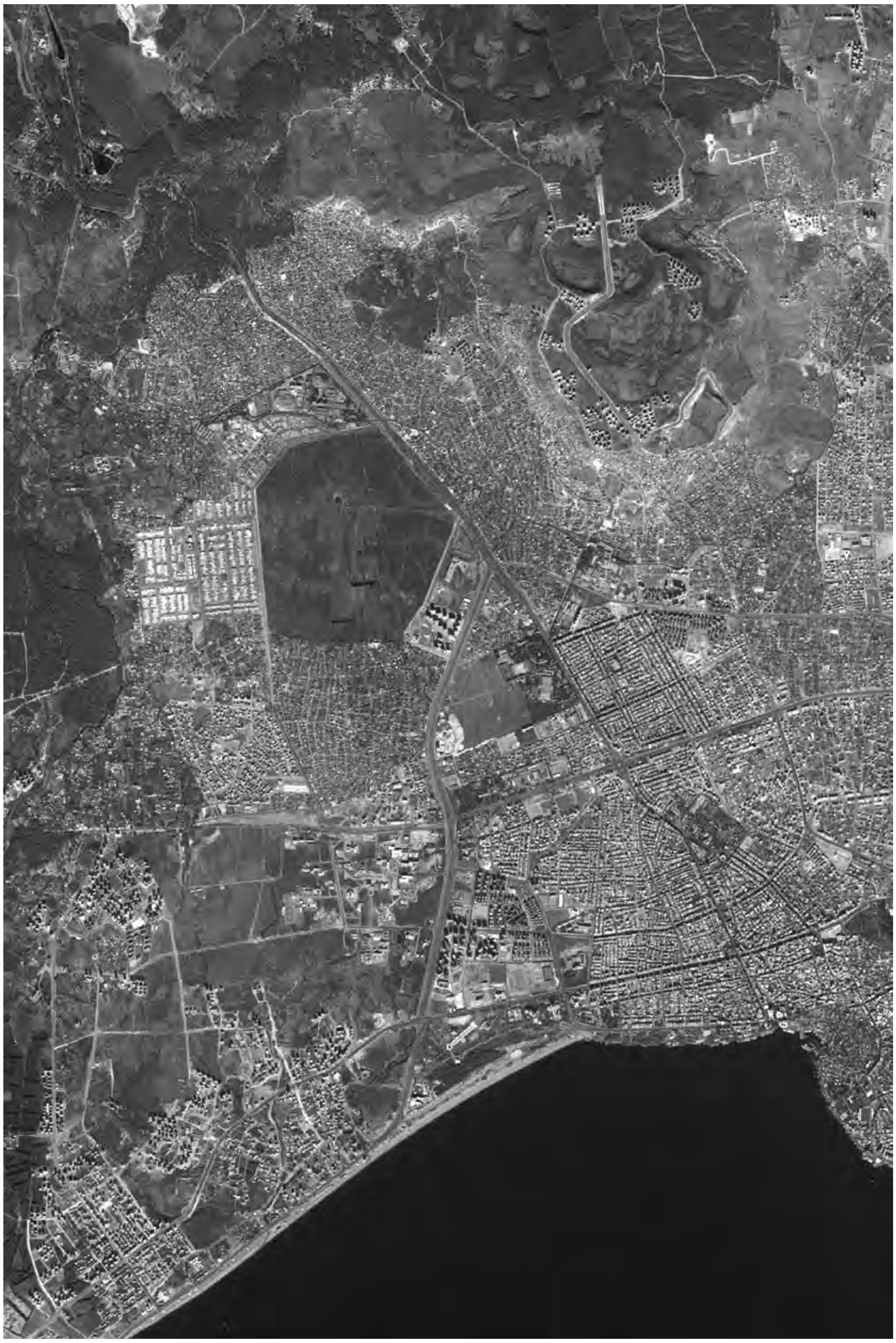


Figure A.9: Single band, grayscale Image (IRS 1C Pan) of size (1600x2400) with spatial resolution 5m

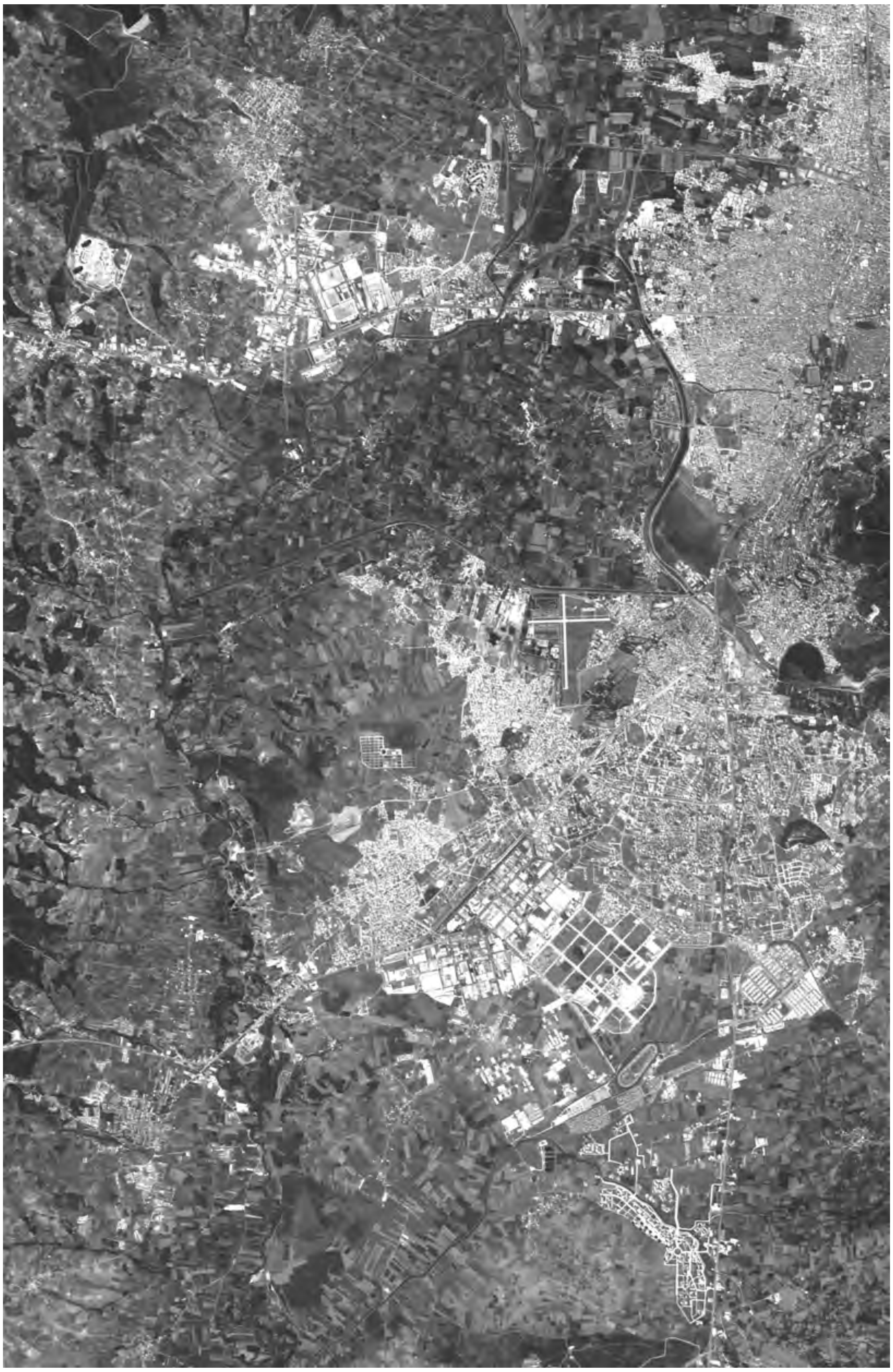


Figure A.10: Single band, grayscale Image (SPOT Pan) of size (4000x2600) with spatial resolution 10m