Republic of Iraq
Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science

Texture Classification Based on Fuzzy Logic

A Thesis
Submitted to the College of Science, Al-Nahrain University in Partial Fulfillment of the Requirements for The Degree of Master of Science in Computer Science

By
Rukaya Ayad Abd Al-Jabar
(B.Sc. 2004)

Supervised By
Dr. Sawsan K. Thamer

1429 **2008**

بِسْمِ اللّهِ الرَّحْمنِ الرَّحِيمِ

وَيَسْأَلُونَكَ عَنِ الرُّوحِ قُلِ الرُّوحُ مِنْ أَمْرِ

رَبِّي وَمَا

أُوتِيتُم مِّن الْعِلْمِ إِلاَّ قَلِيلاً

صدق الله العظيم

سورة الأسراء (الآية٨٥)

SUPERVISOR CERTIFICATION

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Rukaya Ayad Abd Al-Jabar** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Supervisor

Signature:

Name: **Dr. Sawsan K. Thamer**

Title: **Lecturer**

Date: /   / **2008**

The Head of the Department Certification

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name: **Dr. Taha S. Bashaga**

Title: **Head of the department of Computer Science, Al-Nahrain University.**

Date: /   / **2008**

Examining Committee Certification

We certify that we have read this thesis and as an examining committee, examined the student in its content and what is related to it and that in our opinion it meets the standard of a thesis for the degree of Master of Science in Computer Science.

Supervisors Certification

Signature:

Name: **Dr. Sawsan K. Thamer**

Title: **Lecturer**

Date: / / **2009**

Examining Committee Certification

Signature:

Name: **Dr. Laith Alani**

Title: **Assistance Professor (Chairman)**
Date: / / **2009**

Signature:

Name: **Dr. Taha S. Bashaga**

Title:**Lecture (Member)**
Date: / / **2009**

Signature:

Name: **Dr. Bara'a A. Attea**

Title: **Assistant Professor (Member)**
Date: / / **2009**

The Dean of the College Certification

Approved by the Council of the College of Science

Signature:

Name: **Dr. Laith Alani**

Title: The Dean of College of Science**, Al-Nahrain University.**

Date: / / **2009**

DEDICATED TO

My Parents …

My Husband…

My Friends…

To everyone
Taught me a letter

ACKNOWLEDGMENT

Above all else, I want to express my great thanks to my god (Allah) for his uncountable gifts and for helping me to present this wok.

I would like to express my sincere appreciation to my supervisor, Dr. Sawsan K. Thamer, for her guidance, generosity and continued assistance during the development of this work.

Grateful thanks for the Head of Department of Computer Science, Dr. Taha S. Bashaga, Also,I wish to thank the staff of Computer Science Department at Al-Nahrain University for their help.

I would like to say "thank you" to my faithful friends for supporting and giving me advises.

Finally, special thank go to my family, especially my mother and my husband, for their understanding, support, and encouragement during the search period.

Rukaya Ayad

ABSTRACT

The classification process is an important task in many application of computer image analysis for classifying images based on color or texture low-level features.

In this work, a texture classification system is presented which supports querying with respect to texture low-level feature. The fundamental idea is to generate automatically image description by analyzing the image content. The underlying techniques are based on the Gray-Level Co-occurrence Matrix (GLCM) and Gray-Level Run Length Matrix (GLRLM) as statistical approaches to texture analysis. These two techniques are applied in separated manner.

Each class is represented by features vector(s) in the features space and stored in a file. Then, a selection to the best set of features is done using fuzzy concepts (triangular membership functions or trapezoidal membership functions). Given the query image, the system first extracts its features vector, and then compares the selected features with those stored in the file to find the nearest class using fuzzy concept.

During the evaluation process, it was found that; the best results are obtained from combination among features which in turn achieve higher selection rate for features as well as for whole system (gets selection rate nearly 90% for combination among four features and 60% without combination).

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BMP	BitMap Pixel
ClustePro m	Cluster prominence
DifEnt	Difference entropy
DifMom	Difference moment
DifVar	Difference variance
FTCS	Fuzzy-Based Texture Classification System
GLCM	Gray Level Co-occurrence Matrix
GLD	Gray Level Distribution
GLRLM	Gray Level Run Length Matrix
HGRE	High Gray-Level Run Emphasis
Homog	Homogeneity
HSI	Hue, Saturation, and Intensity
HSV	Hue, Saturation, and Value
InfMeasCor	Information measures of correlation
InvDifMom	Inverse difference moment
LGRE	Low Gray-Level Run Emphasis
LRE	Long Run Emphasis
LRHGE	Long Run High Gray-level Emphasis
LRLGE	Long Run Low Gray-level Emphasis
MaxProb	Maximum probability
pdf	Probability Density Function
RGB	Red, Green, and Blue
RLD	Run Length Distribution
RP	Run Percentage
SRE	Short Run Emphasis
SRHGE	Short Run High Gray-level Emphasis
SRLGE	Short Run Low Gray-level Emphasis
STD	Standard Deviations
SumAvg	Sum average
SumEnt	Sum Entropy
SumVar	Sum Variance
TFN	Triangular Fuzz Number

TABLE OF CONTENTS

Chapter Three

(Fuzzy-Based Texture Classification System Implementation)

Chapter Four

(Tests and Results)

Chapter Five

(Conclusions and Suggestions for Future Work)

Chapter One

Overview

CHAPTER ONE

OVERVIEW

1.1 Introduction

Computer vision refers to the field of computer science that is concerned with the design and implementation of algorithms that allow machines to simulate human vision. Various fields related to computer processing of images are categorized according to the type of input which they take and type of results they produce; these categories are [Nib86]:

- Image processing: takes an image as an input, perform some operations on it (such as: enhancement, transformation, rotation, etc.), then produces the processed image as an output.
- Computer graphics: produces images according to received description information (such as: line drawing or use 3 dimensional view of an object with effect for shading, lighting, etc.).
- Pattern recognition and Computer vision: produces descriptive information concerning the received image.

Computer vision is broader than pattern recognition in the sense that is concerned with a complete system.

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified.

These objects will be referred to by using the generic term patterns (i.e usually, it is refer to a pattern as a description of an object which is to be recognized). Objects are described by a set of measurements called also attributes or features [Fri99, The03, and Kun04].

Applications of pattern recognition systems and techniques are numerous and cover a broad scope of activities. A few examples are enumerated only to refer to several professional activities [Mar01]:

- Astronomy:
  Analysis of telescopic images
  Automated spectroscopy
- Biology:
  Automated cytology
  Properties of chromosomes
  Genetic studies
- Engineering:
  Fault detection in manufactured products
  Character recognition
  Speech recognition
  Automatic navigation systems
  Pollution analysis
- Geology:
  Classification of rocks
  Estimation of mining resources
  Analysis of geo-resources using satellite images
  Seismic analysis
- Medicine:
  Analysis of electrocardiograms
  Analysis of electroencephalograms
  Analysis of medical images

- Military:

  Analysis of aerial photography

  Detection and classification of radar and sonar signals

  Automatic target recognition

- Security:

  Identification of fingerprints

  Surveillance and alarm systems

As can be inferred from the above examples, the patterns to be analysed and recognized can be signals (e.g. e1ectrocardiographic signals), images (e.g. aerial photos) or plain tables of values (e.g. stock exchange rates) [Mar01].

A pattern recognition investigation may consist of several stages, enumerated below [Web02]:

1. Formulation of the problem: gaining a clear understanding of the aims of the investigation and planning the remaining stages.

2. Data collection: making measurements on appropriate variables and recording details of the data collection procedure (ground truth).

3. Initial examination of the data: checking the data, calculating summary statistics and producing plots in order to get a feel for the structure.

4. Feature selection or feature extraction: selecting variables from the measured set that are appropriate for the task. These new variables may be obtained by a linear or nonlinear transformation of the original set (feature extraction). To some extent, the division of feature extraction and classification is artificial.

5. Unsupervised pattern classification or clustering. This may be viewed as exploratory data analysis and it may provide a successful conclusion to a study. On the other hand, it may be a means of preprocessing the data for a supervised classification procedure.
6. Apply discrimination or regression procedures as appropriate. The classifier is designed using a training set of exemplar patterns.
7. Assessment of results. This may involve applying the trained classifier to an independent test set of labelled patterns.
8. Interpretation.

There are two main divisions of classification: supervised classification (discrimination) and unsupervised classification (sometimes in the statistics literature simply referred to as classification or clustering).

In supervised classification, we have a set of data samples (each consisting of measurements on a set of variables) with associated labels, the class types. These are used as exemplars in the classifier design.

In unsupervised classification, the data are not labeled and we seek to find groups in the data and the features that distinguish one group from another. Most important feature to be extracted about regions in pattern recognition or image classification problems is through texture analysis [Sam99, Dud00, Web02, and The03].

1.2 Texture analysis

Texture provides a rich source of information about the natural scene. For designers, a texture adds richness to a design. For computer scientists, a texture is attractive not only because it is an important component in image analysis for solving a wide range of applied recognition, segmentation and synthesis problems, but also it provides a key to understand basic mechanisms that underlie human visual perception [Zho06].

Texture analysis is one of the most important techniques used in analysis and classification of images presenting repetition of fundamental image elements. Texture can be recognized when it is seen, but it is a very difficult concept to define [Gui88, Roa87].

The attributes and utilities of textures can be summarized as [Ach05]:

- Textures are repetitive patterns, which characterize the surfaces of many classes of objects. Thus classification of object patterns becomes easy if the textures present in the image are identified and differentiated from each other.
- Textures provide vital information about the arrangement of the fundamental elements of an image.
- The attributes of a texture may be described in qualitative terms such as coarseness, homogeneity, orientation of image structure, and spatial relationships between image intensities or tones. Texture analysis is the quantification and use of such image properties which aid in texture discrimination.

Texture analysis applications have been utilized in a variety of image processing fields such as: automated inspection, medical image processing, remote sensing, and document processing [Sam99].

The various methods for modeling texture and extracting texture features can be applied in four broad categories of problems: texture segmentation, texture classification, texture synthesis, and shape from texture. Among the main tasks of texture analysis are segmentation and classification [Sam99, Sag06]:

- Texture segmentation: Is the process that subdivides an image in to its constituent parts or objects. One does not need to know which specific texture exist in the image in order to do texture segmentation. All that indeed is away to tell that two textures (usually in adjacent regions of an image) are different.
- Texture classification: Texture classification involves deciding what texture category an observed image belongs to. In order to accomplish this, one needs to have a priori knowledge about the classes to be recognized. Once this knowledge is available and the texture features are extracted, then one can use classical pattern classification techniques in order to do the classification.

Texture features could be derived using various approaches such as [Sag06]:

- Structural approach.
- Statistical approach.
- Syntactic approach.

Statistical texture analysis methods deal with the distribution of grey levels (or colures) in a texture. The first order statistics and pixel-wise analysis are not able to efficiently define or model a texture. Therefore, statistical texture analysis methods usually employ higher order statistics or neighborhood (local) properties of textures. The most commonly used statistical texture analysis methods are co-occurrence matrices, autocorrelation function, texture unit and spectrum, and grey level run-length [Mon04].

Fuzzy approaches to pixel classification have found applications in problems where (1) precise knowledge about the pattern classes is not available, (2) large number of pattern samples are not available for statistical estimation of parameters; (3) patterns have partial membership to different classes [Ach05].

In classical two-state logic, an element either belongs or does not belong to a given class. In real life, however, the classes are often ill defined, or overlapping, or fuzzy and a pattern may belong to more than one class. In such a situation, the fuzzy set theoretic techniques have proved to be useful.

There has an increasing use of fuzzy set theory and fuzzy algorithms for image processing implementations. This is motivated by desire to model the ambiguity and noise contained in digitally defined image [Ali99].

1.3 Review of Previous Studies

1. Hirota, et al (1994) [Hir94], have introduced "Implicitly - supervised fuzzy pattern recognition", Introduced a new model of fuzzy pattern recognition where data available about class membership is given implicitly rather than explicitly. While the explicit classification training set conveys complete details about class membership, the implicit format of classification lends itself to more synthetic forms of classification outcomes (such as those expressed in terms of similarities between some pairs of patterns).

2. I. Nedeljkovic (2004) [Ned04], had introduced "Image Classification Based on Fuzzy Logic", an idea to solve the problem of image classification in fuzzy logic manner as well as comparison of the results of supervised and fuzzy classification was the main motivation of this work. Behind this idea was also the question if the possible promising results can give the answer to the question of diminishing the influence of person dealing with supervised classification.

3. M. M. Hoque and S. M. Faizur (2007) [Moh07], had introduced "Fuzzy Features Extraction from BANGLA Handwritten Character". This paper described a method to efficiently detect the meaningful fuzzy features including global features, geometric features and positional features from handwritten Bangla character respectively. The system has been tested for different types of Bangla handwritten alphabets in various styles and they got a successful feature extraction results for most of the test cases.

1.4 Aim of the Work

The aim of this work is to build an image classification system to manipulate an input image and classify it into one of the predefined set of classes. This classification is accomplished by using the following steps:

1. Extract feature sets using statistical approach.
2. Select the best features using fuzzy logic.
3. Classify the query image using fuzzy logic depending on the selected features.

1.5 Thesis Layout

- Chapter two: This chapter describes in some details the definition of texture, application of texture analysis and it introduces how features are extracted from images. It includes also the basic concepts of fuzzy logic.
- Chapter three: In this chapter the implementation steps of the Fuzzy Texture Classification System (FTCS) are presented.
- Chapter Four: In this chapter, the test results are presented and discussed to evaluate the performance of the established system.
- Chapter Five: In this chapter, some of the derived conclusions are listed and a list of suggestions for the future work is given.

CHAPTER TWO

TEXTURE ANALYSIS AND FUZZY LOGIG

Chapter Two

Texture Analysis and Fuzzy Logic

2.1 Introduction

In a typical pattern recognition or object classification process, the first step is the extraction of features or key properties of objects (i.e. mapping from the real world to the feature space). The next step is classification of objects according to their features (i.e. mapping from the feature space to the classification space). The human brain is an excellent classifier which can successfully classify objects in noisy environments even without significant features. However, it still cannot be expected the same performance from our artificial classifiers. Therefore, to work towards a successful classification, extracted features of different objects must show adequate separation in the feature space.

Figure 2.1 illustrates the structure of a traditional pattern recognition system. The two main stages, feature extraction and classification, eventually map the input object into one of the K classes of the classification space [Mon04].

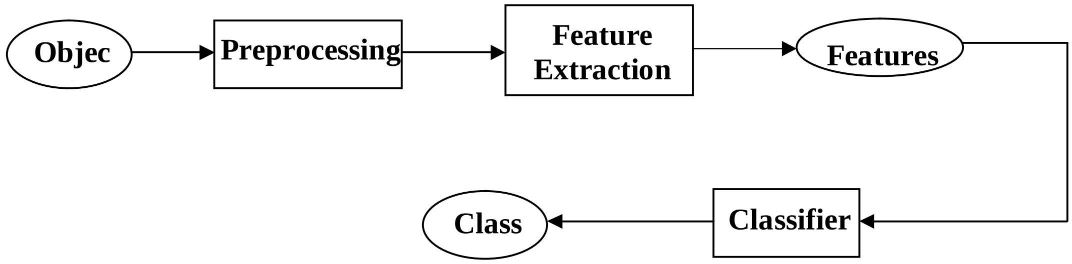


Figure 2.1: A pattern recognition system.

2.2 Texture

Quantitative study of images is often concerned with four types of parameters, which are of fundamental importance. These are Contrast (a very important measure in image processing which often determines the quality of an image), Color (adds more useful discriminatory information to the image), Shape (a measure which is used in recognizing the various object contained in an image), and Texture (Describe the spatial distribution of tonal value within band and provide useful information for performing automatic interpretation and recognition) [Ala96].

Texture is an important characteristic for the analysis of many types of images. It can be seen in all images from multi-spectral scanner images obtained from aircraft or satellite platforms (which the remote sensing community analysis) to microscopic images of cell cultures or tissue samples (which the biomedical community analysis) [Har79].

Although texture can be recognized when it is seen, but it is very difficult to define. This difficulty is demonstrated by the number of different texture definitions attempted by vision researchers. The following are some of these definitions:

- A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic [Tuc98].
- The image texture is nonfigurative and cellular. An image texture is described by the number, and types of its (tonal) primitives, and the spatial organization, or layout of its (tonal) primitives. A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tone surface, there exists a scale such that when the

surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture [Har79].

- Texture is used to describe two dimensional arrays of variations... The elements and rules of spacing or arrangement may be arbitrarily manipulated, provided a characteristic repetitiveness remains [Tra01].
- The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region [Tra01].

Texture may be classified as being artificial or natural. Artificial textures consist of arrangements of symbols, such as line segments, dots, and stars placed against a neutral background. Several examples of artificial texture are presented in figure (2.2). As the name implies, natural textures are images of natural scenes containing semi repetitive arrangements of pixels. Examples include photographs of brick walls, stone, sand, and grass. Figure (2.3) shows several natural texture examples [Pra01].

Uniformity, density, coarseness, roughness, regularity, linearity, directionality, frequency, and phase, are important to describe textures. Some of these perceived qualities are not independent. For example, frequency is not independent of density and the property of direction only applies to directional textures. The fact that the perception of texture has so many different dimensions is an important reason why there is no single method of texture representation which is adequate for a variety of textures. For that, different types of texture may need different features to represent and classify them [Tuc98].

Texture properties are used to discriminate (i) one object from other, (ii) an object from background, (iii) to draw inference about 3D worlds. For that any machine vision system must be able to deal with texture [Kar94].

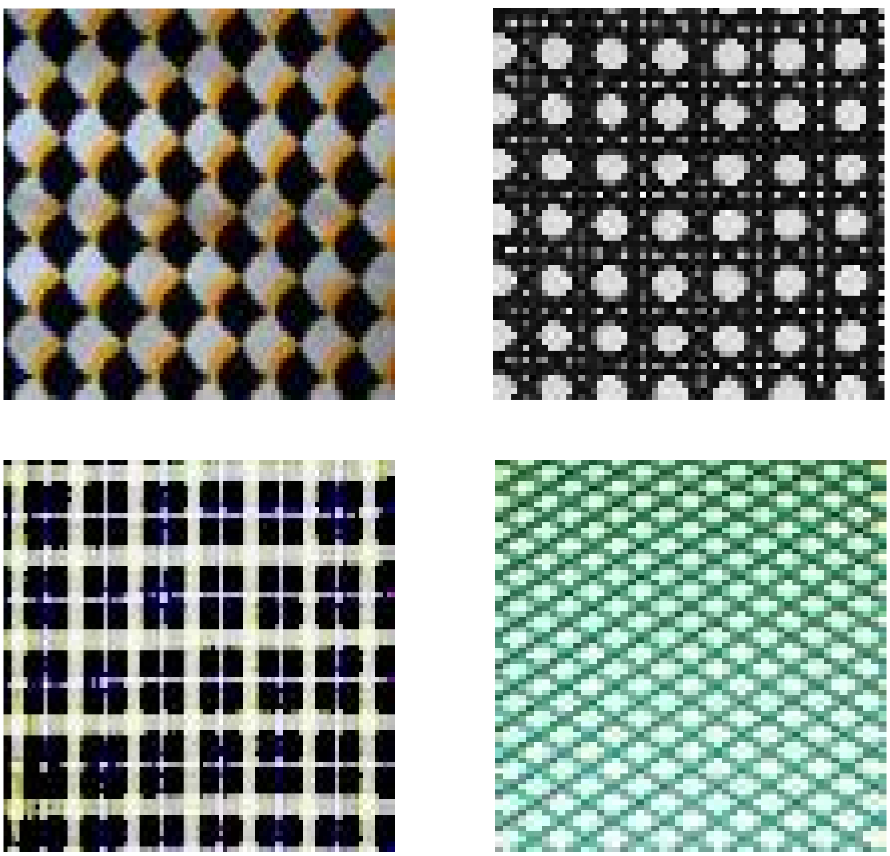


Figure (2.2): Artificial texture.

Figure (2.3): Natural texture.

2.3 Texture Analysis and its Applications

Major goals of texture research in computer vision are to understand, model and process texture, and ultimately to simulate human visual learning process using computer technologies.

Four major application domains related to texture analysis are texture classification, texture segmentation, shape from texture, and texture synthesis [Zho06].

2.3.1 Texture Classification

Texture classification assigns a given texture to some texture classes. There are two main classification methods; they are supervised and unsupervised classification. A supervised classifier is trained using the set to learn a characterization for each texture class. Unsupervised classification does not require prior knowledge, which is able to automatically discover different classes from input textures [Tuc98, Zho06].

Before the classification process done, the training process is done, where some known texture images are used to train the classifier. Training typically involves four major steps. These are:

1. Image pre-processing,
2. Sampling,
3. Feature extraction,
4. Classifier training.

The pre-processing step is used typically for image enhancement and noise removal, although some techniques perform scaling and rotation in this step as well, in order to compensate for variations in the training data.

Image data are often limited in terms of the number of original source images available, so sampling process is done in order to increase the amount of data which divided the images into sub images, either overlapped or disjoint, of a particular window size.

The most important stage of the classification process is the feature extraction stage, at which time the sample image is transformed into a much lower dimensionality feature vector. Many different techniques have been used to handle this stage.

The vectors from all of the training images are then input to the classification system for training. . Again, many different classifiers have been used, although some of them perform slightly better than others, generally the choice of classifier has the least effect on the overall performance of the system. Therefore, speed of training, ease of implementation, and suitability to a given task are more important factors in the choice of a classifier than is its raw performance [Sag06].

2.3.2 Texture Segmentation

Texture segmentation is a difficult problem because one usually does not know a priori what types of textures exist in an image, how many different textures there are, and what regions in the image have which textures. In fact, one does not need to know which specific textures exist in the image in order to do texture segmentation. All that is needed is a way to tell that two textures (usually in adjacent regions of the images) are different.

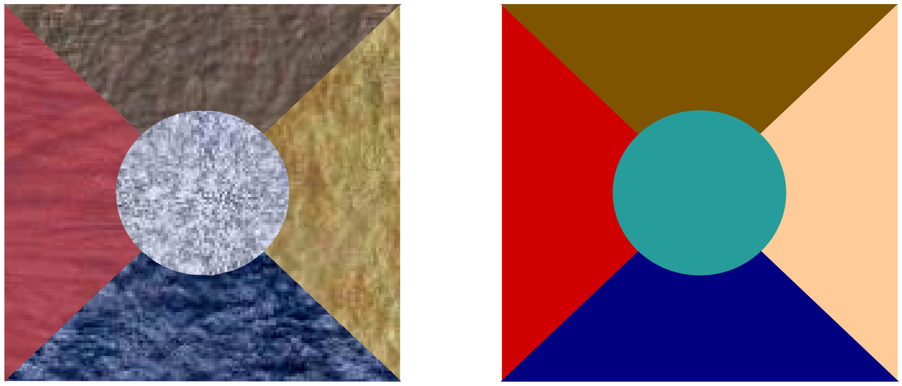
The two general approaches to perform texture segmentation are analogous to methods for image segmentation: region-based approaches or boundary-based approaches.

Region-based approach, one tries to identify regions of the image which have a uniform texture. Pixels or small local regions are merged based on the similarity of some texture property. The regions having different textures are then considered to be segmented regions.

The boundary-based approaches are based upon the detection of differences in texture in adjacent regions. Thus boundaries are detected where there are differences in texture [Tuc98].

Texture segmentation could also be supervised or unsupervised depending on if prior knowledge about the image or texture class is available. Supervised texture segmentation identifies and separates one or more regions that match texture properties shown in the training textures. Unsupervised segmentation has to first recover different texture classes from an image before separating them into regions. Compared to the supervised case, the unsupervised segmentation is more flexible for real world applications despite that it is generally more computationally expensive. Similar to classification, segmentation of texture also involves extracting features and deriving metrics to segregate textures. Figure (2.4) shows an example of texture segmentation.

Partitioning an image into homogeneous regions is very useful in a variety of applications of pattern recognition and machine leaning [Zho06].



(a) **(b)**

Figure (2.4): Example of texture segmentation showing
(a) Original image
(b) Segmented image

2.3.3 Shape from Texture

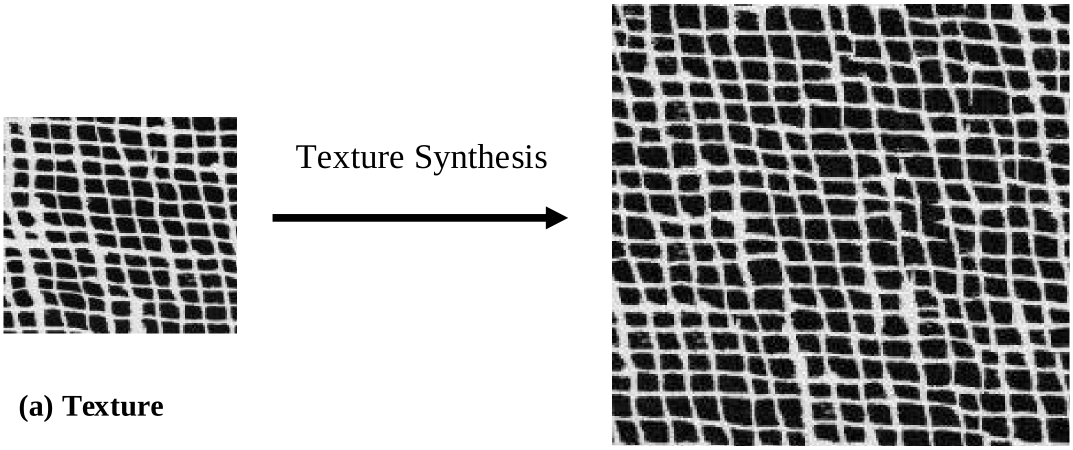
Determining the shape of an object in three dimensional shape is an important task in image processing, and there exist many features in images that allow the viewer to make such a determination, for example variations in intensity on the surface of objects, the relative positions and orientations of edges and corners, and shadowing effects. Texture is another property which can be used to determine the relative orientation of a surface [Sag06].

2.3.4 Texture Synthesis

Computer graphics applications often use textures to render synthetic images. These textures can be obtained from a variety of sources such as hand-drawn pictures or scanned photographs. Hand-drawn pictures can be aesthetically pleasing, but it is hard to make them photo-realistic. Most scanned images, however, are of inadequate size and can lead to visible seams or repetition if they are directly used for texture mapping.

Texture synthesis is an alternative way to create textures. Because synthetic textures can be made any size, visual repetition is avoided. Texture synthesis can also produce tileable images by properly handling the boundary conditions.

The goal of texture synthesis can be stated as follows: Given a texture sample, synthesize a new texture that, when perceived by a human observer, appears to be generated by the same underlying process. Figure (2.5) Show an example of texture synthesis [Wei01].



(a) Texture

(b) Synthesis result

Figure (2.5) Problem Formulations: Given a sample texture (a), our goal is to synthesize a new texture that looks like the input (b). The synthesized texture is tileable can be of arbitrary size specified by the user.

2.4 Feature Selection and Extraction

Any pattern which can be classified in some category must possess a number of features. The first step in the process of classification is to consider the problem, what features to select and how to extract (measure) them [Fri99].

A judicious selection of features for building classifiers is a very crucial aspect of classifier design, and deserves careful consideration. On one hand, there is certainly nothing to lose in using all available measurements in classifier design. On the other hand, too many features make the classifier increasingly complex (sometimes confusing too), in fact, unnecessarily so, in case some of the measurements are redundant.

Feature selection, is essentially the selection of the subset of measurements that optimizes some criterion of separability of classes, since, intuitively, the best set of features should discriminate most efficiently among the classes, that is, enhance the separability among them, while increasing homogeneity within classes at the same time.

Feature extraction, aims to reduce the number of measurements available in a different way by looking for a transformation of the original vector of measurements that optimizes some appropriately defined criterion of separability among classes, possibly leading to fewer features at the same time.

There are various methods of extracting texture features from images, some of these methods are: statistical, structural (or syntactic), and spectral [Fri99, Pal01].

2.4.1 Spectral Approach

The spectral approach to texture analysis deals with images in the frequency domain. Therefore, this approach requires Fourier transform to be carried out on the original images to acquire their corresponding representations in the frequency space.

The two-dimensional power spectrum of an image reveals much about the periodicity and directionality of its texture. For instance, an image of coarse texture would have a tendency towards low frequency components in its power spectrum, whereas another image with finer texture would have higher frequency components.

Fourier transform based methods usually perform well on textures showing strong periodicity, however their performance deteriorates as the periodicity of textures weakens. [Kon02].

Given such performance problems and the high computational complexity of the Fourier transform, the spectral approach is neither a very popular approach among researchers dealing with texture analysis, nor seems to be promising. In fact, Haralick, to whom the early classification of approaches in textual analysis is owed, does not even mention the spectral approach, but sticks to the classification of all methods among the two other approaches: structural and statistical [Kon02].

2.4.2 Structural Approach

Structural approaches (Haralick 1979, Levine 1985) represent texture by well defined primitives (microtexture) and a hierarchy of spatial arrangements (macrotexture) of those primitives. To describe the texture, one must define the primitives and the placement rules. The choice of a primitive (from a set of primitives) and the probability of the chosen primitive to be placed at a particular location can be a function of location or the primitives near the location. The advantage of the structural approach is that it provides a good symbolic description of the image; however, this feature is more useful for synthesis than analysis tasks. The abstract descriptions can be ill defined for natural textures because of the variability of both micro- and macrostructure and no clear distinction between them. It may prove to be useful for bone image analysis, e.g. for the detection of changes in bone microstructure [Mat98].

2.4.3 Statistical Approach

From the statistical point of view, an image is a complicated pattern on which statistics can be obtained to characterize these patterns. The techniques used within the family of statistical approaches make use of the intensity values of each pixel in an image, and apply various statistical formulae to the pixels in order to calculate feature descriptors.

Texture feature descriptors, extracted through the use of statistical methods, can be classified into two categories according to the order of the statistical function that is utilized: First-Order Texture Features and Second Order Texture Features [Tuc98, Kon02].

- First Order Texture Features are extracted exclusively from the information provided by the intensity histograms, thus yield no information about the locations of the pixels. Another term used for First-Order Texture Features is Grey Level Distribution Moments.
- Second-Order Texture Features take the specific position of a pixel relative to another into account. The most popularly used of second-order methods is the co-occurrence matrix method [Kon02, Tuc98].

2.4.3.1 Co-occurrence matrices

The feature set of Haralick is probably one of the most famous methods of texture analysis. It is based on the calculation of the co-occurrence matrix, a second-order statistics of the gray levels in the image window [Jäh99]**.**

The co-occurrence matrix method of texture description is based on the repeated occurrence of some gray-level configuration in the texture; this configuration varies rapidly with distance in fine textures and slowly in coarse textures. Suppose the part of a textured image to be analyzed is an M×N rectangular window. An occurrence of some gray-level configuration may be described by a matrix of relative frequencies $P_{q,d}(i,j)$, describing how frequently two pixels with gray-levels *i,j* appear in the window separated by a distance *d* in direction *θ*. These matrices are symmetric if defined as given in equations (2,1) to (2,4). However, an asymmetric definition may be used, where matrix values are also dependent on the direction of co-occurrence.

Non-normalized frequencies f of co-occurrence as functions of angle θ and distance d can be represented formally as [Son08]:

$$P_{0°,d}(i,j) = |\{[(k,l),(m,n)] \in D : k - m = 0, |l - n| = d, f(k,l) = i, f(m,n) = j\}| \quad \text{.....................} (2.1)$$

$$P_{45°,d}(i,j) = |\{[(k,l),(m,n)] \in D : (k - m = d, l - n = -d) \vee (k - m = -d, l - n = d), f(k,l) = i, f(m,n) = j\}| \quad \text{.....................} (2.2)$$

$$P_{90°,d}(i,j) = |\{[(k,l),(m,n)] \in D : |k - m| = d, l - n = 0, f(k,l) = i, f(m,n) = j\}| \quad \text{.....................} (2.3)$$

$$P_{135°,d}(i,j) = |\{[(k,l),(m,n)] \in D : (k - m = d, l - n = d) \vee (k - m = -d, l - n = -d), f(k,l) = i, f(m,n) = j\}| \quad \text{.....................} (2.4)$$

where $|\{\ldots\}|$ refers to set cardinality and $D=(M \times N) \times (M \times N)$

While a normalized co-occurrence matrix $P_{\theta,d}(i,j)$ is calculated from [Jäh99]:

$$P(i,j) = P_{\theta,d}(i,j) / M \times N \quad \text{.....................} (2.5)$$

where $M \times N$: total number of possible pairs

A set of texture features can be computed from Co-occurrence matrix, these descriptors include [Jäh99, Sam99, Lim04, Dua06, and, Son08]:

1. Energy (Angular second momentum): It describes the uniformity of the texture. When all the matrix elements are almost equal, i.e., when gray level intensities are very close to each other, the value of the energy is small. Thus, the higher the value of the energy, the more irregular the matrix.

$$\text{Energy} = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i, j) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \ (2.6)$$

2. Contrast (Inertia): When the high values of the matrix are further away from the main diagonal, the value of inertia becomes higher. So inertia and the inverse difference moment are measures for the distribution of gray-scales in the image.

$$\text{Contrast} = \sum_{n=0}^{G-1} n^2 \left(\sum_{i=1}^{G-1} \sum_{\substack{j=1 \\ |i-j|=n}}^{G-1} P(i, j) \right) \quad \ldots\ldots\ldots\ldots\ldots\ldots \ (2.7)$$

3. Correlation: It measures the correlation between the elements of the matrix. When correlation is high the image will be more complex than when correlation is low. Haralick's correlation is a measure of gray level linear dependence between the pixels at the specified positions relative to each other.

$$\text{Correlation} = \frac{1}{s_x s_y} \left(\sum_{i=1}^{G-1} \sum_{j=1}^{G-1} (ij) P(i, j) - m_x m_y \right) \quad \ldots\ldots\ldots \ (2.8)$$

Where μ_x, μ_y, σ_x and σ_y are the means and standard deviations of $P_x(i) = \sum_k P(i,k)$ and $P_y(j) = \sum_k P(k, j)$ [Kon02].

4. Variance (Sum of squares): Inform on how spread out the distribution of gray levels is. The variance is expected to be large if the gray levels of the image are spread out vastly.

$$\text{Variance} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-m)^2 P(i,j) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.9)$$

where µ is the mean of the density function $P(i, j)$.

5. Inverse difference moment (InvDifMom): It has a relatively high value when the high values of the matrix are near the main diagonal. This is because the squared difference $(i\text{-}j)^2$ is smaller near the main diagonal, which increases the value of $1/(1+(i-j)^2)$.

$$\text{InvDifMom} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\frac{1}{1+(i-j)^2}P(i,j) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.10)$$

6. Entropy: It measures the randomness of the elements in the matrix. When all elements of the matrix are maximally random, entropy has its highest value. So, a homogeneous image has lower entropy than an inhomogeneous image. In fact, when energy gets higher, entropy should get lower. Entropy has its highest peak when the GLCM is uniform.

$$\text{Entropy} = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1}P(i,j)\log\{P(i,j)\} \quad \ldots\ldots\ldots\ldots \quad (2.11)$$

7. Sum average (SumAvg)

$$\text{SumAvg} = \sum_{i=2}^{2G} i\, P_{(x+y)}(i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.12)$$

where

$$P_{x+y}(i) = \sum_{j,k;\, j+k=i} P(j,k)$$

8. Sum Entropy (SumEnt)

$$\text{SumEnt} = -\sum_{i=2}^{2G} P_{(x+y)}(i) \log\{P_{(x+y)}(i)\} \quad \text{……… (2.13)}$$

9. Sum Variance (SumVar)

$$\text{SumVar} = \sum_{i=2}^{2G} \left(i - \sum_{i=2}^{2G} i\, P_{(x+y)}(i) \right)^2 P_{(x+y)}(i) \quad \text{………. (2.14)}$$

10. Difference variance (DifVar)

$$\text{DifVar} = \sum_{i=0}^{G-1} \left(P_{(x-y)}(i) \left(i - \sum_{j=0}^{G-1} j\, P_{(x-y)}(j) \right)^2 \right) \text{……….. (2.15)}$$

11. Difference entropy (DifEnt)

$$\text{DifEnt} = -\sum_{i=0}^{G-1} P_{(x-y)}(i) \log\{P_{(x-y)}(i)\} \quad \text{……..……. (2.16)}$$

12. Information measures of correlation (InfMeasCor)

$$\text{InfMeasCor1} = \frac{HXY - HXY1}{\max(HX, HY)} \quad \text{………………. (2.17)}$$

$$\text{InfMeasCor2} = \sqrt{1 - e^{-2(HXY2 - HXY)}} \quad \text{………………. (2.18)}$$

where

$$HX = -\sum_{i=0}^{G-1} P_x(i) \log(P_x(i)), \quad HY = -\sum_{j=0}^{G-1} P_y(j) \log(P_y(j))$$

$$HXY = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} P(i,j)\log\left(P(i,j)\right)$$

$$HXY1 = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} P(i,j)\log\left(P_x(i)P_y(j)\right)$$

$$HXY2 = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} P_x(i)P_y(j)\log\left(P_x(i)P_y(j)\right)$$

13. Maximum probability (MaxProb): Result is retrieved from maximum value in the pixel pair that is most predominant in the image. The maximum probability is expected to be high if the occurrence of the most predominant pixel pair is high.

$$\text{MaxProb} = \max_{i,j}\left(P(i,j)\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.19)$$

14. Difference moment (DifMom)

$$\text{DifMom} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\left|(i-j)\right|^{k} P(i,j) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.20)$$

15. Homogeneity (Homog): The homogeneity is expected to be large if the gray levels of each pixel pair are similar.

$$\text{Homog} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\frac{P(i,j)}{1+\left|i-j\right|} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.21)$$

16. Cluster shade (ClusterShade) and Cluster prominence (ClustePropm): They measure the skewness of the matrix, in other words the lack of symmetry. When cluster shade and cluster prominence are high, the image is not symmetric. In addition, when cluster prominence is low, there is a peak in the co-occurrence matrix around the mean values. For the image, this means that there is little variation in gray-scales.

$$\text{ClusterShade} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\left(i + j - m_x - m_y\right)^3 P\left(i, j\right) \quad \ldots\ldots\ldots (2.22)$$

$$\text{ClusterProm} = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\left(i + j - m_x - m_y\right)^4 P(i, j) \quad \ldots\ldots\ldots (2.23)$$

2.4.3.2 Run Length matrices (Primitive Length matrices)

The gray level run length method is a way of extracting higher-order statistical texture features. The technique has been described and applied by Galloway at 1975 and by Chu et al at 1990, which calculates characteristic textural features from gray-level run lengths in different image directions [Alb95, Jäh99].

A large number of neighboring pixels of the same gray-level represents a coarse texture; a small number of these pixels represent a fine texture, and the lengths of texture primitives (run) in different directions can serve as a texture description. A primitive is a maximum contiguous set of constant-gray-level pixels located in a line; these can then be described by gray-level, length, and direction. The texture description features can be based on computation of continuous probabilities of the length and the gray-level of primitives in the texture [son08].

The basis of the calculation of the features is a run-length matrix that is defined as:

$$P(g, r) = \left(a_{g,r}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.24)$$

Where $a_{g,r}$ is the number of occurrences of a connected pixel interval of run length r in the direction θ with all pixel values of the interval being equal to the gray-level value g.

Usually, four run-length matrices for the directions $\theta = 0°; 45°; 90°;$ and 135° are calculated. In order to get sufficiently high run-length values, a reduction of the gray values of an image is performed.

The texture description features can be determined as follows [Jäh99, The03]:

1. Short Run Emphasis (SRE): This feature emphasizes small run lengths, due to the division by r^2.

$$SRE = \frac{1}{N} \sum_{g=1}^{G} \sum_{r=1}^{R} \frac{P(g,r)}{r^2} \quad \text{.........................} \quad (2.25)$$

2. Long Run Emphasis (LRE): This gives emphasis to long run lengths. Thus, we expect SRE to be large for coarser and LRE to be large for smoother images.

$$LRE = \frac{1}{N} \sum_{g=1}^{G} \sum_{r=1}^{R} r^2 P(g,r) \quad \text{.........................} \quad (2.26)$$

3. Gray Level Distribution (GLD): When runs are uniformly distributed among the gray levels, GLD takes small values.

$$GLD = \frac{1}{N} \sum_{g=1}^{G} \left(\sum_{r=1}^{R} P(g,r) \right)^2 \quad \text{.........................} \quad (2.27)$$

4. Run Length Distribution (RLD): Is a measure of run length nonuniformity.

$$\mathrm{RLD}=\frac{1}{N}\sum_{r=1}^{R}\left(\sum_{g=1}^{G}P(g,r)\right)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.28)}$$

5. Run Percentage (RP): where N_{pix} is the total possible number of runs in the image, if all runs had length equal to one, that is, the total number of pixels. RP takes low values for smooth images.

$$\mathrm{RP}=\frac{1}{N_{pix}}\sum_{g=1}^{G}\sum_{r=1}^{R}P(g,r) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.29)}$$

6. Low Gray-level Run Emphasis (LGRE): The LGRE is expected large for the image with low gray level values.

$$\mathrm{LGRE}=\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}\frac{P(g,r)}{g^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.30)}$$

7. High Gray-level Run Emphasis (HGRE): The HGRE is expected large for the image with high gray level values.

$$\mathrm{HGRE}=\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}P(g,r)g^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.31)}$$

8. Short Run Low Gray-level Emphasis (SRLGE): Measures the joint distribution of short runs and low gray level values. The SRLGE is expected large for the image with many short runs and lower gray level values.

SRLGE = $$\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}\frac{P(g,r)}{g^2 r^2}$$ …………………….. (2.32)

9. Short Run High Gray-level Emphasis (SRHGE): Measures the joint distribution of short runs and High gray level values. The SRHGE is expected large for the image with many short runs and high gray level values.

SRHGE = $$\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}\frac{P(g,r)g^2}{r^2}$$ ……………………. (2.33)

10. Long Run Low Gray-level Emphasis (LRLGE): Measures the joint distribution of long runs and low gray level values. The LRLGE is expected large for the image with many long runs and low gray level values.

LRLGE = $$\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}\frac{P(g,r)r^2}{g^2}$$ …………………….. (2.34)

11. Long Run High Gray-level Emphasis (LRHGE): Measures the joint distribution of long runs and high gray level values. The LRHGE is expected large for the image with many long runs and high gray level values.

LRHGE = $$\frac{1}{N}\sum_{g=1}^{G}\sum_{r=1}^{R}P(g,r)g^2 r^2$$ ……………..….. (2.35)

where:

N: number of pixel in image window

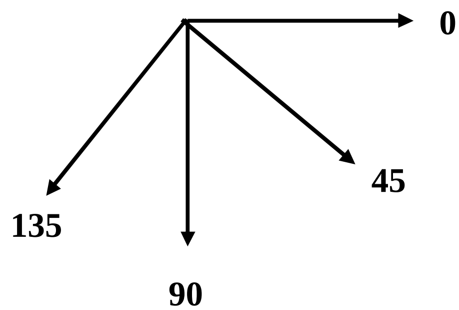
G: number of gray level in image

R: maximum run length in image

N_{pix} : total possible number of runs in the image

Example: Consider a 5x5 sub-image with gray level ranges from 0-3

- Find co-occurrence matrix.
- Find run length matrix.



0	1	1	2	2
1	1	1	3	3
3	3	3	3	3
0	0	0	2	2
1	1	2	2	2

Sub-image

- Co-occurrence matrix when d=1

$$P(0°,1) = \frac{1}{20}\begin{pmatrix} 2 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 1 & 2 & 4 & 0 \\ 0 & 1 & 0 & 5 \end{pmatrix}, \quad P(45°,1) = \frac{1}{16}\begin{pmatrix} 0 & 0 & 0 & 2 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 2 \\ 0 & 4 & 1 & 1 \end{pmatrix}$$

$$P(90°,1) = \frac{1}{20}\begin{pmatrix} 0 & 0 & 0 & 3 \\ 3 & 2 & 0 & 0 \\ 1 & 0 & 2 & 2 \\ 0 & 3 & 2 & 2 \end{pmatrix}, \quad P(135°,1) = \frac{1}{16}\begin{pmatrix} 0 & 0 & 0 & 3 \\ 2 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 2 & 1 & 2 \end{pmatrix}$$

- Run length matrix

$$P(0°)=\begin{pmatrix}1&0&1&0&0\\0&2&1&0&0\\0&2&1&0&0\\0&1&0&0&1\end{pmatrix}, \qquad P(45°)=\begin{pmatrix}4&0&0&0&0\\5&1&0&0&0\\5&1&0&0&0\\5&1&0&0&0\end{pmatrix}$$

$$P(90°)=\begin{pmatrix}4&0&0&0&0\\3&2&0&0&0\\3&2&0&0&0\\3&2&0&0&0\end{pmatrix}, \qquad P(135°)=\begin{pmatrix}4&0&0&0&0\\3&2&0&0&0\\3&2&0&0&0\\3&2&0&0&0\end{pmatrix}$$

2.5 Classification using Fuzzy Logic

Image classification is an area where fuzzy representation and fuzzy reasoning can be successfully applied, mainly for two reasons: (1) ambiguity in the images to be recognized; and (2) the need for fast processing, that is, complicated formulas may not be applicable for a real-time recognition; in this case a fuzzy system may be more convenient.

Different approaches are possible depending on the image recognition tasks, two of them being (1) objects recognition, that is, recognizing shape, distance, and location of objects; and (2) texture analysis, for example, an image X of size m´n pixels can be represented as a set of fuzzy sets and membership degrees to which pixels belong to the fuzzy concepts, such as "brightness," "darkness," "edginess," "smoothness" [Kas96].

2.6 Fuzzy Logic Concepts

The mathematical logic is called classical logic. The classical logic considers the binary logic which consists of truth and false. The fuzzy logic is a generalization of the classical logic and deals with the ambiguity in the logic [Lee05].

Fuzzy logic is relatively young theory. Major advantage of this theory is that it allows the natural description, in linguistic terms, of problems that should be solved rather than in terms of relationships between precise numerical values. This advantage, dealing with the complicated systems in simple way, is the main reason why fuzzy logic theory is widely applied in technique. It is also possible to classify the remotely sensed image (as well as any other digital imagery); in such a way that certain land cover classes are clearly represented in the resulting image [Ned01].

Fuzzy image processing is a kind of nonlinear image processing. The difference to other well-known methodologies is that fuzzy techniques operate on membership values. The image fuzzification (generation of suitable membership values) is, therefore, the first processing step.

Generally, three various types of image fuzzification can be distinguished: histogram-based gray-level fuzzification, local neighborhood fuzzification, and feature fuzzification [Jäh99].

Some of the main characteristics of the fuzzy systems are [Kas96]:

- Fuzzy concepts have to have linguistic meaning; they need to be articulated.
- Membership functions are numerical representations of the linguistic concepts; they can be built either through learning from data, or through experts' opinion, or through both.
- Fuzzy rules can represent vague, ambiguous or contradictory knowledge.
- Fuzzy systems are robust; even if some rules are removed from the rule map, the system could still work properly; fuzzy systems are also robust toward changing conditions in the environment.
- Fuzzy systems are simple to build, easy to realize, easy to explain.

- Fuzzy logic is easy to implement using both software on existing microprocessors or dedicated hardware.

2.7 Fuzzy Set Concepts

Fuzzy set theory, introduced by Lotfi A. Zadeh in 1965, is a generalization of crisp set theory. Fuzzy sets are the tools which convert the concepts of fuzzy logic into algorithms leading to applications. They express precisely what one means by vague expressions [Ibr04].

A fuzzy set consists of objects and their respective grades of membership in the set. The grade of membership of an object in the fuzzy set is given by a subjectively defined membership function. The value of the grade of membership of an object can range from 0 to 1 where the value of 1 denotes full membership, and the closer the value is to 0, the weaker is the object's membership in the fuzzy set [Fri99, Jäh99]..

The traditional way of representing elements *u* of a set A is through the characteristic function:

$\mu_A(u) = 1$, if *u* is an element of the set A, and

$\mu_A(u) = 0$, if *u* is not an element of the set A,

that is, an object either belongs or does not belong to a given set.

In fuzzy sets an object can belong to a set partially. The degree of membership is defined through a generalized characteristic function called membership function:

$\mu_A(u): U \rightarrow [0,1]$

where U is called the universe, and A is a fuzzy subset of U [Kas96].

2.8 Types of Fuzzy Numbers

Fuzzy sets can also be defined by assigning a continuous function to describe the membership either analytically or graphically [Ibr04]. Some commonly used membership functions are illustrated below.

2.8.1 Triangular Fuzzy Number [Lee05]

Among the various shapes of fuzzy number, triangular fuzzy number (TFN) is the most popular one.

Triangular Fuzzy Number It is a fuzzy number represented with three points as follows: A = (a1, a2, a3).

This membership function of this fuzzy number will be interpreted in figure (2.6).

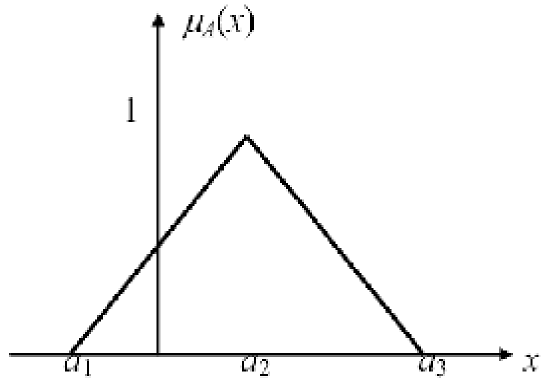


Figure (2.6): Triangular fuzzy number A = (a1, a2, a3).

$$\mu_{(A)}(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases} \quad \text{…………. (2.36)}$$

Some important properties of operations on triangular fuzzy number are summarized:

- The results from Addition or Subtraction between triangular fuzzy numbers result also triangular fuzzy numbers.
- The results from Multiplication or Division are not triangular fuzzy numbers.
- Max or Min operation does not give triangular fuzzy number.

But we often assume that the operational results of Multiplication or Division to be triangular fuzzy numbers as approximation values.

2.8.2 Trapezoidal Fuzzy Number [Lee05]

Another shape of fuzzy number is trapezoidal fuzzy number. This shape is originated from the fact that there are several points whose membership degree is maximum ($\alpha = 1$) for all $\alpha \in [0,1]$.

Trapezoidal fuzzy number we can define trapezoidal fuzzy number A as A = (a1, a2, a3, a4).

The membership function of this fuzzy number will be interpreted figure (2.7).

when a2 = a3, the trapezoidal fuzzy number coincides with triangular one.

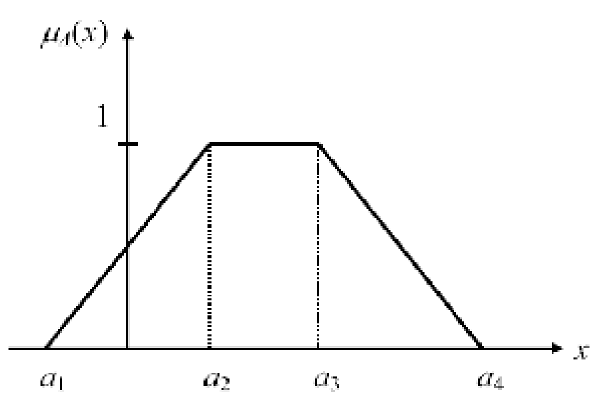


Figure (2.7): Trapezoidal fuzzy number A = (a1, a2, a3, a4).

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \le x \le a_2 \\ 1, & a_2 \le x \le a_3 \\ \dfrac{a_4 - x}{a_4}, & a_3 \le x \le a_4 \\ 0, & x > a_4 \end{cases} \qquad \text{……….} \ (2.37)$$

2.8.3 Gaussian Membership Function [Ibr04, Lee05]

Gaussian or bell shape membership function is often used in practical applications, and its membership function is defined as follows:

$$m_A(x) = \exp\left(\frac{-(x-b)^2}{2s^2}\right), \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.55)$$

The typical shape of this membership function is shown in Figure (2.9).

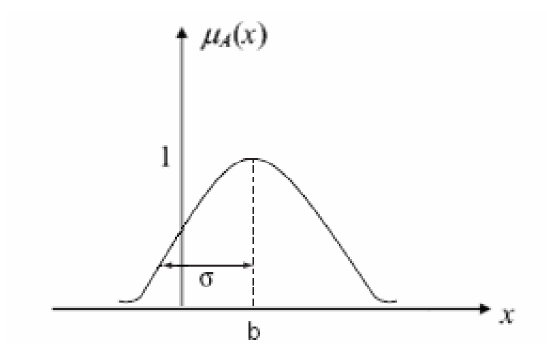


Figure (2.9) The bell shape membership function [Ibr04]

Chapter Three

Fuzzy-Based Texture Classification System Implementation

Chapter Three

Fuzzy-Based Texture Classification System Implementation

3.1 Introduction

This chapter is devoted to describe Fuzzy-Based Texture Classification System (FTCS) implementation to offer the facilities, which may be required to perform the classification process by using fuzzy logic.

The fundamental idea of this work is to implement a program which takes texture image as input and produces image class as output. This task is accomplished by using supervised classification, and statistical approaches have been used to extract texture features of images using co-occurrence and run length matrices. Fuzzy logic is used for computing membership values for all extracted features using triangular and trapezoidal membership functions. Then a combination between features is used to select the best features which satisfy the best selection rate and also using fuzzy logic with combination for comparing and producing the nearest classes to test texture images from training texture images.

3.2 FTCS Structure

Texture classification assigns a given texture to some texture classes. Supervised classification is providing an example for each texture class as a training set. There are two phases to do classification process:

- Learning phase (offline phase), the target is to build a model for the texture content of each texture class presented in the training data, which generally comprises of images with known class labels. The texture content of the training images is captured with the chosen texture analysis method, which yields a set of textural features for each image. These features, could be scalar numbers, discrete histograms or empirical distributions, characterize a given textural properties of the images, such as spatial structure, contrast, roughness, orientation, etc. Figure (3.1) shows the basic modules for learning phase diagram.

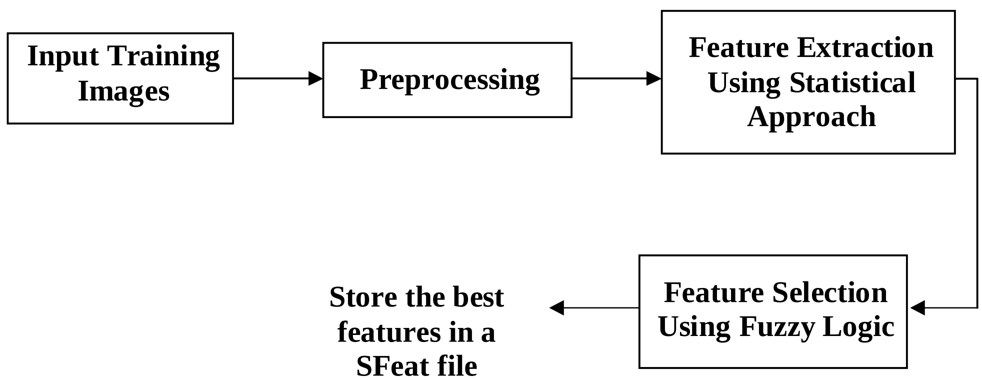


Figure (3.1): Learning Phase.

- Classification phase (online phase), the texture content of the unknown image is first described with the same texture analysis method applied in the learning phase. Then the textural features of the image are compared with those of the training images using classification algorithm and the image is assigned to the category with the best match. Figure (3.2) shows the basic module for classification phase diagram.

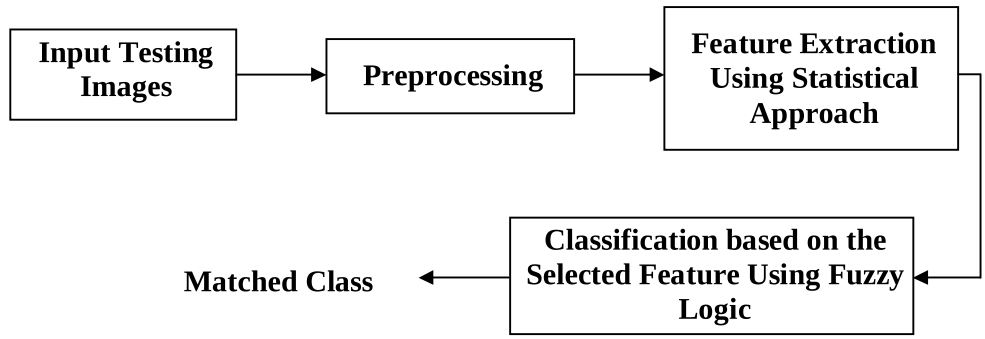


Figure (3.2): Classification Phase.

3.3 Image Preprocessing

The input to this step is an image of bitmap (BMP) type (24 bit/pixel) and the output are quantized gray samples. This stage contains a set of modules. Figure(3.3) shows the basic modules for preprocessing step.

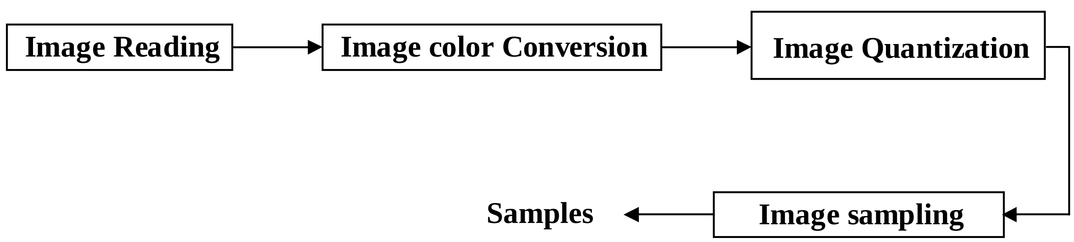


Figure (3.3): Preprocessing Stage.

Scanners are capable of producing image representation in a variety of formats. One of the most popular of these formats is the bitmap (BMP) format. The bitmap bits are the set of bits define the image. In the 2-color, 16-color, and 256-color BMP formats, BMP files consist of three parts. These three parts are header (provides essential information about the image such as image-width, image-height, number of bit/pixel, and a pointer to the beginning of the image-data), color palette (represent the intensities in Red, Green, and Blue (RGB)), and image-data (each entry in the bitmap is an index to the color table). In 16.7 million-color bitmap, where no color table, each entry in image-data is directly specifies a color;

the 3-bytes in each 24-bit entry specify the pixel colors red, green and blue component.

The BMP bits that represent a single line are stored in left-to-right, the same way that the pixels they represent line up on the screen. The first row pixel data in the bitmap responds to the bottom row of pixels on the screen, the second row corresponds the row of pixels second from the bottom, and so on.

- Image Reading module: Extracting required information from image file *(image-width, image-height, and the image-data)*. In this module the color image will split into Red, Green, and Blue components as illustrated in algorithm(3.1), and pass these components and these information to the next modules.

Algorithm (3.1): Read BMP Image

Input:

Img_data // the image file name

Output:

Wid , Higt //The image's width and height
Red(0 to Wid-1, 0 to Higt-1)// Red component of image
Grn(0 to Wid-1, 0 to Higt-1)// Green component of image
Blu (0 to Wid-1, 0 to Higt-1)// Blue components of image

Goal:

Read 24 bit/pixel BMP image file

Step1:

Get from Img_data the Bmph //Bmph is contain the BMP header information
Get from Bmph the Width and Height for the image
Set Wid← Bmph. Width
Set Higt← Bmph. Height

Continue

```
Step2:
        Check if the input image has pixel resolution 24
        If  Bmph. BitPlane = 24 Then
            Set W← Wid* Bmph.BitPlane+31 mod 32
            Img(W–1) // Img contain the raw image's data
            For X=0 to Wid–1
                Img(W–1) ← Get from Img_data file ,
                For Y=0 to Higt–1
                    Set Red(X,Y) ← Img(3*X)
                    Set Grn(X,Y) ← Img(3*X+1)
                    Set Blu(X,Y) ← Img(3*X+2)
                End loop Y
            End loop X
        Else
          Displays massage "The Selected Image's Bitplane is not 24 "
        End if
Step3:
        Return (Red, Grn, Blu, Wid, Higt)
```

- Image Color Conversion module: This module concerned with converting the color image of 24 bit/pixel to gray image of 256 color, as illustrated in algorithm(3.2), and pass the gray image to the next modules.

<u>Algorithm (3.2): Convert BMP Image to Gray Image</u>

<u>Input:</u>

```
        Wid , Higt
        Red (0 to Wid-1, 0 to Higt-1)
        Grn (0 to Wid-1, 0 to Higt-1)
        Blu (0 to Wid-1, 0 to Higt-1)
```

<u>Output:</u>

```
        Gry (0 to Wid-1, 0 to Higt-1) //Gray component of the image
```

<u>Goal:</u>

Convert the color image to gray image

```
Step1:
        For X=0 to Wid–1
            For Y=0 to Higt–1
                Set Gry(X, Y) ← (Red(X,Y)+Grn(X,Y)+Blu(X,Y))/3
            End loop Y
        End loop X
Step2:
        Return (Gry)
```

- Image Quantization Module: The main drawback in using the Co-occurrence and Run Length matrices is the large memory requirement for storing these matrices. Quantization process overcomes this problem by removing some of the information details by mapping groups of data points to a single point. For this reason image quantization process is adopted in this work. As a first step in the quantization operation a lookup table is established, this table is used to convert each gray pixel value from range (0 to 255) to the range (0 to GLevel-1), where GLevel is the number of quantized levels the image will re-quantized to. This lookup table is established once, so that each pixel is mapped into its corresponding quantization value directly without need to recalculate the same re-quantization equation.

  The re-quantization mapping process was performed in different ways, they are:

  § General Equation: Where assuming the image has maximum value is 255 and minimum value is 0.

$$\text{Lookup(I)} = Integer\left(\frac{(\text{Glevel} - 1) * \text{I}}{255}\right) \qquad \ldots\ldots\ldots\ (3.1)$$

  where I = 0, 1, 2, ……., 255.

  § Using traditional (Min-Max): Where *Min* represents the minimum value in the image and *Max* is the maximum value.

$$\text{Lookup(I)} = Integer\left(\frac{(\text{Glevel} - 1)}{Max - Min} * (\text{I} - \text{Min})\right) \qquad \ldots\ldots\ldots\ (3.2)$$

  § Mean and Stander deviation: Where *Mean* is the Mean value and *STD* is the stander deviation value.

$$Mean = \frac{\sum_{x=0}^{Wid-1} \sum_{y=0}^{Higt-1} Gry(x, y)}{Width * Hight} \qquad \ldots\ldots\ldots\ldots\ (3.3)$$

$$STD = \sqrt{\frac{\sum_{x=0}^{Wid-1} \sum_{y=0}^{Higt-1} (Gry(x,y) - Mean)^2}{Width * Hight}} \quad \text{.......... (3.4)}$$

$$Min = Mean - a * STD \quad \text{........... (3.5)}$$

$$Max = Mean + a * STD \quad \text{........... (3.6)}$$

where α is the ratio of the distance from the Mean in terms of STD, and it has values lay within the range [2, 3].

Then the equation (3.2) will be used to compute lookup table.

§ Histogram equalization: The histogram of an image represents the frequency of occurrence of various gray levels in the image. Histogram equalization is a technique used for adjusting the distribution of the gray scale of an image; such that the new gray level histogram of the mapped image is nearly uniform [Ach05]. The core equation of the applied non-linear quantization based on histogram equalization technique is:

$$\text{Lookup(I)} = (\text{Glevel} - 1) * \text{Pr(I)} \quad \text{........... (3.7)}$$

where, P_r is the accumulated probability of the gray level (I), and it is computed according to the following equation:

$$\text{Pr(I)} = \text{Pr(I} - 1) + \frac{His(\text{I})}{\sum_{J=0}^{255} His(\text{J})} \quad \text{....... (3.8)}$$

Algorithm (3.3) shows the calculation steps for *lookup* table and algorithm (3.4) shows the quantization steps.

Algorithm (3.3): Lookup Table

Input:

MappingType // *Mapping type for quantization*
GLevel // *Gray Level the image will quantized into*
Wid, Higt

Output:

Lookup (0 to 255)

Goal:

Generate lookup table to convert pixels range from 0-255 into 0- GLevel-1

```
Step1:
  Case MappingType
        1: // General Equation
          For I=0 to 255 do
                Set Lookup(I) ← Integer((GLevel-1)×I / 255)
          End Loop I
        2: // Traditional Max and Min Values
          Find Min and Max values in the image
          For I=0 to 255 do
              If  I< Min  Then
                Set Lookup (I) ← 0
              If  I>Max  Then
                Set Lookup (I) ←255
              Else
                Set Lookup(I) ← Integer((Glevel-1)/(Max-Min)) × (I-Min)
          End Loop I
          For I=0 to 255 do
              If  Lookup(I) > Glevel-1 Then
                Set Lookup(I) ←Glevel-1
          End Loop I
        3: // Mean and STD Values
          Compute Mean by using equation (3.3)
          Compute STD by using equation (3.4)
          Compute Min and Max by using Mean and STD
          Set Min←Mean-2×STD
          Set Max ←Mean+2×STD
          If Min < 0 Then
            Min = 0
          If Max > 255 Then
            Max = 255
          For I=0 to 255 do
              If  I< Min  Then
                Set Lookup (I) ← 0
              If  I>Max  Then
                Set Lookup (I) ←255
              Else
                Set Lookup(I) ← Integer((Glevel-1)/(Max-Min)) × (I-Min)
          End Loop I
```

Continue

```
        For I=0 to 255 do
            If  Lookup(I) > Glevel-1 Then
                Set Lookup(I) ←Glevel-1
        End Loop I
      4: // Using Histogram Equalization
          Find histogram for image His(0 to 255)
          Find summation for histogram Sum
          Find Probability for each pixel value Pr (0 to 255)
          Set Pr (0) ← His (0)/Sum
          For I=1 to 255 do
              Set Pr (I) ← His (I)/Sum
              Set Pr (I) ← Pr (I) + Pr (I–1)
              Set Lookup(I) ← (GLevel-1) * Pr (I)
          End Loop I
  End Case
Step2:
        Return (Lookup)
```

Algorithm (3.4): Quantize Gry Image

Input:

Wid, Higt, GLevel
Lookup (0 to 255)
Gry (0 to Wid-1, 0 to Higt-1)

Output:

QuantImg (0 to Wid-1, 0 to Higt-1)

Goal:

Quantize the gray image of 256 level for reduce the dimensionality

```
Step1:
  Generate Lookup table  by Call Algorithm(3.3)
  For X=0 to Wid–1
      For Y=0 to Higt–1
          Set QuantImg(X,Y) ← Lookup(Gry(X,Y))
      End loop Y
  End loop X
Step2:
  Return (QuantImg)
```

- Image Sampling Modules: The quantized image will be divided randomly into small sub-images (of size *SLen´SLen)* to increase the amount of data for each image which increase the discrimination between images, where SLen must be greater than 0 and less than width and height of the image. Algorithm (3.5) shows the basic steps for sampling process.

<u>Algorithm (3.5): Get_Sample</u>

<u>Input:</u>

Wid, Higt, SLen

QuantImg (0 to SLen-1,0 to SLen-1)

<u>Output:</u>

Sample (0 to SLen-1,0 to SLen-1)

<u>Goal:</u>

Get sample from image

```
Step1:
  Randomly select the starting point of the sample by Rnd function
  Set Xs← (Wid–SLen)*Rnd
  Set Ys← (Higt–SLen)*Rnd //Rnd return randomly value between 0 and 1
Step2:
  Set the values of the sample from  the array of QuantImg
    For X=0 to SLen–1
          For Y=0 to SLen–1
              Set Sample(X,Y) ← QuantImg(Xs+X,Ys+Y)
          End loop Y
      End loop X
Step3:
  Return (Sample)
```

3.4 Features Extraction

The goal of image analysis is to extract useful data for solving application based problem. This is done by intelligently reducing the amount of image data with the tools have explored. A feature vector is one method to represent an image, or part of an image object, by finding measurements on a set of features. The statistical features is one of the most important features that is used to evaluate the performance of co-occurrence matrices and run length matrices for solving texture classification problem, thus, statistical features are adopted in this work.

The texture feature extraction is applied for each sample get from gray image. The computation of the statistical features can be summarized by the following two modules:

- Module-1: For each sample the co-occurrence matrix *C* is extracted in a four direction and the total for these directions, the co-occurrence matrix is a two dimensional matrix of joint probability *C(I,J)* between pairs of pixels separated by a distance d in a given direction. Twenty one statistical texture features are calculated depending on the extracted co-occurrence matrix *C*. These 21 statistical texture features which are computed in this work are: Energy, Contrast, Correlation, Variance, InvDifMom, Entropy, SumAvg, SumEnt, SumVar, DifVar, DifEnt, InfMeasCor1, InfMeasCor2, MaxProb, DifMom of order 1, 2, 3, and 4, Homogeneity, ClusterShade, ClusterProm. These statistical features are defined in equations from (2.6) to (2.23). Algorithm (3.6) illustrates the steps to calculate the co-occurrence matrices for a given direction (*theta*), where theta represents the directions 0, 45, 90, and 135. The vectors of features for the four directions and for the average are saved in a dedicated file, called *"CocFeatures"*, to be used later.

Algorithm (3.6): Compute Co_occurrence Matrix

Input:

```
SLen
d // distance between neighbor pixels
theta // direction angle
GLevel
Sample (0 to SLen,0 to SLen)
```

Output:

```
C(0 to GLevel–1, 0 to GLevel–1)
```

Goal:

```
Compute gray-level co-occurrence matrix for each sample
```

Step1:

```
//Calculate the array of the co_occurrence matrix
Reset TotNo
Case theta
      0://Horizontal
        For Y =0 to SLen-1 do
            For X=0 to SLen–d-1 do
```

Continue

```
                Set I ← Sample(X, Y)
                Set J ← Sample(X + d, Y)
                Increment C(I, J) and C(J, I) by 1
                Increment TotNo by 2
            End loop X
        End loop Y
      45://Diagonal
        For Y= 0 to SLen –d-1 do
            For X=0 to SLen –d-1 do
                Set I ← Sample(X, Y)
                Set J ← Sample(X + d, Y + d)
                Increment C(I, J) and C(J, I) by 1
                Increment TotNo by 2
            End loop X
        End loop Y
       90://Vertical
        For Y=0 to SLen – d-1 do
            For X=0 to SLen-1  do
                Set I ← Sample(X, Y)
                Set J ← Sample(X, Y + d)
                Increment C(I, J) and C(J, I) by 1
                Increment TotNo by 2
            End loop X
        End loop Y
      135://Sub–Diagonal
        For Y=0 to SLen –d-1 do
            For X=d to SLen-1 do
                Set I ← Sample(X, Y)
                Set J ← Sample(X – d, Y + d)
                Increment C(I, J) and  C(J, I)  by 1
                Increment TotNo by 2
            End loop X
        End loop Y
End Case

Step2:
   Normalize the co_occurrence matrix to convert it to probability
   For X=0 to GLevel–1
      For Y=0 to GLevel–1
          Set C(X, Y) ← C(X, Y) / TotNo
      End loop Y
   End loop X
Step3:
  Return (C)
```

- Module2: For each sample the run length matrix *RL* is extracted in a four direction and the total for these directions; 11 statistical texture features are calculated depending on the extracted run length matrix *RL*. These 11 statistical texture features which are computed in this work are: SRE, LRE, GLD, RLD, RP, LGRE, HGRE, SRLGE, SRHGE, LRLGE, and LRHGE. These statistical features are defined in equations from (2.25) to (2.35). Algorithm (3.7) illustrates the steps to calculate the run length matrices for a given theta, where theta represents the directions 0, 45, 90, and 135. The vectors of features for the four directions and for the average are saved in a dedicated file, called *"RlFeatures"*, to be used later.

Algorithm (3.7): Compute Run Length Matrix

Input:
SLen
theta // *direction angle*
GLevel
Sample (0 to SLen,0 to SLen)
Output:
RL (1 to SampleLen, 0 to GLevel–1)
Goal:
Compute gray-level run length matrix for each sample

```
Step1:
  //Calculate the array of the run length matrix
  Reset Length
  Case theta
        0://Horizontal
          For Y =0 to SLen-1 do
              For X =0 to SLen-1 do
                 If  Sample(X , Y) =  Sample(X + 1, Y) Then
                    Increment Length by 1
                 Else
                    Increment RL (Length, Sample(X,Y)) by 1
                    Reset Length
                 End If
              End loop X
          End loop Y
```

Continue

```
        45://Diagonal
          For Y =0 to SLen-1 do
              For X=0 to SLen-1 do
                If  Sample(X, Y) = Sample(X + 1, Y + 1) Then
                    Increment Length by 1
                Else
                    Increment RL (Length, Sample(X,Y)) by 1
                    Reset Length
                End If
              End  loop X
          End loop Y
        90://Vertical
          For Y =0 to SLen – 1 do
              For X=0 to SLen-1  do
                If  Sample(X, Y )= Sample(X , Y+1) Then
                    Increment Length by 1
                Else
                    Increment RL (Length, Sample(X,Y)) by 1
                    Reset Length
                End If
              End loop X
          End loop Y
        135://Sub–Diagonal
          For Y=0 to SLen –d do
              For X=d to SLen do
                  If  Sample(X, Y) = Sample(X – 1, Y + 1) Then
                  Increment Length by 1
                Else
                  Increment RL (Length, Sample(X,Y)) by 1
                  Reset Length
                End If
              End loop X
          End loop Y
Step2:
  Return (RL)
```

Algorithm (3.8): Image Features Extraction

Input:
Img_data // Image file
NofSample// Number of sample

Output:
CocFeature file //Store Co_occurrence feature for all blocks
RLFeature file//Store Run Length feature for all blocks

Goal:
Extract Features for all samples and stored in a file

***Step1*:** Read BMP file (Color image of 24 bit/pixel). //Call Algorithm (3.1)

***Step2*:** Convert Color image to Gray image. //Call Algorithm (3.2)

***Step3*:** Apply Quantization method on the Gray image. //Call Algorithm (3.4)

***Step4*:** if NofSample <> 0 then Get_Sample //Call Algorithm (3.5)
Else goto Step11

***Step5*:** Calculate the Co_occurrence matrix for four angles and for the average of these angles and normalize it. In this process five matrices are extracted.//Call Algorithm (3.6)

Step6: Extract features for the five Co_occurrence matrices.//Using Equations from (2.6) to (2.23)

Step7: Calculate the Run Length matrix for four angles and for the average of these angles. In this process five matrices are extracted.//Call Algorithm (3.7)

Step8: Extract features for the five Run Length matrices.//Using Equations from (2.25) to (2.35)

Step9: Store the features extracted from Co_occurrence matrices in the "CocFeature" file and Store the features extracted from Run Length matrices in the "RLFeature" file

Step10: decrement NofSample by 1 and goto Step4

Step 11: Exit

Algorithm (3.8) illustrates the basic steps for image preprocessing and extracting features steps for any image (Training Image or Testing Image).

3.5 Fuzzification Process

Before the fuzzification process is done, extracted features are both normalized and quantized. This step is required to unify the dynamic ranges of the extracted features.

The applied normalization process maps the extracted feature's values to the range [0, 1]. This was performed by finding the actual dynamic range (i.e., the highest and lowest values) [f_{min}, f_{max}] of each feature over all classes, taken into consideration that there are many samples in each class. The normalization of feature (*f*) is performed using the following equation:

$$f_{norm} = \frac{f - f_{\min}}{f_{\max} - f_{\min}}$$

The uniform quantization process is used to map the normalized real values of the features to discrete integer indices, whose values lay within the range [0, Nbin-1], where **Nbin** is the number of quantization bins. This was performed by steps declared in algorithm (3.9)**.** The quantized features are saved in a dedicated file, called *"QFeatures"*, to be used later.

<u>Algorithm (3.9): Normalize and Quantize Texture Features</u>

<u>Input:</u>

Nbin // Represent number of bins in the histogram
NofClass//Represent number of training classes
NofSample//Represent number of samples in each class
NofFeat// Represent number of extracted features

<u>Output:</u>

QFeatures file

<u>Goal:</u>

Normalize and quantize all extracted texture features

Continue

```
Step1:
   Read the Co_occurrence or Run Length features and store in array
   F(0 to NofClass–1,0 to NofSample–1,0 to NofFeat–1)
Step2:
   Find Max and Min Values for each feature for all samples in all classes
   Max (0 to NofFeat–1), Min (0 to NofFeat–1)
   For K = 0 To NofFeat–1
      Set Max (K) ←F (0, 0, K)
      Set Min (K) ←F (0, 0, K)
      For I = 0 To NofClass–1
         For J = 0 To NofSample–1
            If F (I, J, K) < Min (K) Then
              Set Min (K) ←F (I, J, K)
            If F (I, J, K) > Max (K) Then
              Set Max (K) ←F (I, J, K)
         End loop J
      End loop I
   End loop K
Step3:
   Normalize and Quantize features
   QuantFeat (0 to NofClass-1, 0 to NofSample-1, 0 to NofFeat-1)
   For I = 0 To ClassNo–1
     For K = 0 To FeatNo–1
        Set M ←  Max(K) – Min(K)
        For J = 0 To SampleNo–1
           Set F(I, J, K) ← (F(I, J, K) – Min(K))/M //Normalize Feature
           Set QuantFeat(I, J, K) ← Int((Nbin–1) * F(I, J, K)) //Quantize Feature
        End loop J
     End loop K
   End loop I
Step4:
  Save array (QuantFeat) in the file QFeatures
```

Fuzzification is the operation of transforming a crisp set to fuzzy set, so each extracted feature is represented by a membership function. Two types of membership functions have been used, they are: triangular and trapezoidal. The fuzzification module is concerned with finding the parameters of the best triangular or trapezoidal membership function that fits the Probability Density Function (***pdf***) of each feature in each class.

Before the near optimal membership functions are computed, the histogram of each feature must be computed for each class as shown in algorithm (3.10). The corresponding probability density function (***pdf***) is computed from the histogram, and then it is used to find the optimum values of the membership function parameters (i.e. **A, B** and **C** values for the triangular membership function and the **A, B, C** and **D** values for the trapezoidal membership function) by using distance measure "c^2".

The distance measure "c^2" was used to find the minimum distance between pdf bin values and the corresponding membership values. c^2 Distance measure is computed as follows:

$$c^2 = \sum_{i=0}^{NBin-1} \left| Pdf(i) - Membership(i) \right| \quad \text{………… (3.1)}$$

The parameters of membership functions that led the minimum sum of absolute differences (i.e., χ^2) have been considered as the optimal values.

Algorithm (3.11) shows the implemented taken to calculate the parameters of the triangular membership functions; algorithm (3.12) shows how to compute the triangular membership value for a specific feature value, algorithm (3.13) shows the implemented taken to calculate the parameters of the trapezoidal membership functions, and algorithm (3.14) shows how to compute the trapezoidal membership value for a specific feature value.

Algorithm (3.10): Compute Normalize Histogram for Quantize features

Input:
Nbin , NofSample
I // Class number
K // Feature number
QuantFeat(0 to ClassNo–1,0 to SampleNo–1,0 to FeatNo–1)

Output:
NHist file //Normalize histogram values

Goal:
Compute histogram for each feature in each class.

Continue

Step1:
//Find histogram for all samples for Feature J in Class I
Hist (0 to Nbin–1)
For K=0 to NofSample **do**
Increment Hist(QuantFeat (I, J, K)) by 1
End loop K
Step2:
//Find Maximum value in the histogram
Set MaxHist ← the maximum value in the histogram
Step3:
//Normalize Histogram
NormHist (0 to Nbin–1)
For J=0 to Nbin **do**
Set NormHist (J) ←Hist(J)/MaxHist
End loop J
Step4:
Save array (NormHist) in the file NHist

Algorithm (3.11): Compute Best Triangular Parameters
Input:
Nbin, NormHist
Output:
TrianMem file //Contain A, B, C, MinG for each feature in each class
Goal:
Find best triangular A, B and C parameters values for each feature histogram which lead to minimum χ^2.

Step1*:*
Find the best matched triangular membership function for histogram
For A1 = 0 **To** Nbin – 3
For B1 = A1 + 1 **To** Nbin – 2
For C1 = B1 + 1 **To** Nbin – 1
Reset G
For I = 0 **To** Nbin – 1
Call Algorithm(3.12)//to produce membership value(Mem) for I in the A1,B1,C1 curve
Set G ←G + Abs (NormHist(I) – Mem)
End loop I
If G < MinG **Then**
Set MinG ← G, Set A ← A1, Set B ← B1, C ← C1
End loop C1
End loop B1
End loop A1
Step2:
Save in the TrianMem file (A,B,C,MinG)

Algorithm (3.12): Compute Triangular Membership

Input:

I // histogram bin
A1,B1,C1 //boundary for traingular

Output:

Mem

Goal:

Compute triangular membership function.

Step1:

```
//Calculate the triangular membership function
If  I <= A1 Or I >= C1 Then Set Mem ← 0
If A1 < I and I <= B1 Then Set Mem ← (I – A1) / (B1 – A1)
If B1 <= I and I < C1 Then Set Mem ← (C1 – I) / (C1 – B1)
```

Step2:

```
Return (Mem)
```

Algorithm (3.13): Compute Best Trapezoidal Parameters

Input:

Nbin
NormHist(0 to Nbin–1)

Output:

TrapMem file //Contain A, B, C, D, MinG for each feature in each class

Goal:

Find best trapezoidal A, B, C and D parameters values for each feature histogram which lead to minimum χ^2.

Step1:

Find the best matched trapezoidal membership function for histogram

```
For A1 = 0 To Nbin – 4
    For B1 = A1 + 1 To Nbin –3
        For C1 = B1 + 1 To Nbin – 2
            For D1=C1+1 To Nbin – 1
                Reset G
                For I = 0 To Nbin – 1
                    Call Algorithm(3.14)//to produce membership value(Mem)
                                          for I in the A1,B1,C1,D1 curve
                    Set G ←G + Abs (NormHist(I) – Mem)
                End loop I
                If G < MinG Then
                  Set MinG←G
                  Set A← A1, Set B ←B1, Set C← C1, Set D← D1
            End loop D1
        End loop C1
    End loop B1
End loop A1
```

Step2:

```
Save in the TrapMem file(A,B,C,D,MinG)
```

Algorithm (3.14): Compute Trapezoidal Membership

Input:
I //histogram bin
A1,B1,C1,D1

Output:
Mem

Goal:
Compute trapezoidal membership function.

Step1:
Calculate the trapezoidal membership function
If I <= A1 or I >= D1 ***Then*** Set Mem ← 0
If B1 < I and I <= C1 ***Then*** Set Mem ← 1
If A1 <= I and I < B1 ***Then*** Set Mem ← (I – A1) / (B1 – A1)
If C1 < I and I <= D1 ***Then*** Set Mem ← (D1 – I) / (D1 – C1)

3.6 Feature Selection:

Finding a specific features vector that has the best discrimination power has been one of the most important problems in the field of texture analysis and image classification. In practice a larger than necessary number of feature candidates is generated and then the best of them is adopted.

In this work, fuzzy logic is used to select the best features by converting the texture features to fuzzy numbers as illustrated in algorithm (3.15) which compute the membership values for each feature in all classes, and find the best features vector by calculating the success rate for each feature. The computation of success rate is done according to the following criteria: if the extracted feature from a class has highest membership value in that class relative to other classes, then the value of success rate of that feature is incremented by 1, as shown in algorithm (3.16). The steps taken to select the good features are shown in algorithm (3.17) which depends on the values for the success rates.

Algorithm (3.15): Compute Membership for each Quantize Feature in each Class

Input:

NofClass
NofSample
NofFeat

Output:

FMem file//contain the membership for each feature in all classes

Goal:

Calculate membership values for each quantized feature in all classes

Step1:

```
FeatMem (0 to ClassNo-1,0 to SampleNo-1,0 to FeatNo-1,0 to ClassNo-1)
For I = 0 To NofClass-1
    For J = 0 To NofSample-1
        For K = 0 To NofFeat-1
            Read from file "QFeatures" Fq for feature K in sample J in class I
            For I1 = 0 To NofClass-1
                If Select Triangular Fuzzy Number Then
                  Read from TrianMem file A,B,C for Feature K in Class I1
                  Call Algorithm (3.12) //to produce membership value
                                        (FeatMem(I,J,K,I1)) for Fq in the A,B,C
                                        curve
                If Select Trapezoidal Fuzzy Number Then
                  Read from TrapMem file A,B,C,D for Feature K in Class I1
                  Call Algorithm (3.14) //to produce membership value
                                        (FeatMem(I,J,K,I1)) for Fq in the
                                        A,B,C,D curve
            End loop I1
        End loop K
    End loop J
End loop I
```

Step2:

```
Save in the FMem file the array (FeatMem)
```

<u>Algorithm (3.16): Compute Success Rate for each Feature</u>

<u>Input:</u>

NofClass
NofSample
NofFeat

<u>Output:</u>

TSuccRate file// Success Rate for each feature

<u>Goal:</u>

Compute success rate for each feature

Step1:

Read from the file membership for each feature in all classes
FeatMem(0 to ClassNo-1,0 to SampleNo-1,0 to FeatNo-1,0 to ClassNo-1)
Set FeatMem ← Read from FMem file

Step2:

```
compute success rate for each feature in each class
SuccessRate(0 to FeatNo-1, 0 to ClassNo-1)
For I = 0 To NofClass-1
    For J = 0 To NofSample-1
        For K = 0 To NofFeat-1
            Set MaxMemFunc ← FeatMem(I, J, K, 0)
            Set Index ← 0
            For I1 = 1 To NofClass-1
                If MaxMemFunc < FeatMem(I, J, K, I1) Then
                  Set MaxMemFunc ←FeatMem(I, J, K, I1)
                  Set Index = I1
                End If
            End I1
              If Index = I Then
                 Increment SuccessRate(K, I) by 1
        End loop K
    End loop J
End loop I
```

Step3:

```
Compute Total Success Rate for each feature in all Classes
TotalSuccRate(0 to FeatNo-1)
For I = 0 To NofClass-1
    For K = 0 To NofFeat-1
        Set TotalSuccRate(K) ←TotalSuccRate(K) + SuccessRate(K, I)
    End K
End I
```

Step4:

save in the file the success rate for each feature
Save in the file TSuccRate the array(TotalSuccRate)

Algorithm (3.17): Feature Selection

Input:

Nbin
NofClass
NofSample
NofFeat
Feature Name //either co_occurrence or run length feature

Output:

SFeat file//store the best features in this file

Goal:

Select the best features

Step1: //either read the co_occurrence feature or run length feature
If Feature = "Co_occurrence" then goto Step2 Else goto Step3

Step2: **//**Find Normalize and Quantize Co_occurrence Features
Normalize and Quantize Texture Features (CocFeature) *Call* Algorithm(3.9)
goto Step4

Step3: **//**Find Normalize and Quantize Run Length Features
Normalize and Quantize Texture Features (RLFeature) *Call* Algorithm(3.9)

Step4: //Compute Normalize Histogram for Quantize features
Read from file "QFeatures" the features and find histogram for each feature in each class then normalize histogram by *Call* Algorithm (3.10)

Step5: //Find the best shape which fit to normalize histogram for all features in each class
Specify Fuzzy Number Shape
If Triangular then
Find the best triangular which fit to normalize histogram
By *Call* Algorithm(3.11)
If Trapezoidal then
Find the best trapezoidal which fit to normalize histogram
By *Call* Algorithm(3.13)

Step6: // Compute Membership for each Quantize features in each class
Call Algorithm(3.15) to compute membeaship values

Step5: **//**Compute Success Rate for each feature in all classes
Call Algorithm(3.16) which produce TotalSuuRate.

Step6: //Select the best feature depend on the value of TotalSuccRate
Search for the features which has high TotalSuccRate values and select it

Step7:
Store the best features (selected) in the SFeat file

Step8: **Exit**

3.7 Classification:

In this process, the classifier is trained to determine the class for each input image based on the obtained measures of the selected features. In this case, a classifier is a function which takes the selected features as input and texture classes as output by using fuzzy logic. To find the match class, first the features is extracted for tested image, compute membership values for each feature in each training class (i.e. using the same parameters for training classes (A, B, C for triangular and A, B, C, D for trapezoidal)), as illustrated in algorithm (3.18), then search for the class which achieve higher membership value for each selected feature and increase the success rates for that class by 1, as illustrated in algorithm (3.19), finally search for the match class which has high success rate. The whole steps for classification process illustrated in algorithm (3.20).

Algorithm (3.18): Compute Membership for Testing Image

Input:

NofClass
NofSample
NofFeat

Output:

FMemT file // store the membership for test class in each training class

Goal:

Compute Membership for Testing Image in each Training Class

Step1:

```
FeatMem (0 to SampleNo-1,0 to FeatNo-1,0 to ClassNo-1)
For J=0 to NofSample-1 do
  For K =0 to NofFeat-1 do
    Read from file Fq for Sample J in the testing class
    For I1=0 to NofClass-1 do
      If Select Triangular Fuzzy Number Then
         Read from TrianMem file  A,B,C for Feature K in the Class I1
         Compute membership for each feature Fq in the A,B,C curve by
         Call Algorithm(3.11) and store in (FeatMem(I,J,K,I1))
```

Continue

```
        If Select Trapezoidal Fuzzy Number Then
           Read from file A,B,C,D for Feature K in the Class I1
           Compute membership for each feature Fq in the A,B,C curve by
           Call Algorithm(3.13) and store in (FeatMem(I,J,K,I1))
        End loop I1
     End loop K
   End loop J
Step2:
   Save FeatMem array  in the FMemT file
```

Algorithm (3.19): Find the nearest Class

Input:

NofClass
NofSample
NofSFeat//Number of selected features

Output:

ClassType
SFeat(0 to NofSFeat-1)//array of selected Features

Goal:

Search for the class which has high success rate which represent the class type for the test image.

```
Step1:
  SuccessRate(0 to NofSFeat -1, 0 to NofClass-1)
  FeatMem(0 to NofSample-1,0 to NofSFeat -1,0 to NofClass-1)
  Increment the content of SuccessRate (K,M) by 1 if feature K has the highest
  membership in the Class M
  FeatMem← Get from FMemT file
  NofSFeat ← Get from SFeat file the number of selected features
  For K = 0 To NofSFeat - 1
      For J = 0 To NofSample-1
          For M = 0 To NofClass-1
              If (M = 0) Or (M > 0 And FeatMem( J, SFeat(K), M) > Max) Then
                 Set Max ←  FeatMem( J, SFeat(K))
                 Set Idx ← M
          End loop M
          Increment SuccessRate(K, Idx) by 1
      End loop J
  End loop K
Step2:
  Compute the success rate for all features to each class
  TotalSuccRate(0 to ClassNo-1)
   For I = 0 To ClassNo-1
      For K = 0 To SFeatNo-1
          Set TotalSuccRate(I) ← TotalSuccRate(I) + SuccRate(K, I)
      End loop K
  End loop I
Step3:
   Return (TotalSuccRate)
```

Algorithm (3.20): Classification Process

Input:

Nbin
ClassNo
SampleNo
FeatNo
Feature Name //either co_occurrence or run length feature

Output:

SFeat file//store the best features in this file

Goal:

Find class type for test image

Step1:
//either read the co_occurrence feature or run length feature
If Feature = "Co_occurrence" then goto Step2 Else goto Step3
Step2:
//Find Normalize and Quantize Co_occurrence Features
Normalize and Quantize Texture Features (CocFeature) and Store in (QFeatTest) file by Call Algorithm(3.9)
goto Step4
Step3:
//Find Normalize and Quantize Run Length Features
Normalize and Quantize Texture Features (RLFeature) and Store in (QFeatTest) file by Call Algorithm(3.9)
Step4:
// Compute Membership for each Quantize features in each class
Call Algorithm(3.18) to compute membeaship values
Step5:
//Compute Success Rate for all features in each classes
Call Algorithm(3.19) which produce TotalSuccRate.
Step6:
//Search for the Class which has the highest TotalSuccRate value toFind which class the test image belong to
Max = TotalSucc(0): ClassType = 0
For I = 1 **To** ClassNo-1
  If TotalSuccRate(I) > Max **Then** Set ClassType ← I
End loop I
Step7:
Return (ClassType)

3.8 Features combinations:

To enhance the performance of the system for selection and classification process, the system will include the combination between two or more features by adding membership values for these features, and then re-determines the success rates for these combined features, and finding the best combined set of features based on success rates to be used in selection and classification process. Various combinations of two, three, and four features have been investigated. Algorithm (3.21) shows the steps taken to find out the success rates for combination between two features. Other (bigger) combinations have been applied in similar way. The same way is applied in classification by comparing the combined features to find the class type.

Algorithm (3.21): Combination between two features

Input:

NofClass, NofSample, NofFeat

Output:

TSuccRateComb file// Success Rate for each feature

Goal:

Compute success rate for combined feature

```
Step1:
    Read from the file membership for each feature in all classes
    FeatMem(0 to NofClass-1,0 to NofSample-1,0 to NofFeat-1,0 to NofClass-1)
    Set FeatMem ← Read from FMem file
Step2:
    Compute success rate for combined features in each class
    SuccessRate(0 to FeatNo-1, 0 to FeatNo-1, 0 to ClassNo-1)
    For K1 = 0 To NofFeat-2
        For K2 = K1 + 1 To NofFeat-1
          For I = 0 To NofClass-1
                For J = 0 To NofSample-1
                    For I1 = 1 To NofClass-1
                        Set Mem←FeatMem(I, J, K1, I1)+FeatMem(I, J, K2, I1)
                        If (I1=0) or (I1>0 and Mem>MaxMem)Then
                          Set MaxMem←Mem
                          Set Index←I1
                        End If
                    Next I1
                    If Index=I Then Increment SuccRate(K1, K2,I) by 1
                End loop J
          End loop I
        End loop K2
    End loop K1
```

Continue

Step3:

Compute Total Success Rate for each feature in all Classes

```
TotalSuccRate(0 to FeatNo-1, 0 to FeatNo-1)
For I = 0 To NofClass-1
   For K1 = 0 To NofFeat-2
      For K2 = K1 + 1 To NofFeat-1
          TotalSuccRate(K1, K2) = TotalSuccRate(K1, K2) + SuccRate(K1, K2, I)
       End loop K2
   End loop K1
End loop I
```

Step4:

save in the file the success rate for combined features

Save in the file TSuccRateComb the array(TSuccRateComb)

CHAPTER FOUR

TESTS AND RESULTS

CHAPTER FOUR

TESTS AND RESULTS

4.1 Introduction

This chapter is devoted to present and discuss the results of the conducted tests to study the classification performance of the suggested FTCS. In section (4.2) system interface is applied, in section (4.3) test material is applied, and at last section, the overall experiments are given.

The FTCS were established using Visual Basic (version 6.0) programming language. The tests have been applied using a personal computer (Pentium 4, processor 1.60 GHz, RAM 1 G-byte).

4.2 FTCS Interface

There are two forms which refer to the two phases of the system. First one represents the training phase and the second one represents the testing phase.

4.2.1 *Training Form*

The Training Form contains many objects which are needed to accomplish the training phase work, these objects are:

1. Menu bar that contains 2 items (*Feature Extraction, Select Features*).
2. Many input variables that should be specified to compute features, these variables are:
   - *Distance*: used to calculate co-occurrence matrix.
   - *Gray level* (Quantize level): used to quantize the gray image.
   - *Length of Sample*: specified the length of each sample.
   - *Number of Sample*: specified the number of samples.

3. Frame which contains the fuzzy membership functions (*Trapezoidal*, *Triangular*). The function should be specified before fuzzy selection is computed.
4. Frame which contains mapping type, this should be specified for lookup table computations.

Figure (4.1) shows the main form of training phase.

Figure (4.1): Training Form.

At first, the user must select from a list, distance, quantization level, length of each sample and number of samples. Then specifies the type of membership function (choose one option from frame1) and specifies the mapping type used to calculate lookup table (choose one option from frame2), then the user will start the process of the training phase by using the Feature Extraction and Select Features operations.

1. ***Feature Extraction***: This item contains two functions, as shown in figure (4.2), (*ComputeCo-occurrence_and_Run-Length_Features*, *Exit*).

Figure (4.2): Feature Extraction.

- ***ComputeCo-occurrence_and_Run-Length_Features*:** when the user calls this function, the following tasks will be performed:
  - § Load the images of 24 bit/pixel for the classes of the systems.
  - § According to the specific number of sample with a specific sample length takes a number of samples from each image.
  - § Compute the co-occurrence matrix for each sample and a set of texture features will be computed from co-occurrence matrix and stored in *CocFeature* file.
  - § Compute the run length matrix for the same sample and a set of texture features will be computed from run length matrix and stored in *RLFeature* file.
- ***Exit:*** when the user calls this function, the program execution will be stopped.

2. ***Select Features*:** This item contains two functions, as shown in figure (4.3) (*BasedOnCooccurrence*,*BasedOnRunLength*).

Figure (4.3): Select Features.

- ***BasedOnCoocurrence*:** when the user calls this function, the following tasks will be performed:
  - § Take the computed co-occurrence features.
  - § Select the best features according to the success rate for the features.
- ***BasedOnRunLength*:** when the user calls this function, the following tasks will be performed:
  - § Take the computed run length features.
  - § Select the best features according to the selection rate for the features.

4.2.2 *Testing Form*

The Testing Form contains the menu bar that has two items (*Feature Extraction*, *Classification*), as shown in figure (4.4).

Figure (4.4): Testing Form.

1. ***Feature Extraction***: this item contains two functions, as shown in figure (4.5) (*Co-occurrence_and_Run-Length, Exit)*.

Figure (4.5): Feature Extraction.

- ***Co-occurrence_and_Run-Length*****:** when the user calls this function, the following tasks will be performed:
  - § Load the image of 24 bit/pixel for the test image.
  - § According to the same number and length of samples giving in the training phase, the samples take from test image.
  - § Compute the co-occurrence matrix for each sample and a set of texture features will be computed from this matrix and stored in a file.

§ Compute the run length matrix for each sample and a set of texture features will be computed from this matrix and stored in another file.

- ***Exit:*** when the user calls this function, the program execution will be stopped.

2. Classification: This item contains two options, as shown in figure(4.6) (*UsingFuzzyBasedOnCo_ocurrence*, *UsingFuzzyBasedOnRun_Length*).

Figure (4.6): Classification.

- ***UsingFuzzyBasedOnCo_ocurrence*:** when the user calls this function, the following tasks will be performed:
  - § Take the computed co-occurrence features.
  - § Find the nearest class according to the success rate for the training class.
- ***UsingFuzzyBasedOnRun_Length*:** when the user calls this function, the following tasks will be performed:
  - § Take the computed run length features.
  - § Find the nearest class according to the success rate for the training class.

4.3 Test Material

In FTCS, 10 textured images are selected to represent 10 different classes. These classes represent the training images for the system. These images have size 256×256 pixels with color resolution 24 bit/pixel, as shown in figure (4.7).

To perform the texture classification testing, 20 textured images are chosen for soft testing (which means that the test images are taken from the training images either part of it or rotate it), as shown in figure (4.8) and 30 textured images are chosen for hard testing, as shown in figure (4.9). These images have different size with color resolution 24 bit/pixel.

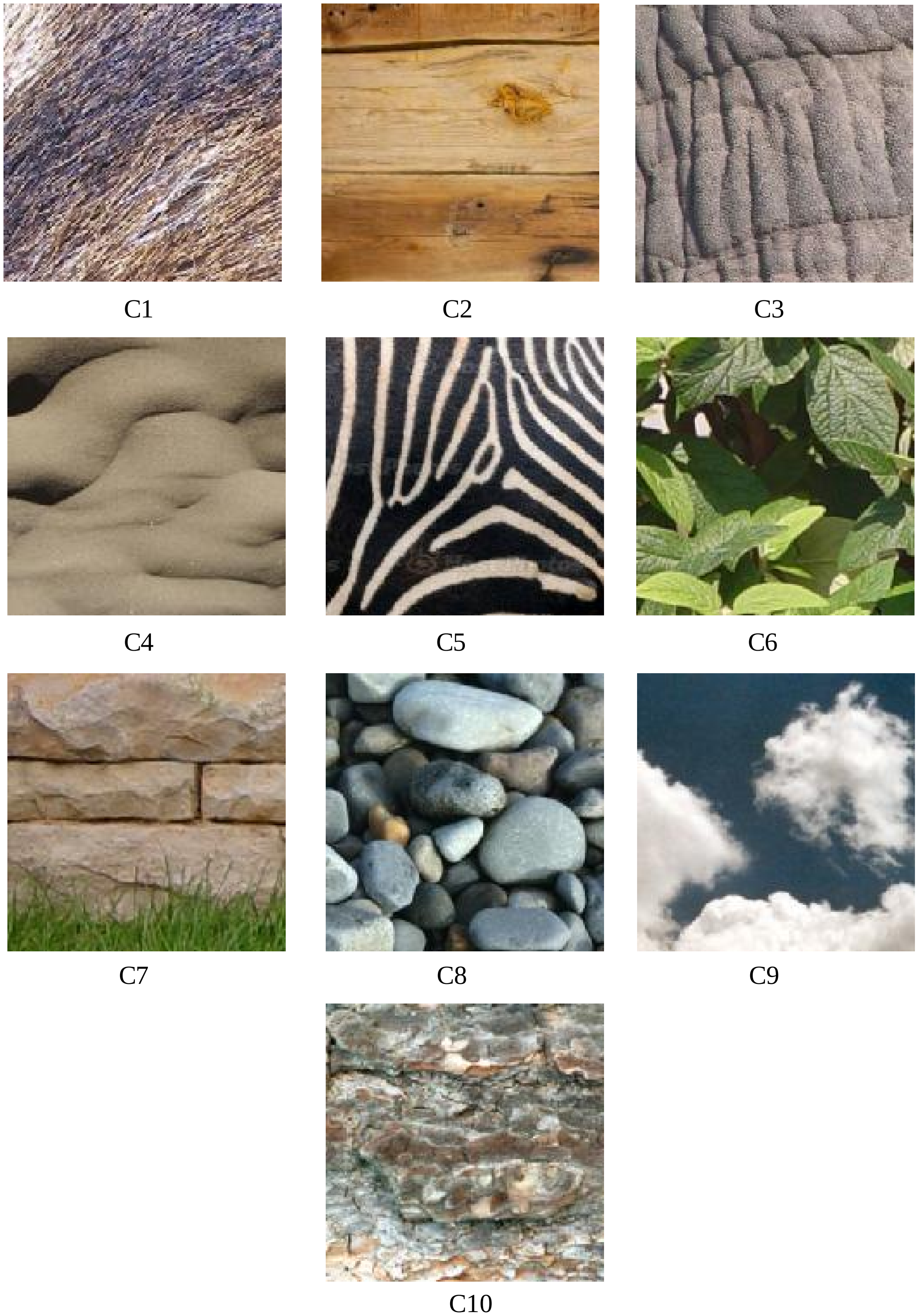
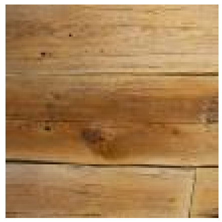
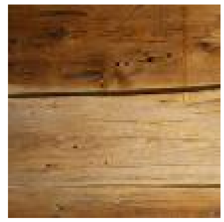


Figure (4.7): Training Images.

Figure (4.8): Test Images using in Soft Testing Process.

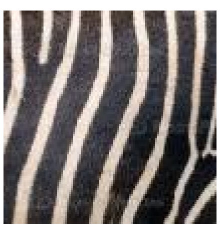
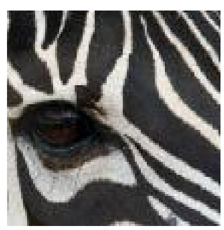
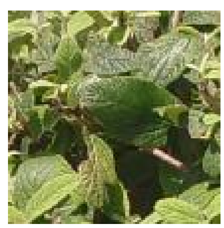
H1 H2 H3 H4

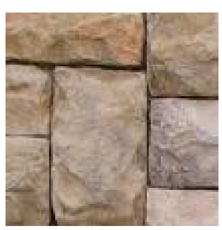
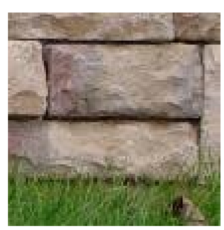
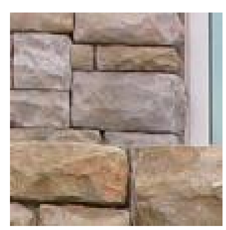
H5 H6 H7 H8

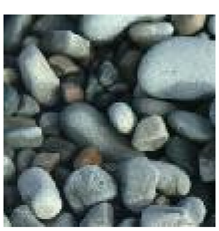
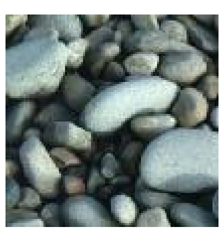
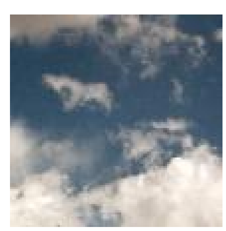
H9 H10 H11 H12

H13 H14 H15 H16

H17 H18 H19 H20

H21 H22 H23 H24


H25


H26


H27


H28


H29

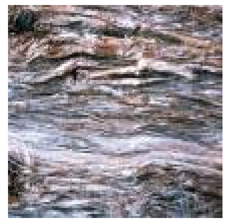

H30

Figure (4.9): Test Images using in Hard Testing Process.

4.4 FTCS Models Analysis

The analysis includes testing various parameter values, number of gray levels was varied to be 16, 32 or 64, the distance between pixels for calculation of co-occurrence matrix was varied to be 1,2, or 3, number of selected sample was varied to be 100, 200, 500 or 1000, the length of each samples was varied to be 10, 25, 50, 75 or 100, which mapping type chose to do quantization also varied to be 1, 2, 3,or 4, and finally determine the membership types which take either trapezoidal or triangular. So the variation in the parameters will cause variation in the results of success rates for the features.

Features computed using co-occurrence and run length matrices are shown in table (4.1) and table (4.2) respectively, where column 1 represent the name of the features, column 2 to column 5 (Th0, Th45, Th90, Th135) represent the theta of the direction, and column 6 (Avg) represent the average for the four direction.

Some of the experiment results are applied according to the parameters mentioned in table (4.3), which declares the cases taken in the results and the parameters which affect the success rate for features where column 1 (CaseNo) represents case number, column 2 (Mtype) represents mapping type, and column 3 (Glevel) represents gray level for quantization.

Experiments with different parameters values are applied in table (4.4) to table (4.9), which contain maximum success rate value for different combinations of features, where column 1 (CaseNo) represent case number, column 2 (single) represent success rates for features with out combination, column 2 to column 4 represent success rates for features with combination (Comb2, Comb3, Comb4), and column 5(Trap/Traing) represent membership function that used in selection features. All of these tables applied results according to distance 1, but different distance is given in table (4.10).

Finally, the best features for the images used in figure (4.7) are obtained from the results for different experiments are shown in table (4.11) with column 1 (Best CocFeat) for best co-occurrence features and column 2 (Best RLFeat) for best run length features.

Table (4.1) The index number of each co-occurrence features

FeatureName	Th0	Th45	Th90	Th135	Avg
Energy	F0	F21	F42	F63	F84
Contrast	F1	F22	F43	F64	F85
Correlation	F2	F23	F44	F65	F86
Variance	F3	F24	F45	F66	F87
InvDifMom	F4	F25	F46	F67	F88
Entropy	F5	F26	F47	F68	F89
SumAvg	F6	F27	F48	F69	F90
SumEnt	F7	F28	F49	F70	F91
SumVar	F8	F29	F50	F71	F92
DifVar	F9	F30	F51	F72	F93
DifEnt	F10	F31	F52	F73	F94
InfMeasCor1	F11	F32	F53	F74	F95
InfMeasCor2	F12	F33	F54	F75	F96
MaxProb	F13	F34	F55	F76	F97
DifMom1	F14	F35	F56	F77	F98
DifMom2	F15	F36	F57	F78	F99
DifMom3	F16	F37	F58	F79	F100
DifMom4	F17	F38	F59	F80	F101
Homog	F18	F39	F60	F81	F102
ClusterShade	F19	F40	F61	F82	F103
ClusterProm	F20	F41	F62	F83	F104

Table (4.2) The index number of each run length features

FeatureName	Th0	Th45	Th90	Th135	Avg
RP	F0	F11	F22	F33	F44
SRE	F1	F12	F23	F34	F45
LRE	F2	F13	F24	F35	F46
RLD	F3	F14	F25	F36	F47
LGRE	F4	F15	F26	F37	F48
HGRE	F5	F16	F27	F38	F49
GLN	F6	F17	F28	F39	F50
SRLGE	F7	F18	F29	F40	F51
SRHGE	F8	F19	F30	F41	F52
LRLGE	F9	F20	F31	F42	F53
LRHGE	F10	F21	F32	F43	F54

Table (4.3) Parameters for specific case

CaseNo	Mtype	Glevel
1	1	16
2	2	16
3	3	16
4	4	16
5	1	32
6	2	32
7	3	32
8	4	32
9	1	64
10	2	64
11	3	64
12	4	64

Table (4.4) Success Rates % for different combinations for Co-occurrence features for 500 samples with length 50

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	51	69	74	76	Traing
1	51	70	74	78	Trap
2	48	64	69	73	Traing
2	46	66	71	75	Trap
3	53	67	72	75	Traing
3	48	68	73	76	Trap
4	44	60	64	66	Traing
4	43	63	67	69	Trap

Table (4.5) Success Rates % for different combinations for Run length features for 500 samples with length 50

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	49	60	67	72	Traing
1	48	62	67	71	Trap
2	48	57	62	67	Traing
2	43	57	65	66	Trap
3	47	61	67	67	Traing
3	46	62	68	69	Trap
4	44	57	57	58	Traing
4	40	54	62	64	Trap

Table (4.6) Success Rates % for different combinations for Co-occurrence features for 500 samples with length 100

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	60	86	89	92	**Traing**
1	60	86	90	93	**Trap**
2	60	83	87	89	**Traing**
2	55	81	89	90	**Trap**
3	64	83	89	90	**Traing**
3	59	83	90	92	**Trap**
4	62	78	82	85	**Traing**
4	58	77	85	87	**Trap**
5	60	85	91	94	**Traing**
5	60	86	92	95	**Trap**
6	60	85	87	90	**Traing**
6	56	81	89	92	**Trap**
7	65	84	90	91	**Traing**
7	59	79	89	92	**Trap**
8	64	77	83	85	**Traing**
8	58	77	83	86	**Trap**

Table (4.7) Success Rates % for different combinations for Run length features for 500 samples with length 100

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	60	76	85	90	**Traing**
1	60	78	86	91	**Trap**
2	58	75	82	87	**Traing**
2	56	79	82	85	**Trap**
3	59	76	81	86	**Traing**
3	56	75	83	89	**Trap**
4	58	73	78	82	**Traing**
4	59	74	82	83	**Trap**
5	55	74	80	83	**Traing**
5	55	74	80	84	**Trap**
6	55	69	78	81	**Traing**
6	52	74	78	81	**Trap**
7	59	76	80	80	**Traing**
7	57	71	80	85	**Trap**
8	53	71	75	76	**Traing**
8	51	68	76	79	**Trap**

Table (4.8) Success Rates % for different combinations for Co-occurrence features for 1000 samples with length 100

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	61	85	89	92	**Traing**
1	60	86	90	93	**Trap**
2	61	83	88	90	**Traing**
2	53	79	88	91	**Trap**
3	62	84	90	93	**Traing**
3	60	85	89	92	**Trap**
4	62	75	82	85	**Traing**
4	57	75	84	86	**Trap**
5	62	86	89	91	**Traing**
5	55	76	80	84	**Trap**
6	61	83	88	90	**Traing**
6	54	74	81	83	**Trap**
7	63	85	88	91	**Traing**
7	55	74	8	84	**Trap**
8	64	77	83	85	**Traing**
8	64	77	83	85	**Trap**
9	64	86	89	93	**Traing**
9	58	83	89	92	**Trap**
10	60	82	86	89	**Traing**
10	52	80	88	90	**Trap**
11	62	82	89	93	**Traing**
11	57	80	90	92	**Trap**
12	61	78	84	88	**Traing**
12	57	77	86	81	**Trap**

Table (4.9) Success Rates % for different combinations for Run length features for 1000 samples with length 100

CaseNo	Single	Comb2	Comb3	Comb4	Trap/Traing
1	59	76	86	90	**Traing**
1	61	79	87	92	**Trap**
2	59	78	83	87	**Traing**
2	55	79	84	87	**Trap**
3	60	77	82	86	**Traing**
3	57	76	83	88	**Trap**
4	46	68	80	82	**Traing**
4	57	74	82	83	**Trap**
5	61	75	82	86	**Traing**
5	56	8	91	92	**Trap**
6	54	77	80	83	**Traing**
6	59	79	89	92	**Trap**
7	60	71	80	83	**Traing**
7	58	77	84	86	**Trap**
8	54	7	76	78	**Traing**
8	50	68	76	80	**Trap**
9	61	74	79	83	**Traing**
9	48	70	79	8	**Trap**
10	55	76	79	83	**Traing**
10	50	72	78	82	**Trap**
11	60	74	79	80	**Traing**
11	53	69	78	78	**Trap**
12	49	68	75	76	**Traing**
12	47	66	74	77	**Trap**

Table (4.10) Success Rates % for different combinations for Co-occurrence features for 500 samples with length 100

CaseNo	distance	Single	Comb2	Comb3	Comb4	Trap/Traing
1	**1**	60	86	89	92	**Traing**
1	**2**	59	81	87	90	**Traing**
1	**3**	55	78	86	88	**Traing**

Table (4.11) Best co-occurrence and run length features

Best CocFeat	Best RLFeat
F5	F0
F9	F1
F10	F3
F23	F6
F26	F17
F31	F22
F40	F23
F41	F25
F44	F26
F46	F28
F50	F29
F51	F34
F52	F36
F53	F39
F54	F40
F60	F45
F65	F47
F83	F50
F86	F51
F94	F52

Chapter Five

Conclusions and Suggestions for Future Work

Chapter Five

Conclusions and Suggestions for Future Work

5.1 Introduction

This chapter is devoted to present the derived conclusions concerned with the performance of the classification methods based on using statistical features as discriminating attributes. Furthermore, some suggestions for future work are also presented in this chapter.

5.2 Conclusions

1. The co-occurrence matrices are calculated for the *distances* 1, 2, and 3. Best results for these selected images are achieved when *distance* between pixels for co-occurrence matrix is 1.
2. Best results are achieved when using *general equation* techniques to generate lookup table.
3. The length and number of the samples play an important role for increasing accuracy for the classification process.
4. Trapezoidal membership yield success rates better than triangular membership but the difference does not give a high improvement (93%, 91% respectively).
5. Co-occurrence method yields selection rates better than run length method (90%, 85% respectively).

6. The performance of the system increases when using combination between features (60% without combination, 75% with combination two features, 85% with combination three features, and 90% with combination four features).
7. Performance results nearly 95% for soft testing and 85% for hard testing.

5.3 Suggestions for Future Work

1. FTCS use image with single texture, so to make the system more flexible for multi texture the pre step for this system can be added, which segment an image into regions with the same texture, i.e. as a complement to grey level or color before classifying it.
2. Using structural attribute in addition to statistical attribute to increase the recognition accuracy.
3. Combine between fuzzy and neural to increase the performance of the system.
4. Using different color models to make the system more flexible, such as HSV (Hue, Saturation and Value) and HSI (Hue, Saturation and Intensity) which derived from RGB color model.

REFERENCES

REFERENCES

1. [Ach05]: T. Acharya and A. K. Ray, "Image Processing: Principles and Applications", Wiley, 2005.
2. [Ala96]: L. Abdul Aziz Al-Ani, "Classification of Digital Satellite Images",(Ph.D.) thesis, College of Science, Al-Nahrain University, Iraq, 1996.
3. [Alb95]: Albregtsen, F., "Statistical Texture Measures Computed from Gray Level Run Length Matrices", Image Processing Laboratory, Department of Informatics, Oslo University, 1995.
4. [Ali99]: R. S. Ali, "Image Segmentation Using Fuzzy and Neural Networks", MSC thesis, College of science, Al-Nahrain University, Iraq, 1999.
5. [Dua06]: M. G. Duaimi, "Development of a Content-Based Image Retrieval System", (Ph.D) thesis, College of science, Al-Nahrain University, 2006.
6. [Dud00]: R. O. Duda, P. E. Hart and D. G. Stork, "Pattern Classification", 2nd edition, Wiley, 2000.
7. [Fri99]: M. Friedman and A. Kandel, "Introduction to Pattern Recognition: Statistical, Structural, Neural and Fuzzy Logic Approaches", World Scientific, 1999.
8. [Gon87]: R. C. Gonzalez and R. E. Gonzalez, "Digital Image Processing", Addison Wesley Publishing Company, 1987.
9. [Gui88]: J. Guibert, D. C. He and L. Wang, "Texture Discrimination Based on Optimal Utilization of Texture Feature", Pattern Recognition, Vol.21, No.2, PP.141-149, 1988.

10.[Har79]: R. M. Haralick, "Statistical and Structural Approaches to Texture", Proceedings of the IEEE, Vol.67, PP.786-804, 1979.

11.[Hir94]: K. Hirota and W. Pedrycz, "Implicitly - supervised fuzzy Pattern recognition", IEEE, 1994.

12.[Ibr04]: A. M. Ibrahim, "Fuzzy Logic: for Embedded Systems Applications", Elsevier Science, 2004.

13.[Jäh99]: B. Jähne, H. Haussecker and P. Geissler, "Handbook of Computer Vision and Applications: Volume2 Signal Processing and Pattern Recognition", Academic Press, 1999.

14.[Jai98]: L. C. Jain and N. M. Martin, "Fusion of Neural Networks, Fuzzy Systems, and Genetic Algorithms: Industrial Applications", CRC Press LLC, 1998.

15.[Kar94]: K. Karu and A. K. Jain, "Learning Texture Discrimination Masks", Proc. IEEE Int'l Conf Neural Networks, June, PP.4374-4397, Orlando, 1994.

16.[Kas96]: N. K. Kasabov, "Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering", Massachusetts Institute, 1996.

17.[Kon02]: E. S. Konak, "A Content-Based Image Retrieval System for Texture and Color Queries", MSC thesis, Computer Engineering, Engineering and Science Bilkent University, 2002.

18.[Kun04]: L. I. Kuncheva, "Combining Pattern Classifier: Methods and Algorithms", Wiley, 2004.

19.[Lee05]: K. H. Lee, "First Course on Fuzzy Theory And Applications", Springer, 2005.

20.[Lim04]: C. Limpsangsri, "Image Retrieval using texture", DePaul University, 2004.

21.[Mar01]: J. P. Marques de Sa, "Pattern Recognition: Concepts, Methods and Applications", Springer, 2001.

22.[Mat98]: A. Materka and M. Strzelecki, "Texture Analysis Methods", Technical University of Lodz, Institute of Electronics, Brussels, 1998.

23.[McN94]: F. M. McNeill and E. Thro, "Fuzzy Logic: A Practical Approach", Academic Press, 1994.

24.[Moh07]: M. H. Mohammed and S. M. Faizur Rahman, "Fuzzy Features Extraction from BANGLA Handwritten Character", International Conference on Information and Communication Technology ICICT, March 2007.

25.[Mon04]: A. Monadjemi, "Towards Efficient Texture Classification and Abnormality Detection", (Ph.D.) thesis, Department of Computer Science, Faculty of Engineering, University of Bristol, October 2004.

26.[Ned01]: I. Nedeljkovic, "Image Classification Based on Fuzzy Logic", Fuzzy Inference System FIS, 2004.

27.[Nib86]: W. Niblack, "An Introduction to Digital Image Processing", Prentice_Hall International, 1986.

28.[Pal01]: S. K. Pal and A. Pal, "Pattern Recognition from Classical to Modern Approaches", World Scientific, 2001.

29.[Pra01]: W. K. Pratt, "Digital Image Processing", 3rd edition, Wiley, 2001.

30.[Roa87]: J. J. Roan, J. K. Aggarwal and W. N. Martin, "Multiple Resolution Imagery and Texture Analysis", Pattern Recognition, Vol.20, No.1, PP.17-31, 1987.

31.[Sag06]: K. H. Sager, "Fractal Based Classification for Color Textural Images", (Ph.D.) thesis, College of Science, Baghdad University, Iraq, 2006.

32.[Sam99]: V. W. Samawi, "An Investigation in to the use of Neural Networks in Texture Classification", (Ph.D.) thesis, College of science, Al-Nahrain University, Iraq, 1999.

33.[Son08]: M. Sonk, V. Hlavac and R. Boyle, "Image Processing Analysis and Machine Vision", Thomson, 3rd edition, 2008.

34.[The03]: S. Theodoridis and K. Koutroumbas, "Pattern Recognition", 2nd edition, Elsevier, 2003.

35.[Tuc98]: M. Tuceryan and A. K. Jain, "Texture Analysis", Chapter 2.1 in Handbook of Pattern Recognition and Computer Vision, 2nd Edition, World Scientific, 1998.

36.[Web02]: A. Webb, "Statistical Pattern Recognition", 2nd edition, Wiley, 2002.

37.[Wei01] : L. Y. Wei, "Texture Synthesis by Fixed Neighborhood Searching", (Ph.D.) thesis, Electrical Engineering, Stanford University, Li-Yi-Wei, 2001.

38.[Zho06]: D. Zhou, "Texture Analysis and Synthesis using a Generic Markov-Gibbs Image Model", (Ph.D.) thesis, Computer Science, Auckland University, 2006.

الخلاصة

تعتبر عملية التصنيف واحدة من أهم العمليات في العديد من تطبيقات الحاسوب لغرض تصنيف الصور بالنسبة الى الصفات المنخفضة المستوى كاللون أوالتركيب النسيجي للصورة.

في هذا العمل يتم عرض نظام التصنيف الذي يدعم الاستفسار بالنسبة الى الخصائص المنالتركيب النسيجي. تتلخص الفكرة الاساسية للعمل في توليد ميزات الصورة اوتوماتيكيا عن طريق تحليل محتوياتها. تعتمد التقنيات المضمنة على اتخاذ مصفوفة الظهور المتلازم للالوان الرمادية و مصفوفة طول السلسلة للالوان الرمادية كوسائل احصائية لتحليل التركيب النسيجي. و قد طبقت هذه الاساليب بحالات منفردة.

يتمثل كل صنف بواسطة متجه او (مجموعة متجهات) في فضاء الصفات و تخزن بملف. لاحقا يتم اختيار افضل مجموعة من المتجهات للصورة باستعمال مفاهيم مضببة (وظائف عضوية مثلثية او وظائف عضوية رباعية). عند الاستعلام عن الصورة, يتم استخلاص متجه الصفات لهذه الصورة و من ثم مقارنة المتجهات المختارة المخزونه في الملف لايجاد اقرب صنف باستعمال القواعد المضببة.

خلال عملية التقييم وجد أن افضل النتائج نحصل عليها من الجمع بين الميزّاتِ و الذي يحقق نسب اعلى لإختيارِ الميزّاتِ بالإضافة إلى النظام ككل (تُصبحُ نسب الاختيار تقريبا 90% عند تجميع بين أربع ميزّاتِ و 60% بدون تجميع).

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

تصنيف التركيب النسيجي بالاعتماد على القاعدة المضببة

رسالة
مقدمة الى كلية العلوم في جامعة النهرين كجزء من متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

رقية اياد عبد الجبار

(بكالوريوس جامعة النهرين ٢٠٠٤)

بإشراف

د. سوسن كمال ثامر

١٤٢٩ ٢٠٠٨