

5-2001

## A neural network-based vision system for automated separation of onion from clod using mechanical harvester

Demian Morquin  
*University of Texas-Pan American*

Follow this and additional works at: [https://scholarworks.utrgv.edu/leg\\_etd](https://scholarworks.utrgv.edu/leg_etd)



Part of the [Manufacturing Commons](#)

---

### Recommended Citation

Morquin, Demian, "A neural network-based vision system for automated separation of onion from clod using mechanical harvester" (2001). *Theses and Dissertations - UTB/UTPA*. 441.  
[https://scholarworks.utrgv.edu/leg\\_etd/441](https://scholarworks.utrgv.edu/leg_etd/441)

This Thesis is brought to you for free and open access by ScholarWorks @ UTRGV. It has been accepted for inclusion in Theses and Dissertations - UTB/UTPA by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact [justin.white@utrgv.edu](mailto:justin.white@utrgv.edu), [william.flores01@utrgv.edu](mailto:william.flores01@utrgv.edu).

A NEURAL NETWORK-BASED VISION SYSTEM  
FOR AUTOMATED SEPARATION OF ONION FROM CLOD  
USING MECHANICAL HARVESTER

A Thesis

by

DEMIAN MORQUIN

Submitted to the Graduate School of the  
University of Texas - Pan American  
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

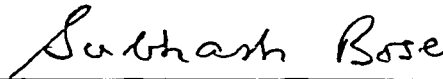
May 2001

Major Subject: Manufacturing Engineering

A NEURAL NETWORK-BASED VISION SYSTEM  
FOR AUTOMATED SEPARATION OF ONION FROM CLOD  
USING MECHANICAL HARVESTER

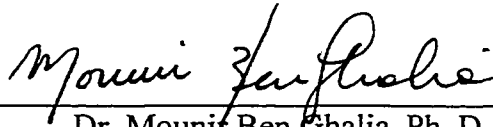
A Thesis  
by  
DEMIAN MORQUIN

Approved as to style and content by:



---

Dr. Subhash Bose, Ph. D., P. E.  
Professor of Manufacturing Engineering  
Co-Chair of Committee



---

Dr. Mounir Ben Ghalia, Ph. D.  
Assistant Professor of Electrical Engineering  
Co-Chair of Committee



---

Dr. Robert Freeman, Ph. D.  
Associate Professor of Mechanical Engineering  
Member of Committee

May 2001

## ABSTRACT

Morquin, Demian, A Neural Network-Based Vision System for Automated Separation of Onion from Clod Using Mechanical Harvester. Master of Science (MS), May 2001, 70 pp.; 7 tables, 31 figures, references, 25 titles.

A neural network classifier for separating clods from onions during harvesting has been developed. The separator consists of a multi-layer feedforward network that maps textural features computed from gray-scale images of onions and clods into the correct object. Texture features were computed from co-occurrence matrices that specify the spatial relationship of pixel values in an image. The textural features selected for this application consist of homogeneity, contrast, variance, and energy. The network was trained using the back-propagation algorithm. Based on the textural features classification, the effect of changing the network configuration on separation effectiveness was investigated. Factors including network topology and combination of textural feature measures forming the inputs of the network were systematically analyzed. Thirty three different network configurations were evaluated. The best separation effectiveness was obtained with three-layer (3-2-1) network with input set consisting of energy, contrast, and homogeneity feature measures. The separation effectiveness for 3-2-1 network topology was 96 percent. An analysis of integration of the neural network-based vision system with a mechanical separator is presented.

## DEDICATION

This thesis is dedicated to an exceptional group of people that have accompanied and supported me along this special journey. From the depth of my heart I dedicate this work to my beloved and amazing wife, Gabriela; my marvelous mother, Diana; my philosopher father, Raul; my sister and friend, Gillian, and her husband, Juan; my beautiful niece, Francisquita; my loving aunt, Mirta; my caring cousins, Henry, Alejandra, Marla, and Andrea; my affectionate uncle, aunt, and cousins Carlos, Stella, Tammy, Romina, and Daniel; and my always present grandparents Elias (☆), Mary, Daniel (☆) and Clara (☆).

Esta tesis esta dedicada a un grupo de personas excepcionales que me han acompañado y apoyado a lo largo de este camino tan especial. Desde lo mas profundo de mi corazon, dedico este humilde triunfo a mi magnifica y amada esposa Gabu, a mi maravillosa madre, Diana; a mi filosofo padre, Raul; a mi hermana y amiga, Gillian, y a su esposo, Juan; a mi hermosa sobrina, Francisquita; a mi bondadosa tia, Mirta; a mis generosos primos, Quique, Alejandra, Marla, y Andrea; a mis afectuosos tios y primos, Carlos, Stella, Tammy, Romina, y Daniel; y a mis siempre presentes abuelos Elias (☆), Mary, Daniel (☆) and Clara (☆).

## ACKNOWLEDGMENTS

I would like to thank my two advisors, Dr. Subash Bose and Dr. Mounir Ben Ghalia for giving me their support, and sharing their knowledge and experience. I like to thank Dr. Robert Freeman for his willingness to serve as a member of my thesis committee and to recognize all the professors in the manufacturing engineering program at UT-Pan American. Finally, I would like to mention Miguel Torres for helping me along the graduate program.

## TABLE OF CONTENTS

	Page
ABSTRACT .....	iii
DEDICATION .....	iv
ACKNOWLEDGEMENTS .....	v
TABLE OF CONTENTS .....	vi
LIST OF TABLES .....	ix
LIST OF FIGURES .....	x
CHAPTER 1. INTRODUCTION .....	1
CHAPTER 2. REVIEW OF LITERATURE .....	3
CHAPTER 3. TEXTURE ANALYSIS .....	11
3.1 Image Quantization .....	12
3.2 The Co-occurrence Matrix .....	14
3.3 Selecting Texture Features.....	16
3.4 Capturing Images .....	17
3.5 Process of Computing Texture Features .....	18
CHAPTER 4. NEURAL NETWORKS .....	21
4.1 Neural Network Justification .....	21
4.2 Architecture of a Neural Networks .....	24
4.2.1 Single-layer feedforward networks .....	24
4.2.2 Multilayer feedforward networks .....	25

4.2.3 Recurrent networks .....	26
4.3 Back-Propagation Algorithm .....	26
4.3.1 Training process .....	28
4.3.2 Function and error signals .....	28
4.4 Motivation for Using Neural Network .....	29
CHAPTER 5. EXPERIMENTAL RESULTS .....	33
5.1 Data Collection for Image Processing .....	33
5.2 Separation Algorithm .....	35
5.2.1 Neural network structure .....	35
5.2.2 Neural network training .....	36
5.3 Separation Results .....	37
CHAPTER 6. MECHANICAL INTEGRATION OF VISION SYSTEM .....	43
6.1 Onion-Clod Separation Concepts .....	43
6.1.1 Rotating Brushes .....	43
6.1.2 Restitution Method .....	44
6.1.3 Rotating Cylinder .....	44
6.2 Harvesting Process .....	45
6.3 Cup Type Conveyor System .....	46
CHAPTER 7. CONCLUSIONS .....	48
REFERENCES .....	50
APPENDIX A:	
Texture Analysis Program .....	53



APPENDIX B

Neural Network Training Program ..... 62

APPENDIX C

Neural Network Testing Program ..... 64

APPENDIX D

Textural Features Data for Clod and Onion..... 66

VITA ..... 70

## LIST OF TABLES

	Page
Table 3.1 Image quantization levels considered in this study .....	13
Table 3.2 Co-occurrence matrices containing the number of pixel pairs at a distance of 1 and direction of 45° and 90° .....	16
Table 3.3 Texture features considered in this study .....	16
Table 3.4 Values obtained for the homogeneity feature for six different orientations of the image of an onion (the distance <i>d</i> is set equal to 1 for all cases) .....	20
Table 4.1 Benefits and characteristics of a neural network structure .....	22
Table 5.1 Image size and resolution values considered in this study .....	33
Table 5.2 Neural network topologies examined in this study .....	35

## LIST OF FIGURES

	Page
Fig. 3.1 Visual perception of the difference in surface texture (a) an onion and (b) a clod .....	12
Fig. 3.2 Images of onion texture before and after using different quantization levels .....	13
Fig. 3.3 Simple surface texture and the matrix containing its corresponding pixel values .....	14
Fig. 3.4 Illustration of the method used to obtain all the different images of onions and clods .....	18
Fig. 3.5 Images showing six different orientation of an onion .....	19
Fig. 3.6 Process of computing the average values of textural features .....	20
Fig. 4.1 Single-layer feedforward networks .....	24
Fig. 4.2 Multi-layer feedforward network .....	25
Fig. 4.3 Recurrent networks .....	26
Fig. 4.4 A backpropagation algorithm model .....	27
Fig 4.5 Scatter plot of textural features contrast and variance .....	30
Fig 4.6 Scatter plot of textural features contrast and homogeneity .....	30
Fig 4.7 Scatter plot of textural features homogeneity and variance .....	31
Fig 4.8 Scatter plot of textural features energy and homogeneity .....	31
Fig 4.9 Scatter plot of textural features energy and contrast .....	32

Fig 4.10 Scatter plot of textural features energy and variance .....	32
Fig. 5.1 Images of (a) onion with 256 gray levels, (b) onion with 16 gray levels, (c) clod with 256 gray levels, and (d) clod with 16 gray levels .....	34
Fig. 5.2 Architecture of the neural network separator .....	37
Fig. 5.3 Separation effectiveness of neural networks with 2 hidden neurons when they are presented with training data .....	38
Fig. 5.4 Separation effectiveness of neural networks with 3 hidden neurons when they are presented with training data .....	39
Fig. 5.5 Separation effectiveness of neural networks with 4 hidden neurons when they are presented with training data .....	39
Fig. 5.6 Separation effectiveness of neural networks with 2 hidden neurons when they are presented with testing data .....	40
Fig. 5.7 Separation effectiveness of neural networks with 3 hidden neurons when they are presented with testing data .....	40
Fig. 5.8 Separation effectiveness of neural networks with 4 hidden neurons when they are presented with testing data .....	41
Fig. 6.1 Rotating Brushes .....	43
Fig. 6.2 Restitution Method .....	44
Fig. 6.3 Rotating Cylinder .....	44
Fig. 6.4 Rotating drum that will be integrated with the cup type Conveyor where the neural network-based vision system will be placed .....	45

Fig. 6.5 Cup type conveyer where vision system will be placed .....	46
Fig. 6.6 Graphic representation of the cup type conveyer .....	47
Fig. 7.1 Flow chart of the onion harvesting process .....	48

## CHAPTER 1

### INTRODUCTION

Mechanical harvesters have been used for different types of fruits and vegetables for more than 10 years. However, there are still some problems with mechanical harvesting that could be improved. For instance, one of the major challenges is handling the separation process of onions and clods. Several improvements have been made in dealing with onion and clod separation; yet, there are still pieces of clods found at the end of this process. This thesis focuses on improving mechanical harvesting of onion using a neural network-based vision system in the process of separation of onion from clod.

According to several research which are reviewed in chapter two, one of the main problems of mechanical harvesting of onion is the lack of removal of pieces of clods that come from a field. These pieces of clods cause damage to onions as well as to the blades of the cutter, during the trimming of roots. By integrating a neural network-based vision system, capable of detecting onion from clod, it is possible to completely eliminate pieces of clod in the separation process before the trimming process takes place.

The vision system is used to capture and analyze images of onions and clods. The analysis of the images includes computing textural features from the digitized images and identifying whether they represent features of an onion or of a clod. Even though, there are many different textural features that could be selected for identification of onion from clod, only four of the significant features are considered in this thesis: homogeneity,

contrast, energy and variance. The four feature measures provide the input data to a multi-layer feed-forward neural network. A back-propagation algorithm is used to train the network by adjusting its weights. Several network topologies are tested and evaluated to determine their efficiency in the separation process.

The outline of the thesis is as follows: in Chapter two, the most relevant literature on texture analysis, vision system, and neural networks are reviewed. Chapter three presents a detailed explanation of texture analysis, image quantization process, co-occurrence matrix generation, and the computation of textural features. Chapter four summarizes the concept of neural networks, various types of neural network architecture, and the back-propagation algorithm. Chapter five discusses the experimental study conducted in this thesis. In Chapter six, the integration of neural network-based vision system with a mechanical onion harvesting mechanism is described. Chapter seven summarizes the results obtained in this study with suggestions for future analysis and improvements of the neural network-based vision system.

## CHAPTER 2

### REVIEW OF LITERATURE

Mechanical harvesting of onions and other agricultural produce from below ground has not been quite successful due to large number of factors affecting the performance of harvesters. Soil conditions, type of agricultural produce, stage of harvesting, to name a few; pose a challenge in using state of the art technology. Maw *et al.* (1998) have developed a mechanical harvester for sweet onions. The features include a lifting head for lifting the tops, gathering wheels, an undercutter, lifting belts, depth gage wheels, a shaker to remove soils from the bulb, a topper to cut tops from the bulb, conveyor to transport onions from lifting head to a container and a conveyor to dispose the tops. An automatic lifting head control adjusts for ground undulations. Several factors affect the performance of a machine. Examples of conditions that help improve machine performance include: sandy loam soil type, proper grasping of the onion top, ground speed of 2.4 *km/h*, a speed lifting belt of equal to 125 percent of ground speed, onion neck length of 40 to 60 mm, and an onion bulb root length of 10 to 40 mm.

LePori and Hobgood (1970) have reported three major types of harvesters available commercially. All these machines use a rod digger chain for lifting bulbs but differ in topping methods. One method of topping uses an air blast directed from below the rod chain to lift tops for cutting. Another method of topping uses counter rotating cylinders that pull tops between the rollers and pinch the top. A modification to this



method uses counter rotating cylinders to position bulbs, while the cylinders direct bulbs to a cutter. Coble *et al.* (1976) have developed an onion harvester which include the following systems: mowing the green tops from the onions, undercutting to break the roots from the soil and soften the soil crust, removing the onions from the row, cutting the remaining tops and the roots, and transporting the onions to packaging shed. However, many of the onion harvesting machines already developed have not been successful in the southern states, particularly in the state of Texas. Some of the reasons behind this lack of success are the type of onion being harvested, the high humidity in the region, and mainly the large formation of clods due to the type of soil characteristics in south Texas. In dealing with significant amount of clods, a mechanical roller system has been designed; yet, the clod percentage is high, and the roller separator decreases its efficiency considerably (Coble *et al.* 1976). Feller *et al.* (1984) have developed a unit that separates onions from soil clods based on the coefficient of restitution property of onion and clod. Since onion has a greater coefficient of restitution, it rebounds to a greater distance than clods when dropped onto a bouncing surface. Efficiency of separation of onions and clods was found to be 93 percent. The effectiveness of separation is calculated based on the recovery of the desired produce, and the rejection of undesired element (Brown *et al.* 1951).

Several prototypes for automated onion harvesting have been tested and some others are currently in use. However, damaging of onions due to the blades used to cut the roots of the onion is still an important issue. Preventing clods from falling on the blades will certainly improve the life of the blades and will increase the speed and quality of the harvesting process (Cuellar *et al.* 2000).

In this thesis, a machine vision system has been developed to help improve the effectiveness of separation between onion and clod. Several studies using machine vision systems have been conducted in many different areas. Digital image processing has a wide variety of applications, such as remote sensing via satellites, image transmission and storage for business, medical image processing, radar, sonar, acoustic, robotics, and automated inspection of industrial parts (Jain 1988, Slater *et al.* 1996, Yamamoto *et al.* 1996). Slater *et al.* (1996) have used a machine vision system to automate the classification and counting of neurons in brain tissue samples. Such a system will improve the time to count and classify neurons, allowing quick and accurate diagnosis. The categorizing process uses pixel classification using color variability to differentiate and identify certain type of neurons. A database management system was used to count the cells in the region provided by the vision system. This research has proven that automated vision system is able to replace manual counting, which it was known to be low, prone to errors, and extremely tedious.

Vision system has also been used to monitor traffic in highways (Zhu *et al.* 1996). A television camera is used to study and monitor traffic in highways. Two types of images are used in this system: a panoramic view image (PVI) and an epipolar plane image (EPI). The developed system was able to count the number of vehicles, estimate their speed, and classify vehicles by length, width, and height. Moreover, the vision system was able to work under diverse light conditions proving a sense of robustness and reliability. The cost of the system is inexpensive since it was effectively tested on a PC 486 using an image frame grabber.

Image processing techniques have also been used for diagnostics of lung cancer (Yamamoto *et al.* 1996). Two types of methods have been used in the study: the maximum intensity projection method (MIP) characterized by its pseudo three-dimensional display, and the Quoit filter method which isolates shadows using a 2-D and 3-D Quoit filters. This new image processing techniques were able to detect cancer by reducing the number of cross sections in the lung.

Motivated by the wide variety of real world applications that have gained from machine vision techniques, the work presented in this thesis shows that such techniques can also benefit agricultural applications. Bolle *et al.* (1996) have developed an automatic produce ID system named Veggie Vision that facilitates the checkout process of fruits and vegetables in food stores. The Veggie Vision system extracts unique features from a produce, such as texture and color, and used them to train the system to recognize each produce. Some of the issues reviewed by Bolle, such as recognition performance, system training, feature recognition, and texture analysis, provided important information for this study.

The representation of a produce item should be invariant to any type of rotation and translation, regardless of the feature being used for classification purposes. During classification and training, color has a crucial effect defined by hue, saturation, intensity, and texture as primary features for proper identification. The results obtained in this study were evaluated using color analysis combined with texture. When only color was used, 79 percent of the produce was correctly selected, and 93 percent of the produce was identified correctly among the four top choices. In case where color and texture were

used together, 84 percent of the time the produce was correctly selected, and 96 percent of the time the correct produce was found among the four top choices.

Techniques used by Haralick *et al.* (1973) for image classification using textural features is applied to this research. Haralick used texture to identify objects or regions of interest in an image. Computable textural features based on gray tone spatial dependencies were used on three different kinds of image data – photomicrograph of sandstone, and aerial photograph of land use categories. The textural features are based on statistics, summarizing the relative frequency distribution of gray levels in an image. Eighteen different textural features were computed and used for classification. Some of these features were sum of squares, entropy, difference variance, sum average, contrast, etc.

Tao *et al.* (1995) have used Fourier-based separation technique for shape grading of agricultural produce using machine vision for automated inspection. The relationship between object shape and its boundary spectrum values in Fourier domain was explored for shape extraction. A shape separator based on harmonics of Fourier transform was defined and tested for potato.

Jain (1988) has described the most important requirement in the process of image analysis during image sampling and quantization. According to Jain, image representation is possible via two-dimensional orthogonal functions known as basis images. These basis images are identified by unitary matrices, known as *image transforms*. The specific characteristics of each image are related to the relationship among its pixels, such as their spatial frequency, band-width, and power spectrum. The analysis of the relation among the pixels is evaluated by filter design, feature extraction,

segmentation, clustering, and many other techniques. For instance, segmentation separates different objects by their boundary while feature extraction evaluates the shape, texture, or moments of certain object.

A significant issue to be considered during texture analysis is the use of proper quantization techniques. Most quantization methods are based on traditional clustering or thresholding operation (Lynn *et al.* 1996). The use of proper quantization techniques will increase the speed of performance without losing the essential characteristics of a produce. A quantizer consists of a mapping procedure in which a continuous variable  $u$  is projected into a discrete variable  $u'$  taking values from a set finite numbers  $\{r_1, \dots, r_n\}$  (Jain 1988). There are many forms of quantization techniques, such as the Optimum Mean Square or Lloyd-Max, the Mean Square Gaussian, the Mean Square Laplacian, the Optimal Uniform Gaussian, the Optimal Uniform Laplacian, and several others. In this thesis, the uniform quantization technique is used. A more detail explanation of this technique is presented in Chapter 3.

In this study, the difference in textural features between onion and clod is used as the basis for separation during harvesting. A neural network is trained to learn the specific textural features of onion and clod. The structure of a neural network consists of simple processing units that are interconnected. The processing units called neurons are capable of storing information that are essential during neural network training. Each processing unit, characterized by a particular weight, carries an output signal that connects to one or many units. The output signal is defined by the activation function which control the amplitude range of the output signal to some finite value (Haykin 1999, Hecht-Nielsen 1989). During the training process of a neural network, the weight values

are constantly updated by analyzing the error signals calculated after each output signal. The learning rate and the number of iterations that a network is executed, control the neural network training process (Haykin 1999).

Roberts *et al.* (1991) have investigated and demonstrated the quantifying ability of a neural network to classify character image data. Because of its fully connected and interrelated structure, a multi-layer neural network is able to cope with inconsistencies and uncertainty that commonly appear on live image data. For pattern recognition, a multi-level backpropagation network (MLBPN) provides consistent and reliable results (Chen *et al.* 1994). The MLBPN architecture or structure is based on the significance of each feature extracted from the image. Chen *et al.* (1994) performed a feature extraction experiment. Using separate training and testing sets, five different types of ships were evaluated. It was proven that a MLBPN model recognized and classified different objects more accurately and efficiently. In addition, selecting the appropriate features will ensure optimal results.

A neural-network classifier has been successfully used for detecting vascular structures in angiograms (Nekovei *et al.* 1995). The study has showed that neural network could be used effectively to classify objects based on the image gray-scale data. The most crucial part of this classification is the preprocessing stage of image, since it is during this stage that the amount of data to be analyzed is systematically and efficiently reduced (Nekovei *et al.* 1995).

For neural network analysis, several factors must be considered: the network topology, the number of iterations during training, the learning rate, the training sample set, and the network initial weights (Haykin 1999, Hecht-Nielsen 1989, Nekovei *et al.*

1995). Nekovei *et al.* (1995) have reported the significance of the number of samples and the network topology that affect the classification performance of a neural network. Some of the concepts presented in this thesis were already used during the development of a neural network to detect blood vessels in angiograms. The neural network structure used by Nekovei consisted of a multilayer feed-forward network trained by the back-propagation algorithm. Pixel values taken from a center of an image were used as input to train the network. . In general, the selection of an appropriate input data set is essential for the success and implementation of a neural network classifier. The results have proved that neural network is an effective method for classification, regardless of the field in which it is applied.

## CHAPTER 3

### TEXTURE ANALYSIS

Because of the importance of selecting reliable data as input information to train a neural network, the textural feature measures computed from digitized images of onion and clod are used as the input data to the neural network separator. A general definition of texture is given as the arrangement or characteristics of the constituent elements of a surface (Landau 1990). A more technical definition useful for image processing defines texture as “an attribute representing the spatial arrangement of the gray levels of the pixels in a region” (IEEE Standard 1990). Jain (1988) defines texture as the spatial repetition of basic patterns. Each pattern is formed by an ensemble of pixels. Figure 3.1 shows an image of an onion and a clod. In the image of an onion, there are quasi-regular gray level transitions resulting in a distinct pattern texture that exhibits a visible quasi-regularity (Figure 3.1a). There is also a variation in gray level in a clod image. However, such variation exhibits a random texture, that is no regular pattern texture (Figure 3.1b). This was the motivation for using texture as the basis for separation of onion from clod.

While random textures can be characterized by statistical properties such as standard deviation and auto-correlation of gray level in an image, pattern textures require additional measurements to quantitatively characterize the nature and directionality of the



pattern. These measurements, called *texture features* (Haralick *et al.* 1973), are computed from the image of an object (onion or clod).

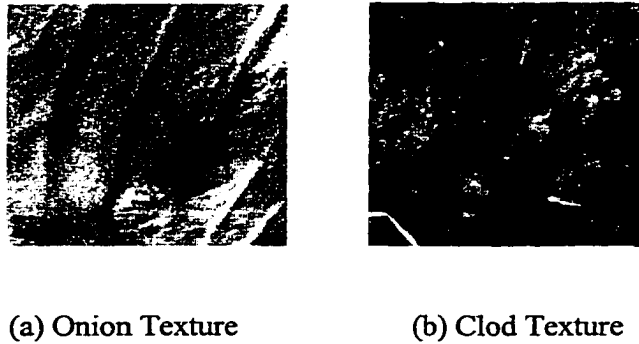


Fig. 3.1 Visual perception of the difference in surface texture between (a) an onion and (b) a clod

### 3.1 Image Quantization

The next step after image digitization is image quantization (Jain 1988). The main role of a quantizer is to map a continuous variable into a discrete one, and the methods to accomplish such a unique task may vary based on the type of quantization algorithm used.

The Optimum Mean Square, also known as Lloyd-Max quantizer, minimizes the mean square error for a given number of quantization (Jain 1988). The idea behind this quantizer is to distribute the mapping as uniformly as possible. In this study, the uniform quantization has been adopted. During uniform quantization a range of pixel values is set to a predetermined value. For example, the pixel values from 0 to 15 are set to 15, values from 16 to 30 are set to 30, and so on. Fig. 3.2 shows images of an onion before and after uniform quantization using different quantization levels.

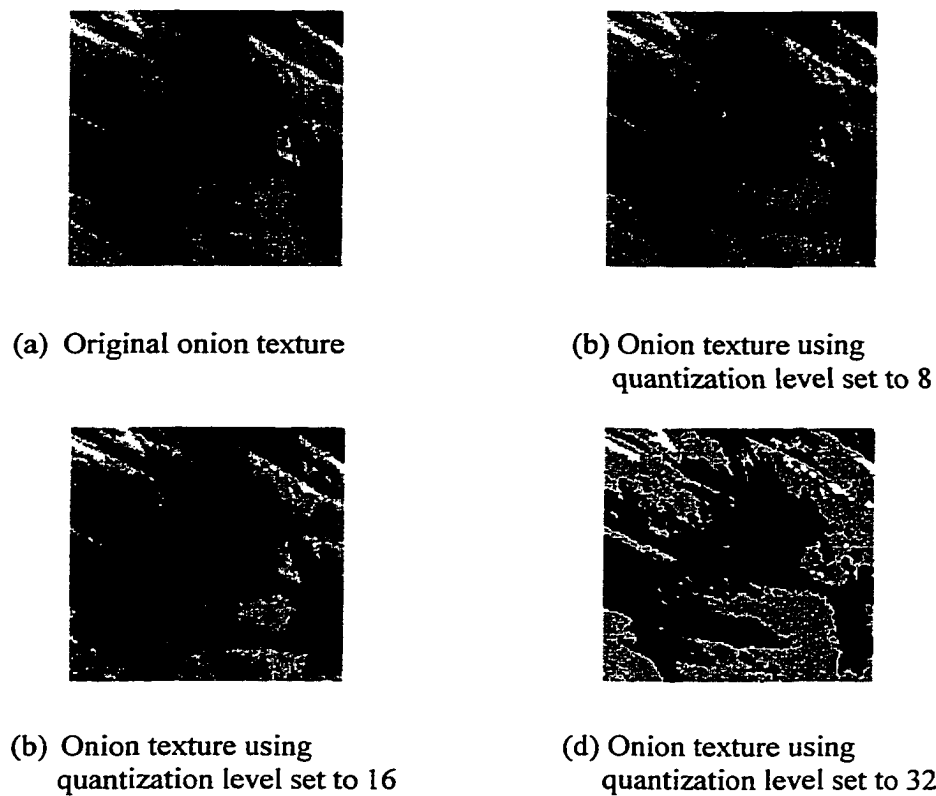


Fig. 3.2 Images of onion texture before and after using of different quantization levels

Since each image contained 256 pixel values (0 to 255), it was possible to select from four levels of quantization as shown in Table 3.1. The idea behind the quantization process is to find the smallest number of quantization levels that will maintain the characteristics of the image, while increasing the speed of image processing.

Table 3.1 Image quantization levels considered in this study

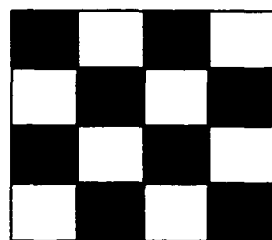
Resolution Values	Quantization Levels
0 to 7	8
0 to 15	16
0 to 31	32
0 to 63	64

For gray scale quantization of monochrome images, an event called contouring is likely to occur. When the number of quantization levels is insufficient, regions of constant gray levels result (Jain 1988).

### 3.2 The Co-occurrence Matrix

Based on the assumption that a pattern texture information of an image  $I$  contained in a spatial relationship between the gray levels of the image, the spatial relationship can be specified by a matrix  $P$ , called the *co-occurrence matrix* (Davis *et al.* 1981). The co-occurrence matrix is a square matrix of dimension  $N \times N$ , where  $N$  is the number of different gray levels in the image  $I$ . An element  $P_{ij}$  of a co-occurrence matrix  $P$  of an image  $I$  represents the relative frequency with which two pixels, one with gray level  $i$  and the other with gray level  $j$ , separated by a distance  $d$  along a direction  $\theta$ , occurs in an image. That is, an element  $P_{ij}$  is computed as the number of times the gray levels  $i$  and  $j$  occur in two pixels separated by the distance  $d$  ( $d=1$  pixel, or 2 pixels, etc.) along the direction  $\theta$  ( $\theta = 0^\circ$ , or  $45^\circ$ , or  $90^\circ$ , or  $135^\circ$  etc.) in an image  $I$ .

For a same image, different co-occurrence matrices can be formed for each combination of distance  $d$  and directional angle  $\theta$ . Once the co-occurrence matrices are formed, texture features can be computed. To illustrate the procedure of computing textural feature measures, a simple example of an image of 4 by 4 is considered (Fig. 3.3). The matrix in which the pixel values are stored contains 4 rows and 4 columns with only two possible values, 0 for black and 1 for white.



(a) Simple Texture

0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

(b) Simple Texture Matrix

Fig. 3.3 Simple surface texture and the matrix containing its corresponding pixel values

While the image represents some type of texture with certain pattern, the matrix stores the original pixel values maintaining the same pattern. Thus, by analyzing the original matrix, unique textural features of that image can be distinguished. To accomplish the task, several *gray-level co-occurrence matrices* are generated. These matrices represent the number of the same pixel pair in certain direction. The size of the co-occurrence matrix depends on the number of pixel values. In this simple example, the size of the co-occurrence matrix is 2 by 2, since there are only two possible pixel values, 0 and 1. Position (1,1) stores the number of pixel-pair with value (0,0), position (1,2) stores the number of pixel-pair with value (0,1), and so on.

Two different algorithms have been developed to count the number of pixel-pairs. The first algorithm is referred to the *symmetric algorithm*. It counts the number of pixel-pairs at a certain angle in both directions (up and down). The second algorithm, referred to as the *non-symmetric algorithm*, counts only the pixel-pairs in one direction (down). For example, in the symmetric algorithm at 45° angle (1,1 direction), there are five (0,0) and four (1,1) pixel-pairs, while there are no (0,1) and (1,0) pixel-pair (Table 3.2). In the non-symmetric algorithm at 45° angle (1,1), there are twice as many pixel-pairs.

Table 3.2. Co-occurrence matrices containing the number of pixel pairs at a distance of 1 and direction of 45° and 90°

Displacement (1,1) at 45° angle				Displacement (0,1) at 90° angle			
Using Non-Symmetric algorithm		Using Symmetric algorithm		Using Non-Symmetric algorithm		Using Symmetric algorithm	
5	0	10	0	0	6	0	12
0	4	0	8	6	0	12	0

### 3.3 Selecting Texture Features

Several co-occurrence matrix-based texture features have been proposed and tested in the literature (Davis *et al.* 1981, Haralick 1979, Tamura *et al.* 1978, Weszka *et al.* 1976). The texture feature measures considered in this study are reported in Table 3.3.

Table 3.3 Texture features considered in this study

Feature Measure	Formula
Homogeneity	$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{P_{ij}}{1+ i-j }$
Energy	$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (P_{ij})^2$
Contrast	$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i-j)^2 P_{ij}$
Variance	$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i-\mu)^2 P_{ij}$

For the variance feature measure, the mean or average gray level value  $\mu$  is given by

$$\mu = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} P_{ij}}{N^2} \quad (3.1)$$

where  $N$  is the total number of gray levels in an image.

Some texture features can be perceived by a human eye, such as contrast and homogeneity. However, many other texture features are insensitive to a human eye. It has been reported that a human eye cannot perceive texture differences higher than second order (Tamura *et al.* 1978). For the onion-clod separation, the selection of the texture features is based on some experimental investigation, the results of which are discussed in Chapter 5. The results show the usefulness of these features for effectively discriminating between an image of an onion and that of a clod.

### 3.4 Capturing Images

To collect enough images, the same onion and clod were used several times. However, the position of an onion and a clod were rotated after taking images for certain orientations. The procedure to capture each image consists of placing an onion or a clod on a 10 degree scale pie chart showing all the angles from  $0^\circ$  to  $360^\circ$ . After taking a picture, the onion or the clod was rotated to a different angle, and another image was taken. Each onion and clod was rotated six times, obtaining twelve different images; six images for onion and six images for clod. For this study, a total of two hundred images were used; one hundred images of onions and one hundred images of clods. Fig. 3.4 illustrates the method used to obtain the images of onions and clods.

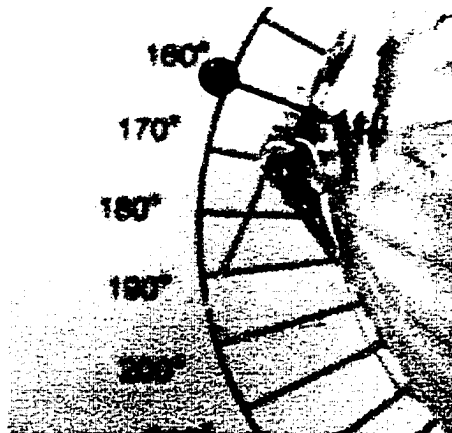


Fig. 3.4 Illustration of the method used to obtain all the different images of onions and clods

### 3.5 Process of Computing Texture Features

Since the texture of an object is independent of object orientation, texture features computed from co-occurrence matrices must produce the same separation results independently of the orientation of an object. To verify this, four different co-occurrence matrices corresponding to four different values of orientation angle  $\theta$  ( $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ ) were established for each image and for a specific value of distance  $d$ . Each feature measure is computed from the co-occurrence matrix. Figure 3.5 shows six images for different orientations of an onion. For each image, four co-occurrence matrices were established for the following combinations of distance and orientation angle ( $1,0^\circ$ ), ( $1,45^\circ$ ), ( $1,90^\circ$ ), and ( $1,135^\circ$ ). The contrast feature is computed for each image from the four co-occurrence matrices. The results are summarized in Table 3.4.

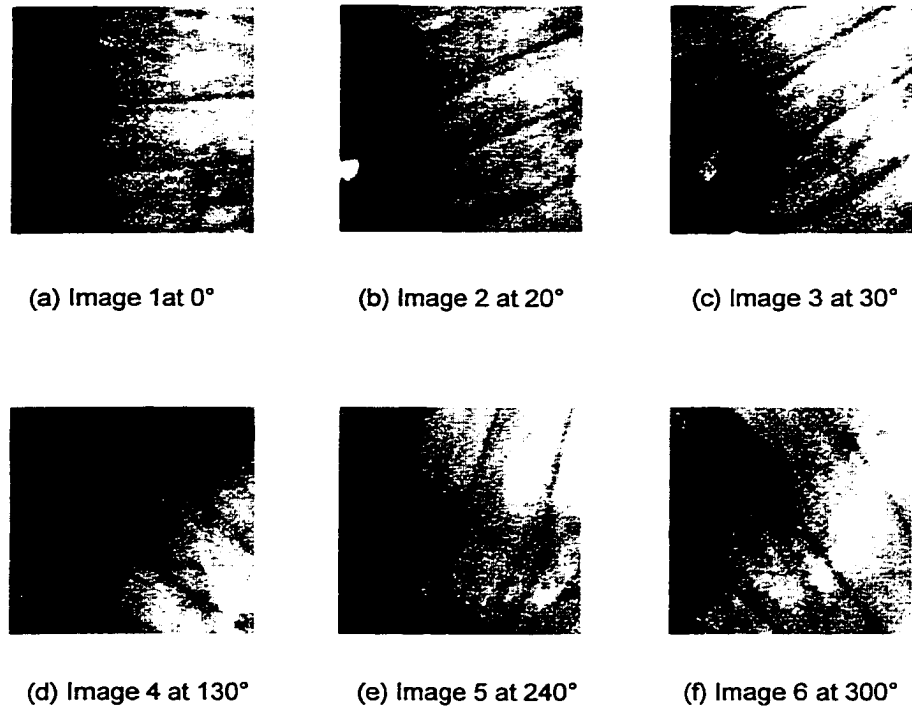


Fig. 3.5 Images showing six different orientation of an onion

It can be seen that for each of the distance and orientation angle combinations, different values of contrast features are obtained for the images of onion. However, the average values of the homogeneity feature for the six images are very close. This means that the average value is independent of orientation of onion. Consequently, for each textural feature, the average value is used as one of the inputs to the neural network separator, and not the four individual values of the textural feature. This is illustrated in Figure 3.6.



Table 3.4 Values obtained for the homogeneity feature for six different orientations of the image of an onion (the distance  $d$  is set equal to 1 for all cases)

Angle	Homogeneity Feature Image 1 at 0°	Homogeneity Feature Image 2 at 20°	Homogeneity Feature Image 3 at 30°
$\theta = 0^\circ$	0.8460	0.8447	0.8504
$\theta = 45^\circ$	0.8384	0.8170	0.8193
$\theta = 90^\circ$	0.8921	0.8872	0.8781
$\theta = 135^\circ$	0.8389	0.8544	0.8608
<b>Mean</b>	<b>0.8538</b>	<b>0.8508</b>	<b>0.8521</b>

Angle	Homogeneity Feature Image 4 at 130°	Homogeneity Feature Image 5 at 240°	Homogeneity Feature Image 6 at 300°
$\theta = 0^\circ$	0.8528	0.8602	0.8661
$\theta = 45^\circ$	0.8525	0.8365	0.8616
$\theta = 90^\circ$	0.8801	0.8723	0.8661
$\theta = 135^\circ$	0.8364	0.8480	0.8344
<b>Mean</b>	<b>0.8555</b>	<b>0.8543</b>	<b>0.8592</b>

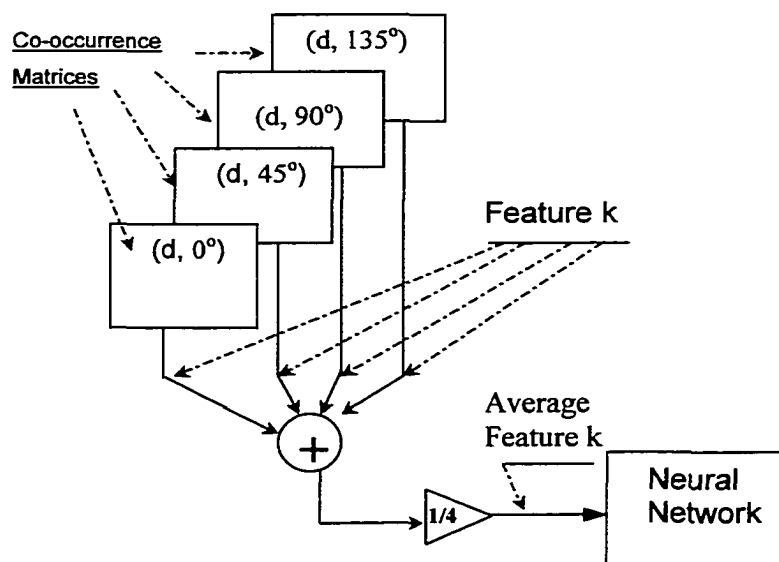


Fig. 3.6 Process of computing average values of textural features.

## CHAPTER 4

### NEURAL NETWORKS

This chapter describes the concepts of neural networks, various structures, and the back-propagation algorithm used in this thesis. This chapter lays the foundation and motivation for using neural networks to identify onion from clod for separation.

#### 4.1 Neural Network Justification

Neural networks have been applied to solve problems dealing with identification and classification. The reliability and versatility of neural network was used in many disciplines, such as digital communication, nuclear engineering, business, and weather forecasting (Chen *et al.* 1994, Haykin 1999, Hecht-Nielsen 1989, Nekovei *et al.* 1995). In medicine, neural network was used to detect blood vessel in angiograms, and in engineering, it has been applied to recognize different shapes of ships (Chen *et al.* 1994, Nekovei *et al.* 1995). Because of the problem presented in this study focuses on feature extraction, identification and separation, the use of a robust and reliable neural network system was thought about.

Some of the main reasons of using neural networks in such diverse fields are based on the characteristics and benefits that emerge from a well designed and robust parallel distributed processing structure. The most relevant aspects are summarized in Table 4.1 (Haykin 1999, Hecht-Nielsen 1989).

Table 4.1 Benefits and characteristic of a neural network structure.

CHARACTERISTICS	DESCRIPTION
Nonlinearity	This attribute makes neural network extremely versatile and allows its use in different fields such as medicine, business, and engineering among others.
I/O Mapping	It is used for the network to be trained and learned to provide a proper response under certain input information.
Adaptivity	The network are designed to adapt its weights to new changes, and constantly improve its capability.
Evidential Response	A robust neural network helps to select and rely on its decision
Contextual Information	Its structure exposes information globally since its response is based on a contextual data.

The concept of neuro-computing comes from the roots of the human brain thinking process (Haykin 1999, Hecht-Nielsen 1989). Unlike conventional digital computers, the human brain operates in a very unique way, where complex information is constantly analyzed generating different types of responses. Basically, the structural elements of the brain are the neurons (nerve cells) with infinite interconnections through which all sensory information flows. In a very general form, a neural network machine is designed to somehow model the human brain, and it is implemented using electronic components or software packages on conventional digital computers (Haykin 1999, Hecht-Nielsen 1989).

Analogously, as the human brain receives, analyzes, stores, and learns information; a neural network system is exposed, trained, and tested with certain type of information to process and recognize similar information. In other words, a neural network has the capability of storing data and making it available for future use. It simulates the human brain by acquiring knowledge through a learning process, and uses inter-neuron connections or connecting links to strengthen its weight.

Perhaps a more functional view of neural network is a signal-flow graph, where directed links are interconnected at certain points named nodes. A direct link starts at

node  $j$  and finishes at node  $k$ . A typical node  $j$  has its respective node signal  $X_j$ , and an associated function in which the input signal is transmitted to an output signal  $Y_k$ . There are three elemental rules that control a signal flow (Haykin 1999):

**RULE 1:** A signal flows along a link in one direction. There are two types of links:

a) *synaptic* (dictated by a linear input-output relation)



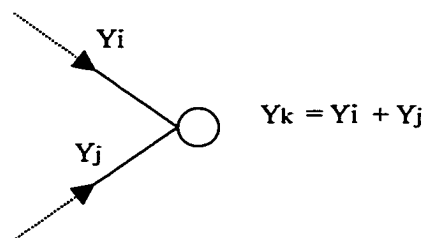
where node signal  $X_j$  is multiplied by a synaptic weight  $W_{kj}$  to produce the output node  $Y_k$ .

b) *activation* (dictated by a non-linear input-output relation)

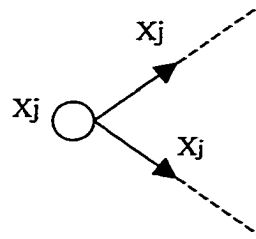


where node signal  $X_j$  is controlled by nonlinear activation function  $f(\cdot)$  to produce the output node  $Y_k$ .

**RULE 2:** A node signal equals the algebraic sum of all signals entering the appropriate node via the incoming links ("fans in").



**RULE 3:** The signal at a node is transmitted to each outgoing link originating from that node, with the transmission being completely independent of the functions of the outgoing links.



The description and rules of a directed graph reflect both the signal flow from neuron to neuron and the signal flow inside each neuron. This is just a simple model that helps to understand a neural network system.

## 4.2 Architecture of a Neural Networks

There are three types of neural network structures discussed in the thesis.

### 4.2.1 Single-layer feedforward network

The single layer feedforward network is the simplest of all. It consists of an input layer of source nodes projecting onto an output layer of neurons in one direction (Fig. 4.1) (Haykin 1999, Hecht-Nielsen 1989).

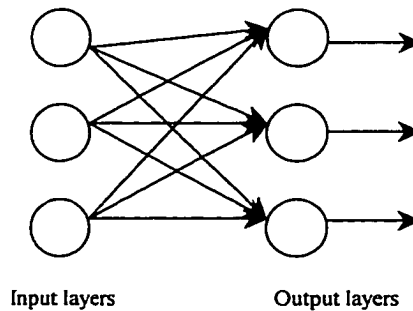


Fig. 4.1 Single-layer feedforward networks

### 4.2.2 Multilayer feedforward networks

The second type of feedforward network is the multilayer networks that differentiates from feedforward single-layer network with hidden layers between input and output layers. These hidden layers allow to model higher-order statistics by adding another dimension to the network (Fig. 4.2).

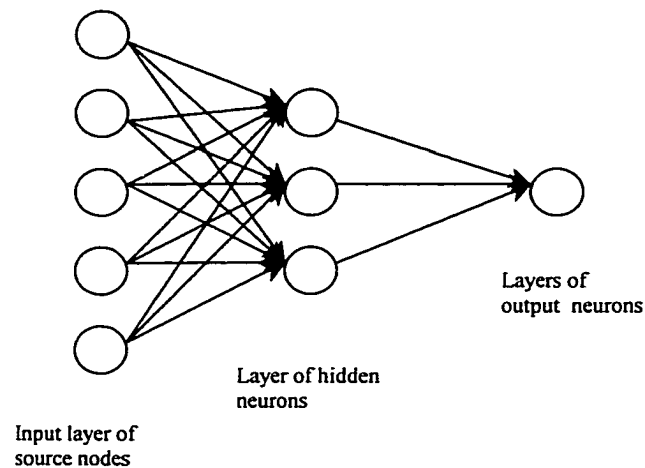
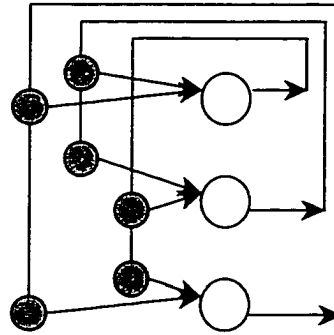


Fig. 4.2 Multi-layer feedforward networks

### 4.2.3 Recurrent networks

A third type of neural networks structure is known as recurrent network in which the output signal rather than feedforward is looped back to input neurons. This form of architecture has a deep impact on the learning capability of the network as well as on its performance.



Recurrent network with self-feedback

Fig. 4.3 Recurrent networks

### 4.3 Backpropagation Algorithm

For problems or applications where the input presented to a network has never been used, backpropagation algorithm is generally used. This selection is based on the fact that for backpropagation algorithm, new input tends to give similar output after the network has been trained with similar input values. These results are based on the type of function signals selected to train the network. In a multilayer neural network there are mainly two types of signals which are essential for the development of a network, the function signal and the error signal.

A backpropagation algorithm using two hidden layers is modeled in Fig. 4.4. The structure of the model is basically the same as the one used in this study,

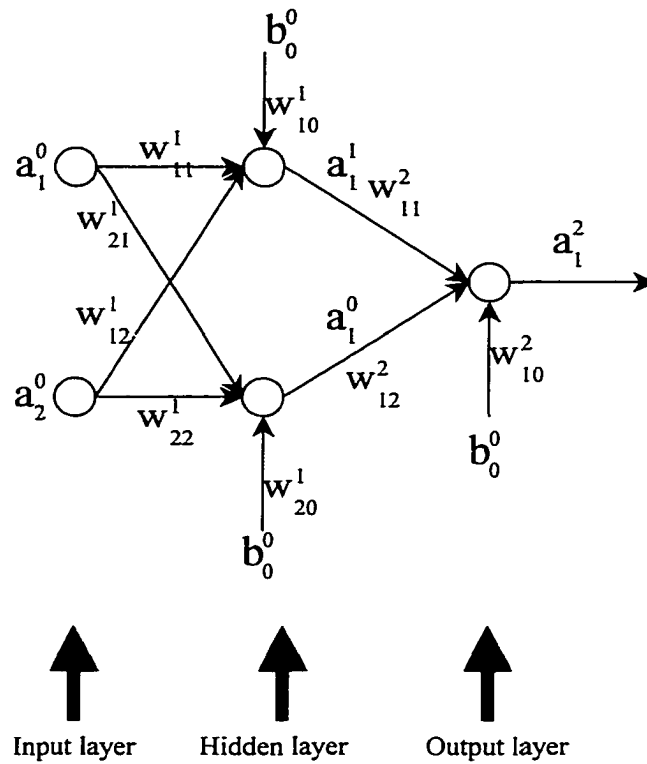


Fig. 4.4 A backpropagation algorithm model

where an input such as  $k$  indicates the layer,  $n$  the number of neurons,  $b$  the bias, and  $a_n^k$  indicates the  $w_{ni}^k$  input weight for each neuron,  $i$  indicates the input number. Thus, the value of a neuron  $n_n^k$  is determined by,

$$n_n^k = b_0^0 w_{n0}^k + a_n^0 w_{ni}^k + a_{n+1}^0 w_{n(i+1)}^k \quad (4.1)$$

and  $a_n^1$  is obtained by,

$$a_n^1 = f(n_n^k) \quad (4.2)$$

where  $f(n_n^k)$  is the transfer function that generates the outputs between 0 and 1.



#### 4.3.1 Training process

During the training phase, the weights were updated until one complete presentation of the entire training, called *epoch*. During the training process, two crucial passes of significant computations take place. The first pass is the forward pass in which the synaptic weights are not modified, and the function signals of the entire network are evaluated neuron by neuron. During the forward pass, the signal error of the desired output is computed. The second pass, known as the backward pass, begins at the output layer and passes the error signals layer by layer throughout the entire network.

The number of training cycles (epochs) as well as the learning rate parameter has a significant effect on the success of the neural network training process. During the course of this study, different epochs and learning rate values as well as numerous network topologies were tested until best results were obtained.

#### 4.3.2 Function and error signals

The function signal is an input signal that is set at the input end of a network. It moves forward through the network to each neuron, and turns out at the output end of the network as an output signal. The signal is related to the weights applied to each neuron (Haykin 1999). There are many different function signals, such as tan-sigmoid transfer function, linear transfer function, log-sigmoid transfer function, etc. The selection of the function signal depends on the type of problem to be solved.

An error signal occurs at the output neuron of a network, and it moves backward via layers through the network. The computation of an error signal originates in every neuron of the network, and it affects the input of the next neuron.

The error signal at the output of a neuron  $j$  at iteration  $n$  is given by

$$e_j(n) = d_j(n) - y_j(n) \quad (4.3)$$

where  $d_j(n)$  is the desired response for neuron  $j$ , and  $y_j(n)$  refers to the function signal at the output of a neuron (Haykin 1999).

The instantaneous value of error for neuron  $j$  is defined by,

$$\varepsilon(n) = \frac{1}{2} \sum_{j \in c} e_j^2(n) \quad (4.4)$$

where  $c$  includes all the neurons in the output layer of a network.

#### 4.4 Motivation for Using Neural Network

After computing the average of textural features for the 200 images of onion and clod, the values obtained were plotted in a scatter plot to determine whether the difference in texture given in Fig 3.1, was also observed in the scatter plot. In order to obtain complete information, combinations of each pair of textural feature were plotted. The scatter plots built for this analysis are shown in Fig. 4.5 through 4.10. The results from the plot indicate that there is difference between the textural features of onion and clod. However, it is also observed that several values overlapped. To successfully separate onion from clod, it is necessary to rely in a process that is capable of classifying new data based on previously learned information. Such a classification must be done accurately and flawlessly. This preliminary study lead to the application of neural networks for identification of onion from clod from vision based images. The next chapter describes the results of the texture based studies of onion and clod separation.

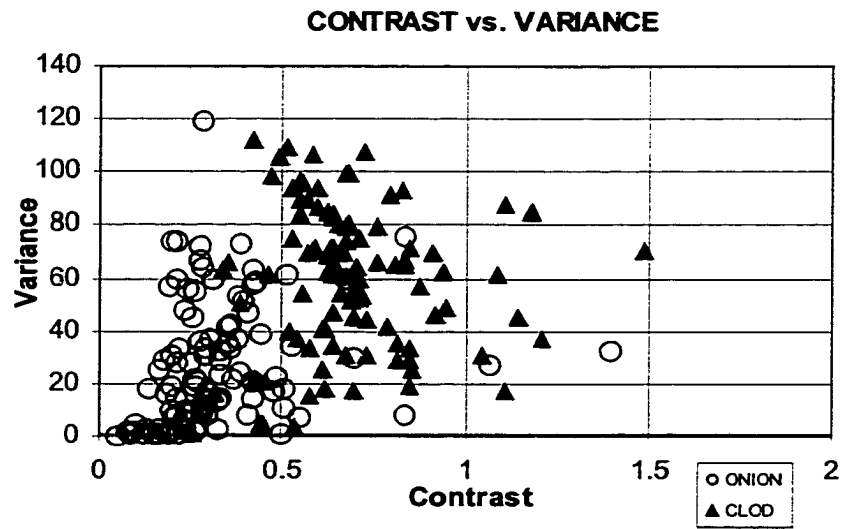


Fig. 4.5 Scatter plot of textural features contrast and variance

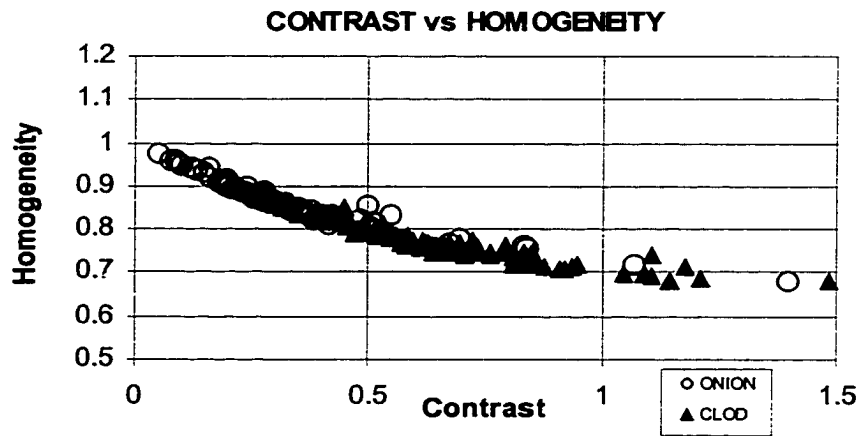


Fig. 4.6 Scatter plot of textural features contrast and homogeneity

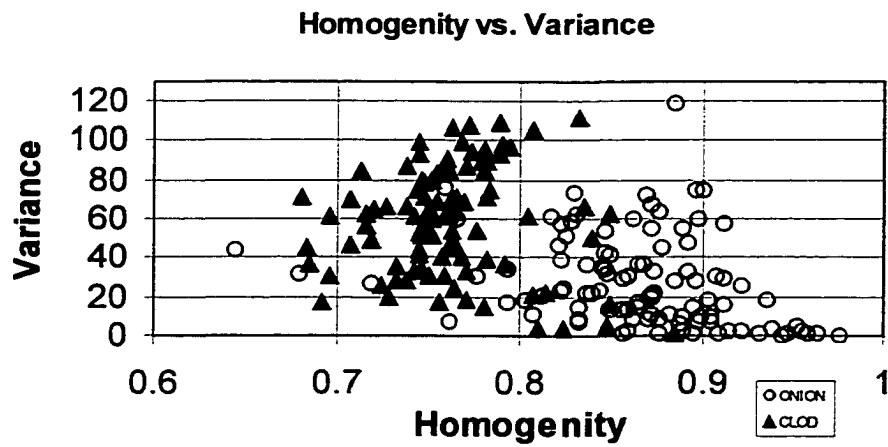


Fig. 4.7 Scatter plot of textural features homogeneity and variance

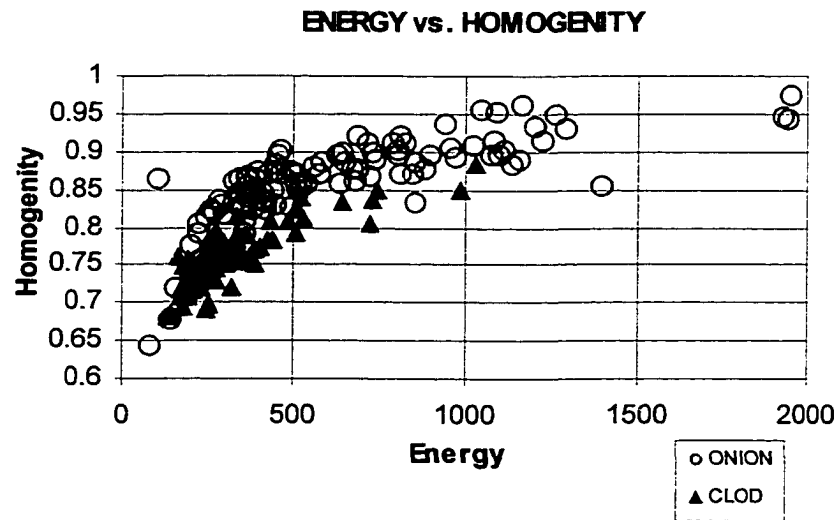


Fig. 4.8 Scatter plot of textural features energy and homogeneity

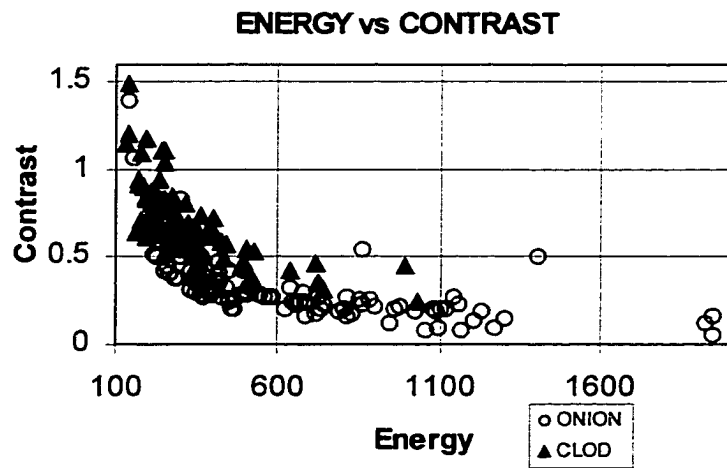


Fig. 4.9 Scatter plot of textural features energy and contrast

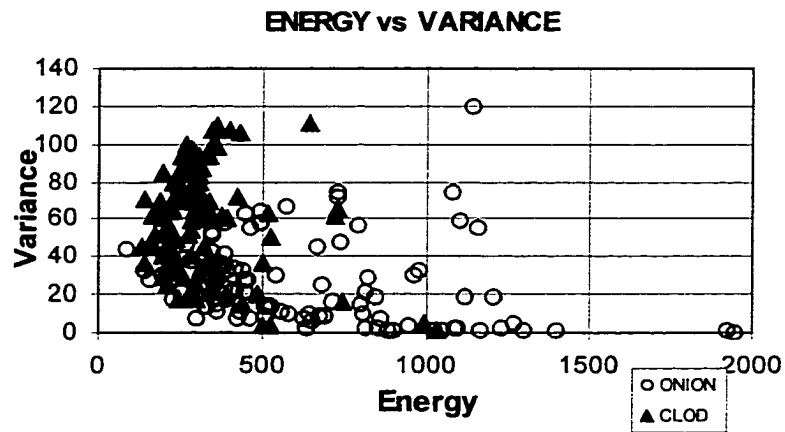


Fig. 4.10 Scatter plot of textural features energy and variance

## CHAPTER 5

### EXPERIMENTAL RESULTS

#### 5.1 Data Collection for Image Processing

Samples of onion and clod were collected after harvesting from a field in the Rio Grande Valley of Texas. One hundred (64 x 64) images of onions and one hundred (64 x 64) images of clods were taken using a vision system. All images were taken at 256 gray levels. To obtain large number of images, several images of onions and clods were taken with different orientations. Since the size of a co-occurrence matrix ( $N \times N$ ) was determined by the number of gray levels of an image, a high number of gray levels lead to excessive computations when evaluating all the entries of the matrix. Thus, an attempt was made to reduce the number of gray levels in the image before the co-occurrence matrices were established. Different quantization levels (8, 16, 32 and 64) were tested. It was found that if the gray levels of the images were quantized into 16 levels, efficient texture-based separation performance was obtained, while reducing processing time. Figure 5.1 shows images of onion and clod before and after quantization.

Table 5.1 Image size and resolution values considered in this study

<b>Image Size</b>	<b>Resolution Values</b>	<b>Quantization Level</b>	<b>Image Resolution</b>	<b>Later Data Analysis</b>
64	0 to 7	8	Excellent	Very Slow
64	0 to 15	16	Very Good	Acceptable
64	0 to 31	32	Poor	Very Fast
64	0 to 63	64	Very Poor	Fastest

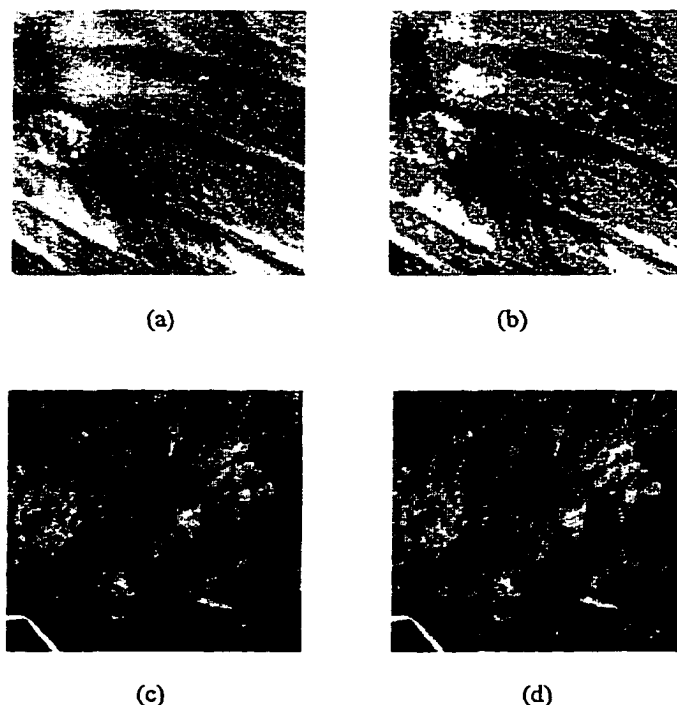


Fig. 5.1 Images of (a) onion with 256 gray levels, (b) onion with 16 gray levels, (c) clod with 256 gray levels, and (d) clod with 16 gray levels

The texture features namely, homogeneity, contrast, energy, and variance were computed for 200 images for the following combinations ( $d=1, \theta=0^\circ$ ), ( $d=1, \theta=45^\circ$ ), ( $d=1, \theta=90^\circ$ ), and ( $d=1, \theta=135^\circ$ ). These combinations were used to form the co-occurrence matrices. Average values of each texture feature were then obtained. The 800 values (400 texture features for onion and 400 texture features for clod) constitute the pool of data for the neural network separator. Half of the 800 values were used for training the neural network, and the other half was used for testing.

## 5.2 Separation Algorithm

### 5.2.1 Neural network structure

For this type of problem in which extracting higher-order statistics is of primary importance, a fully-connected multi-layer feed-forward neural network becomes an appropriate candidate for the separation task. The appearance of hidden layers containing a predefined number of hidden neurons is one of the major characteristics of this form of neural network. These hidden neurons intercede between the inputs and the outputs of the network, allowing flexibility and global perspective, despite their local connections (Haykin 1999). The neural network is fully connected, meaning that every neuron in each layer is connected to all the nodes in the previous layer.

Although in a multi-layer neural network structure, several hidden layers might be included between the input layer and the output layer, for this particular application, only one hidden layer was used. However, several network topologies were considered differing by the number of input nodes, the combination of textural features used as inputs to the network, and the number of neurons in the hidden layer. Thirty three neural network configurations were examined. A summary of the network configurations is given in Table 5.2.

Table 5.2 Neural network topologies examined in this study

Possible combinations of textural features used as inputs to the network	Network Topology
Contrast (Co), Energy (En), Homogeneity (Ho), Variance (Va)	
Co-En-Ho-Va	4-5-1
Co-En-Ho-Va	4-3-1
Co-En-Ho-Va	4-2-1
Co-En-Ho, Co-En-Va, Co-Ho-Va, En-Ho-Va	3-5-1
Co-En-Ho, Co-En-Va, Co-Ho-Va, En-Ho-Va	3-3-1
Co-En-Ho, Co-En-Va, Co-Ho-Va, En-Ho-Va	3-2-1
Co-En, Co-Ho, Co-Va, En-Ho, En-Va, Ho-Va	2-5-1
Co-En, Co-Ho, Co-Va, En-Ho, En-Va, Ho-Va	2-3-1
Co-En, Co-Ho, Co-Va, En-Ho, En-Va, Ho-Va	2-2-1



### 5.2.2 Neural network training

The single most significant aspect in the implementation of neural network is mapping of the training set of data to the output layer. Its significance directly influences the learning behavior and performance of the network. The activation function signal is induced by the log-sigmoid function that generates output between 0 and 1. The error signal is given by,

$$e_k(t) = d_k(t) - y_k(t) \quad (5.1)$$

where  $d_k(t)$  is the desired response for neuron  $k$ , and  $y_k(t)$  is the output signal of neuron  $k$  at iteration  $t$  (Haykin 1999).

Initial weights have a crucial role on the performance of the neural network training. For this study, the initial weight values were randomly assigned, and they were adjusted during training via the back-propagation algorithm (Maw *et al.* 1998).

A set of 400 textural feature values (200 images corresponding to onion and 200 images corresponding to clod) was used for neural network training. The training consists of (i) presenting the network with textural features measure to its input nodes, (ii) compute the corresponding network output, (iii) compute the error between the desired output and the actual output of the network, and (iv) back-propagate the error and adjust the network weights (Figure 5.2). After the neural network is trained and final values for the weights are obtained, its separation performance is tested first by using the training data set, and then, by using the new 400 test data that contains the 200 textural feature values of onion and 200 textural feature values of clod. This process was repeated for all

the 33 different neural network configurations and the results are presented in the next section. For each configuration, the training data set is selected from the 400 textural features (depending on the network topology and the corresponding combination of textural features used as inputs to the network).

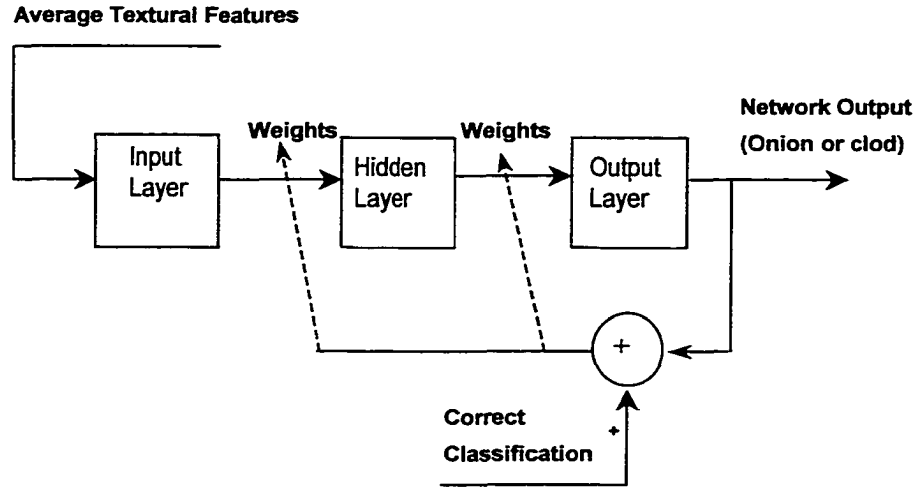


Figure 5.2 Architecture of the neural network separator

### 5.3 Separation results

The objective of the separation process is to recover onions and reject clods. We define the *recovery* variable,  $R_c$ , and the *rejection* variable,  $R_j$ , for onions and clods respectively as ratios given by,

$$R_c = \frac{O_p}{O_p + O_r} = 1 - \frac{O_r}{O_p + O_r} \quad (5.2)$$

and

$$R_j = \frac{C_r}{C_r + C_p} = 1 - \frac{C_p}{C_r + C_p} \quad (5.3)$$

where,

$O_p$  : onions in the product exits

$O_r$  : onions rejected with clods

$C_r$  : clods rejected

$C_p$  : clods remaining with onions.

The *separation effectiveness* ( $SE$ ) is calculated from the following product (Hecht-Nielsen 1989):

$$SE = 100 \times R_c \times R_j \quad (5.4)$$

The performance of the neural network as a separator was evaluated by computing its separation effectiveness,  $SE$ . The latter was evaluated for all the 33 neural network configurations. First, the performance of each neural network was evaluated by running the input data used for training. The results are reported in Figures 5.3-5.5.

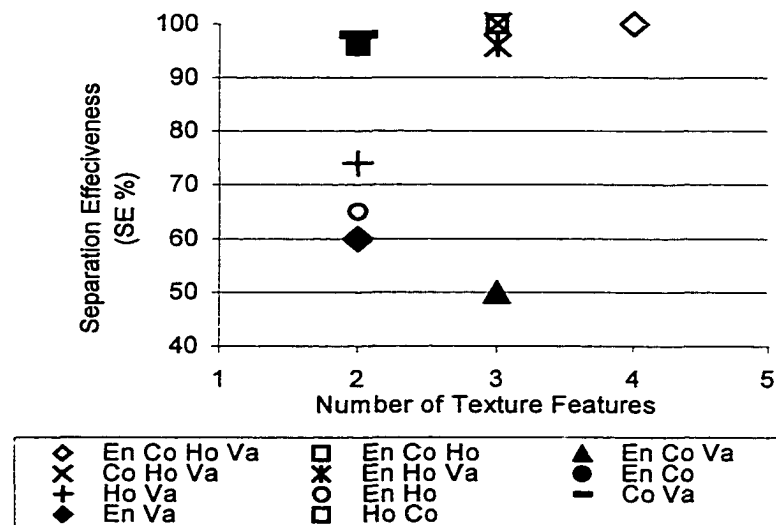


Fig. 5.3 Separation effectiveness of neural networks with 2 hidden neurons when they are presented with training data

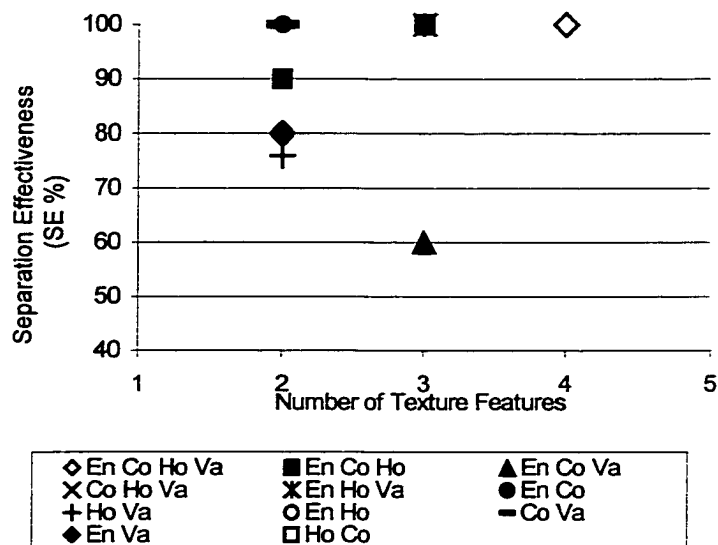


Fig. 5.4 Separation effectiveness of neural networks with 3 hidden neurons when they are presented with training data

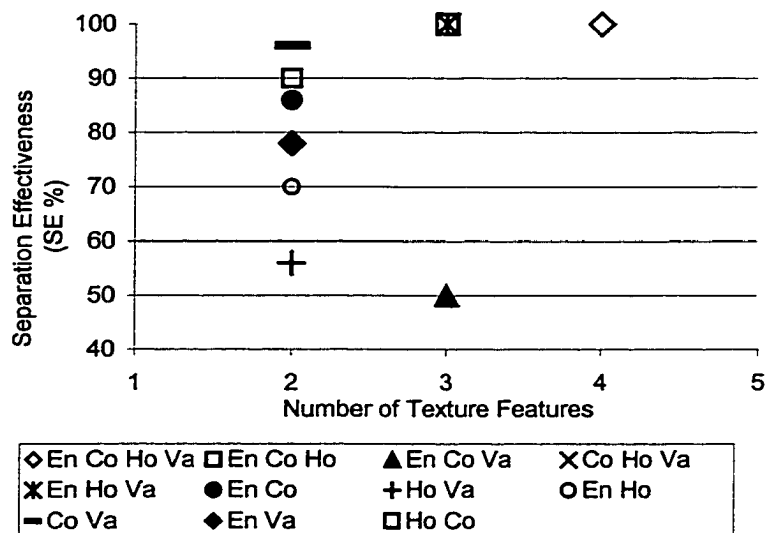


Fig. 5.5 Separation effectiveness of neural networks with 5 hidden neurons when they are presented with training data

The performance of each neural network was evaluated by running the new test data and the results are presented in Figures 5.6-5.8.

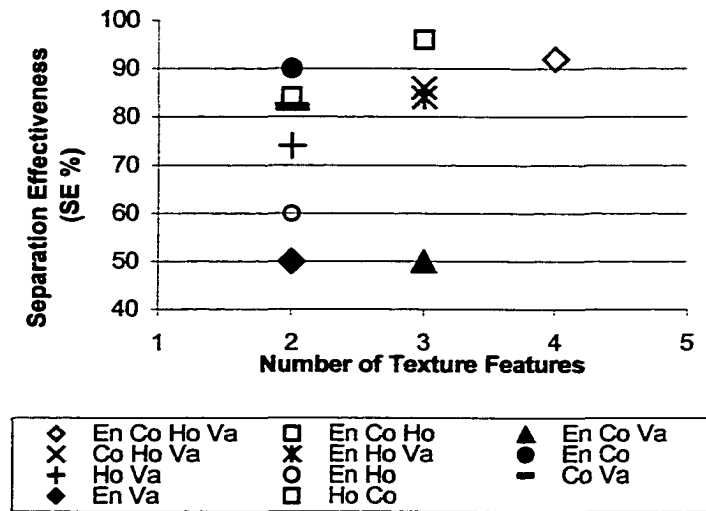


Fig. 5.6 Separation effectiveness of neural networks with 2 hidden neurons when they are presented with new testing data

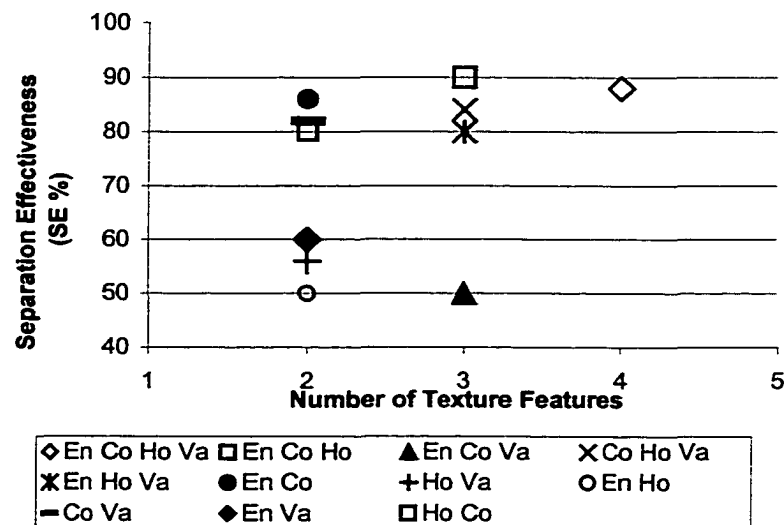


Figure 5.7. Separation effectiveness of neural networks with 3 hidden neurons when they are presented with new testing data

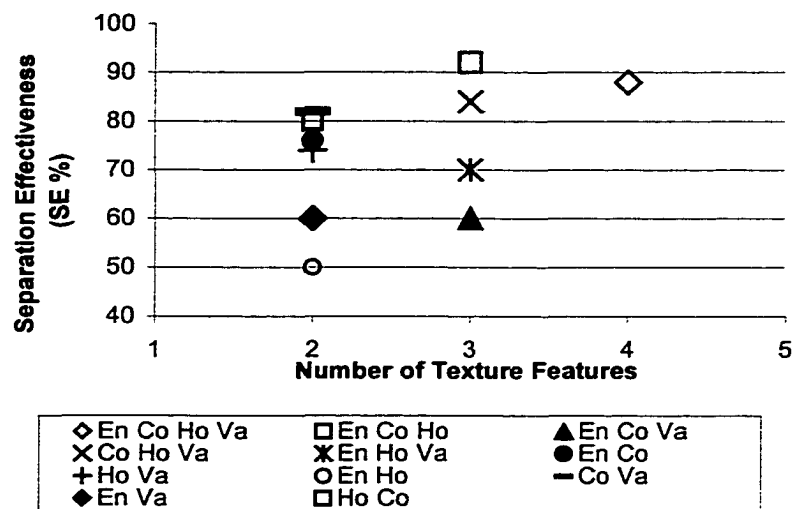


Figure 5.8. Separation effectiveness of neural networks with 5 hidden neurons when they are presented with new testing data

In several cases, separation effectiveness was equal to 100% when some neural networks were presented with the training data. The neural networks that achieved the maximum separation effectiveness were the following (Figures 5.3-5.8): (3-2-1 with inputs En-Co-Ho), (3-2-1 with inputs Co-Ho-Va), (4-2-1 with inputs En-Co-Ho-Va), (2-3-1 with inputs Co-Va), (2-3-1 with inputs En-Ho), (3-3-1 with inputs Co-Ho-Va), (3-3-1 with inputs En-Co-Va), (3-3-1 with inputs En-Ho-Va), (3-5-1 with inputs En-Co-Va), (3-5-1 with inputs En-Ho-Va), and (4-5-1 with inputs En-Co-Ho-Va).

When the neural networks were presented with new test data, the highest separation effectiveness was obtained for the following structure, 3-2-1 with inputs En-Co-Ho (Figure 5.6). The combination of textural features consisting of energy, contrast and homogeneity (En-Co-Ho) used as inputs, provided consistent high separation effectiveness for different network topologies; 3-2-1, 3-3-1, and 3-5-1, with separation effectiveness values of 96%, 90%, and 92% respectively (Figures 5.3-5.8).

The combination of textural features consisting of energy, contrast, homogeneity, and variance (En-Co-Ho-Va) provided the maximum number of inputs to the neural networks. However, the separation effectiveness was only 92% due to the fact that the variance feature generated a noise, rather than contributing additional information to the neural network.

If the separation effectiveness (SE) is evaluated based on the number of input textural features, we see from Figures 5.3-5.8 that in most cases using two textural features lead to low separation effectiveness, especially for the following combinations: En-Va, En-Ho, and Ho-Va. The simulation results also show that the performance of effectiveness for the combination En-Co-Va does not exceed 60%.

If the separation effectiveness (SE) is evaluated based on the number of hidden neurons of the neural network, it can be seen that the highest values of separation effectiveness are obtained with two hidden neurons. In addition, the number of hidden neurons has a significant effect on the network performance.

From all the cases examined, the network topology 3-2-1 with inputs En-Co-Ho gives the highest separation effectiveness with values reaching 100%, when the network was tested with training data, and 96% when network was tested with new test data.

## CHAPTER 6

### MECHANICAL INTEGRATION OF VISION SYSTEM

#### 6.1 Onion-Clod Separation Concepts

The proposed neural network-based vision system is planned to be integrated with a mechanical harvester system that has the capability to separate clods from agricultural produce during harvesting. Some of the different design concepts used to separate onions and clods are described.

##### 6.1.1 Rotating Brushes

As the name indicates, the concept of rotating brush is based on using brushes to “scrub” the onions to separate them from the clods. The brushes are positioned at a predetermined angle to promote forward movement along the axis of the cylinder. A simple schematic sketch of the rotating brushes concept is shown in Fig. 6.1 (Cuellar 2000).

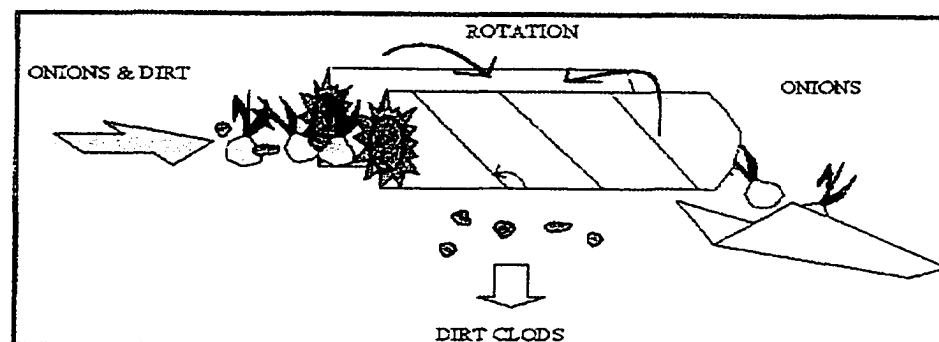


Fig. 6.1 Rotating Brushes



### 6.1.2 Restitution Method

The restitution method uses the concept of coefficient of restitution to separate onions from clods. Since onion has a greater coefficient of restitution than clod, it rebounds farther than clods when they are dropped onto a bouncing surface (Coble *et al.* 1976, Davis *et al.* 1981, Feller *et al.* 1984). A schematic sketch of the restitution method is shown in Fig. 6.2.

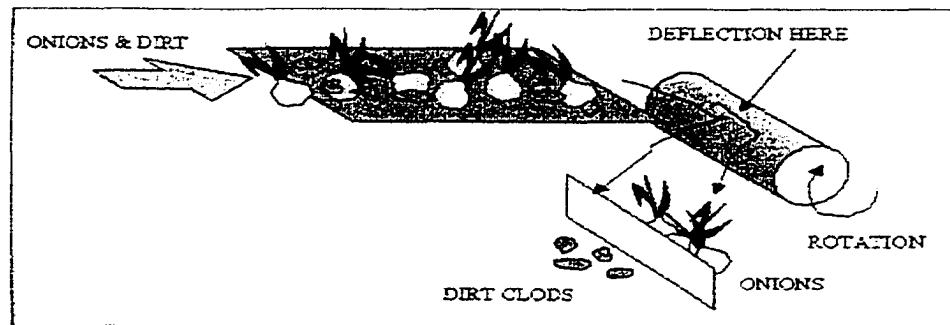


Fig. 6.2 Restitution Method

### 6.1.3 Rotating Cylinders

This concept consists of two rotating cylinders. The cylinders are inclined and have slots cut for the clods to fall through. Onions and clods are dropped at the top end of the cylinder. As the onions and clods rotate and bounce, clods break and fall through the slots. Onions roll down to the bottom of the cylinder (Cuellar 2000).

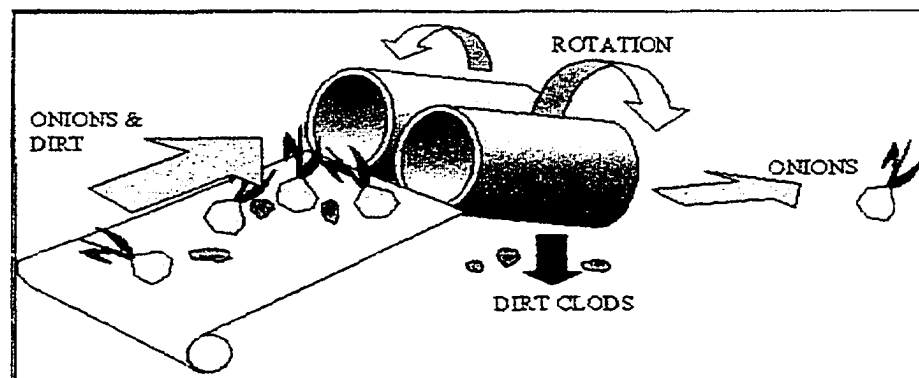


Fig. 6.3 Rotating Cylinder

## 6.2 Harvesting Process

Onions and clods are transported and dropped by a link type chain conveyor to an inclined feeder which has part of the surface solid and the rest has holes for clods to fall through. The impact of drop of onions and clods from the chain conveyor will break some of the clods, letting smaller clods to fall through the holes. The onions and clods are dropped from the feeder to a rotating drum, which has slots cut at predetermined distance. The inner surface of the rotating drum is glued with poron to prevent damage of onions. The length and diameter of the drum are designed such that most of the clods will break and fall through the slots (Coble *et al.* 1976, Cuellar *et al.* 2000) Figure 6.4 shows a 3-D view of the rotating drum.

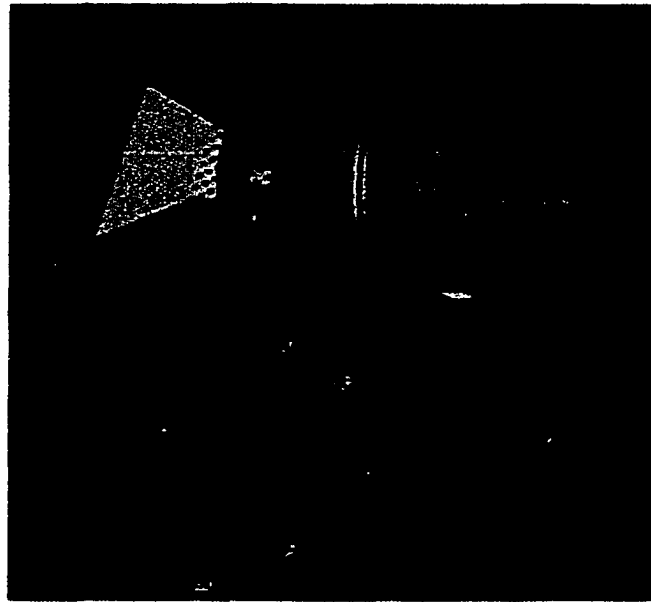


Fig. 6.4 Rotating drum that will be integrated with the cup type conveyor where the neural network-based vision system will be placed.

### 6.3 Cup Type Conveyor System

Any remaining clods are separated by the neural network-based vision system in a cup type conveyor system as they are transported for sorting and bagging. This conveyor will be linked to the rotating drum, shown in Fig.6.4. It consists of several rows of cups that will be filled with either onions or pieces of clods that were not separated by the rotating drum.

The neural network-based vision system will be placed at the end of the cup type conveyor, and it will be used to eliminate any piece of clod still remaining. This classification system will determine if the cup is holding an onion or a clod based on its texture analysis, and will trigger a signal that will open the cup to drop the clod. A 3-D model of the proposed cup type conveyor system is shown in Fig. 6.5.

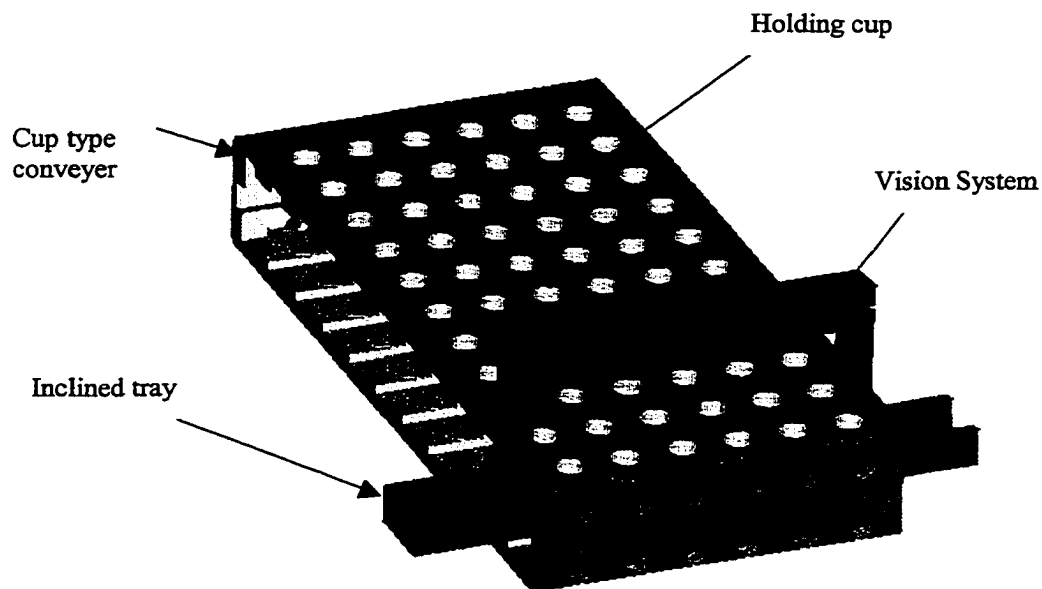


Fig 6.5 Cup type conveyor with a vision system

Although the exact specifications of the cup type conveyor go beyond the scope of this study, it is important to mention some of parameters that will have critical impact in the development phase of the conveyor. Some of these significant parameters are the width and length of the conveyor, the diameter of the cups, the number of cups in the conveyor, the distance between cups, the distance between the vision system and the produce, and the number of cameras to be used in the system. Figure 6.6 shows different views of the significant components to be considered during the design phase of the cup type conveyor.

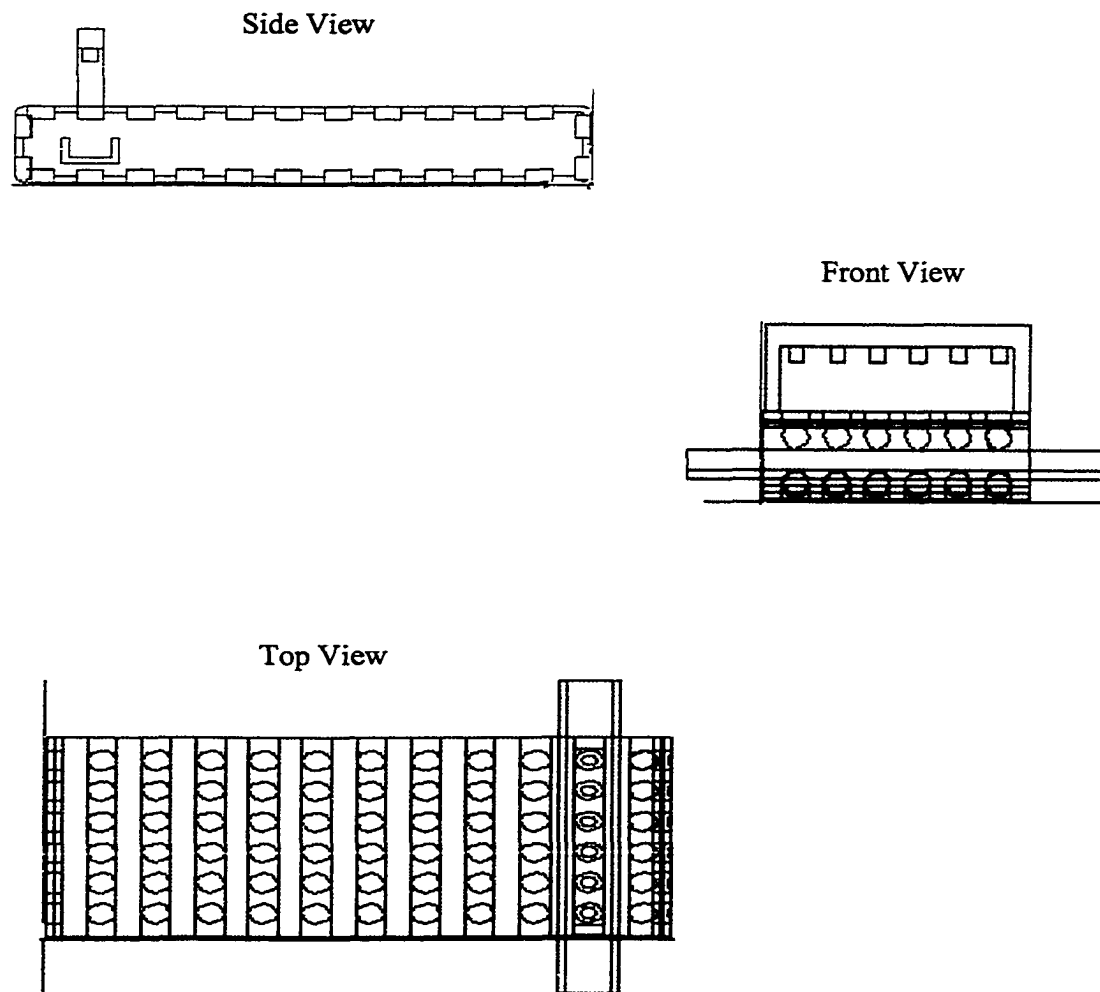


Fig. 6.6 Graphic representation of the cup type conveyor

## CHAPTER 7

### CONCLUSIONS

It has been successfully demonstrated that a neural network-based vision system could be used to separate onions from clods based on textural features. If a prototype of the neural network-based vision system is to be built, it will be added to the end of a mechanical harvester, as shown in the flow chart in Fig. 7.1.

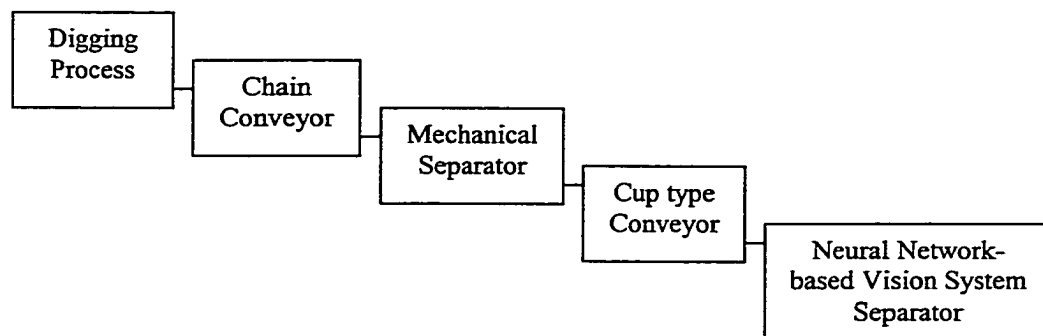


Fig. 7.1 Flow chart of the onion harvesting process

Thirty three network topology configurations were evaluated using combinations of the four textural features: homogeneity, contrast, energy, and variance. The neural network with 3-2-1 topology and inputs: energy, contrast, and homogeneity gave the best separation effectiveness of 96 percent. The least effective configurations were mostly resulted when only two inputs were used. There is a possibility to increase the number of textural features, since only four of the textural features were tested. When adding more

number of inputs to the neural network, the possibility to obtain better results will depend on the information provided by additional inputs, since it could generate unnecessary noise to the neural network as shown in Chapter 5. The cup type conveyor system with a neural network-based vision system should be accurately designed precisely, analyzed, and tested to obtain optimal results.

The proposed neural network-based vision system is intended to work with a new integrated mechanical harvester system. However, the scope of this system goes beyond the field in which it was applied in this thesis.

## REFERENCES

- Bolle, R.M., Connell, J.H., Haas, N., Mohan, R., and Taubin, G., 1996, "VeggieVision: A Produce Recognition System," Technical Report, IBM T.J. Watson Research Center, Yorktown Heights, NY, pp. 244-251.
- Brown, G.G., Katz, D., Foust, A.S., and Schneidewind R., 1951, *Unit Operations*, p.15 John Wiley and Sons, New York.
- Chen, C.Y., and Hwang, C.J., 1994, "A Multi-Level Backpropagation Network for Pattern Recognition System," *IEEE Transactions On Neural Network*, **4**, No. 2, pp. 78-82.
- Coble, C.G., Aldred, W.H., and Dilo, R.C., 1976, "Mechanized Harvesting System for Fresh Market Onions," Winter Annual Meeting of ASAE, Chicago, IL, pp. 14-17.
- Cuellar, E., Fielder, S., and Guidici, E., 2000, *Design of an Onion-Clod Separator*, Senior Design Report, University of Texas-Pan American, Edinburg, TX.
- Davis, L.S., Clearman, M., and Aggarwal, J.K., 1981, "An Empirical Evaluation of Generalized Cooccurrence Matrices," *IEEE Transactions Pattern Analysis and Machine Intelligence*, **2**, pp.214-221.
- Feller, R., Nahir, D., and Coble, C.G., 1984, "Separation of Soil Clods from Onions Using Impact," *Transactions. of ASAE*, **27**, No. 2, pp. 353-357.
- Haralick, R.M., 1979, "Statistical and Structural Approaches to Texture," *Proceedings of the IEEE*, **67**, No. 5, pp.786-804.
- Haralick, R.M., Shanmugam, R., and Dinstein, I., 1973 "Textural Features for Image Classification," *IEEE Transactions on Systems, Man and Cybernetics*, **3**, pp. 610-621.
- Haykin, S., 1999, *Neural Network a Comprehensive Foundation*, Prentice Hall, New Jersey.

- Hecht-Nielsen, R., 1989, *Neurocomputing*, Addison-Wesley Publishing Company, Menlo Park, California.
- IEEE Standard 610.4, 1990, *IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology*, IEEE Press, New York.
- Jain, A. K., 1988, *Fundamentals of Image Processing*, Prentice-Hall, New York.
- Landau S.I., (Ed.), 1990, *Webster Illustrated Contemporary Dictionary*, Doubleday, Garden City, NY.
- LePori, W. and Hobgood, P., 1970, "Mechanical Harvester for Fresh Market Onions," *Transactions of ASAE*, **13**, No. 4, pp. 517-519.
- Lynn Abbott, A., and Zhao, Y, 1996, "Adaptive Quantization of Color Space for Recognition of Finished Wooden Components," *IEEE Transactions on Signal Processing*, **4**, No. 12, pp. 252-256.
- Maw, B.W., Smittle, D.A., Mullinix, B.G., and Cundiff, J.S., 1998, "Design and Evaluation of Principles for Mechanically Harvesting Sweet Onions," *Transactions of ASAE*, **41**, No. 13, pp. 517-524.
- Nekovei, R. and Sun, Y., 1995, "Back-Propagation and its Configuration for Blood Vessel Detection in Angiograms," *IEEE Transactions On Neural Network*, **6**, No. 1, pp. 64-72.
- Roberts, A., and Yearworth, M., 1991, "A Comparison of Pre-Processing Transforms for Neural Network Classification of Character Images," *IEEE Transactions On Neural Network*, **12**, No. 3, pp. 189-192.
- Slater, D., Healey, G., Sheu, P., Cotman C.W., Su, J. Wasserman, A., and Shankle, R., 1996, "A Machine Vision System for the Automated Classification and Counting of Neurons in 3-D Brain Tissue Samples," *Proceedings of IEEE, Workshop on Application of Computer System*, pp. 224-229.
- Tamura, H., Mori, S., and Yamawaki, T., 1978, "Texture Features Corresponding to Visual Perception," *IEEE Transactions of Systems, Man, and Cybernetics*, **8**, pp. 460-473.
- Tao, Y., Morrow, C.T., Heinemann, P.H., and Sommer, H.J., 1995. "Fourier Based Separation Technique for Shape Grading of Potatoes Using Machine Vision," *Transactions of ASAE*, **38**, No. 3, pp. 949-957.
- Weszka, J.S., Dyer, C.R., and Rosenfeld, A., 1976, "A Comparative Study of Texture Measures of Terrain Classification," *IEEE Transactions of Systems, Man, and Cybernetics*, **6**, No. 4, pp. 269-285.



- Yamamoto, S., Jiang H., Matsumoto, M., Tateno, Y., Iinuma T., and Matsumoto, T., 1996, "Image Processing for Computer-Aided Diagnosis of Lung Cancer by CT (LSCT)," *Systems and Computers in Japan*, **25**, No. 2, pp. 236-241.
- Zhu, Z., Yang B., Xu, G., and Shi, D., 1996, "A Real-Time Vision System for Automated Traffic Monitoring Based on 2D Spatio-Temporal Images," *Proceedings of the IEEE, Workshop on Application of Computer System*, pp. 162-167.

## APPENDICES

J

## APPENDIX A

## TEXTURE ANALYSIS PROGRAM

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function COMBINES both the QUANTIFICATION process and %
% calculate the TEXTURAL FEATURES of images of %
% onion and clod %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
=====
square is the original picture matrix
=====
square=imread('c:\eye\closeups\Oni_Picl.tif');
imshow(square)
q_number = input('Enter Quantization number (32,64,128): ');
while (q_number ~= 32) & (q_number ~= 64) & (q_number ~= 128)
    disp ('You have entered a wrong Quantization number ');
    q_number = input('Enter Quantization number (32,64,128): ');
end
%q_number = 64;
quant_m =zeros(q_number,q_number); %(128,128)
limit_matrix = q_number % 128
%imshow(square)
%limit_stats = 17 %128

=====
Reduce is a new matrix that reduces the quantized image
=====
reduce=zeros(q_number,q_number); % 128
reduce_limit = q_number; % 128 % The reduce_limit is the reduce image
center = ((256 - q_number)/2) - 1
tic % BEGIN timing for reducing original image
for row = 1:reduce_limit
    %i=i+1;
    for col = 1:reduce_limit
        reduce(row,col)= square(row+center,col+center);
    end
end
toc % END time for reducing original image
disp (' Above are the seconds it takes to reduce original image')
reduce_pic=(mat2gray(reduce)); % pic1 is the matrix after grayscale
conversion

=====
Rotate reduce image to be a quantized image
=====
answer= input('Do you want to rotate the image? (Enter: 1 for YES ; 0
for NO): ');
if answer == 1
    deg = input('How many degrees do you wish to be rotated: ');
    rot = imrotate(reduce,deg,'nearest','crop');

```

```

rotate_pic = (mat2gray(rot));
for row = 1:reduce_limit
    for col = 1:reduce_limit
        reduce(row,col)= rot(row,col);
    end
end

elseif answer == 0
    disp('There was NO rotation on the image');
    deg = 0;
else
    disp('You have entered an incorrect answer');
end

```

```

=====
This is the loop to quantimize the original matrix (containing
pixel 0 to 255) into a new matrix called TRANS.
Its size still will be 256 x 256, but
the pixel values will only be 51, 102, 153,204, or 255
Then, another statistical matrix of size 5 x 5 will be
generated to analyze pixel patterns.
=====

```

```

tic % BEGIN timing for image QUANTIMIZATION
pixel = 15;
gr_level = input ('Please enter the gray level desired (8,16,32,64):
');
while (gr_level ~= 8) & (gr_level ~= 16) & (gr_level ~= 32) &
(gr_level ~= 64)
    disp ('You have entered a wrong gray level !!! ');
    gr_level = input ('Please enter the gray level desired
(8,16,32,64): ');
end
for row = 1:limit_matrix
    %i=i+1;
    for col = 1:limit_matrix
        %j=j+1;
        %if (col+dy) <= limit_matrix
        %if (row+dx) <= limit_matrix
        if reduce(row,col)<= pixel
            quant_m(row,col)= pixel;
        elseif reduce(row,col)<= pixel + gr_level
            quant_m(row,col)= pixel + gr_level;
        elseif reduce(row,col)<= pixel + 2*gr_level
            quant_m(row,col)= pixel + 2*gr_level;
        elseif reduce(row,col)<= pixel + 3*gr_level
            quant_m(row,col)= pixel + 3*gr_level; %255 with 64 gr_level
        elseif reduce(row,col)<= pixel + 4*gr_level
            quant_m(row,col)= pixel + 4*gr_level;
        elseif reduce(row,col)<= pixel + 5*gr_level
            quant_m(row,col)= pixel + 5*gr_level;
        elseif reduce(row,col)<= pixel + 6*gr_level
            quant_m(row,col)= pixel + 6*gr_level;
        elseif reduce(row,col)<= pixel + 7*gr_level
            quant_m(row,col)= pixel + 7*gr_level; %255 with 32 gr_level
        elseif reduce(row,col)<= pixel + 8*gr_level
            quant_m(row,col)= pixel + 8*gr_level;

```

```

elseif reduce(row,col)<= pixel + 9*gr_level
    quant_m(row,col)= pixel + 9*gr_level;
elseif reduce(row,col)<= pixel + 10*gr_level
    quant_m(row,col)= pixel + 10*gr_level;
elseif reduce(row,col)<= pixel + 11*gr_level
    quant_m(row,col)= pixel + 11*gr_level;
elseif reduce(row,col)<= pixel + 12*gr_level
    quant_m(row,col)= pixel + 12*gr_level;
elseif reduce(row,col)<= pixel + 13*gr_level
    quant_m(row,col)= pixel + 13*gr_level;
elseif reduce(row,col)<= pixel + 14*gr_level
    quant_m(row,col)= pixel + 14*gr_level;
elseif reduce(row,col)<= pixel + 15*gr_level
    quant_m(row,col)= pixel + 15*gr_level; %255 with 16 gr_level
elseif reduce(row,col)<= pixel + 16*gr_level
    quant_m(row,col)= pixel + 16*gr_level;
elseif reduce(row,col)<= pixel + 17*gr_level
    quant_m(row,col)= pixel + 17*gr_level;
elseif reduce(row,col)<= pixel + 18*gr_level
    quant_m(row,col)= pixel + 18*gr_level;
elseif reduce(row,col)<= pixel + 19*gr_level
    quant_m(row,col)= pixel + 19*gr_level;
elseif reduce(row,col)<= pixel + 20*gr_level
    quant_m(row,col)= pixel + 20*gr_level;
elseif reduce(row,col)<= pixel + 21*gr_level
    quant_m(row,col)= pixel + 21*gr_level;
elseif reduce(row,col)<= pixel + 22*gr_level
    quant_m(row,col)= pixel + 22*gr_level;
elseif reduce(row,col)<= pixel + 23*gr_level
    quant_m(row,col)= pixel + 23*gr_level;
elseif reduce(row,col)<= pixel + 24*gr_level
    quant_m(row,col)= pixel + 24*gr_level;
elseif reduce(row,col)<= pixel + 25*gr_level
    quant_m(row,col)= pixel + 25*gr_level;
elseif reduce(row,col)<= pixel + 26*gr_level
    quant_m(row,col)= pixel + 26*gr_level;
elseif reduce(row,col)<= pixel + 27*gr_level
    quant_m(row,col)= pixel + 27*gr_level;
elseif reduce(row,col)<= pixel + 28*gr_level
    quant_m(row,col)= pixel + 28*gr_level;
elseif reduce(row,col)<= pixel + 29*gr_level
    quant_m(row,col)= pixel + 29*gr_level;
elseif reduce(row,col)<= pixel + 30*gr_level
    quant_m(row,col)= pixel + 30*gr_level; %255 with 8 gr_level
end
end
end
toc % END timing for image QUANTIMIZATIONdisp ('Above are
seconds it takes for image QUANTIMIZATION')The building-in function
MAT2GRAY converts atrices to a grayscale intensity
image.quant_pic1=(mat2gray(quant_m)); % pic1 is the matrix after
grayscaleverition


---


%imshow(square), figure, imshow(reduce_pic), figure, imshow(quant_pic1)
if answer == 1
imshow(reduce_pic), figure, imshow(rotate_pic), figure, imshow(quant_pic1),
figure, imshow(square)

```

```

else
    imshow(reduce_pic), figure, imshow(quant_pic1), figure, imshow(square)
end

```

```

=====
The STATS MATRIX will hold the values of the NEW STATISTICAL MATRIX
=====

```

```

limit_matrix = reduce_limit;
% limit size of the statistical matrix
if gr_level == 64
    limit_stats = 4
elseif gr_level == 32
    limit_stats = 8
elseif gr_level == 16
    limit_stats = 16
else
    limit_stats = 32
end
limit_count = input('How many DISPLACEMENT you wish to run ? : ');
for count = 1:limit_count % limit_count holds the times the program
                        %will run
stats=zeros(limit_stats,limit_stats);

```

```

=====
dy and dx will hold the displacement values enter by the user
The displacement intends to reflect any type of patterns that
will be unique from each feature
=====

```

```

dx= input('Enter displacement for rows: ');
dy= input('Enter displacement for columns: ');
tic
while dx > limit_matrix | dy > limit_matrix
    if dx > limit_matrix
        disp('The number of row displacement must be <=
limit_matrix');
        dx= input('Enter displacement for rows : ');
        disp(' ');
    end
    if dy > limit_matrix
        disp('The number of column displacement must be <=
limit_matrix');
        dy= input('Enter displacement for columns: ');
        disp(' ');
    end
end
end

```

```

=====
Stats Matrix Results
=====

```

```

i_range=0; % i_range keeps track of the PIXEL value in the i_row
i=1; % i counts the row position on the statistical matrix
while i <= limit_stats
    j=1; % j counts the column position on the statistical matrix
    j_range=0; % j_range keeps track of the PIXEL value in the j_column
    while j <= limit_stats

```



```

fprintf(sta, 'PICTURE ROTATION:  ');
fprintf(sta, '%5.0f  ', deg);
fprintf(sta, '\n');
fprintf(sta, 'QUANTIZATION NUMBER\n');
fprintf(sta, ' (%1.0f)', q_number);
fprintf(sta, '\n');
fprintf(sta, 'GRAY LEVEL\n');
fprintf(sta, ' (%1.0f)', gr_level);
fprintf(sta, '\n');
end
fprintf(sta, '\n');
fprintf(sta, '\n');
fprintf(sta, 'DISPLACEMENT \n');
fprintf(sta, ' (%1.0f, %2.0f)', displace);
fprintf(sta, '\n');
fprintf(sta, 'MATRIX METHOD 1, NO SYMMETRIC\n');
fprintf(sta, '\n');

for i = 1:limit_stats % printing all row values
    for j = 1:limit_stats % printing all column values
        fprintf(sta, '%5.0f', stats(i, j));
        fprintf(sta, ' ');
    end
    fprintf(sta, '\n');
end

fprintf(sta, '\n');
fprintf(sta, 'MATRIX METHOD 2, SYMMETRIC\n');
fprintf(sta, '\n');

for i = 1:limit_stats % printing all row values
    for j = 1:limit_stats % printing all column values
        fprintf(sta, '%5.0f', co_stats(i, j));
        fprintf(sta, ' ');
    end
    fprintf(sta, '\n');
end
fclose(sta); % CLOSE STATISTICAL EXTERNAL FILE %

toc % END timing for creating STATISTICAL MATRIX
disp (' Above are the seconds it takes for calculate the STATISTICAL
      MATRIX')
stats

=====
Obtaining the values for ENTROPY, ENERGY, CONTRAST, and HOMOGENEITY
Those values are sent to tne SCREEN as well as to an EXTERNAL FILE
=====
norm = 0;
co_norm = 0;
for i = 1:limit_stats
    for j = 1:limit_stats
        norm = stats(i, j) + norm;
        co_norm = co_stats(i, j) + co_norm;
    end
end
end

```



```

Norm_Matrix = 1/norm;
Co_Norm_Matrix = 1/co_norm
Entropy = 0;
Energy = 0;
Contrast = 0;
Homogeneity = 0;
Check = 0;
Co_Energy = 0;
Co_Contrast = 0;
Co_Homogeneity = 0;
Co_Check = 0;

for i = 1:limit_stats
    for j = 1:limit_stats

        Energy = Norm_Matrix*(stats(i,j)^2)+ Energy;
        Contrast = Norm_Matrix*(i-j)^2 * stats(i,j) + Contrast;
        Homogeneity = Norm_Matrix*stats(i,j)/(1+abs(i-j))+ Homogeneity;
        Check = Norm_Matrix*stats(i,j) + Check;
        if stats(i,j) == 0
            log_tropy = log10(0.000001);
        else
            log_tropy = log10(stats(i,j));
        end
        Entropy = - Norm_Matrix*(stats(i,j))*log_tropy;
        -----
        Co_Energy = Co_Norm_Matrix*(co_stats(i,j)^2)+ Co_Energy;
        Co_Contrast = Co_Norm_Matrix*(i-j)^2 * co_stats(i,j) +
            Co_Contrast;
        Co_Homogeneity = Co_Norm_Matrix* co_stats(i,j)/(1+abs(i-j))+
            Co_Homogeneity;
        Co_Check = Co_Norm_Matrix*co_stats(i,j) + Co_Check;
        if co_stats(i,j) == 0
            Co_log_tropy = log10(0.000000000001);
        else
            Co_log_tropy = log10(co_stats(i,j));
        end
        Co_Entropy = - Co_Norm_Matrix*(co_stats(i,j))*Co_log_tropy;

    end
end

TP = Entropy
E=Energy
C=Contrast
H=Homogeneity
Co_TP = Co_Entropy
Co_E=Co_Energy
Co_C=Co_Contrast
Co_H=Co_Homogeneity

=====
    Sending the Data Results to an external file
=====
fid= fopen('Oni_P1_Try.txt','a');

```

```

if count == 1
    fprintf(fid, 'PICTURE ROTATION:  ');
    fprintf(fid, '%5.0f  ', deg);
    fprintf(fid, 'degrees');
    fprintf(fid, '\n');
    fprintf(sta, 'QUANTIZATION NUMBER\n');
    fprintf(sta, '(%1.0f)', q_number);
    fprintf(sta, '\n');
    fprintf(sta, 'GRAY LEVEL\n');
    fprintf(sta, '(%1.0f)', gr_level);
    fprintf(sta, '\n');
    fprintf(fid, '\n');
    fprintf(fid, 'DISPLACEMENT      Entropy      Energy      Contrast
Homogeneity\n');

end
fprintf(fid, '\n');
fprintf(fid, ' (%1.0f, %2.0f)', displace);
fprintf(fid, ' %14.4f', TP);
fprintf(fid, ' %11.4f', E);
fprintf(fid, ' %12.4f', C);
fprintf(fid, ' %10.4f\n', H);
fprintf(fid, ' %21.4f', Co_TP);
fprintf(fid, ' %11.4f', Co_E);
fprintf(fid, ' %12.4f', Co_C);
fprintf(fid, ' %10.4f\n', Co_H);
fprintf(fid, '\n');

fclose(fid);
end

```

## APPENDIX B

## NEURAL NETWORK TRAINING PROGRAM

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEMIAN MORQUIN
%CREATED ON: 3/4/2000
%LAST MODIFIED ON :3/27/200
%
%TITLE: This is the Onion-Clod TRAINING Neural Network
%
% This is a function use to TRAIN the neural networks.
% Its read the TRAINING data from an external file
% containing all the textural feature previously calculated.
% All the values are normalized before feeding the network.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid1 = fopen('Train_Data_OrdEnCoHo.dat','r'); % Read training data
                                         from a file
fid2 = fopen('Tr_Norm_dataECH.dat','w'); % Normalize data and sending
                                         to file
All_Data = fscanf(fid1,'%f'); % Matrix holding training data

N = max(size(All_Data));

l = 1;
for k = 1:4:N, % Sorting data into appropriate group
    ENE(l) = All_Data(k); % Energy data
    CON(l) = All_Data(k+1); % Contrast data
    HOM(l) = All_Data(k+2); % Homogeneity data
    Suc(l) = All_Data(k+3); % Onion (1) or Clod (0)
    l=l+1;
end;
ENE_max = max(ENE) % Find the MAX value of the ENERGY Data
ENE_min = min(ENE) % Find the MIN value of the ENERGY Data

ENEn = (ENE - ENE_min) / (ENE_max - ENE_min); % Normalize the values
                                         of ENERGY Data between
                                         0 & 1

Data=[ENEn; CON; HOM; Suc]
PData=[ENEn; CON; HOM];
TData=[Suc];
fprintf(fid2,'%6.3f %6.3f %6.3f %7.0f\n',Data); % Sending Normalized
                                         data

fclose(fid1);
fclose(fid2);

```

```

=====
CREATING THE NEURAL NETWORK. THIS IS A 3-2-1, using TANSIG function,
LEARNING RATE = 0.95;
INPUTS WEIGHTS, LAYER WEIGHTS, & BIASES are RANDOM;
EPOCHS = 2000;
GOAL = 10^-10;
=====

rand('state',0);
net = newff([0 1;0 1;0 1],[2 1],{'tansig','tansig'};%,'traingdm');
net.layers{1}.initFcn = 'initwb';
net.layers{2}.initFcn = 'initwb';
net.inputweights{1,1}.initFcn = 'rands';
net.layerweights{2,1}.initFcn = 'rands';
net.biases{1,1}.initFcn = 'rands';
net.biases{2,1}.initFcn = 'rands';
net=init(net);
net.trainParam.epochs=800;
net.trainParam.lr=0.90;
%net.trainParam.mc=0.90;
net.trainParam.show=200;
net.trainParam.goal=10^-4;
net= train(net,PData,TData);
a=sim(net,PData)
IW = net.IW{1,1}
LW = net.LW{2,1}
Bias1 = net.b{1,1}
Bias2 = net.b{2,1}
fid4 = fopen('Values_Ob.dat','w');
fprintf(fid4,'InputWeights = IW\n');
fprintf(fid4,'%3.4f \n',IW);
fprintf(fid4,'LayersWeights = LW\n');
fprintf(fid4,'%3.4f \n',LW);
fprintf(fid4,'Biases{1,1} = Bias1\n');
fprintf(fid4,'%3.4f \n',Bias1);
fprintf(fid4,'Biases{2,1} = Bias2\n');
fprintf(fid4,'%3.4f \n',Bias2);
fclose(fid4);

fid3 = fopen('Results_ECH_2hd.dat','w');
fprintf(fid3,'%1.4f\n',a); % Sending Normalized data
fclose(fid3);

```

## APPENDIX C

## NEURAL NETWORK TESTING PROGRAM

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEMIAN MORQUIN
%CREATED ON: 3/4/2000
%LAST MODIFIED ON :3/27/200
%
%TITLE: This is the Onion-Clod TESTING Neural Network %
%
% This is a function use to TEST the neural networks. %
% Its read the TESTING data from an external file %
% containing all the textural feature previously calculated.%
% All the values are normalized before feeding the network. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid1 = fopen('Test_Data_EnCoHo.dat','r'); % Read testing data from a
file
fid2 = fopen('TS_Norm_ECHdata.dat','w'); % Normalize data and sending
to file
All_Data = fscanf(fid1,'%f'); % Matrix holding training
data

N = max(size(All_Data));

l = 1;
for k = 1:4:N, % Sorting data into
appropriate group
    ENE(l) = All_Data(k); % Energy data
    CON(l) = All_Data(k+1); % Contrast data
    HOM(l) = All_Data(k+2); % Homogeneity data
    Suc(l) = All_Data(k+3); % Onion (1) or Clod (0)
    l=l+1;
end;
ENE_max = max(ENE)
ENE_min = min(ENE)
ENEn = (ENE - ENE_min) / (ENE_max - ENE_min);

Data=[ENEn; CON; HOM; Suc]
PData=[ENEn; CON; HOM];
TData=[Suc];
fprintf(fid2,'%6.3f %6.3f %6.3f %7.0f\n',Data); % Sending Normalized
data

fclose(fid1);
fclose(fid2);

=====
Neural Network TESTING
=====
net = newff([0 1;0 1;0 1],[2 1],{'tansig','tansig'})%,'traingdm');

```

```

net.layers{1}.initFcn = 'initwb';
net.layers{2}.initFcn = 'initwb';
net.IW{1,1}= [484.5 -172 -2234.4;-18.6 113.6 -110.9];%[-0.7326 3.8766
23.7110;14.4053 18.4210 -69.7602;-2.5320 2.5671 30.5420];%[341.2 -121.1
-1576;-79.2 168.9 -39];%[670.3 -238.1 -3091.5;-21.6 177.4 -50.2];%[-
0.3097 -0.3252 -0.5820;0.8535 0.7620 0.5775;0.8447 -0.8774 -
0.9614];%[0.0283 0.7751 -0.5519;0.9142 3.9069 -1.4578];%[-57.0555 -
22.6698 -77.1848;5.6151 23.7439 -5.9540;61.3307 25.8418 73.5409;-9.3047
21.9858 91.4168;-4.6100 23.8665 96.7689];(Tansig);
net.LW{2,1}= [-113.7983 -6.8028];%[-25.97 0.1681 53.4458];%[-65.9741 -
2.9137];%[-188.9312 -3.1826];%[-0.3219 0.0606 -0.4363];%[0.8386 -
2.1771];%[-179.7782 -31.4636 -179.8332 -347.5578 339.5677];(Tansig);
net.b{1,1}= [1799.5;70.8];%[-24.6135;40.2970;-
27.6758];%[1269.9;20.6];%[2489.2;-0];%[0.4990;0.4176;-
0.6257];%[0.4560;0.0373];%[86.1326;-2.7873;-86.2285;-86.2285;-
93.6938];(Tansig);
net.b{2,1}= [120.6202];%[27.6758];%[68.9579];%[192.1879];%%[-
0.003];%[1.6788];%%[23.4365];(Tansig);
net=init(net);
%net.trainParam.epochs=1000;
%net.trainParam.lr=0.95;
%net.trainParam.show=200;
%net.trainParam.goal=10^-20;
%net= train(net,PData,TData)
a=sim(net,PData);
for i= 1:100
    fid3 = fopen('TestResults_EnCoHo_h2.dat','w');
    if a(i) <= 0.6
        output(i) = 0;
        fprintf(fid3,'%1.0f\n',output);%Sending adjusted data result
    else
        output(i) = 1;
        fprintf(fid3,'%1.0f\n',output);%Sending adjusted data result
    end
end
fid4 = fopen('Test_a_Results_EnCoHo.dat','w');
fprintf(fid4,'%1.4f\n',a); % Sending actual data result
fclose(fid4);
fclose(fid3);

```

## APPENDIX D

TEXTURAL FEATURES DATA  
FOR CLOD

#	ENERGY	CONTRAST	HOMOGENEITY	VARIANCE
1	718.5612	0.4642	0.8045	61.4548
2	724.9751	0.3519	0.8361	65.7615
3	372.7242	0.6249	0.7612	62.2533
4	250.9673	0.5237	0.7839	75.3392
5	323.466	0.6226	0.7663	68.9289
6	280.3443	0.7095	0.7518	59.5805
7	291.3216	0.5454	0.7826	89.3343
8	346.1635	0.5818	0.7634	106.8427
9	273.5763	0.548	0.7812	84.3505
10	344.5827	0.6374	0.7601	61.5854
11	429.4928	0.4893	0.807	105.7077
12	337.1694	0.522	0.7827	39.4142
13	250.4478	1.0435	0.6965	30.9102
14	638.8095	0.4221	0.8329	111.5517
15	228.6157	0.8476	0.7419	33.6736
16	512.706	0.3354	0.8492	63.0805
17	294.0656	0.6356	0.7635	71.7299
18	178.0358	1.0888	0.6959	61.5928
19	219.3381	0.7611	0.7381	66.2645
20	209.5203	0.8747	0.715	56.9111
21	420.7048	0.5877	0.7831	71.4867
22	389.4334	0.6724	0.7501	60.1548
23	141.3657	1.4863	0.6805	70.783
24	228.8446	0.8462	0.7483	71.0771
25	195.6635	1.1791	0.7129	84.5702
26	356.9236	0.7319	0.7587	30.3886
27	202.0833	0.7299	0.7452	44.0746
28	275.5815	0.6869	0.7526	51.3829
29	502.9622	0.5437	0.792	37.1234
30	167.8411	0.9378	0.7151	62.1852
31	194.0336	0.8363	0.7199	64.7978
32	233.5895	0.9478	0.7187	49.1863
33	243.9755	0.629	0.7661	70.9878
34	269.7039	0.5705	0.7694	69.221
35	260.9232	0.6386	0.7593	84.01
36	332.8701	0.5269	0.7893	93.769
37	248.7738	1.1069	0.7384	87.5791
38	194.4835	0.8316	0.7265	66.6719
39	265.3631	0.7921	0.7601	91.1517
40	363.1337	0.514	0.7899	109.2036
41	303.7211	0.6495	0.7516	80.0333
42	243.9437	0.6756	0.7524	79.3381
43	250.5278	0.8304	0.7451	93.4119
44	314.5505	0.5951	0.7715	86.6324
45	277.9423	0.6252	0.7614	84.917
46	316.2946	0.8087	0.7192	65.3068

	ENERGY	CONTRAST	HOMOGENEITY	VARIANCE
47	308.4548	0.5976	0.7741	93.924
48	190.1943	0.909	0.7071	69.5302
49	399.5499	0.7245	0.7729	107.4966
50	275.5253	0.7106	0.7435	74.6789
51	338.9216	0.6699	0.7541	69.11
52	263.2887	0.6821	0.7448	99.2229
53	289.2371	0.5564	0.7817	95.3309
54	224.7824	0.7007	0.7511	64.5405
55	242.4447	0.6756	0.7454	74.3892
56	304.9091	0.6301	0.7645	63.1745
57	520.5531	0.3871	0.8396	50.6889
58	741.4699	0.319	0.849	16.0177
59	234.7713	0.7594	0.748	79.0517
60	208.4736	0.713	0.7453	52.9858
61	195.1987	0.7039	0.742	60.9065
62	279.6895	0.6617	0.7633	54.4163
63	207.2689	0.6931	0.7498	55.2262
64	322.7725	0.6934	0.7658	45.1063
65	340.895	0.6707	0.7687	99.018
66	302.0633	0.6412	0.7548	83.4233
67	256.0238	0.5833	0.7664	70.1414
68	291.5922	0.6433	0.76	64.2786
69	278.6849	0.5452	0.7951	96.571
70	227.4159	0.6798	0.7466	80.0687
71	297.8983	0.5714	0.7795	89.2926
72	359.4097	0.471	0.7904	98.3153
73	175.5498	0.709	0.7465	51.9441
74	282.2568	0.5572	0.7773	54.2472
75	210.7514	0.857	0.7236	25.2503
76	388.5284	0.6177	0.7709	18.3009
77	258.3324	0.8112	0.7327	28.6438
78	197.6927	0.8461	0.7376	28.5863
79	138.3788	1.2078	0.6852	36.7462
80	208.7629	0.8126	0.7326	35.4851
81	296.4984	0.6972	0.7561	17.5828
82	165.1034	0.6421	0.761	46.9822
83	245.4117	1.1074	0.6918	17.3308
84	524.6874	0.5331	0.8103	3.9134
85	130.9072	1.1466	0.6832	45.267
86	435.3058	0.573	0.7813	15.1298
87	990.1651	0.4502	0.8483	4.8636
88	502.4659	0.439	0.8244	4.2872
89	273.4109	0.8487	0.7285	19.1252
90	306.06	0.6768	0.75	30.7308
91	1030.3289	0.2504	0.8846	0.7642
92	335.1038	0.4305	0.8153	22.0666
93	277.5689	0.6182	0.7681	40.6216
94	293.5765	0.6105	0.7648	24.8586
95	167.6965	0.9212	0.707	46.0255
96	221.3961	0.6408	0.7464	34.7689
97	315.2105	0.5742	0.7714	33.6675
98	227.4302	0.7857	0.7445	41.6495
99	485.4915	0.4528	0.808	20.4279
100	195.3459	0.6124	0.758	40.956



TEXTURAL FEATURES DATA  
FOR ONION

#	ENERGY	CONTRAST	HOMOGENEITY	VARIANCE
1	461.2366	0.2615	0.8722	55.1006
2	295.1193	0.5104	0.8172	61.7682
3	341.0161	0.3918	0.8263	51.4447
4	408.4915	0.3677	0.8369	21.8098
5	730.0752	0.1996	0.9002	74.4959
6	1099.7807	0.2077	0.8962	59.6831
7	785.7397	0.1873	0.9112	57.0508
8	845.5414	0.2596	0.8714	19.3948
9	569.9889	0.2729	0.8712	66.9653
10	489.415	0.2813	0.8754	64.0216
11	1080.8552	0.2105	0.8957	74.449
12	1157.6632	0.2366	0.8887	55.7744
13	799.8471	0.2114	0.8945	14.7864
14	231.0239	0.837	0.7583	76.3207
15	810.4945	0.2671	0.8723	20.9485
16	350.3564	0.384	0.8248	24.745
17	388.2727	0.2981	0.8646	17.5408
18	384.258	0.3508	0.8491	41.4832
19	340.8092	0.3596	0.8465	42.4271
20	343.0831	0.3764	0.846	53.398
21	723.4346	0.2719	0.8686	72.3162
22	538.6203	0.2904	0.859	30.8246
23	359.6037	0.5059	0.8079	10.8198
24	294.8199	0.8318	0.7611	7.8263
25	153.4732	1.0681	0.7187	27.5127
26	682.933	0.1591	0.9206	25.3852
27	822.5324	0.1785	0.9107	29.2138
28	300.2307	0.6732	0.7663	59.8316
29	361.3523	0.2878	0.8643	14.6367
30	685.3386	0.2497	0.8762	9.1858
31	326.9176	0.3079	0.8626	59.6809
32	448.7731	0.4203	0.8314	62.8036
33	973.7845	0.2196	0.8915	33.0756
34	415.3757	0.4812	0.8226	22.9773
35	673.6074	0.2974	0.8615	8.2678
36	554.8776	0.2786	0.8812	11.1934
37	736.8728	0.2329	0.8907	48.0164
38	381.2363	0.3528	0.8455	35.3307
39	667.6915	0.2509	0.878	44.8782
40	1263.6564	0.0987	0.9507	4.9584
41	579.6406	0.2783	0.888	9.9678
42	435.6618	0.3304	0.8482	32.2131
43	354.3381	0.4806	0.7936	17.4748
44	809.2444	0.1609	0.9201	2.7862
45	896.9673	0.2175	0.8947	1.16
46	344.5799	0.3005	0.8645	37.0557

#	ENERGY	CONTRAST	HOMOGENEITY	VARIANCE
47	284.3894	0.3791	0.8363	37.2359
48	504.8665	0.2947	0.8598	13.2615
49	455.0888	0.2117	0.8949	27.6139
50	1088.1655	0.1973	0.9141	1.902
51	327.4504	0.4238	0.8323	14.2564
52	1295.6626	0.1475	0.9304	1.0105
53	364.8716	0.3619	0.8392	21.5274
54	412.5534	0.3556	0.8464	33.8381
55	1203.1394	0.13	0.935	18.2936
56	381.8784	0.332	0.8443	23.3401
57	490.2953	0.4247	0.8279	58.4447
58	82.7128	2.5948	0.645	44.4942
59	1230.0939	0.1916	0.9137	1.9385
60	257.9965	0.4446	0.8227	39.144
61	1117.5684	0.1985	0.9023	18.6174
62	521.5808	0.3263	0.8545	13.4385
63	649.9416	0.2298	0.8872	6.2937
64	511.7614	0.3316	0.8488	14.0339
65	1943.2586	0.1581	0.9423	0.534
66	142.1463	1.3987	0.68	32.4007
67	946.8991	0.1293	0.937	3.1589
68	1923.9843	0.1179	0.9457	0.7519
69	362.0975	0.3304	0.8564	28.8669
70	426.2654	0.2787	0.8711	11.4484
71	1398.9531	0.5004	0.8558	0.8472
72	448.977	0.2606	0.8735	21.6473
73	623.6946	0.2114	0.897	7.7883
74	1947.7252	0.0503	0.9748	0.321
75	464.078	0.2006	0.9039	7.2936
76	713.5554	0.1817	0.9108	16.2713
77	959.025	0.1996	0.9064	30.5625
78	379.6271	0.4223	0.8232	57.4432
79	290.6593	0.3874	0.8297	73.1315
80	197.9952	0.6963	0.7768	30.0981
81	858.4797	0.5464	0.8322	7.6406
82	641.1246	0.2411	0.8983	9.5747
83	1050.5551	0.0864	0.9568	1.2224
84	883.8392	0.26	0.8764	1.6995
85	802.9143	0.1986	0.9039	9.5297
86	392.6223	0.2897	0.8725	33.0505
87	505.2815	0.2812	0.8705	8.9012
88	265.9359	0.4076	0.821	46.7186
89	853.2931	0.2323	0.888	2.6365
90	446.3145	0.2473	0.8845	28.2762
91	418.0852	0.4051	0.8326	8.1681
92	1138.5702	0.2777	0.8837	119.5143
93	224.1864	0.5025	0.8034	18.1951
94	1094.0735	0.0914	0.9543	2.0128
95	365.135	0.275	0.8671	36.4069
96	1166.9927	0.0759	0.9621	1.3734
97	246.2769	0.418	0.8131	20.8695
98	1025.9877	0.1913	0.9083	1.31
99	632.995	0.3222	0.859	2.4563
100	361.3523	0.2878	0.8643	14.6367

## VITA

Demian Morquin obtained a bachelor of science degree with a major in computer science and a minor in mathematics in December, 1993. He graduated from the Honors Program at the University of Texas-Pan American, and pursued a challenging career in the education field as an educator and a coach. He has been a successful math and computer science teacher at Sharyland and McAllen High Schools. In addition, he has contributed to the sports field as the head boys soccer coach at both the high schools. He has excelled as a student as well as an athlete throughout his entire educational career. He was recognized as an All American Scholar in 1991 and 1992. He will be graduating from the Engineering Master's Program with a major in manufacturing in May of 2001.

Permanent Address: 3712 Howard Dr. #19, McAllen, TX 78503.