PATTERN RECOGNITION OF LENTIL SEEDS USING MACHINE VISION AND NEURAL NETWORKS

A Thesis Submitted to the College

of Graduate Studies and Research

in Partial Fulfilment of the Requirements

for the Degree of

Master of Science

in the Department of Electrical Engineering

University of Saskatchewan

Saskatoon

By

Philip W. Winter

Spring 1997

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5L7

ABSTRACT

Machine vision systems, computer-based visual pattern recognition, has been proven a useful tool in many industries, mostly in the automated inspection of manufactured items. In the production of grain in the agricultural industry, a similar inspection process occurs for the quality inspection of the seeds. This current grading process requires highly trained human inspectors and entails many repetitive measurements which sometimes leads to inconsistencies. A machine vision tool would be very beneficial in the grain inspection process in improving efficiency and objectivity.

Lentil seeds are presently graded visually based on the Canadian Grain Commission's established criteria: seed colour, size, damage, and foreign material. In this work, a machine vision system was developed to differentiate commercial samples of lentils (Laird variety) into three classes of "good", "discoloured", and "broken and peeled". A neural network was used to analyse 21 morphological and colour features of a sample and provide an output between -1 and 1. Two architectures of neural networks were tested: the multi-layer neural network (MLNN) and the multi-structure neural network (MSNN). The best system performance obtained was as follows: 94% success in recognition of good lentils, 97% for discoloured lentils, and 95% for broken and peeled lentils, for an overall recognition rate of 95.6%. This result was obtained using an MLNN network. The MSNN network gave comparable results with an overall accuracy rate of 95%. The features used were also analysed for their contribution to the pattern recognition problem. They were ranked using the accumulated correlation coefficient (ACC) method and by eigensystem analysis. A reduced set of optimal features was used to train an MLNN, with an overall accuracy of 93%. Finally, the projected areas of the lentils, and the mass of the lentils were correlated and a linear equation was developed so that results of the pattern recognition could be presented in the form of a percentage by weight, as is the industry standard.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to Dr. Hugh Wood of the Department of Electrical Engineering, and Dr. Shahab Sokhansanj of the Department of Agricultural and Bioresource Engineering. Their supervision and support during the preparation of this thesis was deeply appreciated.

Special thanks are also extended to Mr. Bill Crerar, support personnel, and fellow students of the Agricultural Processes Laboratory for their friendship, encouragement, and patience. Appreciation is also given to friends and colleagues who supported and encouraged the author throughout his M. Sc. work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION AND OBJECTIVES

1.1 General

In the past few decades, many attempts have been made to develop true thinking machines. However, despite incredible advances in computer technology, this goal has continued to elude researchers. It is still not possible to build machines which can form widely varied and abstract concepts like the human brain can. On the other hand, computer algorithms can be developed which give the computer a form of intelligence which allows it to perform very specific tasks and to contribute to the efforts of the human brain. These types of systems can be coupled with optical sensors, in effect becoming *machine vision systems*.

The term machine vision refers to a system of computer hardware and software which can perform some visual tasks which would normally be in the domain of human workers. The machine vision hardware normally consists of a video camera, a video signal digitiser, and a host computer. The software of the system consists of routines to control the digitiser, image analysis routines to measure features in the image, and pattern recognition algorithms to make judgments about objects within the image.

Typical uses of machine vision in the past were restricted to sectors usually associated with high technology. Machine vision has been used successfully by military organizations for automated recognition of aircraft. Many electronic part manufacturers have incorporated this technology in their plants for the purposes of product inspection and quality assurance. These past uses of machine vision have generally been expensive due to the need for sophisticated computer hardware, thus limiting their wide-spread use. In the past few years, however, the computer technology needed for a machine vision system has decreased dramatically in price. This, coupled with the increasing speed and capabilities of computers and vision technology points toward an expansion for machine vision in many areas, including the agriculture industry.

1.2 Application to the Agriculture Sector

The manufacturing industry has long used machine vision in inspection and quality-assurance processes. Machine vision has provided a consistent and quick method of inspection. The agriculture industry has similar requirements to manufacturing in that materials are produced which must be inspected for quality before being delivered to consumers. Many quality attributes of agricultural products are assessed on visual attributes, for example, size, shape, colour, and texture. The product often is graded based on visual inspection by a quality control worker. Machine vision inspection methods can be applied to these products as well.

Some examples of machine vision applications in agriculture include a beef carcass inspection system [Park, et al., 1996], an automatic vehicle guidance system for farm vehicles [Wood, et al., 1990], and a mushroom growth inspection system [Murray, et al., 1996]. A correlation has been developed between the color of alfalfa grind measured on the L,a,b scale, and the carotene content in the grind [Patil, et. al, 1995]. Patil also used machine vision to characterize the particle size in alfalfa cubes [Patil, et. al, 1992].

The above examples demonstrate how a vision system can be applied to agricultural processes. The most promising application, however, is in grain inspection. One of the earliest attempts at visual discrimination of grain seeds was by Segerlind and Weinberg (1973). This research did not incorporate a video camera but relied on manually tracing seed profiles onto paper. A Fourier series expansion allowed for the discrimination of wheat, oats, barley, and rye with a 99% accuracy. Subsequent researchers have followed, developing more automated systems of grain discrimination. A Bayesian classifier was used to discriminate a variety of grain seeds using morphological features [Brogan and Edison, 1974]. Wheat varieties were the subject of a recognition system developed at the University of Manitoba [Neuman, et al., 1986]. This system used Fourier descriptors [Zhan and Roskies, 1972] to separate 14 wheat varieties.

Colour features have also been used in wheat classification [Saperstein, et. al, 1989]; red, green, and blue components were used for this project. A discriminator for canola and mustard seed was presented by Jeff Hehn (1990) at the University of

Saskatchewan. Reflectance characteristics of the seeds were used as features, along with morphological measurements.

Barley variety was the subject of two projects at the University of Saskatchewan. Romaniuk (1994) used Fourier descriptors and neural networks to discriminate three varieties of barley. Shrestha (1996) used morphological and grayscale values as the basis to discriminate four varieties of malting barley. Other machine vision systems reported include a dockage tester for borage [Li, 1996], a popcorn quality tester [Winter, et. al, 1996], and a grading system for pistachio nuts [Ghazanfari, 1996].

Using machine vision for grain inspection is desirable for a number of reasons. Most commercial grains grown in Canada are inspected for quality by the Canadian Grain Commission before they go to market. The inspection process is dependent upon human visual judgments which could lead to inconsistencies. Given the large geographic separation of producers and buyers in Canada, it is not possible to have one human grader doing all seed inspection. A grain shipment is typically graded when the farmer delivers it to an elevator, and again when it arrives at market, and it is not uncommon to have different grades put on the same shipment by different graders. Also, grain quality varies season to season, region to region, and with the development of new varieties, all of which combine to make the job of grading very difficult. The subjectivity of the grading process sometimes leads to disagreements between the product trading partners and causes difficulties in the efficient processing of grains. As a tool for the human grader, a machine vision system would help to overcome these problems. By using computers to aid

1.3 Application to Lentil Inspection

The challenges with the current grain inspection system discussed above are even more of a problem with non-traditional crops such as lentils. For example, the colour of lentil seed is a very important factor in grading. The seed colour, however, is very much dependent upon growing and storage conditions. Acceptable levels of colour for each grade change every year. Because of this variability, a means of continuously adapting the grading methods to the latest conditions becomes an important task.

Figure 1.1 is the standard lentil grading chart prepared and distributed to graders by the Canadian Grain Commission in 1996. In the top row are the grading classes into which a sample of lentils is divided into, while each column gives the maximum allowed percentages of each class for the sample to fit into one of the four grades. Some of the classes are easily sorted by eye, such as stones and foreign particles. These classes require only a yes/no judgment: they either are stones or they're not. Other classes, however, do not lend themselves to such an easy decision. A broken lentil, for example is a lentil with more than one quarter of the seed missing. Judging what is exactly one quarter of the seed is not easy to do by eye. Figure 1.2 shows a series of examples of lentils that the Canadian Grain Commission publishes. The pictures show a large difference between stained and unstained lentils. But many of those lentils fall midway between these two easy-to-see extremes.

Samples of lentils are also graded based on colour using a "fuzzy" scale with labels of "good", "reasonable", and "fair". Figures 1.3 and 1.4 are grading charts to which a human inspector compares a sample to determine its colour grade. If the overall sample colour is closest to Figure 1.3, then a colour grade of "good" is applied. If the colour is closest to Figure 1.4, then the sample is classified as "fair". A sample judged "fair" in colour is assigned a maximum grade of No. 3 Canada as defined in Fig. 1.1, before any other quality aspects of the sample are tested. It is very easy to misclassify a sample which lies at the midpoint between two colour classes, and two separate graders will often give different results.

1.3 Thesis Objectives

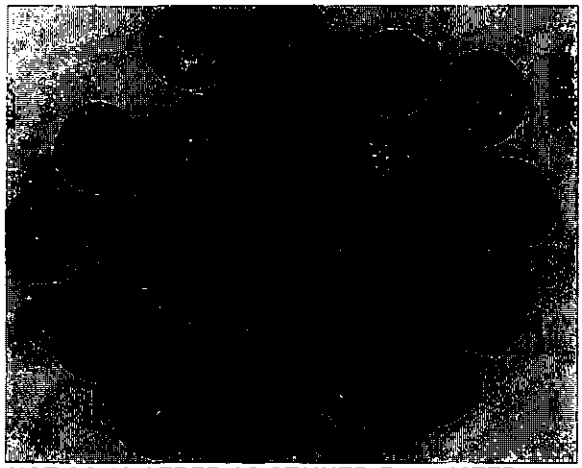
The objective of this work is to assemble the necessary components of a vision system and to develop a recognition procedure to discriminate between three classes of Laird lentils: good, discoloured, and peeled and broken. These classes are not meant to coincide with the grading system used by the Canadian Grain Commission but can be used as a means of pre-identification of grain quality by persons who are in the cleaning and packaging business. A neural network technique will be optimised as the discriminator and will provide an objective, mathematical model of the lentil classes. An investigation will be done into the best physical and colour features of lentils to use in the discrimination process. In order that results from the vision system can be reported as a percentage-by-mass, as is done by industry, a mathematical relationship between lentil area and mass will be developed.

Grade Name	Degree of Soundness	Maximum Limits								
			Damage				Foreign Material			
		Stained	Heated	Peeled, Split and Broken	Other Damage	Total	Stones	Ergot	Sclerotia	Total Foreign Material
No. 1 Canada	Uniform in size of good natural colour	1.0%	about 0.2%	2.0%	1.0%	2.0%	about 0.1%	0.05%	0.10%	about 0.2%
No.2 Canada	Uniform in size of reasonably good natural colour	4.0%	about 0.5%	3.5%	2.0%	3.5%	about 0.2%	0.05%	0.10%	about 0.5%
Extra No. 3 Canada	Uniform in size of fair colour	7.0%	about 0.5%	5.0%	5.0%	5.0%	about 0.2%	0.05%	0.10%	about 0.5%
No. 3 Canada	Poor Colour	--	1.0%	10.0%	10.0%	10.0%	about 0.2%	0.05%	0.10%	1.0%
If specs for No.3 Canada are not met, grade:			Lentils, Sample Canada, Account Heated	Lentils, Sample Canada, Account Damaged	Lentils, Sample Canada, Account Damaged	Lentils, Sample Canada, Account Damaged	up to 2.5%: grade: Lentils, Rejected (grade) Account Stones over 2.5%: Lentils, Sample Salvage	Lentils, Sample Canada, Account Ergot	Lentils, Sample Canada, Account Admixture	Lentils, Sample Canada, Account Admixture

Figure 1.1 Lentils (Canada) - Primary and Export Grade Determinants [Canadian Grain Commission, 1996].

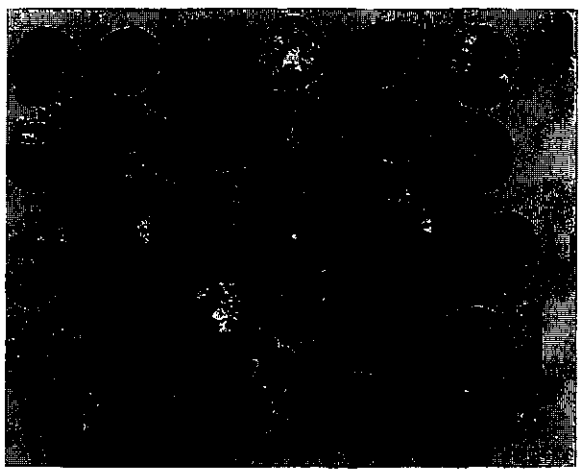
MOTTLED-CONSIDERED AS STAINED

NOT STAINED



NOT CONSIDERED AS STAINED-EVALUATED ON COLOR

STAINED



DAMAGED-ASCOCHYTA
ALSO TO BE CONSIDERED AS STAINED

DAMAGED-FROST

Figure 1.2 Visual grade determinants of lentils [Canadian Grain Commission, 1995a]. Not to be used for grading purposes.

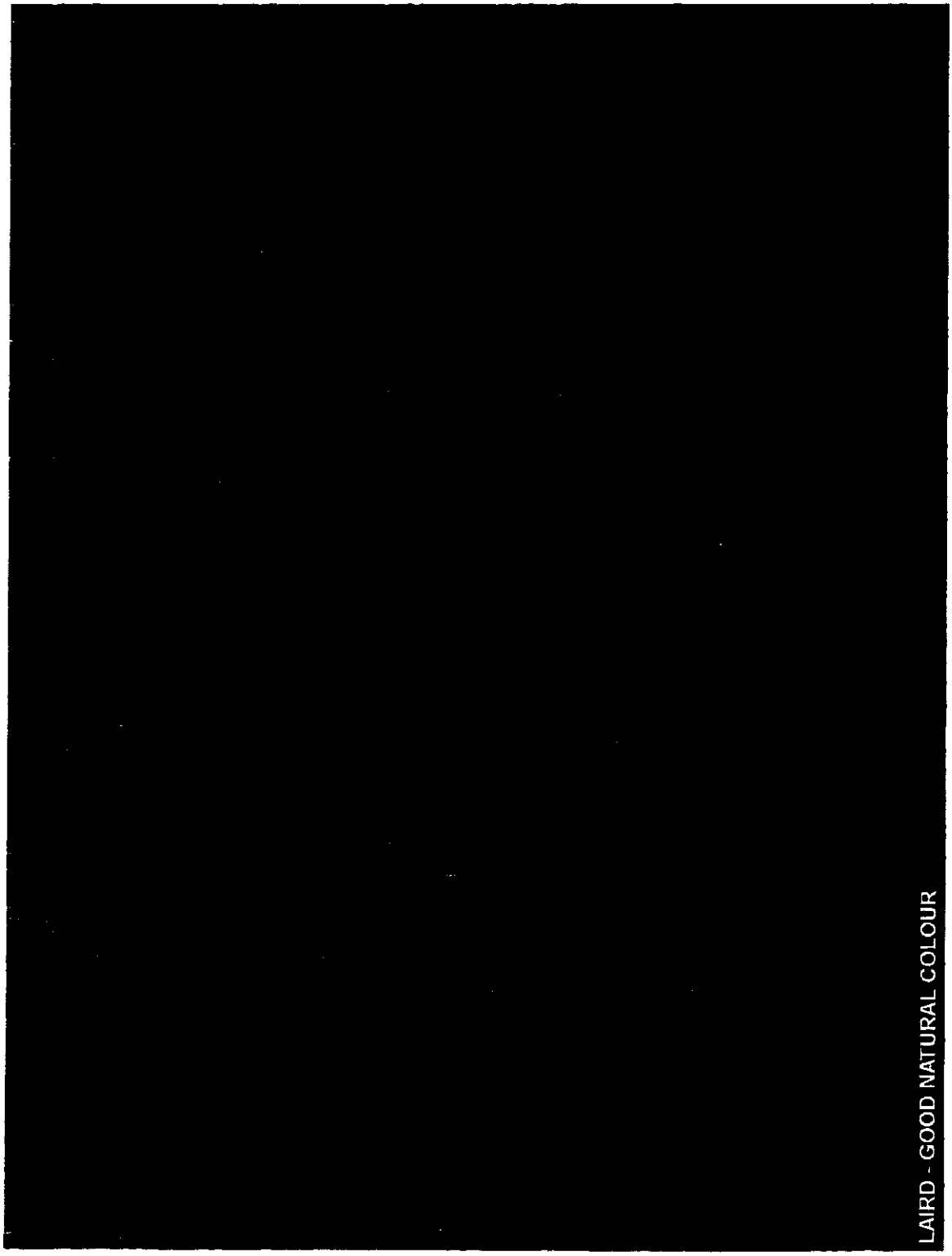


Figure 1.3 Laird lentils with "good natural colour" [Canadian Grain Commission, 1995b]. Not to be used for grading purposes.

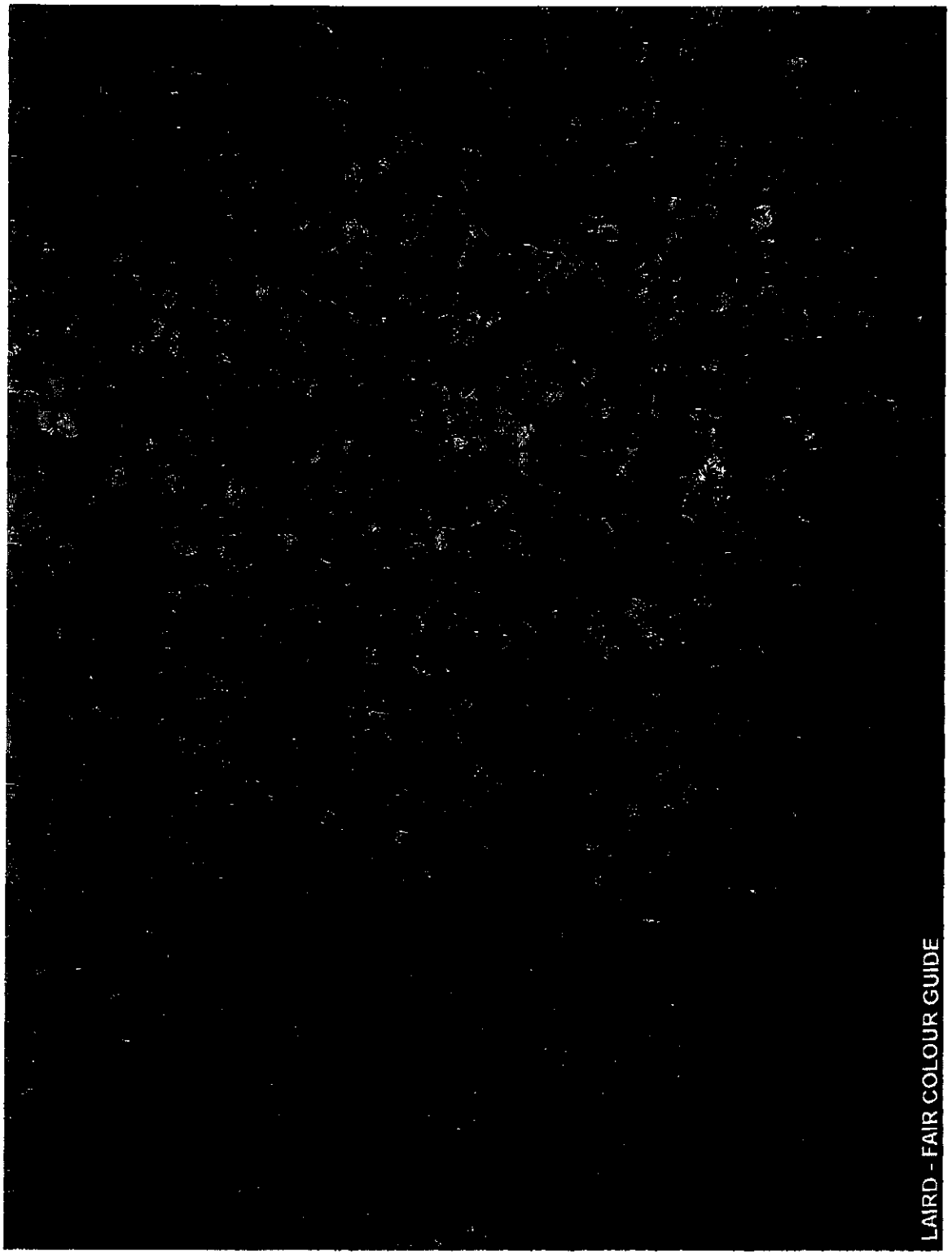


Figure 1.4 Laird lentils with "fair" colour [Canadian Grain Commission, 1995c]. Not to be used for grading purposes.

1.4 Thesis Outline

The material in this work is presented in five chapters. This first chapter has given a brief background on machine vision and examples of its application to the agriculture industry. The goals of the project are also given. Chapter 2 will give the theoretical background on the pattern recognition techniques used for this project, including the background on neural networks and the architectures which have been used.

Chapter 3 gives details on the hardware and software of the machine vision system as well as an explanation of their operation. The methods followed in feature extraction, and the development of the recognition routines are also given. Chapter 4 presents the results obtained in feature selection, neural network testing, and area/mass correlation. Finally, Chapter 5 gives a summary of the project including the main conclusions. Recommendations for future research are also given in Chapter 5.

Hardware specification can be found in Appendix A. Appendix B presents the histograms and fitted normal curves for the measured features of lentils. Appendix C is a mathematical derivation of the neural network model and back-propagation learning algorithm.

2. PATTERN RECOGNITION AND NEURAL NETWORKS

2.1 Introduction

This chapter will introduce the principles of pattern recognition as applied to this work. The concept of a neural network will be introduced. Two configurations of neural networks which are used in this work will be analysed and compared. Finally, it will be shown how a neural network can be applied to pattern recognition.

2.2 Pattern Recognition

Pattern recognition is a broad term which applies to the problem of how we perceive our surroundings. When a child first learns to walk, it determines what sequence of muscle movements will create a stable step, and which will cause loss of balance. This can be considered a form of pattern recognition. When the time comes to learn to read, the same learning and pattern recognition techniques are applied to distinguishing a letter 'b' from a letter 'd'. These learning processes of the human mind can be emulated to some extent by mathematical methods which can take problems such as these and create a formal structure around them.

There are two basic aspects to pattern recognition: the development of a decision rule, and the use of that decision rule [Meisel, 1972]. The "pattern" is developed during the making of the decision rule, or the "training" stage. "Recognition" is done in the use of the decision rule. Figure 2.1 shows the process for creating a pattern recognition system. The first step is identifying the physical system. The system must be clustered or grouped into similar objects, where each of these clusters is called a class. A system may be composed of easily defined, discreet classes; an example would be a classroom of children, where the classes might be "male" and "female". The students naturally fall into either one class or the other. A system may also be continuous, where classes are artificially imposed. An example of this would be baked bread, where the classes are a grading system of "good", "fair", and "poor" based upon the size of holes in the bread. The classes are not naturally occurring but are based upon an imposed set.

The next step is to measure the properties of the system. Properties are any measurable features of the system. In the case of visual pattern recognition, features are those properties which can be measured with the use of a video camera. These can include any morphological features, such as area and perimeter, or any colour measurements, such as red, green and blue content. The raw data is referred to as the "measurement space". A point in measurement space corresponds to a sample of the

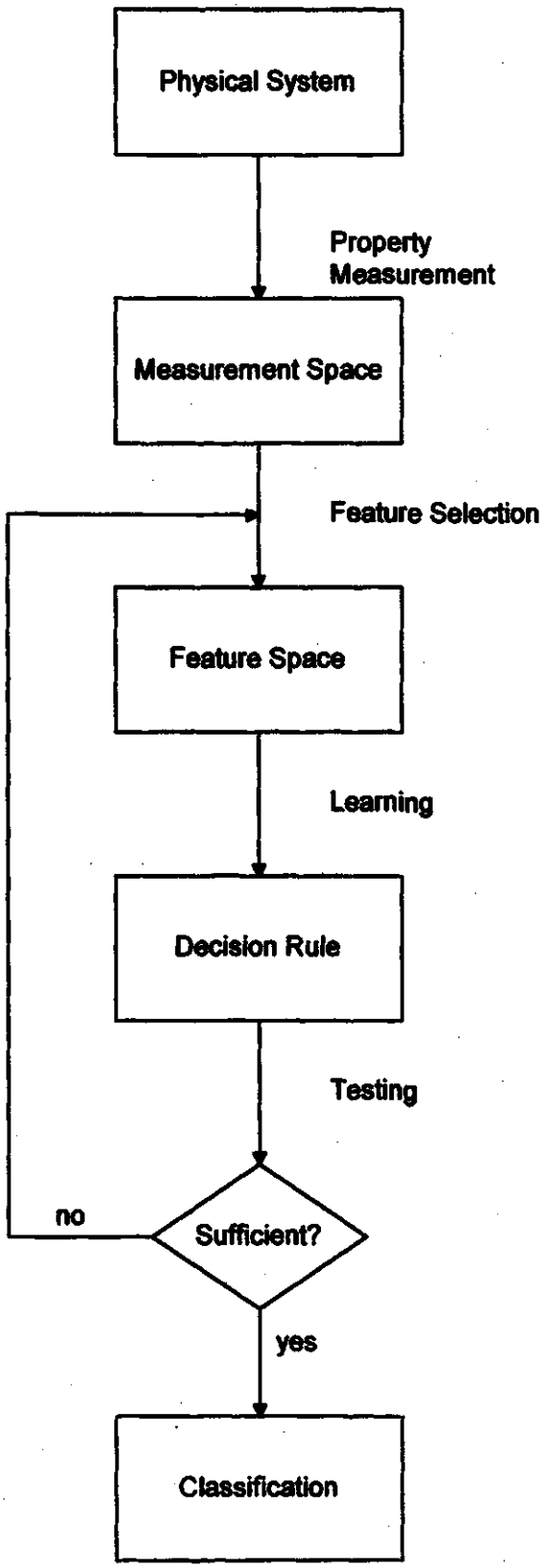


Figure 2.1 A flowchart outlining a pattern recognition system.

system, with a number of dimensions equal to the number of features measured. Thus a sample or vector in measurement space is:

$$\mathbf{x}=(x_1,x_2,...,x_n) \tag{2.1}$$

where x is the sample vector,

x_n is the n^{th} feature, and

n is the number of features measured.

The process of choosing the features best suited for recognition is called "feature selection". Typically, more features of the system can be measured than are needed to create a working decision rule. Generally, the fewer features that are used, the less accurate the recognition will be. However, more features indicates a more complicated decision rule, and more computational power is needed to find that rule. Therefore, a median point must be found to balance accuracy and ease-of-calculation. Feature selection is done by deciding which features are truly necessary for solving the problem. Some features will not correlate with the classes of the system, and some features will be redundant with other features. These features should be removed to create an optimised decision rule. Once features are selected, the resulting data points are referred to as the "pattern space".

The next step in the process is to use this data to find a decision rule through a learning process, also referred to as pattern classification [Meisel, 1972]. There are many methods of creating a decision rule; risk analysis, Gaussian classification, and decision trees to name only a few. The method used will depend upon the particular problem. For

example, Gaussian classification is best used only if the features of the system follow a normal distribution. Other density curves can be used, but these must be known beforehand. The method used in this work is a technique called neural networks. The advantages of neural networks are that they can create very complex non-linear decision rules, and do not depend upon a prior knowledge of the feature distributions. Neural networks will be discussed later in this chapter.

Finally, once the decision rule is found, it must be tested. This is done by presenting new pattern samples to the decision rule and verifying the results against the desired output. Typically, an equal number of sample vectors is used for training as is used for testing, since testing the decision rule is equally important as creating it. If the results from the testing show a satisfactory recognition accuracy, the decision rule may then be used for pattern recognition. If however, the accuracy rate is not satisfactory, or the decision rule is too computationally expensive, then further feature selection may be necessary.

2.2.1 Feature Selection

To be suited for pattern recognition, a feature must supply significant information about the subject. That is, in pattern space, it should separate the classes by a substantial amount (see Fig. 2.2). It is usually understood that a large distance between points in pattern space corresponds to a large difference in the actual subjects. Selected features should have as little correlation to each other as possible to maximise the amount of

information each feature supplies to the separation process. Features should also be directly relevant to the quality to be recognised. For example, in classifying different varieties of seeds, the angle of orientation of the seeds with respect to the camera is not a relevant feature [Brogan, 1974].

There are many methods of choosing features. Most methods aim at finding which features have the qualities listed above. The methods used in this project are the Average Correlation Coefficient Technique (ACCT), and the eigensystem analysis technique [Mucciardi, et al., 1971]. The features are first ranked in order of suitability by the ACCT method, which determines how redundant each feature is. The eigensystem analysis, then, is used to decide how many features are needed to create an accurate decision rule.

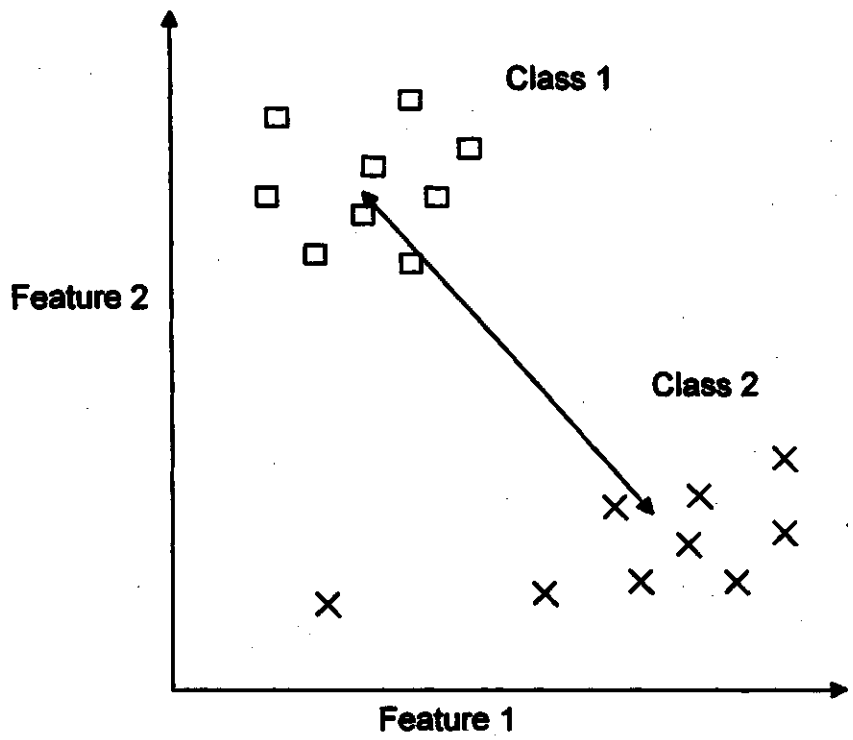


Figure 2.2 A two-class, two-feature pattern space.

2.2.1.1 Average Correlation Coefficient Technique

One method to rank features according to their recognition suitability is called Average Correlation Coefficient Technique (ACCT) [Mucciardi, et al., 1971]. This method examines the similarities between the features. The features are ranked according to how correlated they are to the other features; a high correlation translates into a high redundancy. Redundant features may be removed from the system.

In ranking the features, ACCT first selects one of the features to place at the top of the list. This selection may be done at random, but it is better to base the selection on some knowledge of the features. In this work, the first feature was chosen based upon probability of error (POE). The POE of a feature is the amount of overlap of its distribution curves for each class (see Fig. 2.3). If the curves are greatly overlapped (large POE), the features will correspond to a small class separation distance in pattern space. This means that the classes will be hard to separate using this feature. The top ranked feature, then, is the feature with the lowest POE compared to each of the other features.

Next, the correlation coefficients between features are found. A high correlation between two features indicates that one of the features is largely redundant. ACCT takes this correlation into account when ranking the features. The second feature chosen, after the one with the lowest POE, is the feature with the smallest correlation coefficient to the first feature. The third feature is chosen such that its average correlation with the first two is smaller than that of all the remaining features, and so on. The n^{th} choice on the list will

depend on its average correlation to the previous n-1 features. The end result is a list of all features ranked in order of decreasing usefulness for creating a decision rule. Table 2.1 summarises the ACCT method of ranking features.

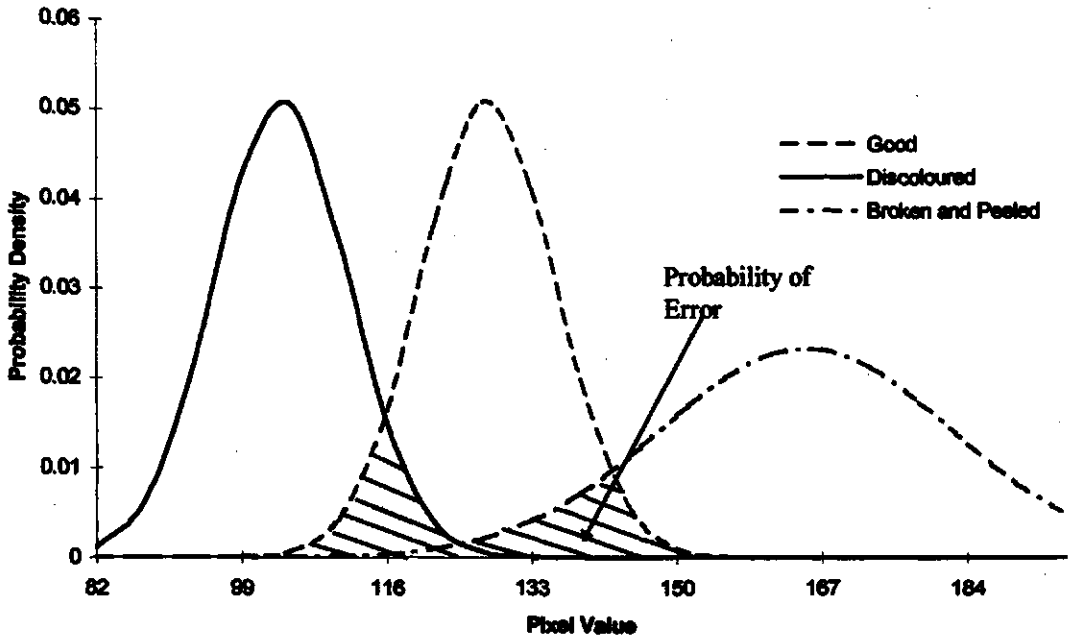


Figure 2.3 The probability of error for a three class problem. The feature described by these curves is "Pixel Value", and the classes represented are "Good", "Discoloured", and "Broken and Peeled".

Table 2.1 Ranking features by ACCT.

Feature rank	Feature found by:
#1	feature with lowest POE
#2	feature with lowest correlation to feature #1
#3	feature with lowest average correlation to features #1 and 2 found by calculating the following for each remaining feature: $\frac{(\text{correlation with feature \#1} + \text{correlation with feature \#2})}{2}$
#4	feature with lowest average correlation to features #1, #2, and #3 found by calculating the following for each remaining feature: $\frac{(\text{corr. with feature \#1} + \text{corr. with feature \#2} + \text{corr. with feature \#3})}{3}$

2.2.1.2 Eigensystem Analysis

Once the features have been ranked according to suitability, one must be able to determine how many features are needed to perform the recognition to a desired accuracy. A method for this is eigensystem analysis. This method was used by Mucciardi and Gose (1971) to determine the number of features needed from a EKG waveform to reach a 90% accuracy (ratio of correctly identified waveforms to entire sample set). It was also used by Shrestha (1996) to select features for machine vision discrimination of barley varieties.

In eigensystem analysis, the eigenvalues are found from the feature correlation matrix (a symmetric matrix containing the correlation coefficients of each feature to every

other feature). An eigenvalue is the characteristic value of the matrix which relates the contribution of each vector to the entire system. A system with five features, for example, will have five eigenvalues, each relating how much variance is accounted for by that particular vector. In total, the variance accounted for by all vectors will add up to 100%. Thus, if a particular variance is desired, the number of features (vectors) needed to reach that variance can be found by summing the eigenvalues.

The eigenvalues are first found from the features correlation matrix and placed in order according to decreasing value. Each eigenvalue is associated with an eigenvector in n dimensional space, where n is the number of features. The largest eigenvalue is associated with the axis accounting for the largest variance, and so on. The sum of all eigenvalues will add up to the number of vectors:

$$\sum_{j=1}^{n} e_j = n \qquad (2.3)$$

where e_j is the j^{th} eigenvalue,

and n is the number of features.

Each vector is responsible for an amount of variance in the system given by:

$$\sigma_j = \frac{e_j}{n} \times 100\% \qquad (2.4)$$

where σ_j is fraction of variance accounted for by the j^{th} axis,

and the accumulated variance accounted for by the first i largest eigenvalues is:

$$\sigma_i^{acc} = \sum_{j=1}^{i} \frac{e_j}{n} \times 100\% \qquad (2.5)$$

Figure 2.4 shows a sample plot of Equation 2.5. On this graph, the y axis is the total variance accounted for by the number of features, plotted on the x axis. From this graph, it can be seen that 4 features are necessary to expect a total accuracy of 90%

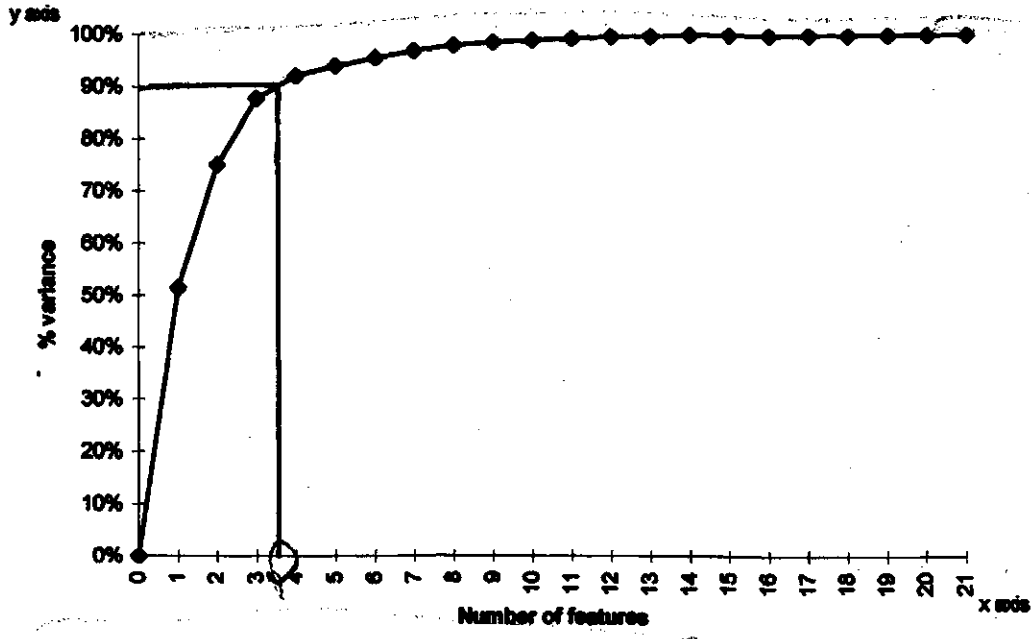


Figure 2.4 Sample variance plot for eigensystem analysis.

2.3 Neural Networks

In creating the decision rule in this project, a neural network was used. In this section, the basis for a neural network will be explained, as well as how it is used for pattern classification. The three types of network configurations used will also be discussed.

The neuron is a mathematical construct which is modelled according to the neuron in the biological nervous system. A system of artificial neurons, also known as an artificial neural network (ANN) has some of the same properties as the biological brain, which is a network of biological neurons. The ANN has learning and adaptive functions similar to the biological model.

In the following sections, the mathematical basis for the neuron and the ANN will be discussed. A learning algorithm called back propagation will be shown. Three types of network architectures, feed forward with single output, feed forward with multiple outputs, and a Multiple Structure Neural Network (MSNN) will be described.

2.3.1 Neurons

The neuron is the building block of the neural network and is often referred to as a node, unit, or processing element (PE). The first simplified computational neuron was proposed by McCulloch and Pitts in 1943 [McCulloch and Pitts, 1943]. A revised version, called the perceptron, was invented by psychologist Frank Rosenblatt in the late 1950s [Freeman, 1991]. The neuron performs two basic operations. The first function of the neuron is to perform a weighted sum of the neuron inputs while the second function is to apply an output function to the summed inputs.

The structure of the single neuron can be seen in Fig. 2.5 [Demuth, 1995]. The inputs to the neuron are multiplied by a weight W. A unity bias is often included and multiplied by a weight b. These weighted inputs are then summed. If the weights are a row vector, and the inputs a column vector, then the sum is the dot product of the two (Equation 2.6). The neuron stores information in the weights W. By changing these weights, the inputs to the neuron can generate different values on the output function. A neural network "learns" by changing the weights so that a desired output is achieved for a given input.

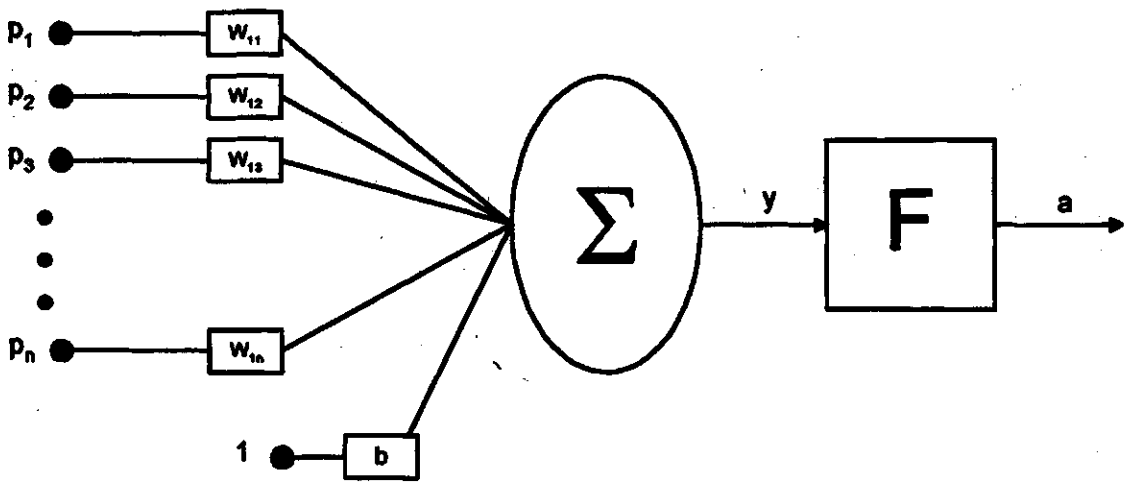


Figure 2.5 The structure of an artificial neuron.

Once the net input is calculated, it is converted to an activation value by applying an output, or activation function. The output of the neuron is given by Equation 2.7.

$$y = W \cdot p + b \tag{2.6}$$

$$a = F(W \cdot p + b) \tag{2.7}$$

The output function can be chosen from a wide range of functions (see Fig. 2.6). The choice depends on the application and the desired output. If the input is linear, then the linear function can be chosen. If however, the system in inherently non-linear, then a non-linear activation function should be used as this is the only place that a non-linearity can be introduced into the output equations. The typical non-linear function used is the sigmoid function, which is used in this work. The hard-limit function is used if a binary output is desired. It cannot be used with some learning algorithms which demand a differentiable function.

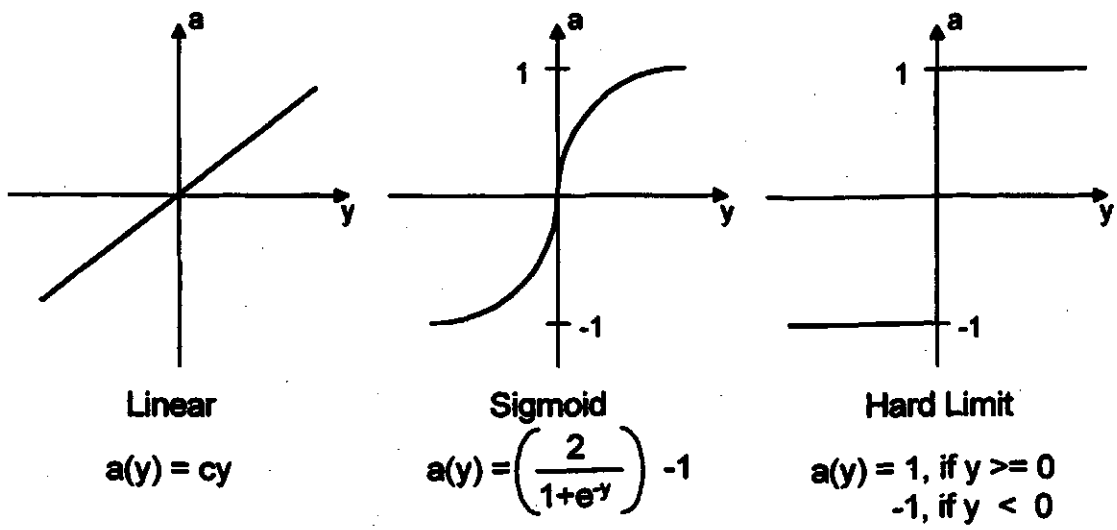


Figure 2.6 Three typical neural activation functions.

2.3.2 Feed-Forward Neural Networks

The single neuron is capable of mapping inputs onto many different output functions. A multi-layered neural network (MLNN), however, is capable of producing a

much more complex output equation. The two general types of networks are the recurrent network, and the feed-forward network. The recurrent network has connections which loop back, forming a feedback loop. These networks are generally used for controls applications and other problems related to dynamics. The feed-forward network has no feed-back loops, however, in both type of networks, the nodes are arranged in layers, each layer's output feeding into the next layer's inputs, until an output is produced.

Figure 2.7 shows a common feed-forward network. This figure shows a three layer network. The input vector feeds into the network via the input layer. This layer is sometimes referred to as the fourth layer, but does not contain any neurons. The inputs are fed to the first layer of neurons, which in turn feeds its outputs directly to the neurons in the second layer. These two layers are called hidden layers because they do not have any direct connection to the outside world. The final layer is called the output layer, from which the final output from the network is received.

In referring to the layers of a network and the number of neurons in it, the following terminology will be used: # of inputs: # of nodes in 1st hidden layer: # of nodes in 2nd hidden layer: # of nodes in output layer. For example, a network with 16 inputs, 15 nodes in the first hidden layer, 30 nodes in the second hidden layer, and one node in the output layer would be referred to as 16:15:30:1.

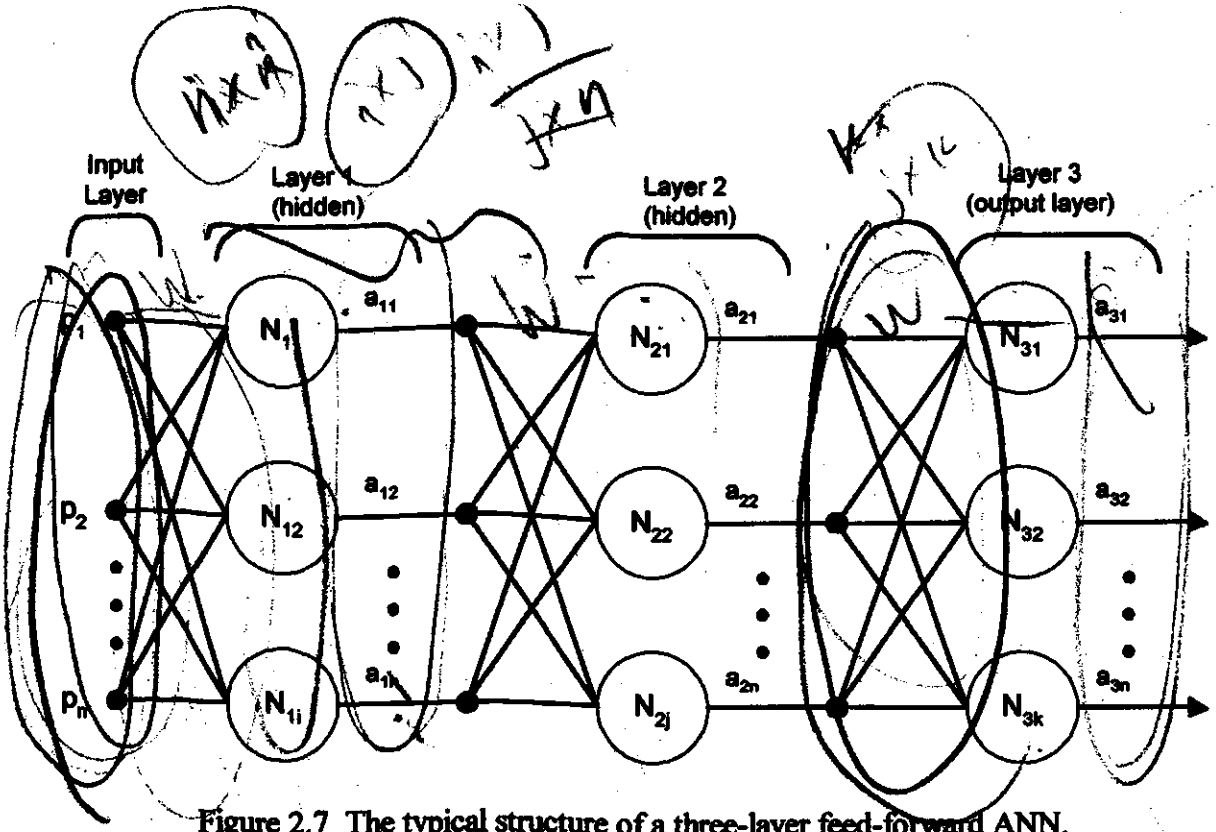


Figure 2.7 The typical structure of a three-layer feed-forward ANN.

The network in Figure 2.7 has n inputs, i neurons in the 1st hidden layer, j neurons in the 2nd hidden layer, and k neurons in the output layer. It is common for networks to have different values for i, j, and k. Each layer feeds into consecutive layers. The outputs of each layer can be found from

$$a_1 = F_1(W_1 \cdot p + b_1) \tag{2.8}$$

$$a_2 = F_2(W_2 \cdot a_1 + b_2) \tag{2.9}$$

$$a_3 = F_3(W_3 \cdot a_2 + b_3) \tag{2.10}$$

and therefore,

$$a_3 = F_3(W_3 \cdot F_2(W_2 \cdot F_1(W_1 \cdot p + b_1) + b_2) + b_3) \tag{2.11}$$

where a is the column vector of outputs of each respective layer,

F is the activation function for each neuron,

p is the input column vector,

W is the row vector of weights for each respective layer, and

b is the bias weight column vector for each layer.

2.3.3 Back-Propagation Training

The true power of a neural network is in its ability to learn. This is accomplished by the automatic adjustment of the weights and biases until the desired output is achieved. This is a form of supervised learning in which the correct output is known, and the network learns by adjusting to reduce the error between the predicted and the actual outputs. The output of the network is changed by adjusting the weights until an acceptable error between the desired and the actual output is reached.

The back-propagation (BP) approach was formulated in part by Parker in 1985, and by Rummelhart and McClelland (1986). In BP, the networks is trained by comparing the input-output pairs through a two-phase propagate-adapt cycle [Freeman and Skapura, 1991]. The input vectors are applied to the input layer, and signals propagate through the hidden layers of the network until an output is generated. An error value is found by subtracting the expected output from the actual output. The summed squared error (SSE) is found for the entire set of input vectors. The SSE will change for different values of weights, and can be thought of as an error surface if plotted against the weights. Figure 2.8 is a sample error surface for the case with only two weights. As can be seen from this figure, an error surface can have many local minima and maxima. The purpose of the BP

technique is to find the weights where the SSE is at the global minimum. This is done by calculating the direction of steepest descent on the error surface from the current SSE. The weights are then adapted so that the SSE follows this steepest descent path towards a minimum.

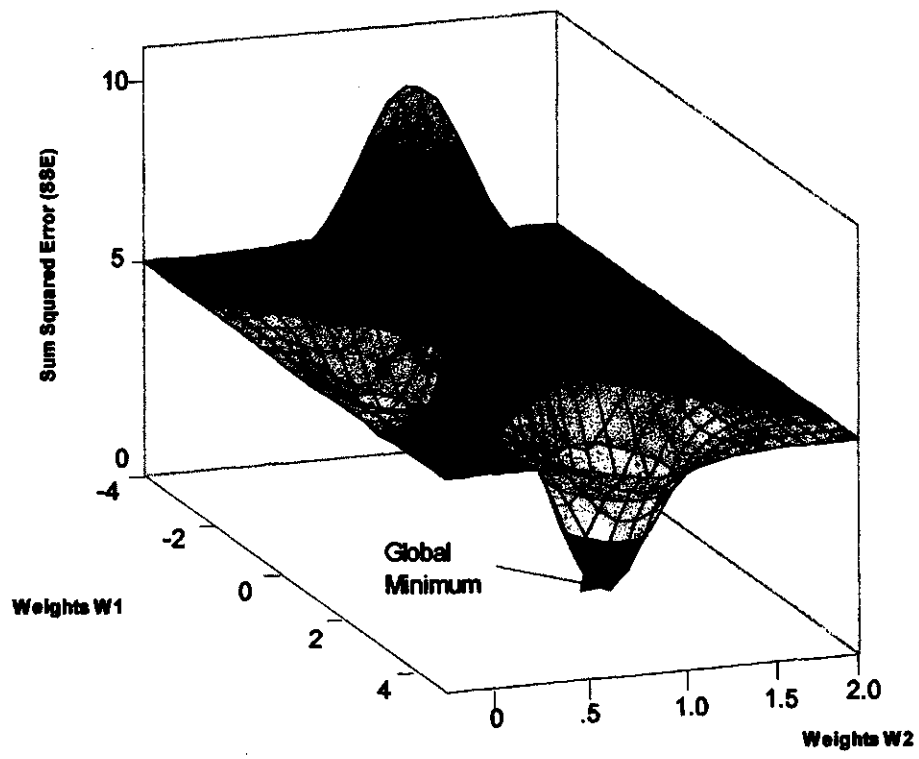


Figure 2.8 A sample error surface for a two-weight neural network [The Mathworks Inc. 1995].

The final results of this process is a network which maps the training input vectors onto specific points on the output functions; the network has found a way to internally represent the given training inputs as the desired outputs. For example, a neural network may be trained to output a value of +1 if given the input features of a wheat kernel, and -1 if given the input features of a wild oat seed. If the network is presented with a vector

which is the same or similar to a training class, the network will output a value which is close to the output value assigned to that class.

2.3.3.1 Update of Network Weights

The weights in the neural network are changed in such a way that the SSE decreases by changing them in the direction of the negative gradient of the error surface. The amount they are changed depends upon the magnitude of the SSE. Equation 2.12 is the formula for updating the output layer weights. The full derivation of this equation can be found in Appendix C. The symbols in Equation 2.12 refer to a neural network shown in Figure 2.9:

$$w^{o}_{kj}(t+1) = w^{o}_{kj}(t) + \eta\Delta w^{o}_{kj}(t)$$

$$= w^{o}_{kj}(t) + \eta(y_{pk} - o_{pk})f^{o'}_{k} i_{pj} \qquad (2.12)$$

where w^{o}_{kj} is the weight of the k^{th} output node for the j^{th} input to that node,

t signifies the current iteration,

y is the expected output of the of the k^{th} output node due to the p^{th} net input,

o is the actual output of the k^{th} output node due to the p^{th} net input,

$f^{o'}_{k}$ is the derivative of the activation function for the k^{th} output node,

i_{pj} is the output of the j^{th} neuron of the previous network layer due to the p^{th} network input, and

η is a learning rate constant.

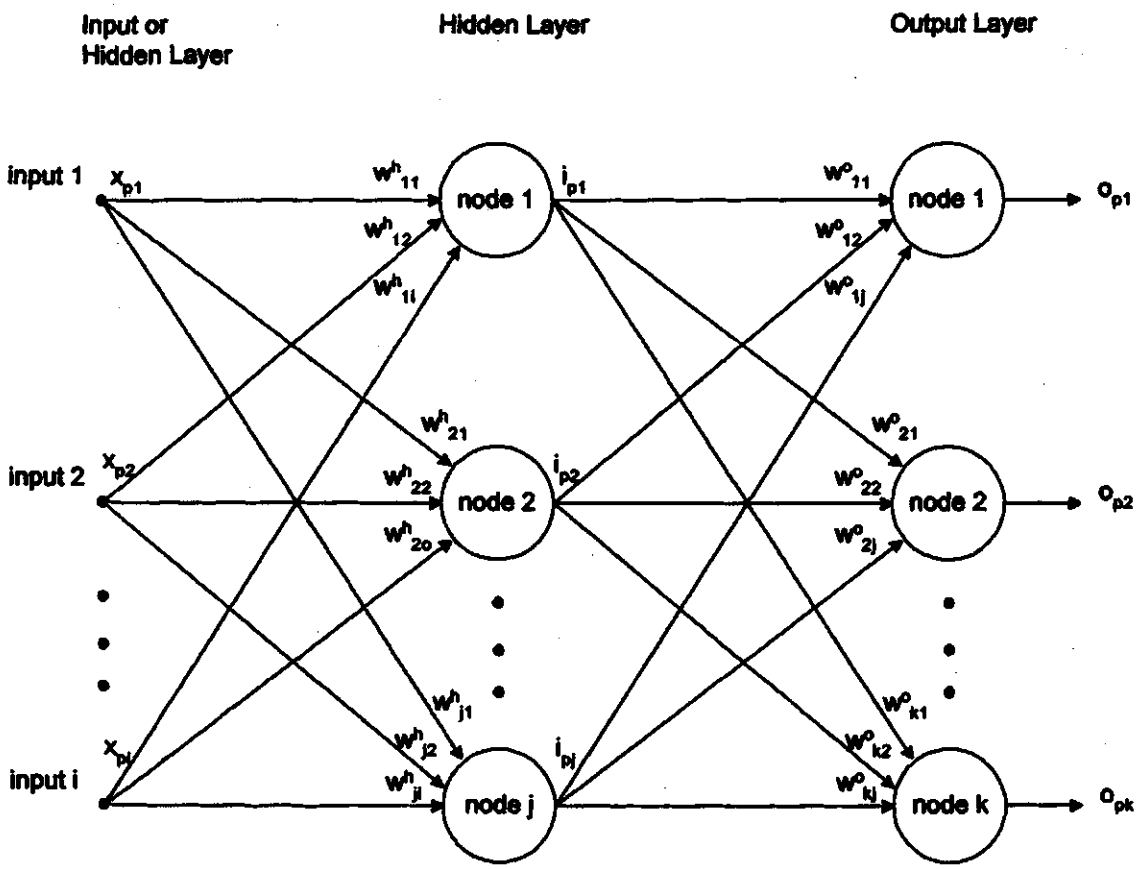


Figure 2.9 The location of weights and outputs used in the BP equations.

If a sigmoid function is used as the activation function, then

$$f_k^{o'} = \frac{1}{2}(1 - o_{pk}^2) \tag{2.13}$$

and Equation 2.12 becomes:

$$w^{o}_{kj}(t+1) = w^{o}_{kj}(t) + \eta(y_{pk} - o_{pk})\frac{1}{2}(1 - o^{2}_{pk})i_{pj} \qquad (2.14)$$

Equation 2.14 is the equation to update the weights in the output layer. However, the change in the weights depends upon the fact that the error surface for this layer can be found directly by finding the difference between actual and desired outputs. This method will not work on the hidden layers, however, because the output of the hidden neurons is not known. The updates to the hidden layers are found by calculating the contribution of each node to the error. Equation 2.15 is the equation which can update the weights in the hidden layers. The symbols are referenced from Fig. 2.9.

$$w^{h}_{ji}(t+1) = w^{h}_{ji}(t) + \eta f^{h'}_{j} x_{pi} \sum_{k} (y_{pk} - o_{pk}) f^{o'}_{k} w^{o}_{kj} \qquad (2.15)$$

where w^{h}_{ji} is the weight on the j^{th} hidden node for the i^{th} input to that node,

$f^{h'}_{j}$ is the derivative of the activation function of the j^{th} hidden node, and

x_{pi} is the output of the i^{th} neuron of the previous layer due to the p^{th} network input.

To summarise the back-propagation learning technique, it is an iterative, two stage process of trial and adaptation. The first stage is to test the network with data for which the output is known. An error is then calculated from this known output. Equation 2.14 is then used to adapt the weights of the output layer. The weights of the hidden layers are

then updated using Equation 2.15. Thus, the error value is propagated backward through the network until all weights have adapted to decrease this error. Another iteration of trial and adapt then begins. The process continues until a pre-set error is reached or until a maximum number of iterations have passed.

2.3.3.2 Learning Rate

The factor η in Equations 2.14 and 2.15 is called the learning rate. It is the learning parameter which controls the step size of change in the weights and biases. This factor can be critical in the performance of the network in terms of the speed of learning and the network's ability to converge. The learning rate is usually chosen to be some small value of around 0.05 to 0.25 [Freeman, 1991]. In general, a larger learning rate will result in a faster-training network. However, if the factor is too large, it will result in the network "bouncing" between minima on the error surface. This will cause the network to become unstable and it will not converge on the global minimum. A small learning rate will make it more likely that the network will converge to the lowest point on the error surface. The smaller the value, however, the more iterations the network must make to settle on an acceptable solution. This will increase the training time and load on the computer.

The dilemma in choosing the learning rate can be compensated for by using an *adaptive learning rate*. This method will automatically decrease the learning rate when

the network is bouncing between points on the error surface, and increase the rate when the change of error is small. To implement an adaptive learning rate, the SSE is compared to the SSE from the last iteration. If the ratio of new error to old exceeds a set value (typically 1.04) then the learning rate is decreased by multiplying it by a value less than one (typically 0.7)[Demuth, 1992]. The new weights and biases are discarded and the iteration is repeated with the new learning rate. Similarly, if the new error is less than the old error, the rate is then increased by a typical factor of 1.05.

2.3.3.3 Momentum

As was previously shown, the goal of the back-propagation algorithm is to find the global minimum on the error surface. Some error surfaces, however, have many local minima in which the network may become trapped. The sample error surface in Fig. 2.8 has two minima of which one is deeper than the other. The shallower minimum will not give as good a result as the deeper minimum, therefore it is more desirable to find the weights at the global minimum on the error surface.

Since the weight changes depend on the gradient of the current point on the error surface, if the current point happens to fall to zero inside a local minimum, the weights will cease to change, and the network will become trapped. If this local minimum is too far away from the global minimum, the network will not provide an adequate mapping function. It then is desirable to build in to the learning equations the ability to jump over small depressions in the error surface. This is called *momentum*.

To add momentum to the BP algorithm, the weight changes become equal to the sum of the new change obtained from the gradient and a fraction of the last weight change. A momentum factor *P* is used to regulate the amount of influence the previous weights have on the new weights. It is typically set at 0.95. The weight change equation is then:

$$w(t+1) = w(t) + \eta\Delta w(t) + L\Delta w(t-1) \qquad (2.16)$$

where w is the weight being changed,

t signifies the iteration,

$\Delta w(t)$ is the calculated weight change for this iteration,

$\Delta w(t-1)$ is the weight change from the previous iteration, and

L is the momentum factor.

2.3.4 Neural Networks as Pattern Classifiers

The standard function of a neural network is to map inputs onto an output function. This can be applied directly to the problem of pattern recognition. The quantifiable features from the classes can be used as the inputs in the networks. The network can find patterns in the inputs and change the weights through back-propagation so that the output differentiates between the classes.

To illustrate this principle, let us look at a case with 2 classes and 2 features. Figure 2.10 shows the pattern space of the problem. The two classes can be separated by

a single line. Input vectors which occur above the line are classified as Class 1, while those which fall below the line are Class 2. The pattern classifier is a single neuron with a hard-limit output which forces the output to be either 0 or 1. If the two features are used as the inputs (x_1 and x_2) for a single neuron, the weights in the neuron can form the needed discriminatory line. From Equation 2.7, the output of the neuron is:

$$a = \text{HardLim}[w_1x_1 + w_2x_2 + (-1)(1)] \tag{2.17}$$

When the correct weights are found which will create the proper decision boundary, the output becomes:

$$a = \text{HardLim}[-x_1 + x_2 - 1] \tag{2.18}$$

Thus, when the input to the neuron is above the decision line, the output of the neuron is +1, and when the input is below the line the output of the neuron is -1. Now, when the neuron is given a new, unknown input, the output will either be +1, which will indicate Class 1, or -1, which will indicate Class 2.

The same principle can be applied to more complex problems. The number of features is limited only by computer capacity. If a non-linear function is used on the output, such as a sigmoid function, any non-linear discriminatory line can be formed. This means that the inputs can be very complex and multi-dimensional in nature, as is usually the case in pattern recognition problems.

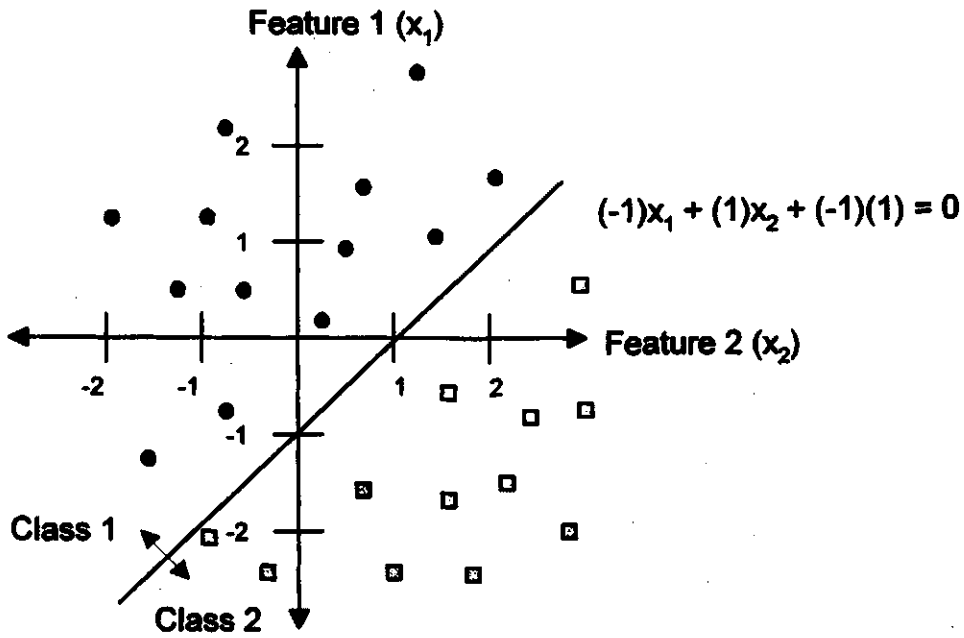


Figure 2.10 A two dimensional pattern space with two classes which can be discriminated with a simple linear line.

Figure 2.11 shows how a sigmoid output is used in forming a decision rule. The classes are trained to converge to points along the sigmoid line. In Fig. 2.11 there are three classes, which may have training goals of +0.8, 0, and -0.8 respectively. However, since the sigmoid is a continuous function unlike the hard limit function, the outputs will not necessarily converge to the exact output goals. They will fall within a range near to the goals. Therefore, secondary decision lines must be chosen to accommodate this range of outputs. In Fig. 2.11, class 1 may include all outputs which fall between 0.4 and 1.0, class 2 between -0.4 and +0.4, and class 3 between -0.4 and -1.0.

How complex the network must be to solve a classification problem depends on the specific situation. A two layer network is capable of forming any non-linear decision rule. However, a three layer network may be more powerful, and therefore smaller, than a

large two layer network. No more than two hidden layers should be needed for any classification problem [Lippman, 1987]. There is no known method of determining the correct number of neurons to use. The first hidden layer is generally chosen to have the same number of neurons as there are inputs. The size of the second hidden layer is sometimes chosen randomly and optimised by comparing the results. It has been suggested that the number in the second hidden layer should be three times as many as in the first hidden layer [Lippmann, 1987]. The output layer can either be a single neuron, or it can have one neuron for each class to be discriminated, depending on the form of the desired output.

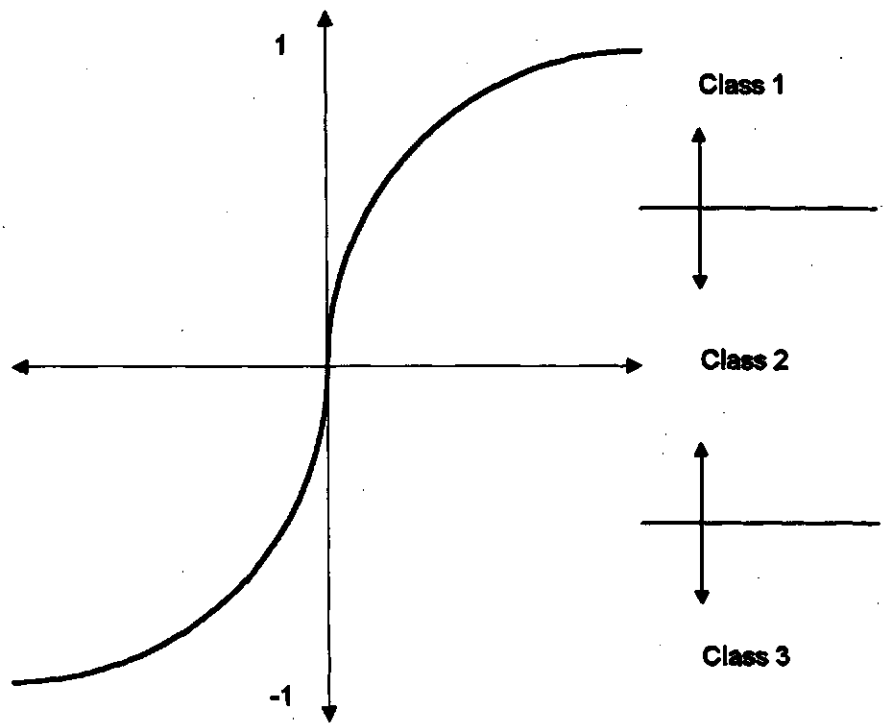


Figure 2.11 Secondary discriminating lines drawn on a sigmoid activation function.

2.3.5 Multi-Structure Neural Network

A multi-structure neural network (MSNN) is a network which was designed specifically as a pattern classifier. It was introduced by Ghazanfari and Irudayaraj [Ghazanfari, 1994; Ghazanfari, 1996] and is a modification of a method proposed by Gupta and Upadhye [Gupta, 1991] for shape classification.

The MSNN consists of separate pre-classifiers, one for each class (see Fig. 2.12). These classifiers are separate feed-forward neural networks which have been optimised for their designated class. The outputs of the pre-classifiers are passed to a maximum selector. The class whose pre-classifier has the largest output becomes the chosen class.

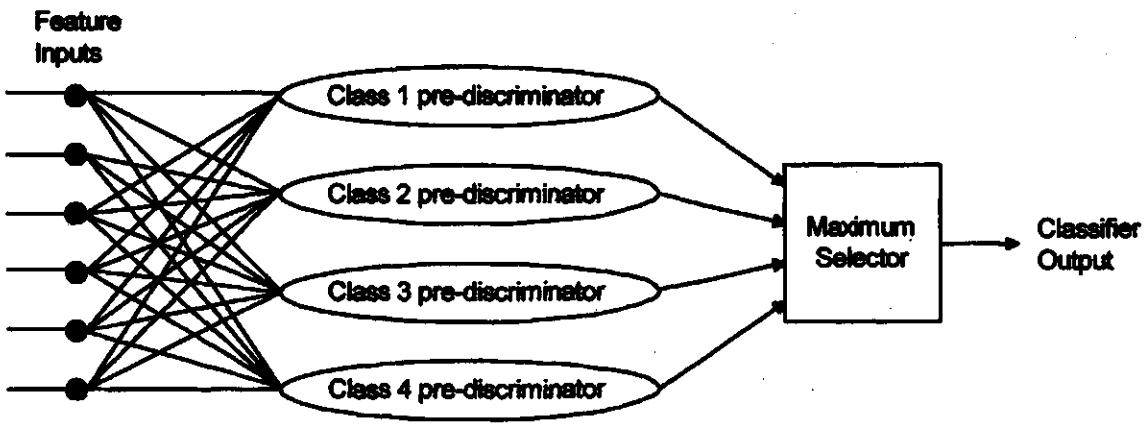


Figure 2.12 The structure of a MSNN with six inputs and four classes.

The pre-discriminators are individually selected and trained by a method referred to as bias training. In bias training, the pre-discriminator sees more data of the biased class than any of the other classes. The pre-discriminators are two class networks; the

class which the pre-discriminator is biased towards makes up one class, while all others are grouped into the other class. The pre-discriminator is trained to give a high positive output when presented with its biased class, and a low output when presented a pattern of any other class. In this way, the pre-discriminator should give the highest output of all pre-discriminators when its biased class is presented, thus allowing it to be selected by the maximum selector.

Ghazanfari (1996) used the MSNN to discriminate four classes of pistachio nuts. He concluded that the MSNN was slightly more suited than the simple feed-forward network for recognition purposes. The bias training required a significantly lower number of training iterations than the feed-forward networks since there are only two classes, and generally fewer neurons. However the total training and processing time of the MSNN was greater due to the sequential processing of each pre-discriminator.

2.4 Summary

This chapter presented the concepts of pattern recognition and neural networks. The basic concepts of pattern recognition were introduced by developing the theory of system features, pattern space, and feature selection. A neural network was then introduced by presenting the concepts and equations behind the artificial neuron. The neurons can be placed in an interconnected structure called a feed-forward neural network. It was shown how the neural network could be made to learn by manipulating neural weights on an error surface. An adaptive learning rate and a momentum factor

were introduced and it was shown how they could speed neural learning. The application of a neural network to the problem of pattern recognition was shown. Finally, the MSNN, a neural network architecture designed specifically for pattern recognition, was introduced.

3. EQUIPMENT AND METHODS

3.1 Introduction

This chapter will introduce and explain the hardware and software used. A new imaging system was assembled for the purposes of this project since the existing system was not capable of processing colour information from an image, and colour is an important aspect in lentil grading. In assembling the equipment, the following system capabilities were considered:

- extraction of a wide range of shape features from the subjects;
- extraction of colour features;
- hardware and software compatibility;
- fast data transfer using state-of-the-art technology (i.e. PCI-bus based);
- reasonably inexpensive.

Figure 3.1 shows a system configuration diagram. The complete system consisted of the following:

Software

- image capture/feature extraction software;
- pattern recognition software;

Hardware

- host computer;
- image capture hardware (computer expansion card used to digitise an image);
- video camera;
- lighting.

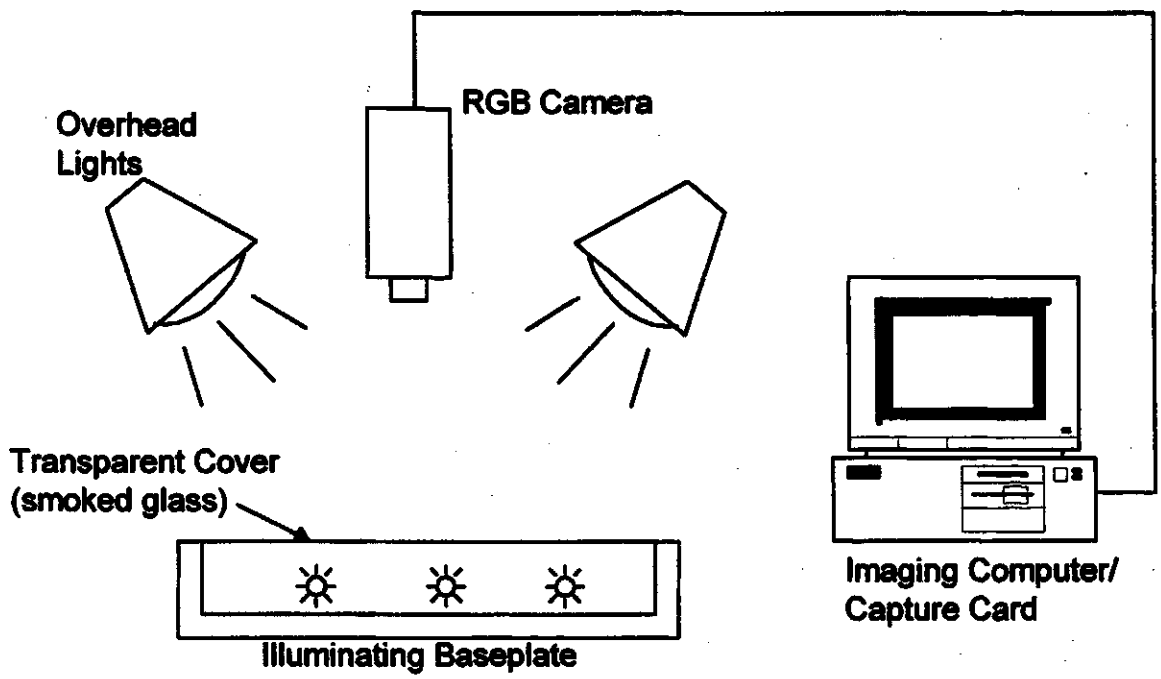


Figure 3.1 The imaging system configuration.

3.2 Software Description

There are two software systems used in this project: capture software, and a pattern recognition package. The capture software is used to interface the video-capture card and the computer. It allows for the digitising and saving of subject images. It also can extract and save quantifiable features from the subjects. The pattern recognition software takes the data saved from feature extraction and uses it in classifying the subjects into desired categories.

3.2.1 Capture Software

The capture software is the most important part of an imaging system. It is responsible for controlling the digitiser card, displaying and saving images, processing the image, and extracting features from the image. There are many commercial software packages available for every type of computer operating system. The widest selection are available for the Microsoft Windows operating system (Microsoft Corporation, Redmond, WA), which was therefore selected for its ease of use and compatibility with most popular application software today.

The first criterion in choosing the capture software was that it had to meet the requirements of the system, namely it had to be able to extract a wide variety of features and be able to operate with colour. These functions, however, are common in most

modern software packages. The selection criterion then became price and hardware compatibility. Most commercial packages, such as Optimas (Optimas Corporation, Bothell, WA) and Image Pro Plus (Media Cybernetics, Silver Spring, MA) were very similar in functionality and use. These packages have a very high price however, because they include extra processing and image filtering functionality which was not needed in this work. The package which was chosen was Inspector version 1.7 (Matrox Electronic Systems Ltd., Montreal, PQ). This software had all of the functions needed, and the price was low since it did not have many extra features. Matrox Inspector also guaranteed hardware compatibility with a digitiser card also made by Matrox.

3.2.1.1 Blob Analysis

Blob analysis is a function of the capture software and is the procedure which allows for identification of connected regions of pixels within a grayscale image, and the calculation of selected features of those regions. The connected regions are known as blobs and correspond to objects in an image, such as lentil seeds in this project.

The following steps are performed when doing a blob analysis:

1. capture or load the image;
2. calibrate pixels to millimetres;
3. convert image to grayscale or extract a single colour component;

4. segment the image;
5. select blob features to calculate; and
6. calculate blob features.

Step one is to either load an image from disk or digitise an image from a camera. This procedure places an image into computer memory where it can be processed. Step number two is the calibration of the image. This is the spatial calibration of the image which relays how many millimetres there are per pixel (image element). Calibration is done by capturing an image of an object of known length (usually a ruler), and then measuring how many pixels make up that length. Inspector has a utility to measure a length and to set the calibration of an image automatically.

Step three in the blob analysis processes is to convert the colour image to a grayscale representation. This is a necessary step since segmentation (explained below) cannot be performed on a colour image. If morphological features are being measured, the image can be converted to grayscale (an averaging of the red, green, and blue intensities). If the colour of the objects is of interest, the red, green, and blue components of the colour image can be extracted into three monochrome images.

The fourth step in blob analysis is to identify which pixels of an image are background pixels and which are foreground, that is pixels making up an object; this is done by segmenting the image. A pixel value (between 0 and 255) is chosen to represent a threshold. Since we are working on a white background (pixel value of 255), all pixels

above the threshold value are considered background pixels, and those below are foreground pixels. Figure 3.2 shows a typical image histogram. The rightmost peak in this figure is the background peak, and the left is the foreground peak. The threshold is chosen between these peaks, in this case at a value of 185. There is typically a background peak and a foreground peak in an image, so the threshold can be chosen between these two peaks. In a colour image, there are three sets of intensity values per pixel, one for each colour component (red, green, and blue). Thus, it is not possible to find a single threshold value on a colour image. The image must be converted to grayscale where only one set of intensity values exists.

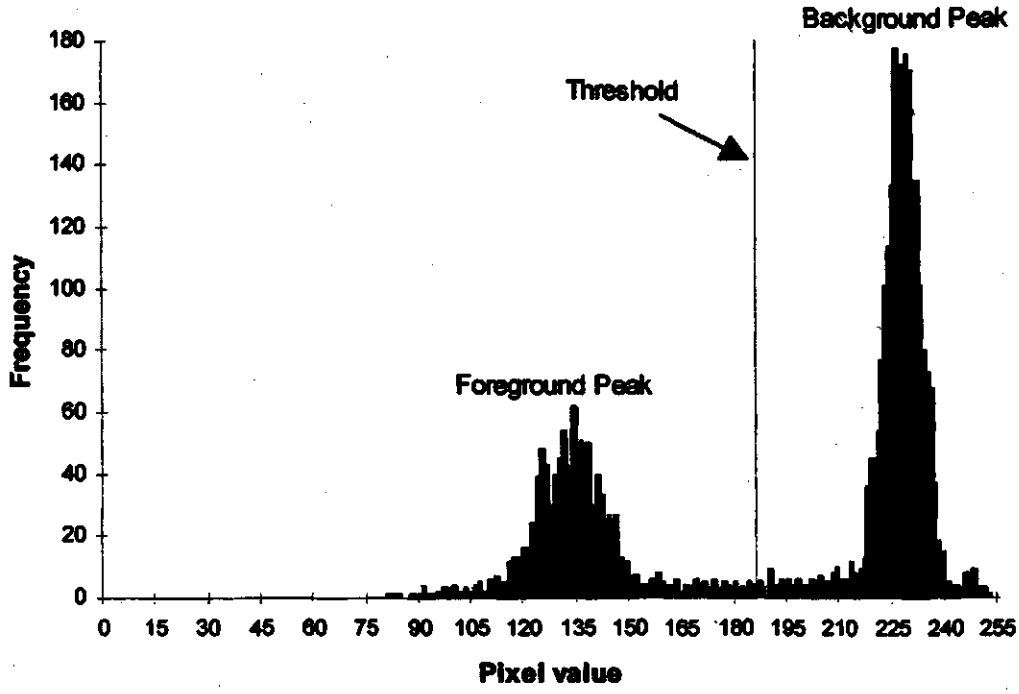


Figure 3.2 A typical image grayscale histogram.

Matrox Inspector has a graphical utility for doing segmentation (see Fig. 3.3). The image is shown in a segmentation window, and the threshold is changed via a scroll bar until all seeds have been highlighted. Once the threshold value is set, the blobs are identified by finding touching foreground pixels in the image. Since small particles of dust or dirt could also be under the threshold value and be foreground pixels, they might be mistaken for blobs. This source of error is removed by setting a minimum number of touching pixels before being classified as a blob. For example, a dirt particle might be made up of a total of 20 touching pixels and be passed over as a blob, while a lentil seed, made up of 500 pixels, is labelled as a blob.

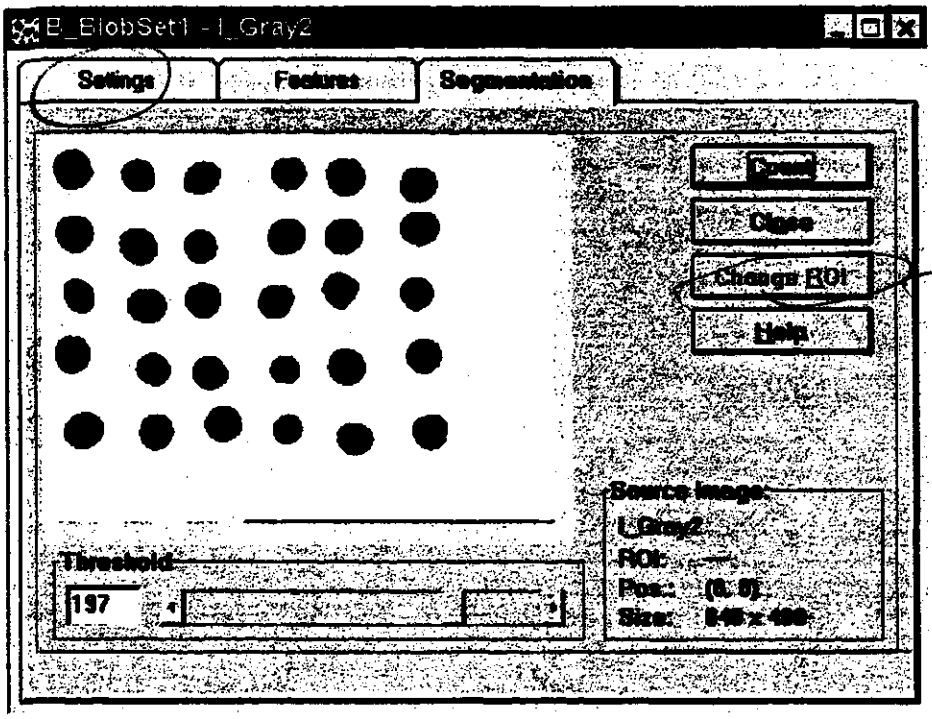


Figure 3.3 The segmentation tool in Matrox Inspector.

When finding blobs in an image, Matrox Inspector allows for choosing whether two diagonally adjacent pixels are considered touching by setting the *connectivity*. Images are represented by a square lattice of pixels (see Fig. 3.4). If connectivity is set to "4 neighbours", only the vertically and horizontally adjacent pixels are considered to be touching. If connectivity is set to "8 neighbours", diagonally adjacent pixels are also considered to be touching. Thus, in Fig. 3.5, there is one blob if connectivity is set at "8 neighbours", and 2 blobs if set at "4 neighbours".

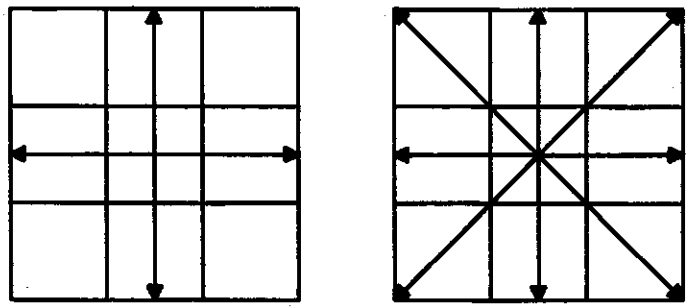


Figure 3.4 A lattice with 4 neighbours (on the left) and a lattice with 8 neighbours (on the right). This figure is taken from the Matrox Inspector User's Manual.

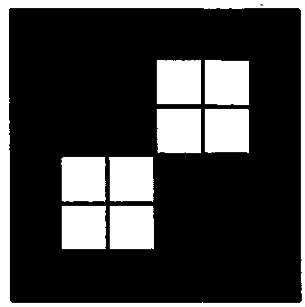


Figure 3.5 The effects of the number of neighbours on blob segmentation. The image has 2 blobs when 4 neighbours are used and 1 blob when 8 neighbours are used. This figure is taken from the Matrox Inspector User's Manual.

After finding the blobs in an image the next steps are to decide which features of the blobs to measure, and then perform the measurements. The measurable features are described in the next section. Generally, morphological features, such as area and perimeter, are measured on a grayscale image, while the colour measurements are taken from the extracted colour component images. After measurements are taken, the features are displayed in a table, which can then be saved as an ASCII text file for use in other programs.

3.2.1.2 Blob Feature Descriptions

Matrox Inspector has over 60 features which it can measure from a blob. Only a few of these features are useful for pattern recognition purposes however. The features described below were the features used in this work.

Area

This is the number of foreground pixels in a blob, converted to square millimetres. This area does not include any holes in the blobs. To insure that there are no holes in the blob, the threshold must be chosen such that the entire seed is covered.

Perimeter

This is the total length of edges in a blob (including the edges of any holes). An allowance is made for the staircase effect which occurs when diagonal edges are digitised; Fig. 3.6 shows how this error in the perimeter occurs. In quantising an image, the objects in the

image are made up of square pixels. Thus, the diagonal sides of an object are made up of what looks like steps in a staircase. If the perimeter was measured with these steps, a substantial error would build up. To allow for this error, a factor of 1.414, rather than 2.0, for two adjacent edges, is used.

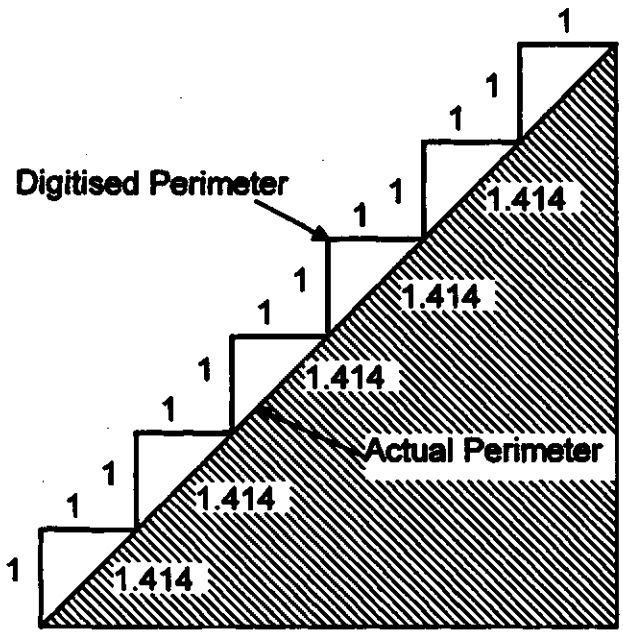


Figure 3.6 The staircase effect in digital perimeter measurement.

Convex Perimeter

This is the perimeter of the convex hull of a blob (see Fig. 3.7). It is the perimeter as if the blob had no concave curves in its perimeter (curves in toward the centre of the blob). It is always equal to or smaller than the actual blob perimeter.

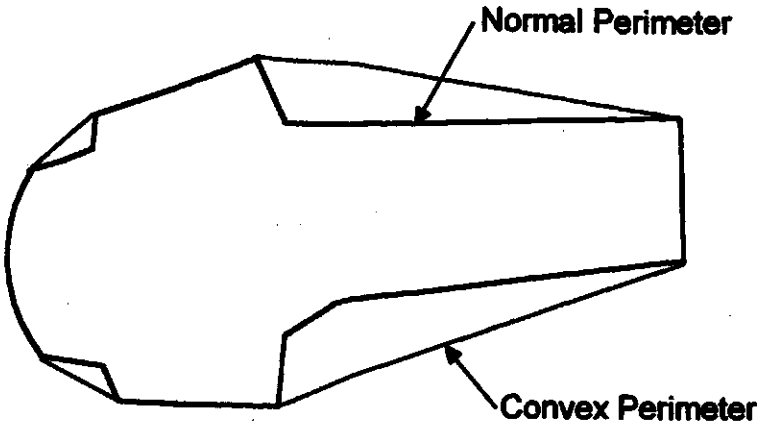


Figure 3.7 The convex perimeter of a blob.

Compactness

A measure of roundness, called compactness, is derived from the perimeter (p), and area (A) of a blob. A perfect circle has a compactness of 1.0. All other shapes have a compactness greater than 1.0.

$$\text{Compactness} = \frac{p^2}{4\pi A} \tag{3.1}$$

Roughness

Roughness is a measure of how smooth the perimeter of a blob is. A smooth convex object has the minimum roughness value of 1.0.

$$\text{Roughness} = \frac{\text{Perimeter}}{\text{Convex Perimeter}} \tag{3.2}$$

Length

Length is the size of the major axis of a blob (see Fig. 3.8). This feature is derived from area and perimeter measurements, using the assumptions that:

$$\text{Area} = \text{length} \times \text{breadth} \tag{3.3}$$

and

$$\text{Perimeter} = 2(\text{length} + \text{breadth}). \tag{3.4}$$

Breadth

Breadth is the size of the minor axis of the blob (see Fig. 3.8).

Elongation

This is the ratio of length of the blob to its breadth.

$$\text{Elongation} = \frac{\text{length}}{\text{breadth}} \tag{3.5}$$

Intercept 0

This is the number of times that a transition from background to foreground occurs in the horizontal direction for the entire blob. It is a colour feature and is a measure of how variable the surface colour of the blob is.

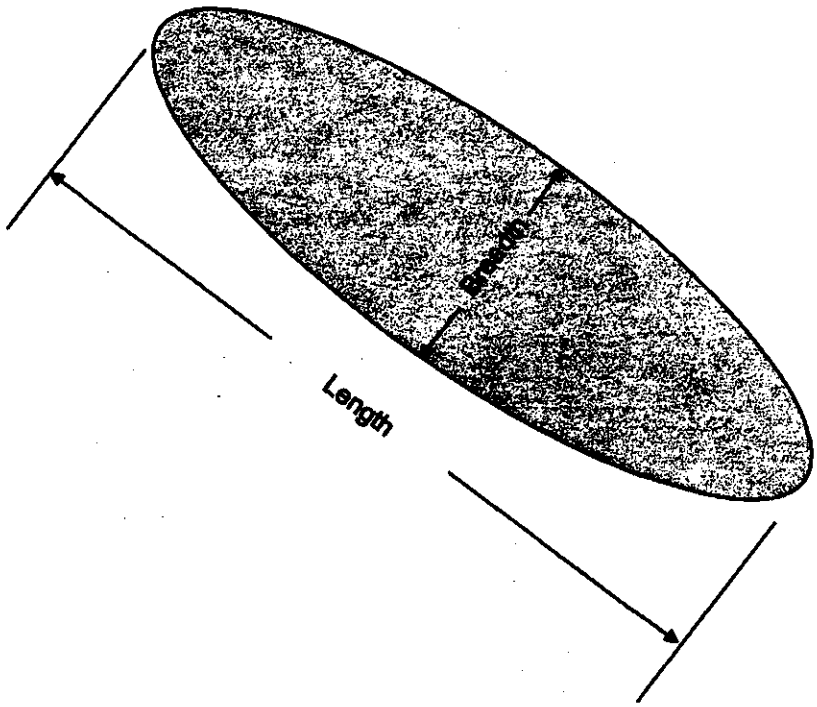


Figure 3.8 The length and breadth of a blob.

Mean Pixel

This is the average pixel value found in a blob. It is a value between 0 and 255 and can be found for each of three colour components; red, green, and blue. A value of 0 corresponds to the lowest intensity possible (solid black in a grayscale image), and a value of 255 corresponds to the maximum intensity possible (solid white in a grayscale image).

Maximum Pixel

This is the maximum pixel value found in the blob for the three colour components.

Minimum Pixel

This is the minimum pixel value found in the blob for the three colour components.

Standard Deviation of Pixels

The standard deviation of pixel values in a blob (for each colour component).

$$S.D. = \sqrt{\frac{\sum p_i^2 - (\sum p_i)^2 / N}{N}} \tag{3.6}$$

where N = the number of pixels, and

P = the pixel value.

3.2.2 Pattern Recognition Software Description

After features have been measured by Inspector, the data are saved as a text file. The software used to perform recognition on this data was Matlab with the Neural Network Toolbox (The Mathworks Inc., Nattick, Mass.) on which was implemented a number of neural networks of different size and configuration. Since this procedure was computationally intensive, it was run on a Sparcstation 2 workstation (Sun Microsystems Inc., Mountain View, CA).

The Matlab program accepts the training data in a text format, along with the desired neural outputs. The number of neurons per layer, number of iterations, and desired error goal can be set by the user. After training, the weights can be saved, and

used to run a test file. The final output is an array of the neural outputs, which can be compared to the desired outputs to determine the percent correctly identified.

3.3 Hardware Description

3.3.1 Host Computer

The main platform for this system was a Pentium personal computer. The computer had a speed of 133 MHz and 32 megabytes of on-board memory. The operating system was Windows '95. The video display was a 15" colour monitor. This system was chosen for this project because it had the widest range of image hardware and software available for it. The system is equipped with a Peripheral Component Interconnect (PCI) bus which allows for high speed transfer of data from the capture card into computer memory. The PCI bus allows for up to 132 Megabytes per second (MB/s) data transfer. For live video from a capture card, 30 frames per second must be transferred at 1 MB per frame [Matrox, 1996]. Therefore a sustained data transfer rate of over 30 MB is needed. Most modern capture cards made for the small computer today are based on the PCI bus.

Along with the image capture card, the Matrox Inspector program resided in the host computer. A copy of Matlab also ran on this computer, however, for the purposes of training the neural networks, the data was transported to a workstation due to the limitations of computational speed in the personal computer.

3.3.2 Image Acquisition Board

The image acquisition board, or framegrabber, is a peripheral add-in card which digitises the analogue signal from a camera so that it can be operated on by a computer. The Matrox Meteor/RGB (Matrox Electronic Systems Ltd., Montreal, PQ) card was used in this project. Complete specifications can be found in Appendix A.

The Meteor card was installed in the host computer in one of the PCI slots. Figure 3.9 shows a block diagram of the Meteor capture card. It was connected to the camera with an RGB (red-green-blue) interface. This means that the card had an input for each of the three colour channels. The Meteor converted the analogue signals into 8 bits of digital data which translates into 256 levels of reflectance intensities. For each RGB channel, the capture card digitised 8 bits of data, for a composite image of 24 bits. As can be seen on the block diagram, the Meteor card also has inputs for 4 composite channels, and an S video input, which were not used in this project. The Meteor was capable of capturing an image of 640 pixels by 480 pixels. For an image width of 110 mm, this gives a resolution of 0.17 mm/pixel. A lentil is approximately 7 mm wide, therefore an image of a lentil would be approximately 41 pixels wide. This resolution was considered to be sufficient for this work as other projects had successfully used similar resolutions [Li, 1996; Shrestha, 1996].

Cables connecting the card to the camera were built in the laboratory using shielded cable and pin-outs supplied by the camera and card manufacturers. The complete cable wiring can be found in Appendix A.

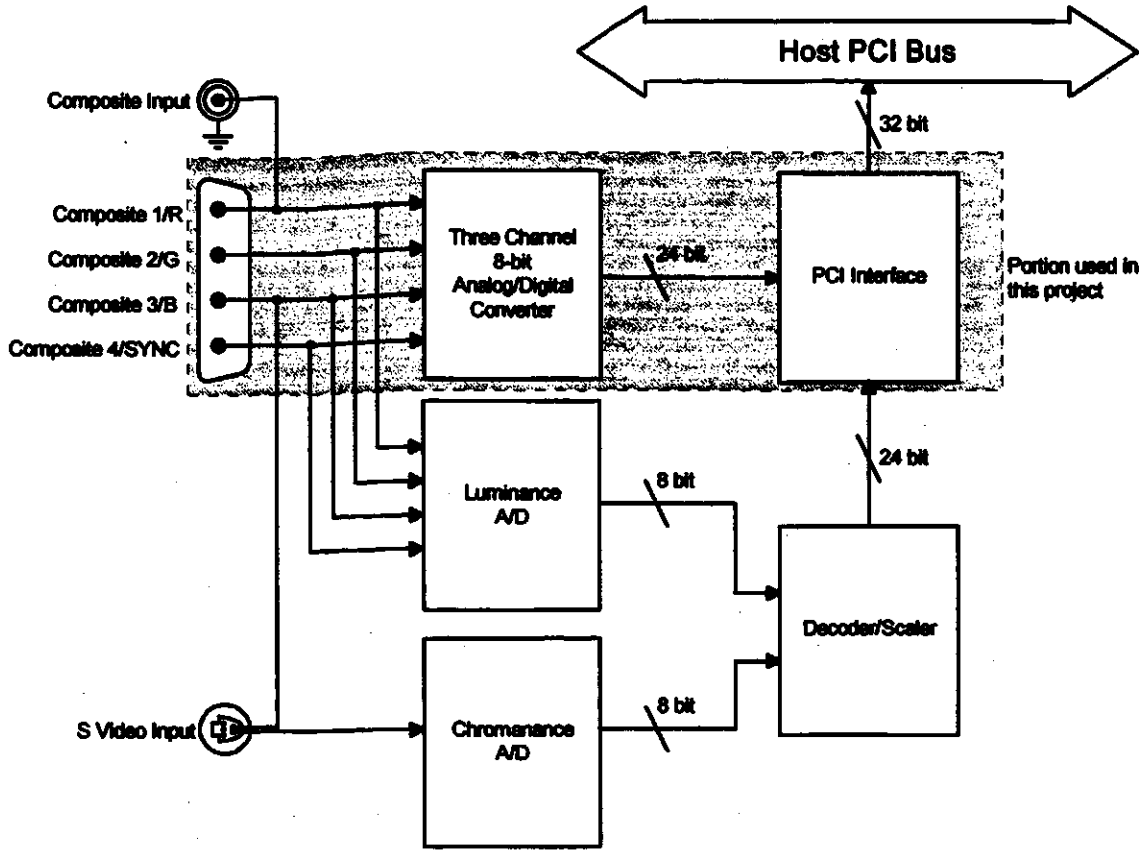


Figure 3.9: The block diagram of the Matrox Meteor image capture card. This diagram was taken from a Matrox Meteor specification sheet.

3.3.3 Camera

A Sony DXC-151A CCD colour video camera (Sony of Canada Ltd., Willowdale, ON) was used for imaging the lentils. This camera has a sensing array of 768 horizontal

elements and 493 vertical elements. The camera had an RGB output via a D-sub 9-pin connector. There was also a composite video output via a BNC connector which can be used to simultaneously connect a television monitor for easy adjustment of focus.

This camera had four modes of white balance selectable by the user. A white balance calibrates the three colour outputs of the camera to give a desired white value under the given operating conditions. The mode which was used was Auto mode, which allows for the automatic setting of the white balance. In this mode, the operator presses the AUTO button, and the white balance is adjusted according to the colour temperature of the subject. This balance value is stored in memory until the AUTO button is pressed again. In this way the output of the camera will remain constant assuming the external lighting remained the same. The Sony DXC-151A also offers a gain control and a shutter speed adjustment. Gain control is used to boost the output of the camera in poor light conditions, and shutter speed is used to capture images from moving objects. Both were turned off in these tests.

The lens used was a telephoto lens with manual aperture control. This lens could give variable viewing area from the camera without adjusting camera height. A +2 lens and a +4 diopter lens were used to decrease the focal length so that the image could be brought into focus within the desired viewing area.

3.3.4 Lighting

Both a top-lighting system and a back-lighting system were used in this work. The top-lighting was supplied by two incandescent bulbs of 150 watts each. The top-lighting was necessary to extract colour information from the lentils. The second lighting system was provided by a lightbox. Normally, a lightbox is used for back-lighting purposes where only a silhouette of an object is needed. However, the lightbox in this project served to reduce shadow effects. The shadows created by the overhead lighting fell on the smoked bottom surface of the glass instead of the top surface where they would interfere with the segmentation process. Also, the fluorescent lighting inside the lightbox made the smoked surface a good white contrast from the subjects.

3.4 Summary

This chapter described the hardware and software used in developing a lentil discrimination system. The software was the Matrox Inspector program which provided the image capture and feature extraction capabilities of the system. A description of the features which could be measured by Inspector and which were used in this work was given. Matlab with the Neural Network Toolbox was the software with which the neural network pattern recognition algorithms were implemented. The hardware was a Pentium PC running Windows '95. The capture card was a PCI-bus Matrox Meteor which was connected to a Sony DXC-151A RGB video camera. Two lighting systems were used to

illuminate the objects: two overhead incandescent bulbs to give colour and shape features, and a lightbox to reduce shadow effects.

4. IMPLEMENTATION AND RESULTS

4.1 Introduction

The previous chapters explained the methods and equipment behind the inspection and pattern recognition systems used in these experiments. This chapter will present the experimental data. The objective was to discriminate between three grades of lentils: good, discoloured, and broken and peeled. The process of extracting feature data and creating training and testing sets will be described. The results of training the two neural network configurations will be shown and compared. Finally, the implementation and results of the mass-area correlation will be presented.

4.2 Lentil Discrimination

Figure 4.1 shows the procedure for creating the pattern recognition system needed for this project. A random sample of Laird lentils is separated manually into the three desired classes. These lentils are imaged and the data extracted. This data is divided equally into a training set and a test set. The training set is used to train the network, while the test set is used to evaluate the network after the weights have been set in the training procedure.

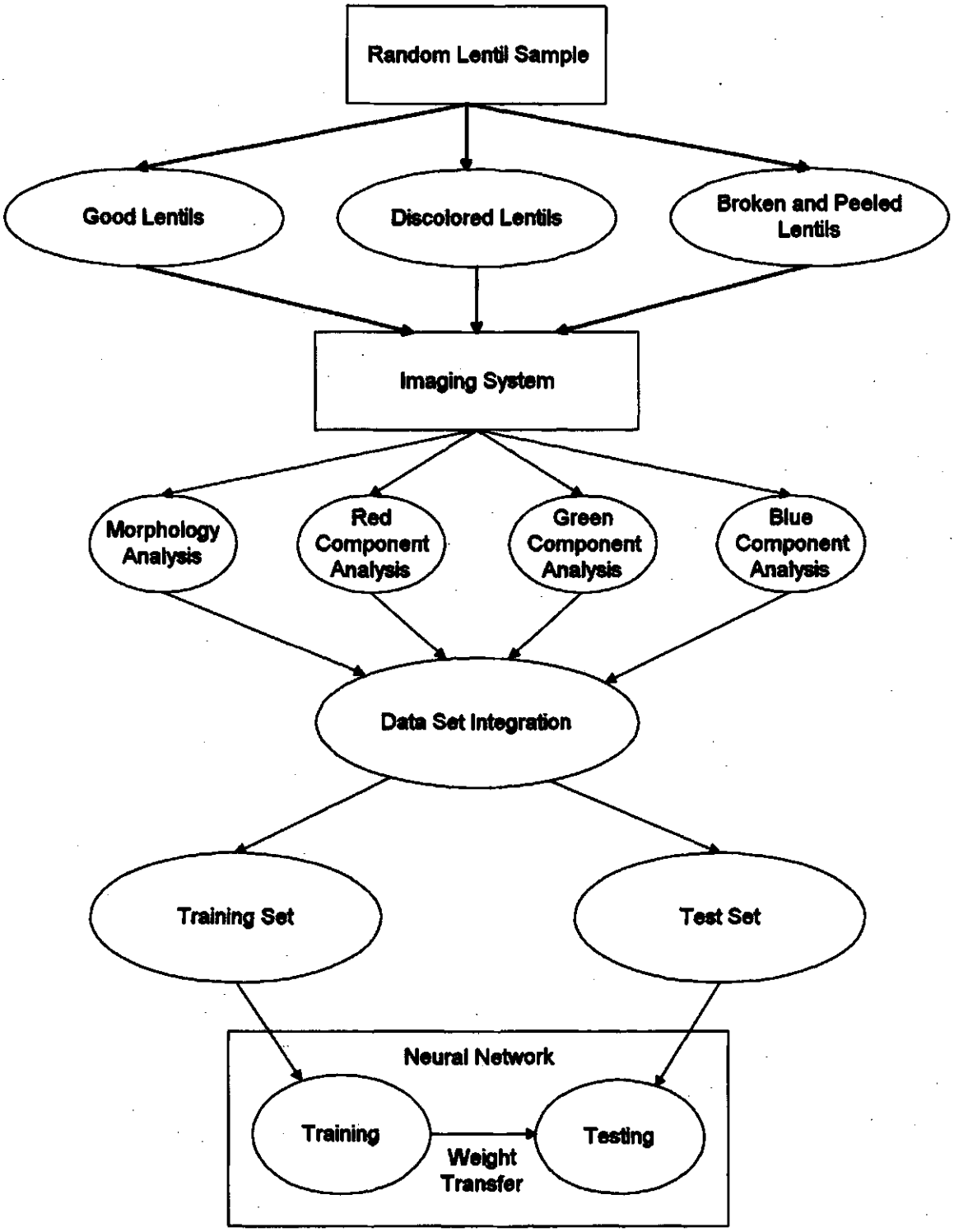


Figure 4.1 Procedural Flowchart for Creation of the Recognition System

4.2.1 Feature Extraction

A 45 kg bag of Laird lentils was received directly from a Saskatchewan producer, and several random samples were taken from it. The samples contained all three desired classes of lentils as well as foreign particles such as wild oats. Each sample was divided into 200 gram groups and passed through two roundhole sieves, a No. 12 (12/64 inches) and a No. 15 (15/64 inches). These sieves had the effect of removing almost all foreign particles from the sample.

The sample, now free from foreign particles, was divided by hand into the three classes, good, discoloured, and broken and peeled. The *discoloured* sample was defined as having darker or whiter spots, or being completely off-coloured to a "significant" degree. This amount of "significance" was arbitrarily set at the beginning of sample separation, and was kept the same throughout the separation process. Figure 4.2 shows a typical sample of discoloured lentils. The *broken and peeled* class consisted of lentils with less than three-quarters of the entire lentil remaining and/or less than half of the seed coat intact. Figure 4.3 is a typical sample of broken and peeled lentils. The lentils that did not fit into these first two classes were grouped into the *good* class. Figure 4.4 shows a sample of good lentils. The process of cleaning and dividing the 200 gram samples was continued until an adequate number of kernels of each class was obtained. This resulted in many more *good* lentils than *discoloured* or *broken and peeled.* To get equal amount of data for each class, random selections of kernels were taken from the *good* class.

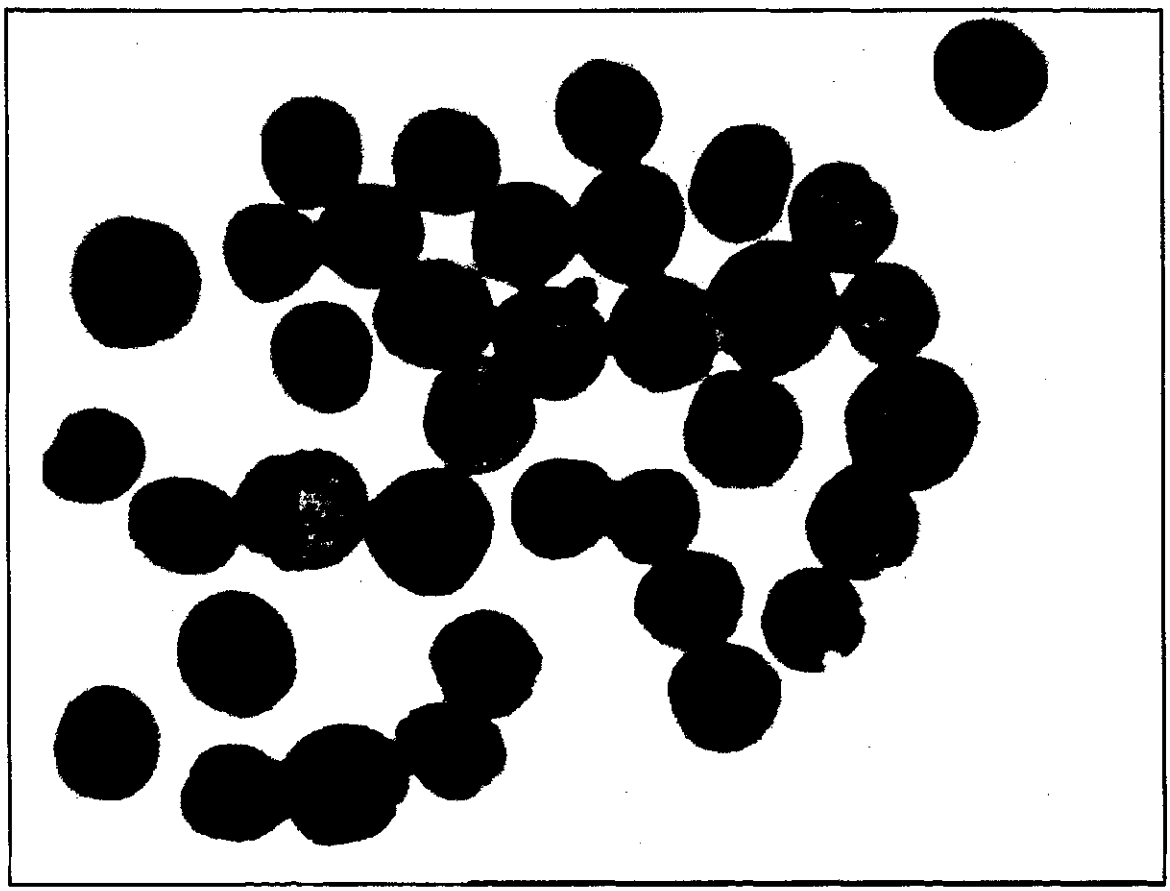


Figure 4.2 A typical sample of discoloured *Laird* lentils.

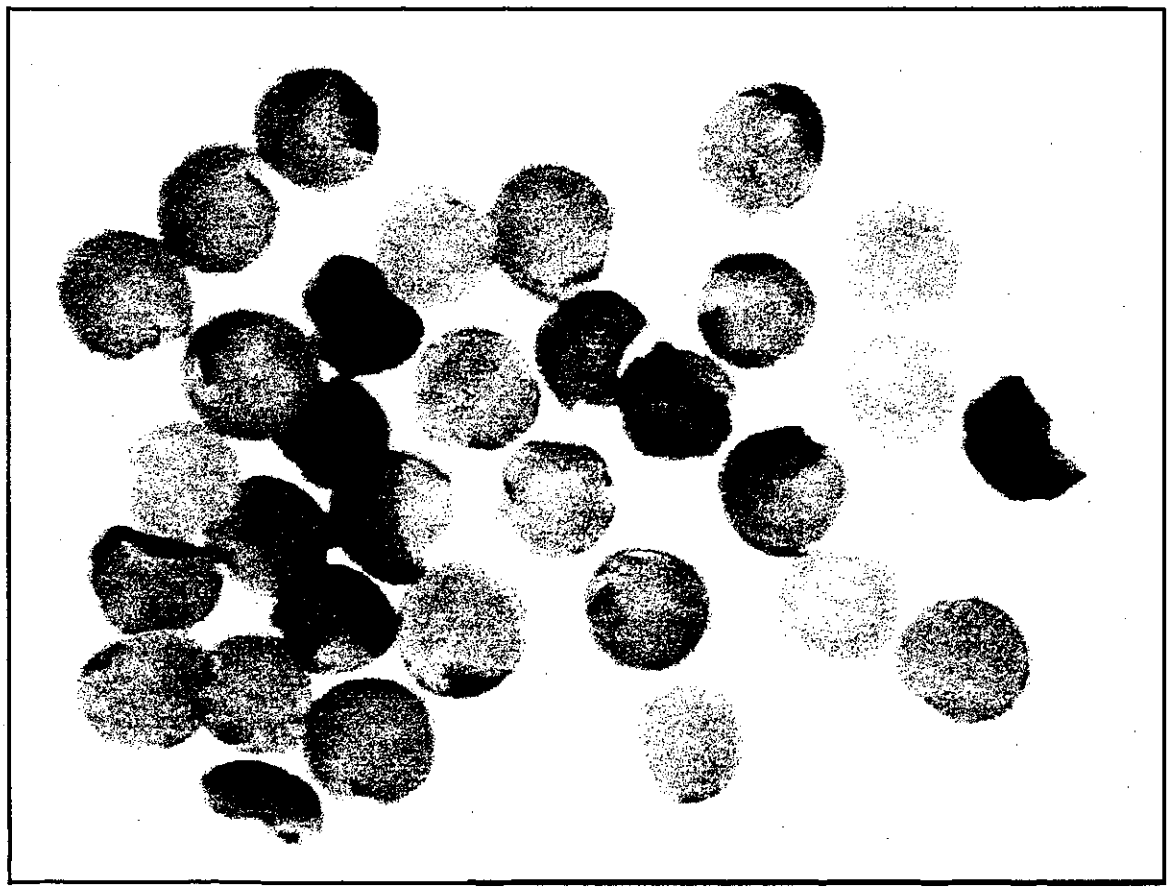


Figure 4.3 A typical sample of *broken and peeled* Laird lentils.

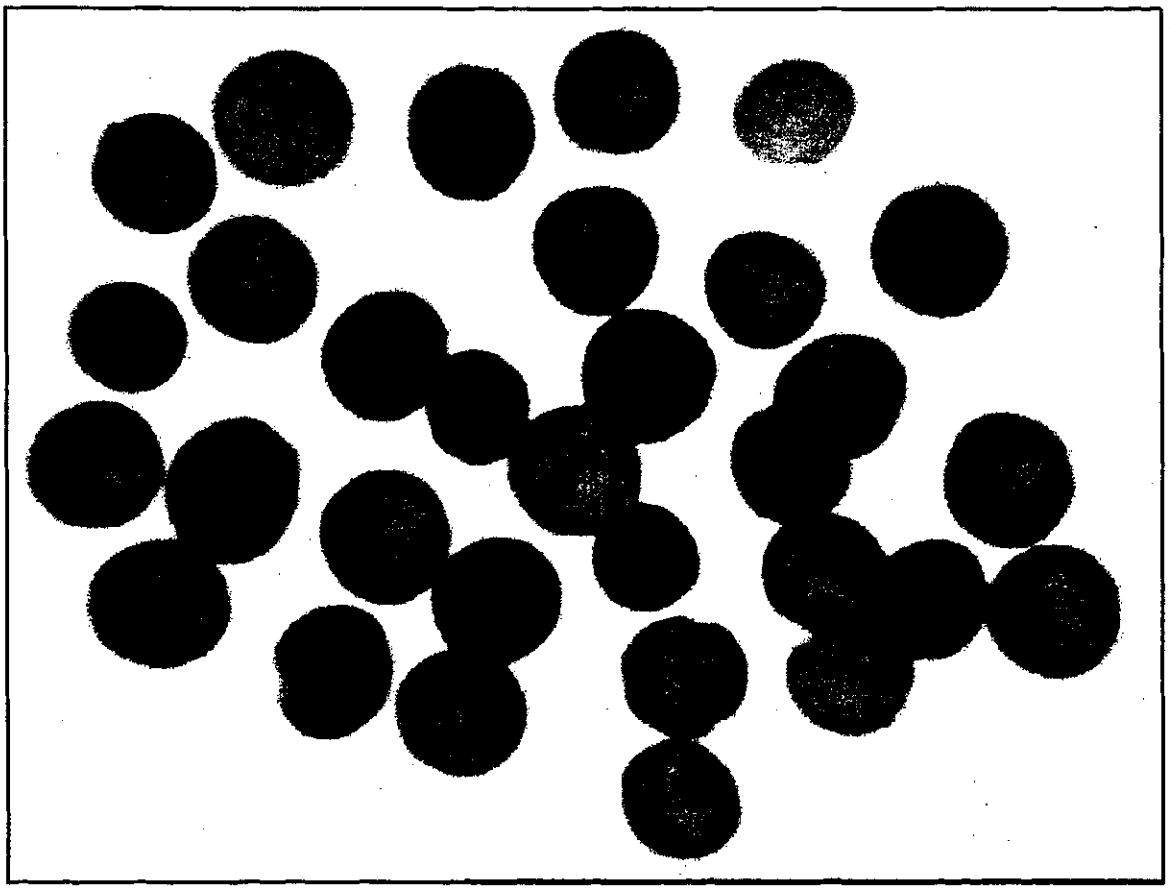


Figure 4.4 A typical sample of *good* Laird lentils.

4.2.1.1 Imaging

The next step in feature extraction was to take the digital images of the kernels. The camera was placed at a height of 78 cm from the base. A x4 and a x2 lens were added to the zoom lens of the camera so that it was able to focus on the kernels. The field of view was 11 cm long by 8 cm wide. Two diagonally opposing top-lights were used, as well as the lightbox for backlighting. The kernels were arranged in the field of view in such a way that no two kernels were touching each other. Twenty kernels at a time were imaged.

The seeds were imaged and the data were stored as uncompressed files on the host computer hard drive. To prevent possible changes to the equipment between images, which would upset the calibration, all images were taken consecutively and saved to disk. Processing was performed on the images later.

4.2.1.2 Blob Analysis

Once all the images were taken and stored, the next step was to perform blob analysis on them. The procedure for blob analysis was described in Chapter 3. The image was converted to grayscale to perform the morphology analysis. Colour analysis was done when the composite image was broken into its three colour component images.

The extraction of the red colour data required a special procedure. The mean value of the red pixels on the peeled lentils was sometimes very close to the background value. This made normal segmentation impossible. To be able to segment these images, the background had to be deleted. This was accomplished by converting the grayscale image to a binary image. This made the blobs completely white, and the background completely black. This binary image was then AND'ed with the composite image using Inspector's mathematical capabilities. The resulting image had the kernels on a solid black background. The red component could then be extracted and segmented as normal.

4.2.2 Training and Testing Data

The data obtained by the feature extraction procedure was now in many ASCII text files, each containing the data for 20 kernels. These files were then put together to form three text files, one for each class of lentils. At this point, each file contained 480 individual data vectors.

The next step was to create a training data set and a test data set. The three files were split equally into six files, two of each class. To create the training set, three of the files, representing the three classes, were shuffled together. The new file consisted of 720 vectors with each vector alternating classes. This shuffling was necessary in the training set so that the neural network does not train on one class for a long time. The test set was then the remaining three files appended to one another. It had 720 vectors with each entire class set coming one after another. Shuffling was not necessary in this file since it

was only used for testing the neural network and no training was done with it. Tables 4.1 and 4.2 show the makeup of the training and test sets. These text files are ordered so that each column is a different feature, and each row makes up an entire vector (all the features of one lentil seed).

Table 4.1 Training Set Format

Training Set	
Vector No.	Vector Class
1	good
2	discoloured
3	broken and peeled
4	good
5	discoloured
6	broken and peeled
⋮	
718	good
719	discoloured
720	broken and peeled

Table 4.2 Test Set Format

Test Set	
Vector No.	Vector Class
1	good
2	good
3	good
⋮	
241	discoloured
242	discoloured
243	discoloured
⋮	
481	broken and peeled
482	broken and peeled
⋮	
720	broken and peeled

4.2.3 Feature Ranking and Selection

The first step in ranking the features was to find their probability of error (POE), the overlap of the distribution curves for the population. In this work it was first assumed that the features would follow a standard normal distribution which is described by Equation 4.1. However a Chi-squared test of fit showed that most of the features did not

fit well to the normal distribution. Table 4.3 shows the results of a partial list of the features. To be a good fit, the observed χ^2 must be less than the calculated χ^2. The colour features seem to be close to a normal distribution, and only the mean red pixel value feature was the only feature to pass the chi-squared test. This meant that the measured features did not follow a normal distribution, given by Equation 4.1;

$$f(x,\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \tag{4.1}$$

where x is the value for which the distribution is being found,

μ is the mean of the distribution, and

σ is the standard deviation of the distribution.

Table 4.3 A chi-squared test of fit to a normal distribution of features of good lentils.

	Degrees of Freedom	Observed χ^2	$\chi^2_{0.95}$ (from table)
Mean Red	10	8.95	18.31
Mean Blue	7	17.08	14.07
Perimeter	9	21.87	16.92
Mean Green	10	28.85	18.31
Area	10	46.06	18.31
Convex Perimeter	11	52.24	19.68
Breadth	7	64.11	14.07
Intercept 0	9	67.48	16.92
Length	11	157.07	19.68
Roughness	10	174.08	18.31
Compactness	9	178.38	16.92
Elongation	3	558.08	7.81

Even though it was shown that the features did not follow a normal distribution, it was left for future work to derive the exact distributions of the lentil features. The POE

was found using normal curves as an approximation. This caused little error in the final result of feature selection since POE was used only as the first step in ranking features. It is used to choose the first feature for ranking by the Accumulated Correlation Coefficient method, and was a refinement from simply choosing the feature at random. Table 4.3 shows a very large discrepancy for some of the features. For example, the observed chi-squared value for Compactness was 178.38, while the calculated chi-squared value was 16.92. This would seem to indicate that Compactness is far from being a normal distribution. However, if one looks at the histogram for Compactness (Fig. B4 in Appendix B), the main curve does look Gaussian. Only a few outlying points cause an increase in the chi-squared value. Thus, even though the features were not strictly Gaussian, and should not be represented as such in more extensive modelling, approximating them as normally distributed should not cause significant difference in feature ranking.

The POE was found by numerically integrating under the distribution curves. Table 4.4 shows the approximated POE for each feature along with the mean and standard deviation value for the feature. The features are sorted according to the POE with the smallest POE coming first. The POE in this table is the overlap of the three curves describing the three classes of lentils. The feature with the best (least) POE was the mean of the green component with a POE of 0.074. The normal distribution curves for this feature can be found in Fig. 4.5. The distribution curves for the rest of the features may be found in Appendix B.

Table 4.4 Lentil Features Sorted by Probability of Error.

	Good		Discolored		Broken and peeled		
Feature	Mean	St. Dev.[¥]	Mean	St. Dev.	Mean	St. Dev.	P.O.E.
Mean of Green Pixels[§]	128.0	7.9	103.9	7.8	165.2	17.2	0.074
Mean of Red Pixels	154.7	8.2	130.2	10.3	204.0	18.1	0.080
Standard Deviation of Green Pixels	20.4	2.6	29.9	3.3	13.1	5.4	0.144
Standard Deviation of Red Pixels	17.3	2.3	26.2	3.6	12.1	5.3	0.152
Maximum Red Pixel Value	217.2	3.7	215.1	4.9	233.7	6.9	0.169
Maximum Green Pixel Value	204.3	3.1	203.3	3.3	208.6	3.9	0.177
Standard Deviation of Blue Pixels	23.7	2.6	31.0	3.0	18.5	4.5	0.200
Maximum Blue Pixel Value	200.3	4.5	203.9	5.0	191.1	6.9	0.275
Mean of Blue Pixels	110.2	6.3	97.5	6.1	120.5	9.8	0.294
Minimum Red Pixel Value	99.6	14.4	65.9	17.3	157.6	33.3	0.310
Minimum Blue Pixel Value	61.7	14.5	37.4	15.1	88.6	19.2	0.379
Breadth (mm)	5.78	0.46	5.18	0.46	5.02	0.80	0.422
Length (mm)	5.93	0.64	5.23	0.47	5.98	0.74	0.426
Perimeter (mm)	23.42	1.65	20.83	1.75	22.00	1.50	0.431
Convex Perimeter (mm)	22.09	1.43	19.74	1.63	20.51	1.42	0.447
Area (mm^2)	36.57	4.61	29.12	4.96	30.54	4.74	0.448
Intercept 0	43.04	3.13	36.17	3.49	40.08	3.62	0.448
Minimum Green Pixel Value	77.8	14.3	43.5	13.4	126.3	32.5	0.470
Compactness	1.20	0.06	1.19	0.04	1.27	0.08	0.512
Roughness	1.06	0.02	1.06	0.01	1.07	0.02	0.556
Elongation	1.03	0.18	1.01	0.08	1.24	0.37	0.556

¥St.Dev.=standard deviation based on n=480

§Pixel values are based on a number from 0 to 255 (black=0, white=255)

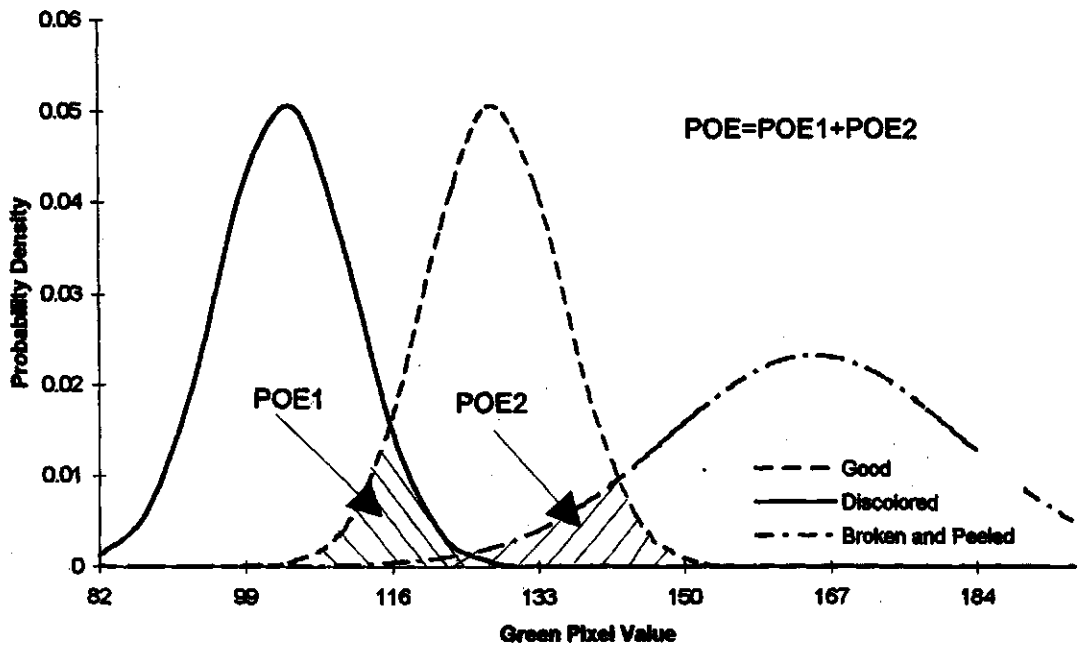


Figure 4.5 The normal distribution functions for the green component feature.

The POE gives no information on which features can be correlated to other features however. If a particular feature has a high correlation to another feature, then only one of the two is contributing toward pattern recognition. Therefore, the next step in ranking the features was to perform a correlation analysis on them. This was accomplished in an Excel spreadsheet on all the data gathered, and the results are given in Table 4.6. These results show the correlation values of one feature to another feature. A value of 1.0 would indicate 100% correlation while a value of 0 would be no correlation. This table shows that features such as area and perimeter were highly correlated to one another (0.95) while colour features and morphological features correlated very little to one another. For example, area and green pixel value had a correlation value of 0.02.

To rank the features, the method of Accumulated Correlation Coefficient (ACC) was used, as explained in Section 2.2.1.1. This method starts with the feature with the lowest POE and ranks the remaining features in order of the lowest average correlation with the previously ranked features. The results of this analysis are shown in Table 4.5 which shows features ranked according to increasing ACC.

The final step in feature selection is to determine how many features were necessary to obtain a desired degree of accuracy. This is done by eigensystem analysis as described in Section 2.2.1.2. This process involves finding the eigenvalues from the correlation matrix of the features (see Table 4.6), and then using Equation 2.5 to find the approximate error to be expected if the first n features are used for recognition. The eigenvalues were found by using the Matlab program, and are given in Table 4.7. The

graph obtained by plotting Equation 2.5 is given in Fig. 4.6. This graph shows that an accuracy of 99% can be expected if the first 10 features shown in Table 4.5 are used.

Table 4.5 Features ranked by the Accumulated Correlation Coefficient method.

Feature	Accumulated Correlation
Mean of Green Pixels	
Area	0.020
Roughness	0.235
Max Green Pixel	0.308
Convex Perimeter	0.261
Max Blue Pixel	0.314
Elongation	0.341
Intercept 0	0.314
Min Blue Pixel	0.346
Perimeter	0.378
Max. Red Pixel	0.377
Compactness	0.392
Mean Blue Pixel	0.412
Breadth	0.459
Length	0.443
Std. Dev. of Red Pixels	0.450
Min. Green Pixel	0.481
Std. Dev. of Green Pixels	0.510
Std. Dev. of Blue Pixels	0.534
Min. Red Pixel	0.551
Mean Red Pixel	0.582

Table 4.6 Correlation matrix of the measured lentil features.

	Area	Perim.	Conv. Per.	Comp.	Rough.	Length	Broth	Elong.	Int. 0	Min Red	Max Red	Red	SD Red	Min Green	Max Green	Green	SD Green	Min Blue	Max Blue	Blue	SD Blue
Area	1.00	0.95	0.99	-0.26	-0.05	0.50	0.84	-0.24	0.91	-0.02	-0.09	-0.01	-0.18	-0.03	-0.03	-0.02	-0.10	-0.02	0.10	-0.03	-0.15
Perimeter	0.95	1.00	0.98	0.04	0.22	0.72	0.68	0.02	0.91	0.12	0.05	0.14	-0.29	0.11	0.08	0.13	-0.23	0.09	-0.02	0.10	-0.26
Conv. Perim.	0.99	0.98	1.00	-0.15	0.01	0.56	0.79	-0.14	0.92	0.04	-0.03	0.05	-0.23	0.03	0.01	0.05	-0.16	0.03	0.05	0.02	-0.20
Compact.	-0.26	0.04	-0.15	1.00	0.86	0.65	-0.63	0.90	-0.11	0.45	0.45	0.46	-0.34	0.45	0.35	0.47	-0.37	0.37	-0.36	0.41	-0.33
Roughness	-0.05	0.22	0.01	0.86	1.00	0.68	-0.39	0.72	0.09	0.40	0.41	0.41	-0.32	0.40	0.34	0.42	-0.35	0.31	-0.33	0.35	-0.32
Length	0.50	0.72	0.56	0.65	0.68	1.00	-0.02	0.70	0.56	0.35	0.30	0.36	-0.40	0.34	0.25	0.37	-0.38	0.30	-0.23	0.33	-0.39
Breadth	0.84	0.68	0.79	-0.63	-0.39	-0.02	1.00	-0.71	0.72	-0.19	-0.24	-0.19	0.00	-0.20	-0.15	-0.20	0.07	-0.18	0.22	-0.21	0.04
Elongation	-0.24	0.02	-0.14	0.90	0.72	0.70	-0.71	1.00	-0.12	0.36	0.36	0.37	-0.27	0.36	0.26	0.38	-0.30	0.33	-0.30	0.36	-0.29
Intercept 0	0.91	0.91	0.92	-0.11	0.09	0.56	0.72	-0.12	1.00	0.10	0.03	0.11	-0.26	0.09	0.05	0.10	-0.20	0.08	0.00	0.06	-0.23
Min. Red	-0.02	0.12	0.04	0.45	0.40	0.35	-0.19	0.36	0.10	1.00	0.60	0.95	-0.92	0.95	0.53	0.94	-0.91	0.92	-0.80	0.86	-0.90
Max. Red	-0.09	0.05	-0.03	0.45	0.41	0.30	-0.24	0.36	0.03	0.60	1.00	0.87	-0.67	0.78	0.65	0.83	-0.74	0.71	-0.61	0.67	-0.72
Mean Red	-0.01	0.14	0.05	0.46	0.41	0.36	-0.19	0.37	0.11	0.95	0.87	1.00	-0.89	0.92	0.55	0.97	-0.91	0.88	-0.81	0.87	-0.92
S.D. Red	-0.18	-0.29	-0.23	-0.34	-0.32	-0.40	0.00	-0.27	-0.26	-0.92	-0.67	-0.89	1.00	-0.89	-0.43	-0.86	0.92	-0.86	0.74	-0.85	0.93
Min. Green	-0.03	0.11	0.03	0.45	0.40	0.34	-0.20	0.36	0.09	0.95	0.78	0.92	-0.89	1.00	0.52	0.96	-0.95	0.89	-0.78	0.87	-0.88
Max. Green	-0.03	0.08	0.01	0.35	0.34	0.25	-0.15	0.26	0.05	0.53	0.65	0.55	-0.43	0.52	1.00	0.54	-0.44	0.41	-0.30	0.40	-0.40
Mean Green	-0.02	0.13	0.05	0.47	0.42	0.37	-0.20	0.38	0.10	0.94	0.83	0.97	-0.88	0.96	0.54	1.00	-0.96	0.88	-0.79	0.89	-0.91
S.D. Green	-0.10	-0.23	-0.16	-0.37	-0.35	-0.38	0.07	-0.30	-0.20	-0.91	-0.74	-0.91	0.92	-0.95	-0.44	-0.96	1.00	-0.88	0.75	-0.87	0.92
Min. Blue	-0.02	0.09	0.03	0.37	0.31	0.30	-0.18	0.33	0.06	0.92	0.71	0.88	-0.88	0.89	0.41	0.88	-0.88	1.00	-0.71	0.92	-0.93
Max. Blue	0.10	-0.02	0.05	-0.36	-0.33	-0.23	0.22	-0.30	0.00	-0.80	-0.61	-0.81	0.74	-0.78	-0.30	-0.79	0.75	-0.71	1.00	-0.70	0.77
Mean Blue	-0.03	0.10	0.02	0.41	0.35	0.33	-0.21	0.36	0.06	0.86	0.67	0.87	-0.85	0.87	0.40	0.89	-0.87	0.92	-0.70	1.00	-0.94
S.D. Blue	-0.15	-0.26	-0.20	-0.33	-0.32	-0.39	0.04	-0.29	-0.23	-0.90	-0.72	-0.92	0.93	-0.88	-0.40	-0.91	0.92	-0.93	0.77	-0.94	1.00

Table 4.7 Eigenvalues and the variance accounted for by n features.

Feature Number	Eigenvalue	Fraction of Variance Accounted For
1	10.800	0.514
2	4.938	0.749
3	2.688	0.877
4	0.895	0.920
5	0.401	0.939
6	0.291	0.953
7	0.259	0.965
8	0.178	0.974
9	0.127	0.980
10	0.120	0.986
11	0.088	0.990
12	0.080	0.994
13	0.043	0.996
14	0.036	0.997
15	0.017	0.998
16	0.015	0.999
17	0.011	0.999
18	0.010	1.000
19	0.002	1.000
20	0.000	1.000
21	0.000	1.000

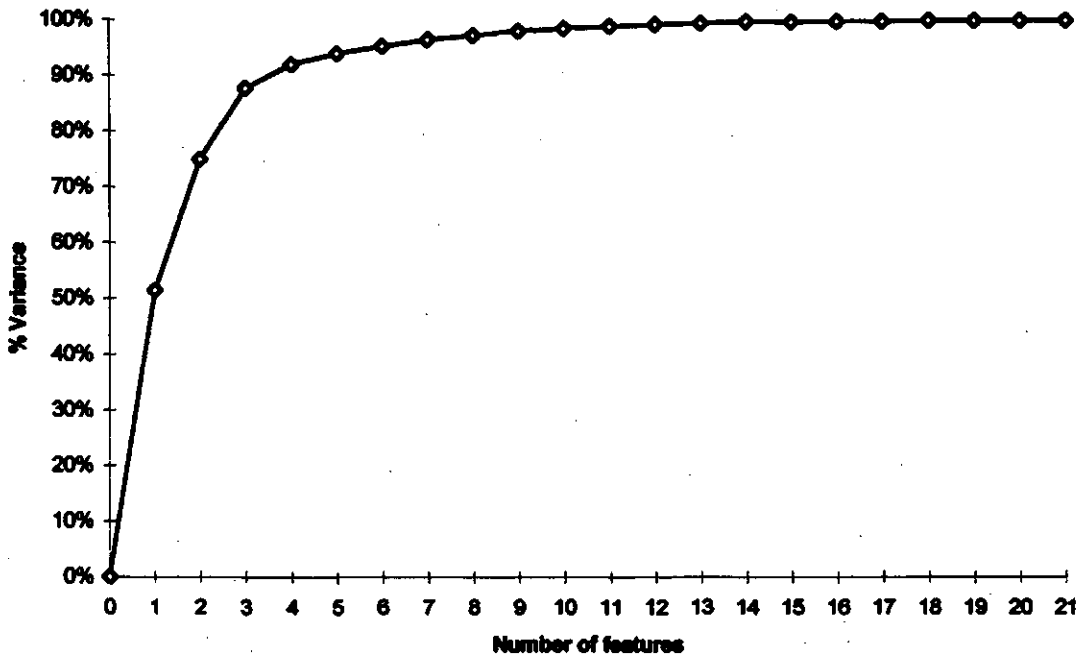


Figure 4.6 Plot of the eigensystem analysis showing the variance expected for n features.

4.2.4 Neural Network Training and Testing

Once the training and testing sets were complete, the next step in the creation of the recognition system was to train a neural network. Two network architectures were tested. The first is a back-propagation network with a single neural output. This structure was also tested with three outputs, one for each class. The second network structure tested was an MSNN network. The networks were trained using different numbers of layers and neurons, to determine the optimal size of the network. Also, the back-propagation network with the single output was trained with both the full set of features and the set of ten features chosen by feature ranking. The other networks were only tested with the full set of 21 features.

A SparcStation 20 workstation was used to test the single output back-propagation network. The other two networks were trained on a Pentium 133 MHz personal computer. The networks were constructed using the Matlab Neural Network Toolbox.

4.2.4.1 Back Propagation Network With a Single Output Neuron

This network was implemented on a SparcStation 20 workstation. The size of the network was varied from 10 neurons per layer to 35 neurons per layer, in 5 neuron increments. Networks with a single hidden layer, and with two hidden layers were tested.

The networks were trained until a summed squared error (SSE) of 0.1 was reached, or until 100,000 epochs had passed. No network reached the error of 0.1; training time on the SparcStation ranged from 2 hours to 48 hours. These networks were tested with both the full 21 features and the top 10 features.

Figure 4.7 shows the average accuracies of the trained networks (all 21 features) tested with the test data set. Table 4.8 tabulates these overall accuracies along with the accuracy obtained for each class. The network which obtained the best overall accuracy was the 35-15-1 (first layer - second layer - output layer) with an accuracy of 95.6%. This network correctly identified the good lentils with a 94.2% accuracy, the discoloured lentils with a 97.1% accuracy, and the broken and peeled lentils with a 95.4% accuracy. Several networks had the lowest possible accuracy of 33.3%, corresponding to all inputs giving the same output value, thus classifying all lentils as the same class.

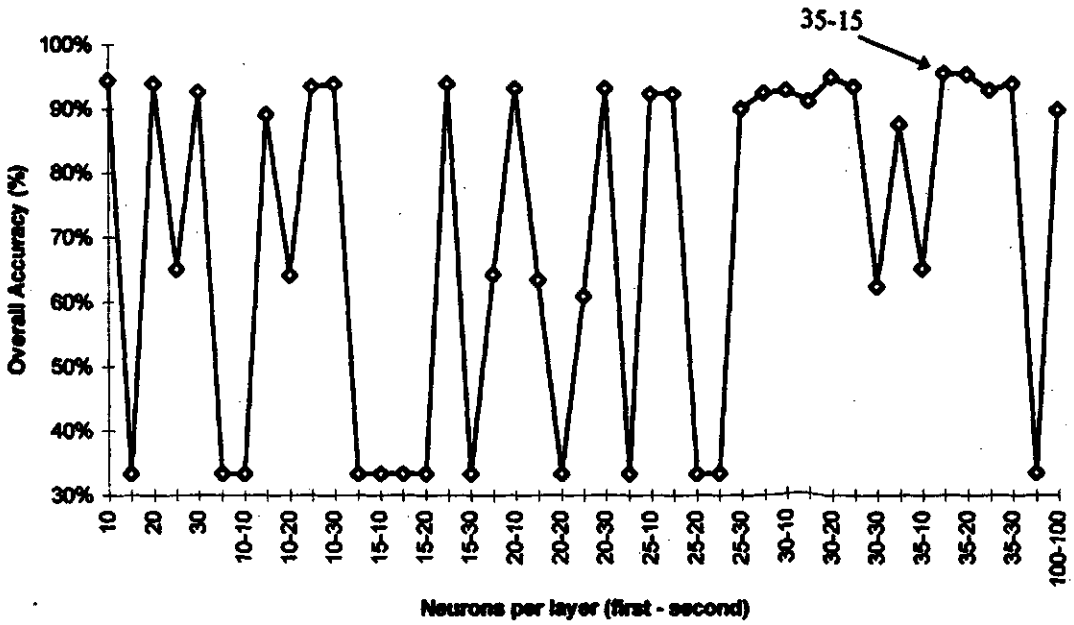


Figure 4.7 Overall accuracies of the back-propagation networks (21 features) with a single output.

Table 4.8 The accuracies obtained from the single output neural networks (21 features).

	Neurons per Layer (first - second)	Accuracy for a Single Class			Overall Accuracy
		Good	Discolored	Broken and Peeled	
1 hidden layer	10	0.938	0.950	0.946	0.944
	15	0.000	1.000	0.000	0.333
	20	0.954	0.975	0.888	0.939
	25	0.000	1.000	0.954	0.651
	30	0.954	0.954	0.871	0.926
	35	1.000	0.000	0.000	0.333
2 hidden layers	10-10	0.000	1.000	0.000	0.333
	10-15	0.946	0.967	0.763	0.892
	10-20	0.000	0.996	0.929	0.642
	10-25	0.946	0.963	0.900	0.936
	10-30	0.929	0.950	0.942	0.940
	10-35	0.000	1.000	0.000	0.333
	15-10	0.000	1.000	0.000	0.333
	15-15	0.000	0.000	1.000	0.333
	15-20	0.000	1.000	0.000	0.333
	15-25	0.942	0.958	0.921	0.940
	15-30	0.000	1.000	0.000	0.333
	15-35	1.000	0.000	0.929	0.643
	20-10	0.954	0.938	0.904	0.932
	20-15	0.000	0.996	0.908	0.635
	20-20	0.000	1.000	0.000	0.333
	20-25	0.000	1.000	0.829	0.610
	20-30	0.963	0.933	0.900	0.932
	20-35	0.000	0.000	1.000	0.333
	25-10	0.950	0.967	0.854	0.924
	25-15	0.950	0.967	0.854	0.924
	25-20	0.000	1.000	0.000	0.333
	25-25	0.000	1.000	0.000	0.333
	25-30	0.913	0.929	0.858	0.900
	25-35	0.938	0.967	0.871	0.925
	30-10	0.963	0.900	0.921	0.928
	30-15	0.946	0.933	0.850	0.910
	30-20	0.963	0.979	0.908	0.950
	30-25	0.963	0.950	0.892	0.935
	30-30	1.000	0.000	0.871	0.624
	30-35	0.971	0.963	0.696	0.876
	35-10	0.000	0.992	0.963	0.651
	35-15	0.942	0.971	0.954	0.956
	35-20	0.963	0.979	0.921	0.954
	35-25	0.938	0.958	0.892	0.929
	35-30	0.942	0.958	0.917	0.939
	35-35	0.000	1.000	0.004	0.335
	100-100	0.908	0.958	0.829	0.899

The output of the 35-15-1 network is shown in Fig. 4.8. This graph shows the neural output of the network for the 720 inputs of the test set. Good lentils were defined as a neural output of 0.8, discoloured were 0.0, and broken and peeled were -0.8. Two secondary discriminating lines were drawn at 0.35 and -0.2 to be able to interpret the outputs, which is a continuous range between -1 and +1. Outputs above 0.35 were considered to be classified as good, below -0.2 were classed as broken and peeled, and those in between were discoloured. With this network, only 14 good lentils, 7 discoloured, and 11 broken and peeled were mis-classified out of a total of 240 of each class.

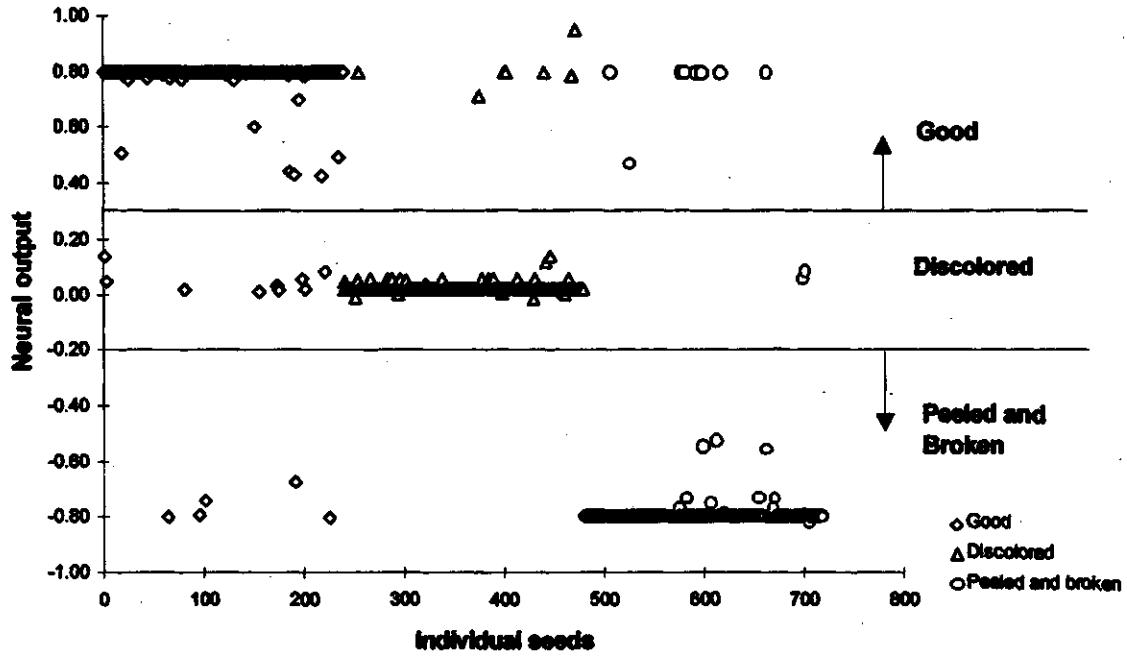


Figure 4.8 Test output for the 35-15-1 neural network (21 features).

Figure 4.9 shows the SSE of the 35-15-1 network. This is the difference between the actual network output and the desired output, summed and squared over all input vectors. Figure 4.9 shows a quick decrease in the error, from a value of around 160 with random weights, to a value of 10.67 within 20,000 iterations. This error decreases slowly, until around 98,500 iterations, when another sharp drop in the error takes the SSE to 5.005 by the one hundred thousandth iteration. Figure 4.10 shows the final 10,000 iterations so that this final decrease can be seen clearly.

This network was implemented using an adaptive learning rate. The initial rate was set at 0.001. The learning rate for the full run of the network is given in Figure 4.11. The value does not increase above 0.001.

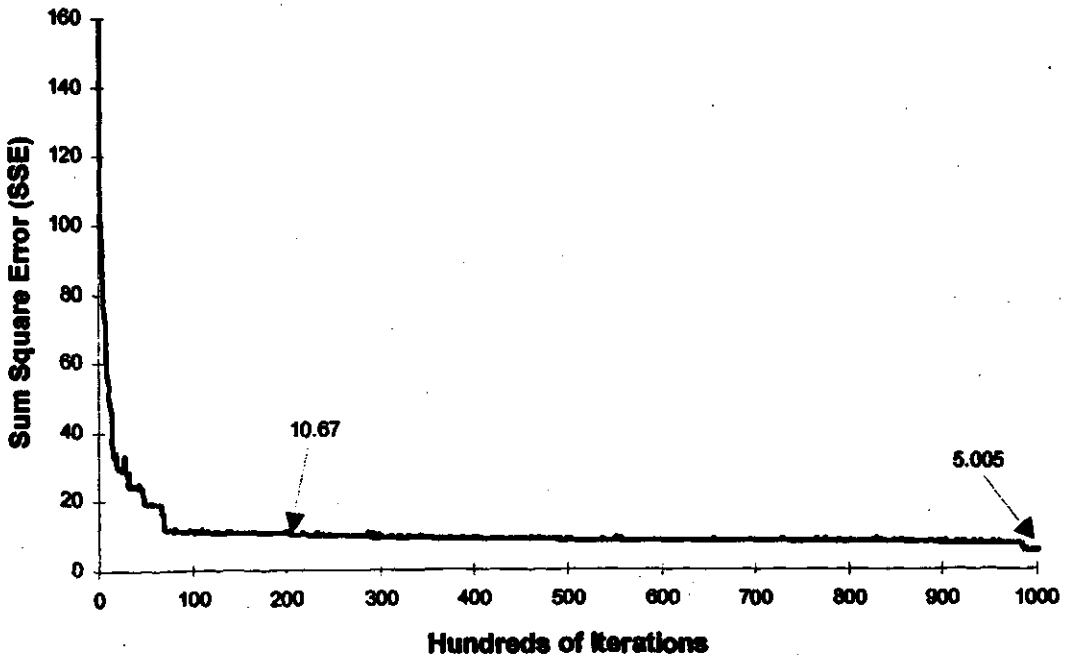


Figure 4.9 Sum Squared Error (SSE) plot for the 35-15-1 neural network (21 features).

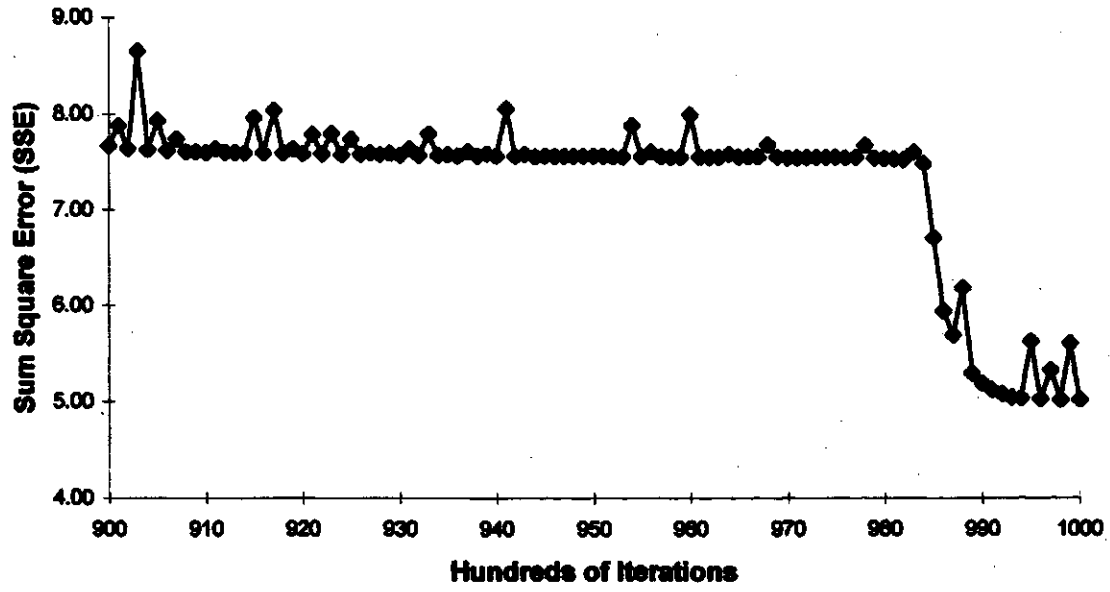


Figure 4.10 SSE for the final 10,000 iterations of the 35-15-1 neural network (21 features).

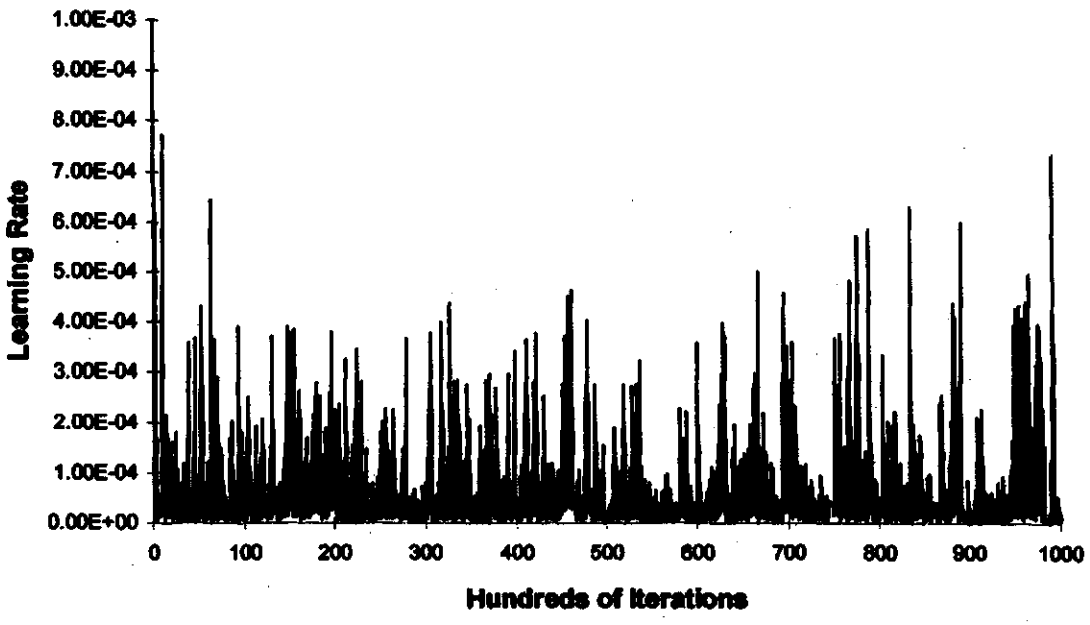


Figure 4.11 Learning rate for the 35-15-1 neural network (21 features).

The single output networks were also trained and tested with the data sets compiled with the top ten features obtained by the ACC method. Figure 4.12 shows the overall accuracies of the networks, ranging in size from 10 neurons to 35 neurons per hidden layer, with 5 neuron steps. Table 4.9 shows these overall accuracies along with the per-class accuracy for each network size.

The best overall accuracy obtained was 93.2% from the 30-30-1 network. The accuracies by class for this network were 92.5% for good, 97.9% for discoloured, and 89.2% for broken and peeled. Figure 4.13 gives the neural outputs for the test set.

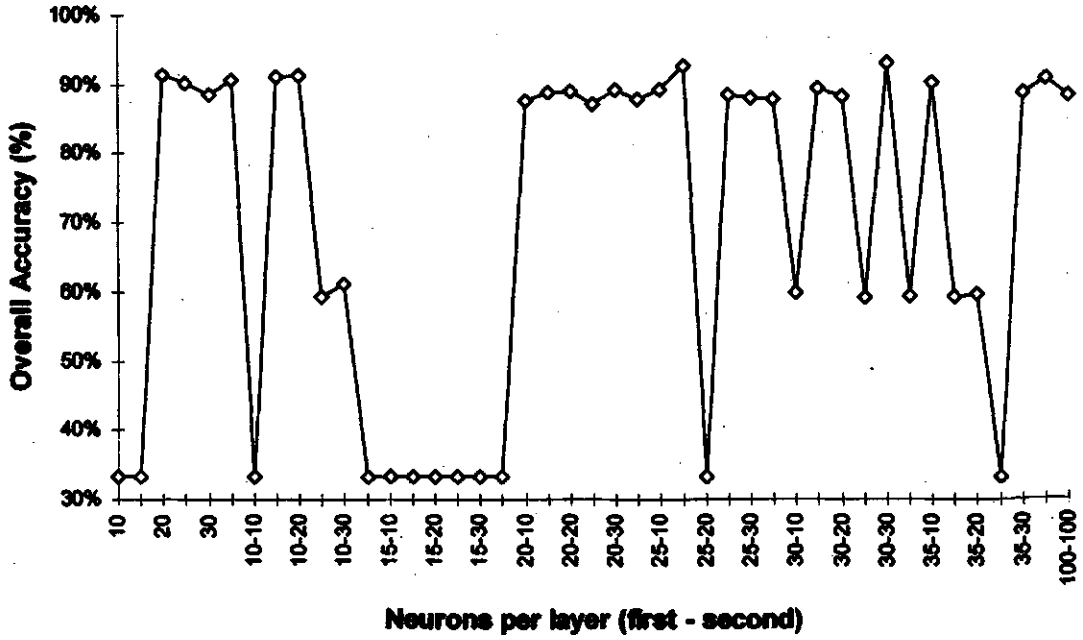


Figure 4.12 Overall accuracies of the back-propagation networks with a single output (10 features).

Table 4.9 The accuracies obtained from the single output neural networks (10 features).

	Neurons per Layer (first - second)	Accuracy for a Single Class			Overall Accuracy
		Good	Discolored	Broken and Peeled	
1 hidden layer	10	0.000	1.000	0.000	0.333
	15	0.000	1.000	0.000	0.333
	20	0.938	0.979	0.825	0.914
	25	0.883	0.967	0.854	0.901
	30	0.904	0.975	0.779	0.886
	35	0.950	0.975	0.792	0.906
2 hidden layers	10-10	0.000	0.000	1.000	0.333
	10-15	0.950	0.946	0.838	0.911
	10-20	0.954	0.946	0.838	0.913
	10-25	1.000	0.000	0.775	0.592
	10-30	0.000	1.000	0.833	0.611
	10-35	0.000	1.000	0.000	0.333
	15-10	0.000	1.000	0.000	0.333
	15-15	0.000	0.000	1.000	0.333
	15-20	0.000	1.000	0.000	0.333
	15-25	0.000	0.000	1.000	0.333
	15-30	0.000	1.000	0.000	0.333
	15-35	0.000	1.000	0.000	0.333
	20-10	0.888	0.992	0.750	0.876
	20-15	0.942	0.971	0.750	0.888
	20-20	0.942	0.979	0.750	0.890
	20-25	0.888	0.950	0.775	0.871
	20-30	0.946	0.979	0.750	0.892
	20-35	0.917	0.979	0.742	0.879
	25-10	0.925	0.938	0.813	0.892
	25-15	0.921	0.971	0.888	0.926
	25-20	0.000	1.000	0.000	0.333
	25-25	0.954	0.938	0.763	0.885
	25-30	0.963	0.933	0.746	0.881
	25-35	0.892	0.942	0.804	0.879
	30-10	1.000	0.000	0.800	0.600
	30-15	0.917	0.971	0.796	0.894
	30-20	0.963	0.942	0.746	0.883
	30-25	1.000	0.000	0.779	0.593
	30-30	0.925	0.979	0.892	0.932
	30-35	1.000	0.000	0.783	0.594
	35-10	0.879	0.975	0.858	0.904
	35-15	1.000	0.000	0.779	0.593
	35-20	0.996	0.000	0.796	0.597
	35-25	0.000	1.000	0.000	0.333
	35-30	0.863	0.954	0.846	0.888
	35-35	0.942	0.950	0.833	0.908
	100-100	0.929	0.954	0.767	0.883

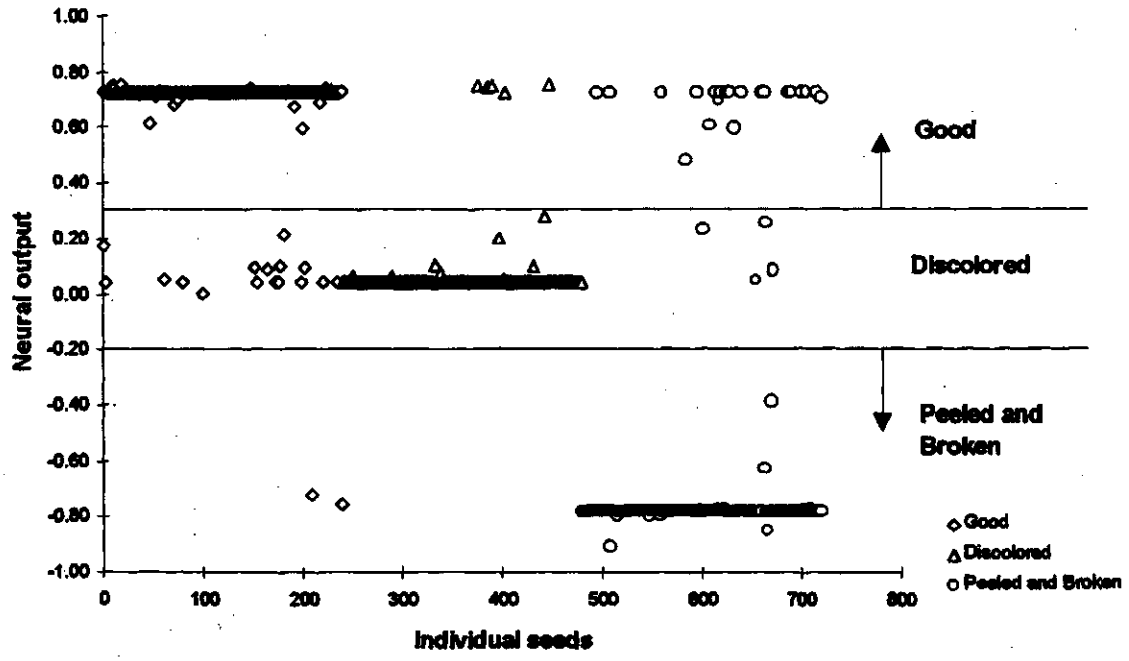


Figure 4.13 Test output for the 30-30-1 neural network (10 features).

4.2.4.2 Back Propagation Network with Three Output Neurons

In the single neuron output networks, the three classes were assigned an arbitrary value on the single output sigmoid function. With three output neurons, one for each class, the assigned outputs are binary. The set {+0.8 -0.8 -0.8} was assigned to the good class. In other words, the first output neuron should give a value of +0.8, while the second and third neurons should give -0.8. Similarly, the discoloured class was assigned {-0.8 +0.8 -0.8} and broken and peeled was {-0.8 -0.8 +0.8}.

The networks were trained using Matlab on a Pentium 133 PC with 32 MB RAM. Network sizes were from 5 neurons per layer to 30 neurons per layer, in 10 neuron steps.

Training time ranged from 1 hour to 24 hours. Only the full training set of 21 features was used. As before, the network trained for 100,000 epochs, or until an SSE of 0.1 was reached. As with the single output networks, the SSE of 0.1 was never reached, and all networks trained for 100,000 epochs. The overall accuracies are given in Fig. 4.14, and the per-class accuracies are given in Table 4.10. The best networks were the 10-3, the 30-3, and the 10-30-3 networks with an overall accuracy of 94%.

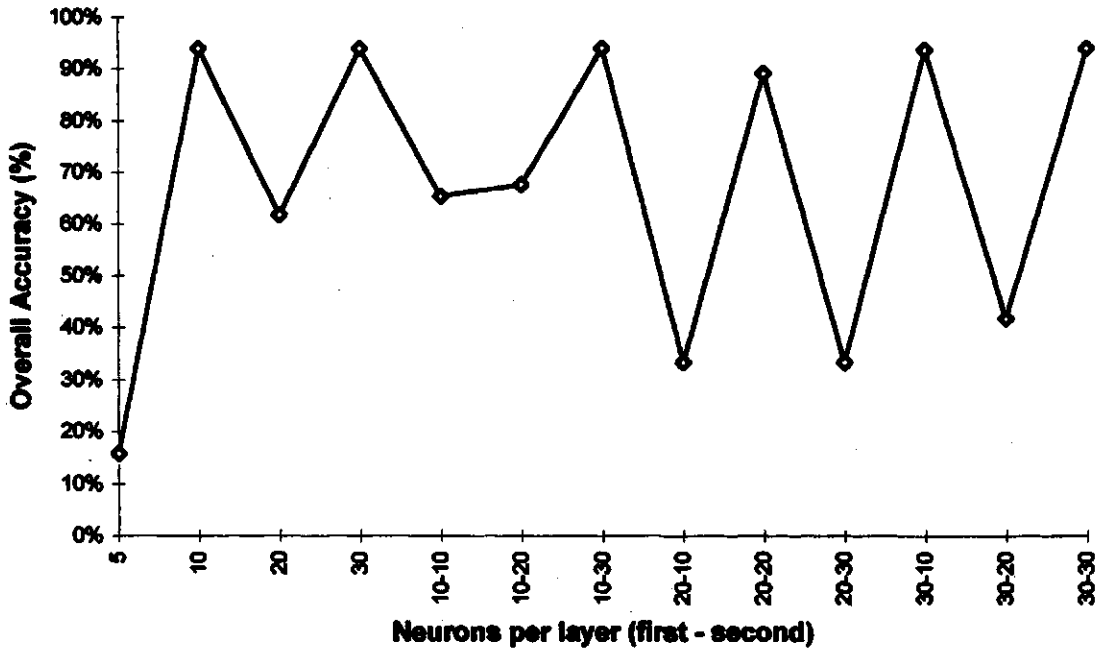


Figure 4.14 Overall accuracies of the three-output networks.

Table 4.10 The accuracies obtained from the three-output networks.

	Neurons per Layer (first - second)	Accuracy for a Single Class			Overall Accuracy
		Good	Discolored	Broken and Peeled	
1 hidden layer	5	0.475	0.000	0.000	0.158
	10	0.954	0.971	0.896	0.940
	20	0.004	1.000	0.846	0.617
	30	0.963	0.971	0.888	0.940
2 hidden layers	10-10	0.008	0.983	0.971	0.654
	10-20	0.075	0.988	0.967	0.676
	10-30	0.958	0.954	0.908	0.940
	20-10	0.000	0.000	1.000	0.333
	20-20	0.946	0.967	0.763	0.892
	20-30	0.000	1.000	0.000	0.333
	30-10	0.938	0.958	0.908	0.935
	30-20	0.375	0.871	0.008	0.418
	30-30	0.938	0.988	0.888	0.938

4.2.4.3 Multi-Structure Neural Network

The MSNN trials were completed on the SparcStation 20. As was shown in Chapter 3, the MSNN has one discriminator for each class, and each discriminator is trained to either accept or reject a vector. The discriminators were trained with bias so that there were more training vectors for the desired class than the reject class. The training files had 240 vectors for the bias class, and 120 vectors from the other two classes for a total of 480 vectors. The discriminators were trained to give a +0.8 if the test vector is from the class the discriminator was trained to recognize, and -0.8 if it is from a different class. The discriminator with the largest output shows which class the input vector belongs to.

Each discriminator was trained to an SSE of 0.1 or a maximum of 100,000 epochs.

Neither the good discriminator nor the discoloured discriminator reached an SSE of 0.1, however several of the broken and peeled discriminators did reach 0.1 after 20,000 to 30,000 epochs. The discriminators were trained sequentially, and took 10 hours to 24 hours each.

Figure 4.15 is a graph of the overall accuracies of the MSNN trials, and Table 4.11 has the accuracy obtained for each class. The size of the networks ranged from 5 neurons to 30 neurons per layer. The best network was the network with a single hidden layer with 30 neurons with an overall accuracy of 94.7%.

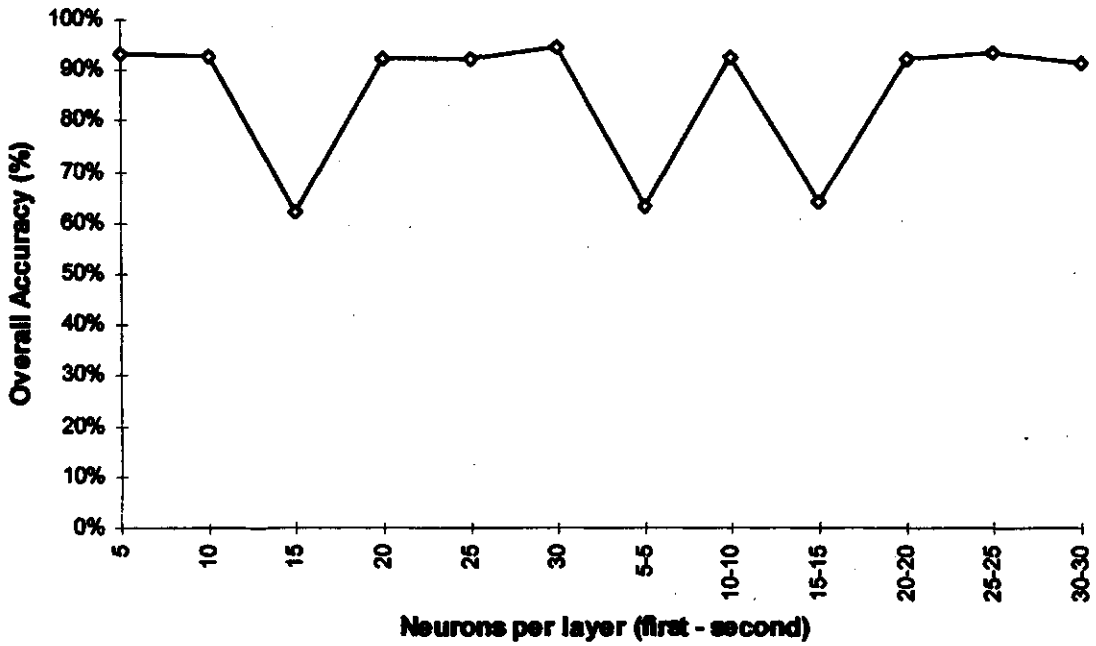


Figure 4.15 Overall accuracies of the multi-structure neural networks.

Table 4.11 The accuracies obtained from the multi-structure neural networks.

	Neurons per Layer (first - second)	Accuracy for a Single Class			Overall Accuracy
		Good	Discolored	Broken and Peeled	
1 hidden layer	5	0.967	0.954	0.875	0.932
	10	0.921	0.958	0.908	0.929
	15	0.992	0.000	0.883	0.625
	20	0.900	0.983	0.896	0.926
	25	0.950	0.958	0.867	0.925
	30	0.925	0.967	0.950	0.947
2 hidden layers	5-5	0.938	0.958	0.004	0.633
	10-10	0.967	0.933	0.883	0.928
	15-15	0.950	0.975	0.000	0.642
	20-20	0.908	0.946	0.921	0.925
	25-25	0.958	0.921	0.929	0.936
	30-30	0.908	0.971	0.871	0.917

4.3 Area/Mass Correlation

The projected (plan) area of good lentils was measured by the vision system. The mass of the lentils was then measured on a Ohaus GA200D scale. Kernels were measured in batches from 1 seed to 10 seeds, in increments of 1, and from 10 to 40 in increments of 5 seeds. Different lentil kernels were used for each trial. A linear regression was performed on this data using Excel, assuming that the fitted line goes through the origin. The result of the regression is given in Equation 4.1.

$$M = 0.002130 \cdot A \tag{4.2}$$

where M is the total mass of good lentils (grams), and

A is total plan area (mm^2).

The area, and observed mass of the kernels are tabulated in Table 4.12, along with the calculated mass based on equation 4.2. Figure 4.16 shows this data in graphical form.

Table 4.12 Lentil projected area/mass correlation.

Trial #	# of kernels	Area (mm^2)	Observed Mass (g)	Calculated Mass (g)
0	0	0	0	0
1	1	41.8	0.098	0.089
2	2	73.4	0.140	0.156
3	3	112.9	0.240	0.240
4	4	139.2	0.293	0.296
5	5	187.4	0.410	0.399
6	6	217.3	0.459	0.463
7	7	250.9	0.525	0.534
8	8	302.2	0.671	0.644
9	9	323.4	0.677	0.689
10	10	365.8	0.805	0.779
11	15	565.4	1.204	1.204
12	20	737.6	1.575	1.571
13	25	871.1	1.811	1.855
14	30	1098.0	2.312	2.339
15	35	1257.7	2.698	2.679
16	40	1433.6	3.074	3.053

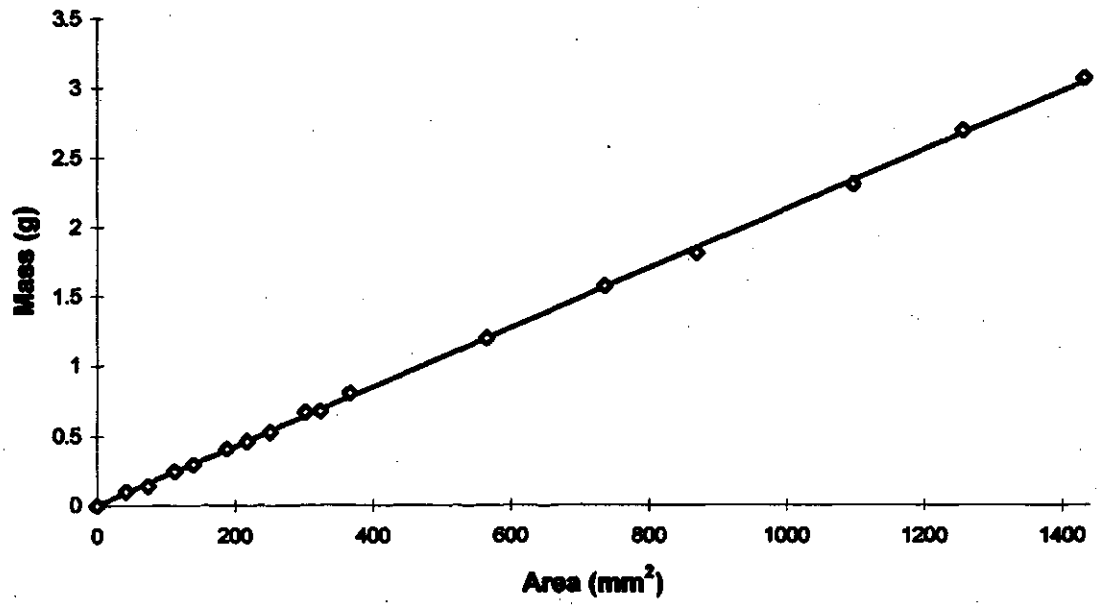


Figure 4.16 Projected area of lentil kernels versus the mass of the kernels.

4.4 Discussion of Results

The first step in this work was to rank the measured features of the lentils according to their suitability to creating a decision rule. Table 4.6 summarises the results of this process, giving the relative ranking of all measured features. From this table, we see that three of the top six features are colour features, including the best feature, Mean of Green Pixels. This seems to make sense since colour is the easiest feature of the three classes to recognise by eye. Discoloured lentils are darker than others, Broken and Peeled lentils are yellower, and Good lentils are greener. Three morphological features also show up in the top six. This also makes sense, due to the ACC method, which ranks features according to their correlation with other features. Shape features will have very little correlation with colour features.

The best neural network result was the 35-15-1 network, with an accuracy of 95.6%. This means that this network only misclassified 38 lentils out of 720. Figure 4.7 shows the recognition accuracies of all sizes tried for the single output network. There is no discernible pattern in this graph to correlate the number of neurons to recognition accuracy. A network which had a large number of neurons (100 per layer) was also tried to see the effect on accuracy. This network only gave an accuracy of 89.9%, thus suggesting that large sizes are not necessary for high recognition rates. Figure 4.9 is the SSE of the 35-15-1 network. This graph shows that the SSE dropped sharply from approximately 8.0 to approximately 5.0 at the very end of the 100,000 iterations. The SSE had been fairly constant for about 70,000 iterations until then, which demonstrates

the difficulty in choosing the maximum number of training iterations. A network should be trained as long as possible, even if the SSE appears to be unchanging.

The single output networks were also trained with a reduced number of features obtained by the ACC method. Figure 4.12 is the accuracies of these networks. The best network was the 30-30-1 network with an accuracy of 92.5%. This was slightly lower than the networks trained with the full feature set, however the lower accuracy is offset by quicker training time.

The feed-forward network with three output neurons was tested next. Figure 4.14 is the recognition accuracies for these networks. The best networks were the 10-3, 30-3, and 10-30-3 networks with an overall accuracy of 94%. Again, this was only slightly lower than the 35-15-1 network, suggesting no significant difference in the performance of these two types of structures.

The final type of network tried was the MSNN. This network is basically three feed-forward networks operating in parallel, each trained to recognise a single class. Figure 4.14 gives the accuracies for all sizes of network tried. The best result was 94.7% achieved by the 30-1 network. This is almost the same accuracy as the 35-15-1 feed-forward network, however the MSNN took nearly three times as long to train due to the three separate discriminators. The MSNN structures were the only networks which reached the minimum SSE of 0.1, although only with the Broken and Peeled discriminators. This suggests that the networks can identify a system of two classes easier

than all three classes together. If the discriminators for the Good, and Discoloured classes were optimised by trying different numbers of neurons, these classes might also be made to converge to a minimum SSE. This, however, would require extensive testing to customise each discriminator, and the extra accuracy gained might not be worth the extra time.

The correlation of lentil mass to projected area is presented in Fig. 4.16. This graph shows the fitted line relating mass to area, and is given by Equation 4.2. The area of lentils on the graph is the sum of the projected area of a group of lentils. Data was taken on the area of up to 40 lentils. The mass was measured collectively for each group of lentils. The equation was based on one moisture content of the lentils, and is not likely to hold for other moisture contents, as both size and mass is likely to change, not necessarily in equal proportions. Further research is necessary to model the area and mass for varying moisture contents if this system is to be used in commercial applications.

4.5 Summary

This chapter has presented the methods used in these experiments to take images and extract features from lentils. It was shown how the data from these features were grouped into separate text files for the purposes of training and testing a neural network. The accuracies obtained from the neural network were shown. The best accuracy was 95.6% obtained from the 35-15-1 back propagation network. Finally, a relationship was developed between the projected area of the lentils to their mass.

5. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

5.1 Summary

The objective of this project was to develop a machine vision system to discriminate three quality classes of Laird lentils: good, discoloured, and broken and peeled. The system hardware consisted of a video camera lighting system, digitizing board, and computer host. The software consisted of a display and analysis program, and a pattern recognition program. The camera was a Sony DXC-151A. The digitizing board was a Matrox Meteor/RGB. A Pentium-133 PC with 32 MB RAM was the host computer for the system. This computer housed the digitizing board and ran the display and analysis program Matrox Inspector. The pattern recognition was also partly run on the host computer using Matlab. A Sun SparcStation 20 also ran some of the pattern recognition routines. Two different architectures of neural networks were trained and compared for their pattern recognition of the lentils.

The lentils were separated by hand into good, discoloured, and broken and peeled. A total of 240 kernels were used from each class. This set was then divided into 120 kernels for network training, and 120 kernels for testing. Using the analysis program, 21

features were measured from the lentils. These features consisted of 8 morphological features (area, perimeter, convex perimeter, breadth, length, elongation, roughness, and compactness), and 13 colour features (minimum pixel, maximum pixel, mean pixel, and standard deviation of pixels for each red, green, and blue component, and intercept 0). All of these features were position invariant to eliminate the orientation of the kernels with respect to the camera as a factor in their measurements.

Neural networks were used for the pattern recognition of the three lentil classes. The network performed a mapping operation of the inputs (features) onto a non-linear output function. This was accomplished by adjusting weights within each neuron through a back-propagation process. This process attempts to minimize the error obtained by comparing the computed output of the network to a desired output. The networks in this project were implemented with Matlab's Neural Network Toolbox. An improved back-propagation algorithm was used which incorporated momentum and an adaptive learning rate to decrease training time.

Two architectures of neural networks were tested. The first network was a simple feed-forward neural network with a single output. This network was implemented on a Sun SparcStation 20. Different numbers of neurons in each layer were used. The number of neurons ranged from 10 to 35 in each layer. No network reached the desired minimum error of 0.1, therefore all took the set maximum number of iterations (100,000) to train. The network which had the best accuracy rate was the network with 35 neurons in the first layer, and 15 in the second layer (35-15-1). This network reached an accuracy of

95.6%. The feed-forward network was also used with three outputs, one for each class. This network was run on a Pentium-133 PC. The size of the network was varied from 5 neurons per layer to 25 neurons. The best accuracy was 93% achieved by the 15-25-3 network after 100,000 iterations. Again, all networks took the maximum number of iterations to train.

The second network configuration used was the multi-structure network (MSNN). This architecture was actually multiple feed-forward networks (called discriminators) operating in parallel, with one discriminator for each class; each discriminator was trained to classify only one class. The discriminator which gave the largest output defined the class for a given input. The size of the network was varied from 5 neurons to 30 neurons per layer. The best accuracy achieved was 94% by the 15-20-1 networks. The network was trained on a SparcStation 20, and each discriminator was trained sequentially. The discriminators for the good class, and the discoloured class each reached 100,000 epochs before reaching the minimum error. However, the broken and peeled discriminator reached an error of 0.1 after approximately 20,000 epochs.

The features were then analyzed for their contribution to the recognition process. The features were ranked according to probability of error (POE), and by the accumulated correlation coefficient (ACC) method. Using eigensystem analysis to determine how much variance was accounted for by each feature, it was determined that only the top 10 features were needed for an maximum accuracy of 99%. Thus the data sets were re-made to include only the following features: area, mean green pixel, length, breadth, mean red

pixel, standard deviation in green pixel, standard deviation in red pixel. The feed-forward neural network with a single output was then used to test these features. Again, the size of the networks was varied from 10 neurons to 35 neurons per layer. The 20-25-1 network achieved the best result with a 92% accuracy.

Since results from grain grading are usually reported as a percentage by mass, the area of the lentil kernels was correlated with their weight. The area of the lentils was found in batches, ranging in size from 1 to 10 in increments of 1 lentil kernel, and 15 to 45 in increments of 5 kernels. These batches were then weighed. The result was a linear correlation of mass vs. area.

5.2 Conclusions

This work tested the process of machine vision for use in the discrimination of lentil seeds. The high recognition accuracy obtained by the neural networks for the three classes of lentils shows that machine vision can be successfully used in the quality analysis of lentils. The 95.6% accuracy achieved by the feed-forward neural network is high enough to be of use commercially.

The best network in terms of accuracy was the 35-15-1 feed-forward network using 21 features. However, smaller networks, and other architectures also had accuracy almost as high. The smaller networks can save many hours in training time. This may be useful in some situations, however it is likely that a vision system would only have to be

trained once off-line. The extra accuracy with the larger networks would make it worthwhile to take a longer training time.

The best architecture for this recognition problem appeared to be the feed-forward network with a single output neuron. This network obtained the highest accuracy over the other networks. Plots of the size of the network versus the accuracy of the network did not show any trend. It is likely that the size of the network must be determined by trial and error for every different application. This could include changing the type of classes or number of classes the system is set up to discriminate.

The MSNN structures functioned well in terms of accuracy, however they also took longest to train. This is due to the sequential nature of the computer software running the network algorithm. Although the discriminators are designed to operate in parallel, they must each be trained and tested sequentially. This, in effect, triples the time needed to train an MSNN with three classes in comparison to a simple feed-forward network. The fact that each discriminator only has two classes to train to does decrease the learning time a little, but not enough to offset the sequential problem. If an MSNN was implemented in hardware with a true parallel structure, this network might be preferable to the feed-forward network. The other problem with the MSNN is that each discriminator operates completely separately from the others. Each discriminator has different classes to train on, which would make it necessary to find the best size of each discriminator separately. This would make the MSNN difficult to design for recognition problems with many classes.

The correlation between mass and area of the lentil kernels is an important result from the use of a vision system. Current grade reporting methods use a percentage-by-weight method for results. Without a mass/area equation, the kernels would have to be weighed manually. By weighing the sample before placing under the camera, the correlation will allow a vision system to simply add the area of the lentil kernels, and use the correlation equation to report the percentage of foreign particles and the percentage of good lentils as a fraction of the total sample by weight.

Although the three classes used in the work might have been somewhat arbitrary, the results showed that the machine vision system could accurately separate a random lentil sample into a number of classes. The methods presented in this thesis can now be applied to the more commercially useful classes of lentils, such as size, colour, and diseased kernels.

5.3 Recommendations

The following recommendation are made for the continuing development of a machine vision system for the purposes of lentil discrimination:

1) Consultations with industry should be made to determine which classes of lentils are needed. These classes may be standard classes identified in the Canadian Grain

Commission grading handbook, or they may be specialized classes which are driven by industry demand.

2) Initial sample set construction should be done by an experienced grain grader. Since the training sets in this work were made by a researcher who is inexperienced in handling grain, the inconsistencies in the training and testing sets will lead to decreased accuracy in the system.

3) A method of colour calibration should be developed. No colour calibration was used in this work. This does not affect the pattern recognition since all values are compared relative to each other. However, if the colour values are to be reported directly, they must be calibrated against a known value. Also, if the neural weights developed on one system are to be transferred to a new system, the new system must be calibrated against the original system or the weights will not be correct.

4) A method of automatic distance calibration must be incorporated into the system. The present method called for imaging a ruler, manually determining how many pixels are in a given distance, and then entering this value into the calibration window. To simplify this process, and to eliminate potential human errors, this system should be automated. A plate with holes of known diameter can be made. The vision system could then measure these holes and automatically analyse them for size and set the calibration. This method would allow for calibration in only a few steps, which reduces the chance for error.

5) The program routines used in this work should be incorporated into a single, user friendly package running on one computer. The digitiser controller, blob analysis, and neural network routines can be written into one program. The program should also include the ability to easily change the neural weights, which can be developed on a faster, dedicated computer.

6) A more appropriate lighting system should be found. The current system, composed of four single-point incandescent bulbs, illuminates the seeds in such a way that significant shadows are formed. These shadows are included in the blob after segmentation and can lead to inaccuracies. Power fluctuations can also lead to varying intensities in the light, and therefore to changes in the measured colour values. One possible solution is the incorporation of round fluorescent tubes which fit around the lens of the camera. While these tubes will not eliminate shadows, they will reduce them significantly. Luo et al. (1996) did an in-depth study on various lighting sources for use in machine vision.

7) An automated seed presentation system must be designed. Figure 6.1 shows a possible configuration for this.

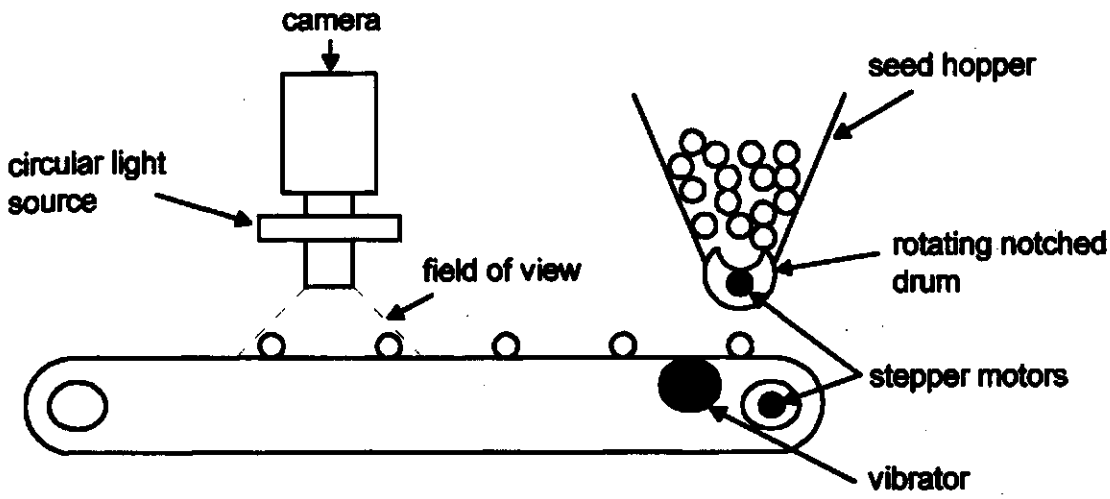


Figure 5.1 Automated imaging system

- The system should be completely enclosed to prevent external light sources from interfering with the image.

- A seed hopper and a rotating notched drum could drop one seed at a time onto a conveyor belt. The drum may have more than one notch to speed up the imaging. The belt should be made of a white material to make segmentation possible.

- The rotation of the drum and the belt could be controlled by the host computer through stepper motors. The number of steps could be counted so that one field-of-view can be moved under the camera at a time. The computer can then stop the belt and trigger the digitiser to grab an image.

- A small vibrator could be added under the belt to decrease the instances of touching seeds.

- To speed up the process, a line scan camera could be used. These cameras grab only one line of an image at a time and can therefore digitize continuous images (as opposed to frames). This would eliminate the need to stop the belt for each frame. A faster method of image acquisition and analysis would be needed. Necessary components for this are currently commercially available in the form of high-speed VXI bus components.

8) An investigation should be done into the true density distribution of the lentil features. As was shown in Section 4.2.3, the features do not follow a normal distribution. The actual distribution would be useful for refining the feature selection, and for using probability classifiers for pattern recognition.

REFERENCES

Brogan, W. L., and A. R. Edision, "Automatic Classification of Grains Via Pattern Recognition Techniques." Pattern Recognition, Vol. 6, 1974, pp. 97-103.

Broten, G. S., and H. C. Wood, "A Neural Network Approach to Analysing Multi-Component Mixtures." Meas. Sci. Technol., Vol. 4, 1993, pp. 1096-1105.

Canadian Grain Commission, *Grain Grading Handbook for Western Canada*, Canadian Grain Commission, 643-303 Main Street, Winnipeg, Manitoba, R3C 3G8, 1989.

Canadian Grain Commission, *Lentil Grade Determinant Chart, Series ISD95-6*, Canadian Grain Commission, 643-303 Main Street, Winnipeg, Manitoba, R3C 3G8, 1995a.

Canadian Grain Commission, *Laird - Good Natural Colour, Series ISD95-6*, Canadian Grain Commission, 643-303 Main Street, Winnipeg, Manitoba, R3C 3G8, 1995b.

Canadian Grain Commission, *Laird - Fair Colour Guide, Series ISD95-6*, Canadian Grain Commission, 643-303 Main Street, Winnipeg, Manitoba, R3C 3G8, 1995c.

Canadian Grain Commission, *Grain Grading Determinants*, Canadian Grain Commission, 643-303 Main Street, Winnipeg, Manitoba, R3C 3G8, 1996.

Demuth, H., and M. Beale, *Neural Network Toolbox for Use with Matlab*, The Mathworks, Inc., Nattick, MA, 1995.

Egelberg, O. O. Mansson, and C. Peterson, "Assessing Cereal Grain Quality with a Fully Automated Instrument Using Artificial Neural Network Processing of Digitized Color Video Images." Proceedings of SPIE's International Symposium on Optics in Agriculture, Forestry, and Biological Processing, November 1994, SPIE Vol. 2345, pp. 146-158.

Freeman, J. A., and D. M. Skapura, *Neural Networks - Algorithms, Applications, and Programming Techniques*, Addison and Wesley Publishing Company Inc., New York, 1991.

Ghazanfari-Moghaddam, A., *Machine Vision Classification of Pistachio Nuts Using Pattern Recognition and Neural Networks*, Ph.D. Thesis in Agriculture Engineering, College of Graduate Studies and Research, University of Saskatchewan, 1996.

Ghazanfari, A., and J. Irudayaraj, "A Multi-Structural Neural Network for Classification of Pistachio Nuts." Presented at the 1994 International ASAE Winter Meeting, Paper No. 946579. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Gupta, M. M., *Special Series of Lectures on Neural Computing Systems: Introduction to Theory and Application of Neural and Neural-Fuzzy System*, Intelligent Systems Laboratory, College of Engineering, University of Saskatchewan, 1995.

Gupta, L., and A. M. Upadhye, "Non-linear Alignment of Neural Net Outputs for Partial Shape Classification." Pattern Recognition, Vol. 24, No. 10, 1991, pp. 943-948.

Hehn, J. L., *Measurement of Seed Features Using Macintosh Based Imaging*, M. Sc. Thesis in Agriculture Engineering, College of Graduate Studies and Research, University of Saskatchewan, 1991.

Hehn, J. L., and S. Sokhansanj, "Canola and Mustard Seed Identification Using Macintosh Based Imaging." Presented at the 1990 International ASAE Winter Meeting, Paper No. 903534. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Li, Y., Machine Vision Determination of Dockage and Quality of Borage Seed, M. Sc. Thesis in Agriculture and Bioresource Engineering, College of Graduate Studies and Research, University of Saskatchewan, 1996.

Li, Y., S. Sokhansanj, and H. C. Wood, "Machine Vision Based Dockage Tester." Presented at the 1995 North-Central Inter-Sectional ASAE/CSAE Meeting, Paper No. SD 95-115. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Lippmann, R. P., "An Introduction to Computing with Neural Nets." IEEE, Acoustics, Speech and Signal Processing Magazine, Vol. 4, No. 2, 1987, pp. 4-22.

Luo, X., D. S. Jayas, T. G. Crowe, and N. R. Bully, "Evaluation of Light Sources for Machine Vision." Presented at the 1996 Agricultural Institute of Canada Annual Conference, Paper No. 96-600. CSAE, Box 381, RPO University, Saskatoon, SK, Canada, S7N 4J8.

Neuman, H., H. D. Sapirstein, E. Shwedyk, and W. Bushuk, "Discrimination of Wheat Class and Variety by Digital Image Analysis of Whole Grain Samples." Journal of Cereal Science, Vol. 6, 1986, pp. 125-132.

Majumdar, S., D. S. Jayas, J. L. Hehn, and N. R. Bulley, "Classification of Various Grains Using Optical Properties." Canadian Agricultural Engineering, Vol. 38, No. 2, April/May/June 1996, pp. 139-144.

Mathworks, The, *Matlab Users Guide*, The Mathworks, Inc., Nattick, MA, 1995.

Matrox Electronic Systems Ltd, *Matrox Inspector User Guide*, Montreal, PQ, 1996.

McCulloch, W. S., and W. H. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity." Bulletin of Mathematical Biophysics, Vol. 5, 1943, pp. 115-133.

Mucciardi, A. N., and E. E. Gose, "A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties." IEEE Transactions on Computers, Vol. C-20, No. 9, September 1971, pp. 1023-1031.

Murray, T. A., and D. S. Humburg, "Evaluation of a Machine Vision System for 3-D Location of Harvestable Mushrooms." Presented at the 1996 North-Central Intersectional ASAE/CSAE Meeting, Paper No. MANSASK 96-108. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Meisel, W. S., *Computer Oriented Approaches to Pattern Recognition*, Academic Press, New York, 1972.

Park, B., Y. R. Chen, and M. Nguyen, "Multispectral Image Analysis using Neural Network Algorithm." Presented at the 1996 Annual International ASAE Meeting, Paper No. 963034. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Patil, R. T., and S. Sokhansanj, "Particle Size Characterization in Alfalfa Cubes Using Machine Vision and NIR." Presented at the 1992 International ASAE Winter Meeting, Paper No. 92-6541. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Patil, R. T., S. Sokhansanj, and D. Pulkinen, "Correlation of Protein and Carotene with Reflectance Characteristics of Alfalfa Grind." Presented at the Agriculture Institute of Canada Annual Conference, 1995, Paper No. 95-304. CSAE, Box 381, RPO University, Saskatoon, SK, Canada, S7N 4J8.

Romaniuk, M. D., *Pattern Recognition of Barley Seeds Using Fourier Descriptors and Neural Networks*, M. Sc. Thesis in Electrical Engineering, College of Graduate Studies and Research, University of Saskatchewan, 1994.

Romaniuk, M. D., and S. Sokhansanj, "Barley Seed Recognition Using a Multi-Layer Neural Network." Presented at the 1993 International ASAE Winter Meeting

Rumelhart, D. E. , G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation." Parallel Distributed Processing, Vol. 1, Chapter 8, MIT Press Cambridge, MA, 1985.

Rumelhart, D., J. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Vols. 1 and 2*, MIT Press, Cambridge, MA, 1986.

Saperstein, H. D., M. R. Neuman, E. Shwedyk, and W. Bushuk, "Wheat Grain Color Analysis by Digital Image Processing I. Methodology." Journal of Cereal Science, Vol. 10, 1989, pp. 175-182.

Segerlind, L. J., and B. Weinberg, "Grain Kernel Identification by Profile Analysis." Transactions of the American Society of Agricultural Engineering, Vol. 16. No. 2, 1973, pp. 324-327.

Shrestha, B. L., *A Neural Network Based Machine Vision System for Identification of Malting Barley Varieties*, M. Sc. Thesis in Electrical Engineering, College of Graduate Studies and Research, University of Saskatchewan, 1996.

Winter, P. W., S. Sokhansanj, and H. Wood, "Machine Vision Methods for use in Grain Variety Discrimination and Quality Analysis." Proceedings of SPIE's International Symposium on Optics in Agriculture, Forestry, and Biological Processing, November 1996, SPIE Vol. 2907.

Winter, P. W., S. Sokhansanj, H. C. Wood, and W. Crerar, "Quality Assessment and Grading of Lentils Using Machine Vision." Presented at the 1996 Agricultural Institute of Canada Annual Conference, Paper No. 96-310. CSAE, Box 381, RPO University, Saskatoon, SK, Canada, S7N 4J8.

Winter, P. W., W. Yang, S. Sokhansanj, and H. C. Wood, "Discrimination of Hard-to-Pop Popcorn Kernels by Machine Vision and Neural Network." Presented at the 1996 North-Central Intersectional ASAE/CSAE Meeting, Paper No. MANSASK 96-107. ASAE, 2950 Niles Road, St. Joseph, MI 49085-9659 USA.

Widrow, B., and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, Vol. 78, NO. 9, September 1990, pp. 1415-1441.

Wood, H. C., N. H. Brown, and J. N. Wilson, "Image Analysis for Vision-Based Agriculture Vehicle Guidance", ASAE Paper No. 901623, International Winter Meeting, Hyatt Regency Chicago, Chicago, December 18-21, 1990.

Wood, H. C., S. Sokhansanj, and B. L. Shrestha, "Optical Seed Recognition by a Neural Network." Unpublished paper, Dept. of Electrical Engineering, Dept. of Agriculture and Bioresource Engineering, University of Saskatchewan, Saskatoon, SK, S7N 5A9.

Zhan, C. T., and R. Z. Roskies, "Fourier Descriptors for Plane Closed Surfaces." IEEE Transactions on Computers, Vol. C-21, No. 3, 1972, pp. 269-281.

APPENDIX A

HARDWARE SPECIFICATIONS

A-1 Camera Specifications

Sony DXC-151A

Pickup Device

Pickup device	Interline-transfer 2/3-inch CCD
Color filter	Primary color filter
Picture elements	380,000
	768 × 493 (horizontal × vertical)
Sensing Area	Equivalent to a 2/3-inch pickup tube

Optical system an miscellaneous

Lens mount	C mount
Signal system	EIA standards, NTSC color system
Scanning system	525 lines 2:1 interface, 30 frames/sec
Synchronization	Internal/external (switched automatically)
External synchronization signal	
	VBS or BS signal
	(Sync level 0.3 Vp-p)
	(Burst level 0.3 Vp-p)
Horizontal resolution	460 TV lines
Minimum illumination	1.3 lux F1.4 (GAIN: +12 dB, SHUTTER: OFF)
Sensitivity	2000 lux, over F5.6(GAIN 0 dB)
Video output	RGB: 0.7 Vp-p, 75 ohms, sync negative
	Y: 1 Vp-p, 75 ohms
	C: C level is in accordance with VBS.
Video S/N ratio	48 dB or over (GAIN: 0 dB)
Electronic shutter	10 modes
	OFF, FL, 1/250, 1/1000, 1/2000, 1/4000, 1/10000 sec
	CCD IRIS mode
	(Standard mode, Backlight mode, Spotlight mode)
White balance	4 modes
	AUTO, 3200, 5600, ATW
Gain control	4 modes
	AGC, 0 dB, 6 dB, 12 dB
Input and output connectors	GEN LOCK IN: BNC type
	DC IN: 12-pin connector
	VIDEO OUT: BNC type
	RGB SYNC: D-sub 9-pin connector
Power requirements	12 V DC
Power consumption	7 W
Operating temperature	0°C to 40°C (32°F to 104°F)
Storage temperature	-20°C to 60°C (-4°F to 140°F)

Operating humidity	20% to 80% (no condensation allowed)
Storage humidity	20% to 90% (no condensation allowed)
Shock resistance	Less than 686 m/s^2 (70 g)
Dimensions	65 × 50 × 170 (w/h/d) (2 5/8 × 2 × 6 ¾ inches) excluding projecting parts
Mass	Approx. 520 g (1 lb. 2 oz.)

Table A1 Pin assignments for the RGB SYNC connect (D-sub 9 pin female)

Pin No.	Signal
1	GND (earth)
2	GND (earth)
3	R output
4	G output
5	B output
6	Y output, composite video output, or no signal
7	Sync output
8	GND (earth)
9	C output or no signal

A-2 Framegrabber Specifications

Matrox Meteor/RGB

Acquisition
- Software selectable video input (up to four channels)
- Programmable sharpness, hue, brightness, contrast, and saturation
- Standard color and monochrome video and RGB
  (Gain and offset adjustable on each channel)
- Pixel jitter: ± 6 ns (typical)

Data transfer
(depending on system and VGA card used:)

- up to 45 MB/sec

Note: Real-time acquisition to system or display memory requires:
- 9 MB/sec for 640 × 480 × 8 @ 30 fps RS-170
- 11 MB/sec for 768 × 576 × 8 @ 25 fps CCIR
- 35 MB/sec for 640 × 480 × 24 @ 30 fps NTSC or RGB
- 42 MB/sec for 768 × 576 × 24 @ 25 fps PAL or RGB

Interface and connectors

- PCI 32-bit interface
- Video input connector (DB9 for composite or RBG inputs)
- Separate Y/C input connector (4 pin mini din)
- Phono Jack for composite input

Environmental

- Power consumption < 7.5 Watts
- Card size is 23.55 cm × 10.55 cm (9.3" × 4.2")
- Operating temperature from 0 to 55°C (32 to 131°F)

Table A2 Pin assignments for connectors on the Matrox Meteor/RGB.

Pin No.	Signal
Video in (DB9):	
1	Composite 1 or Red
2	Composite 2 or Green
3	Composite 3/or Luma in or Blue
4	Composite 4 or Composite Sync
5	Ground
6	Ground
7	Ground
8	Ground
9	Ground
Y/C in (4 pin mini din):	
1	Ground
2	Ground
3	Y (Luma) In
4	C (Chroma) In
Phono Jack in:	
1	Composite 1
2	Ground

A3 Cable Interconnections

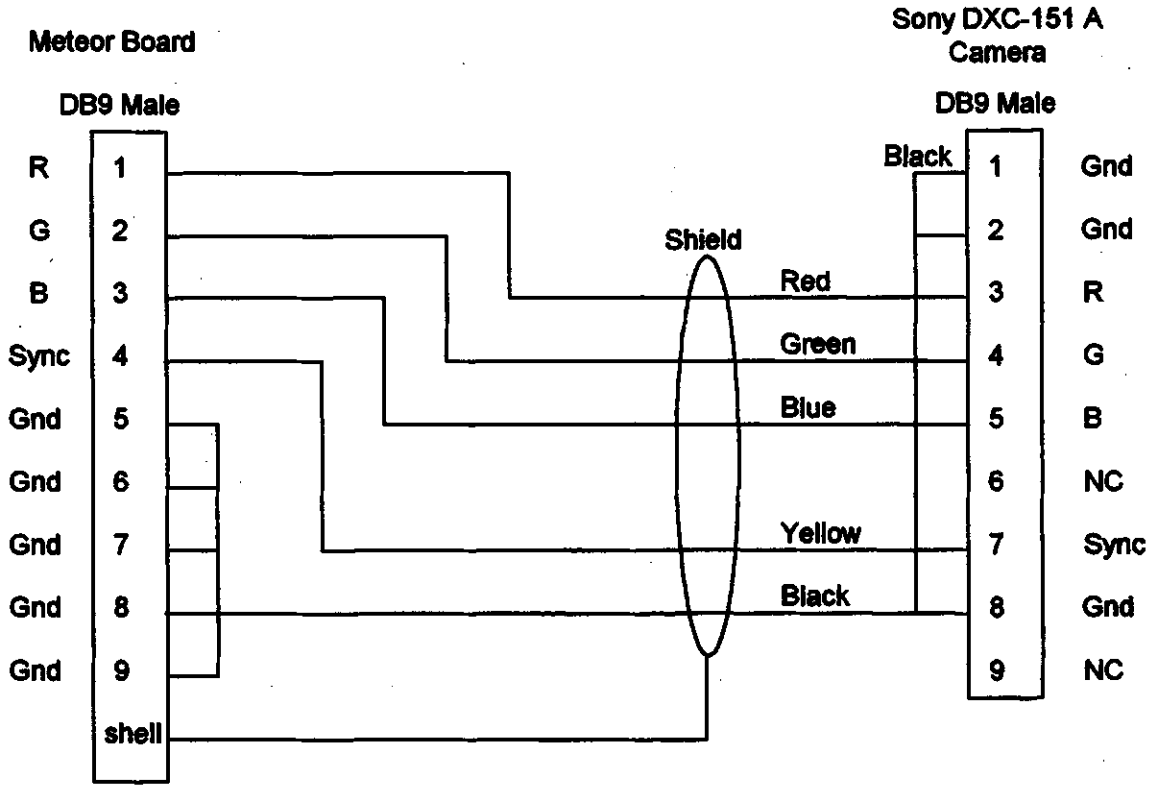


Figure A1 Cable interconnections between video camera and capture board.

APPENDIX B

FEATURE HISTOGRAMS AND DENSITY CURVES

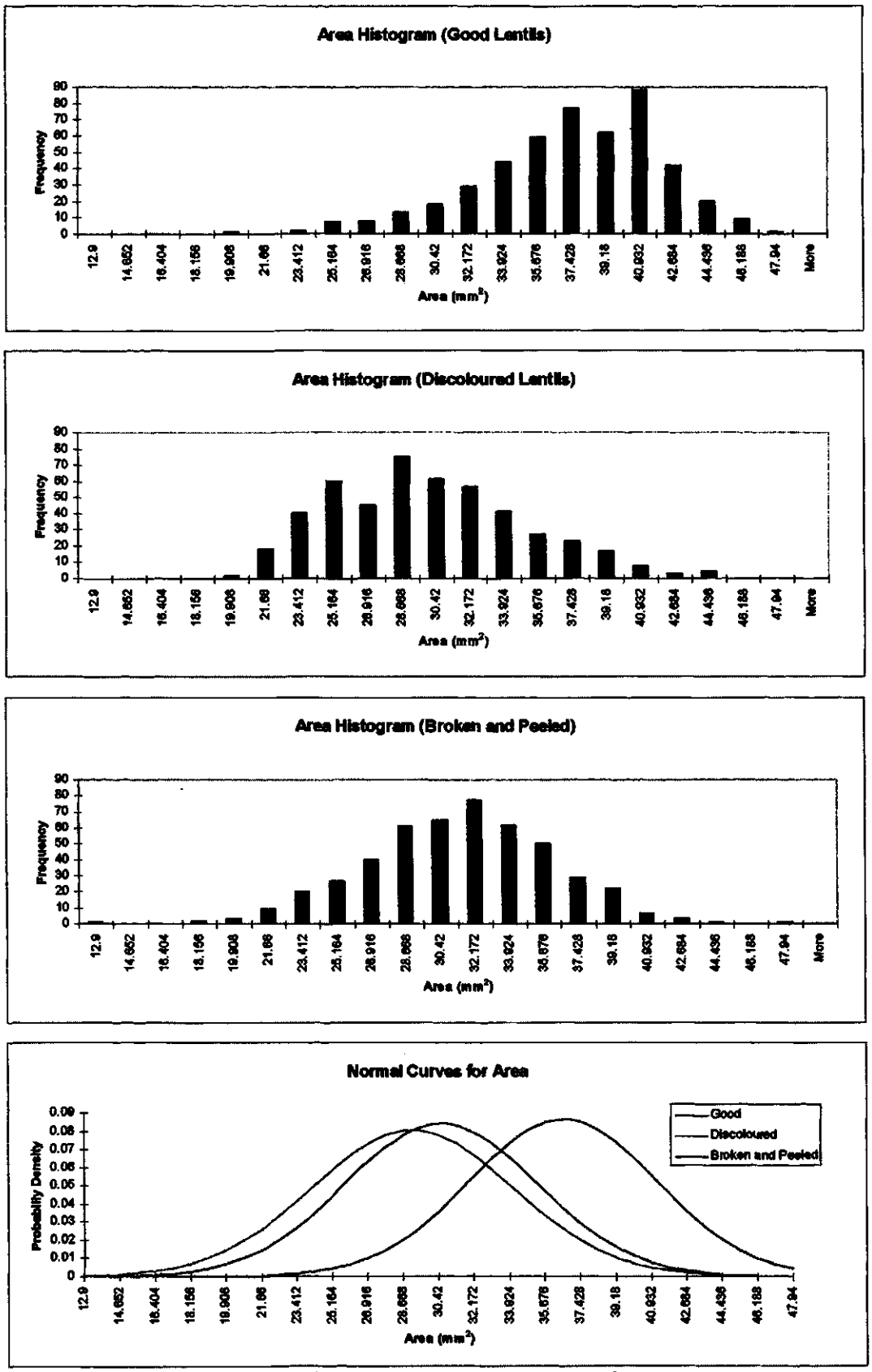


Figure B1 Histogram and normal curves for Area.

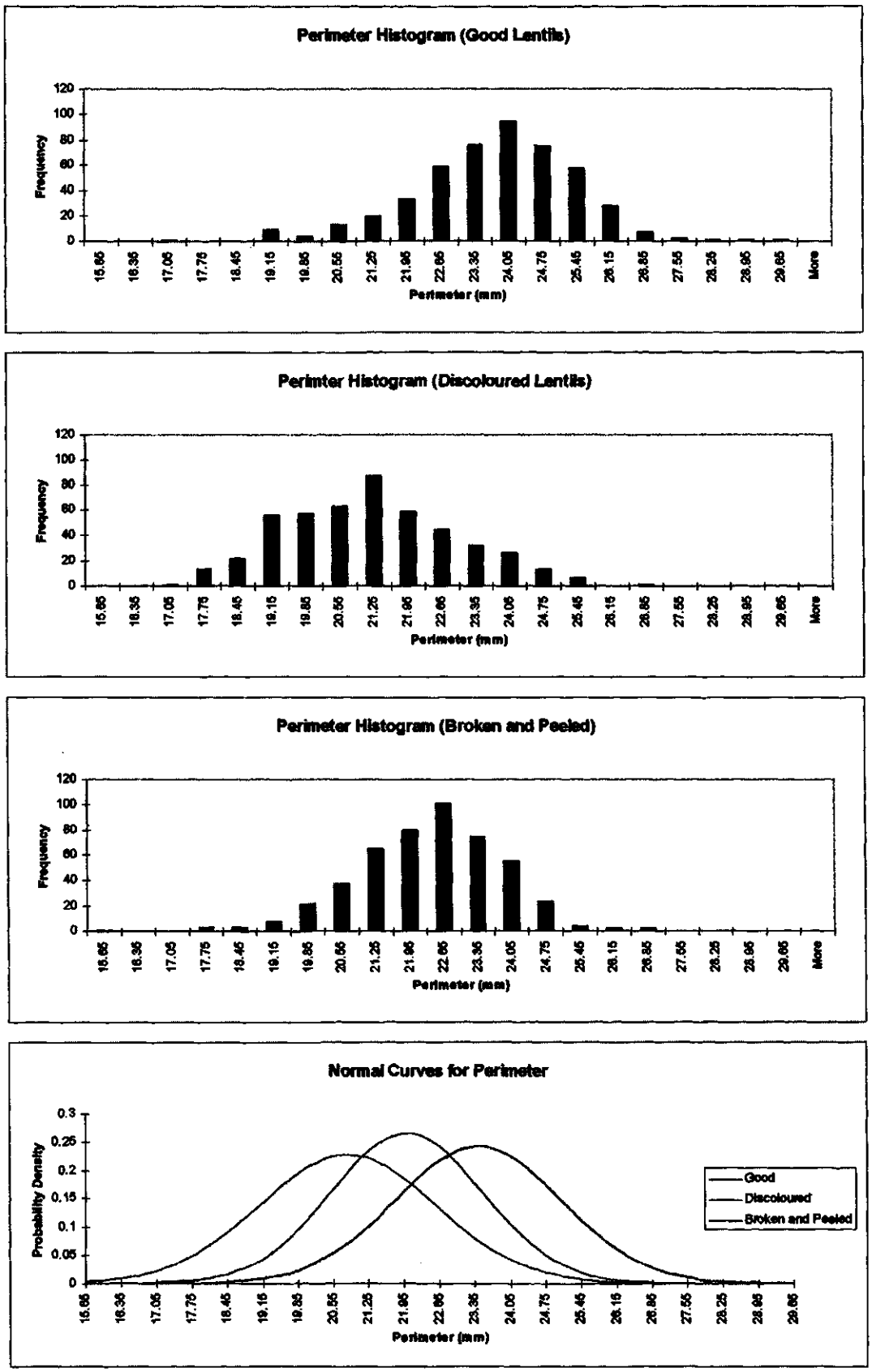
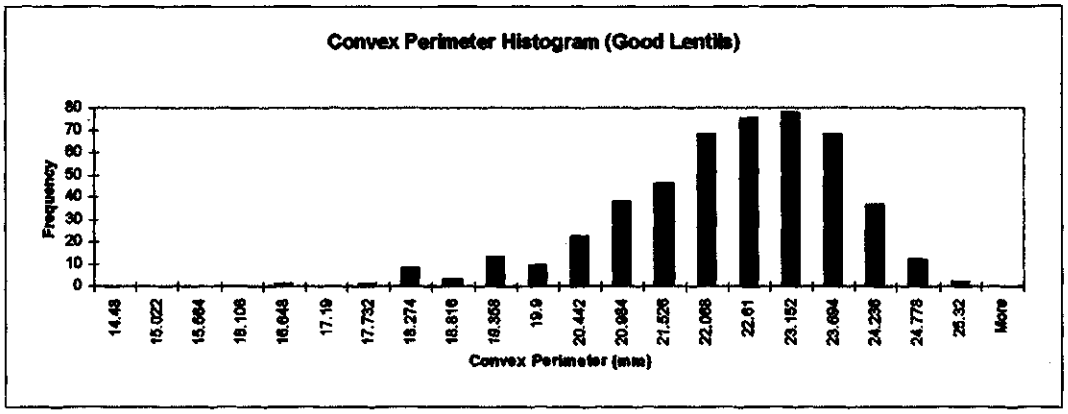
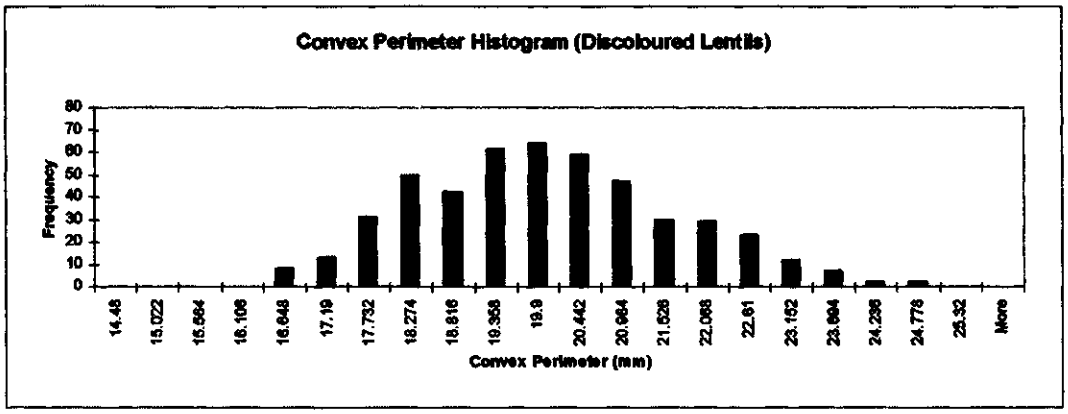
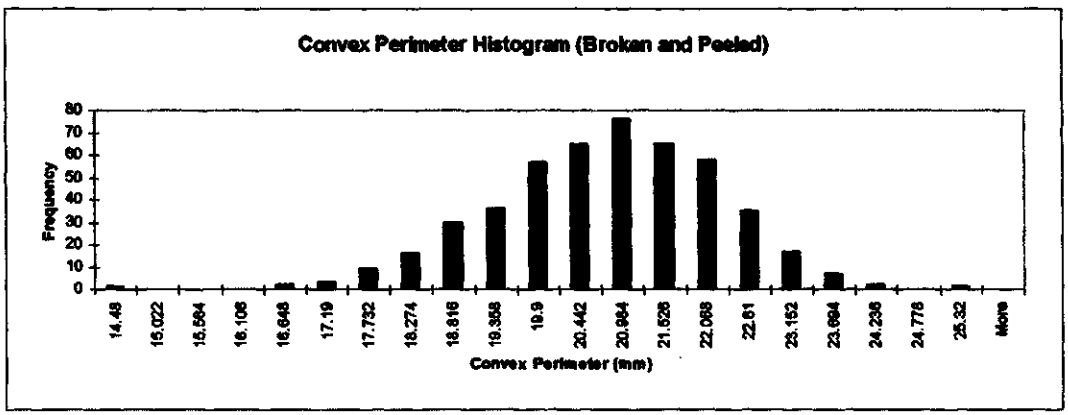


Figure B2 Histogram and normal curves for Perimeter.







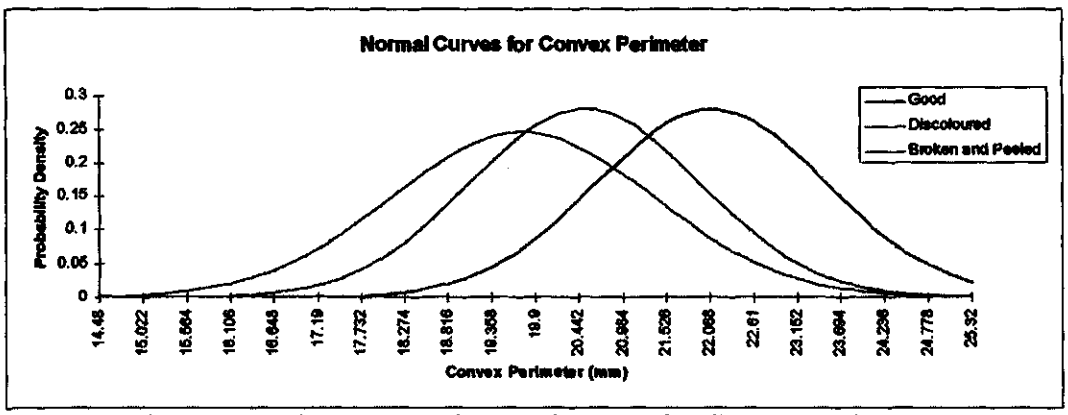


Figure B3 Histogram and normal curves for Convex Perimeter.

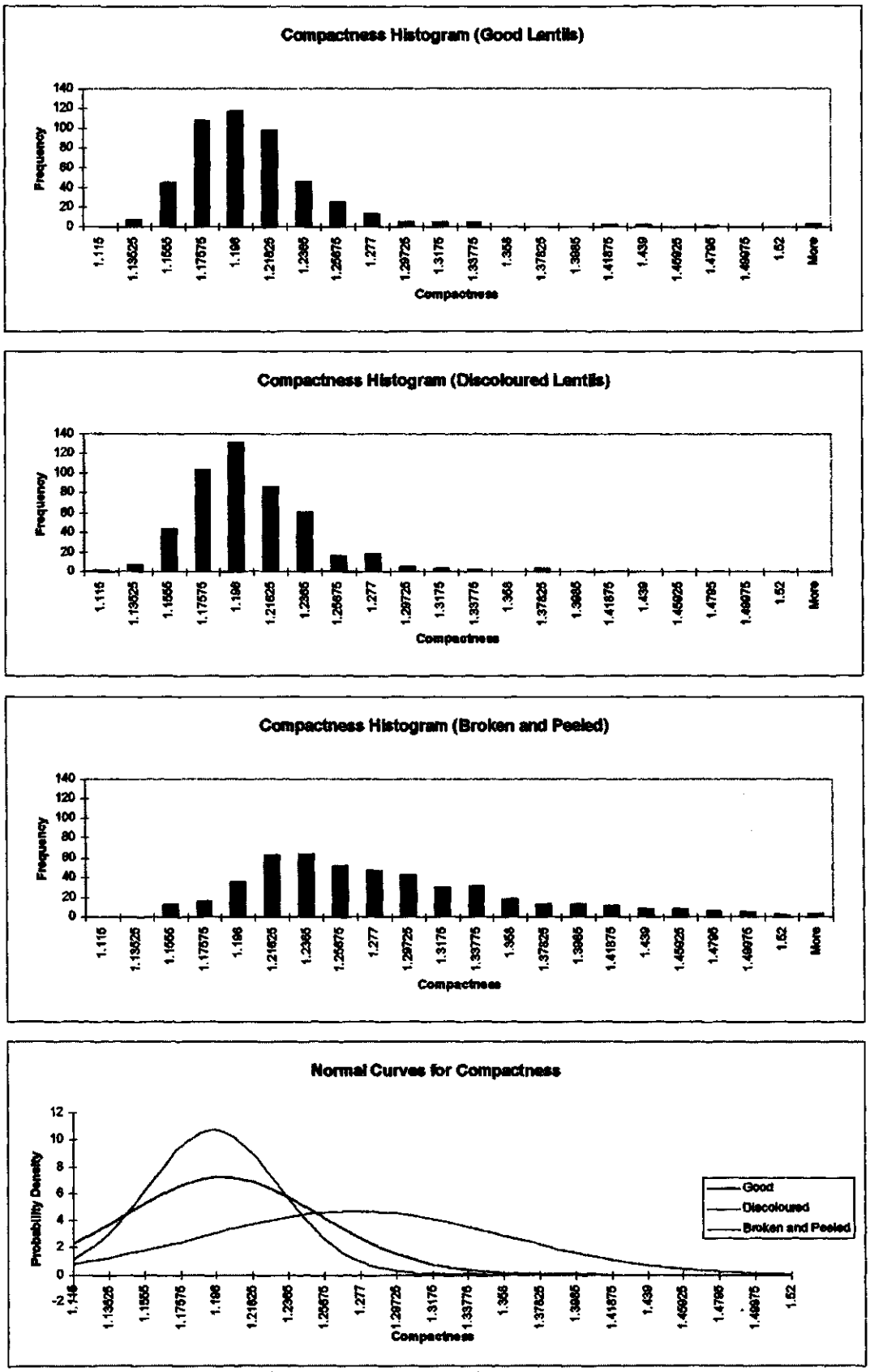
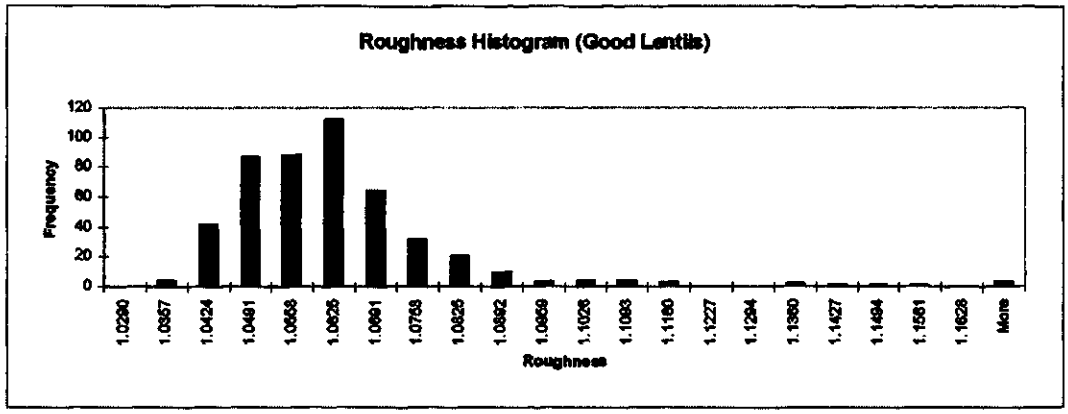
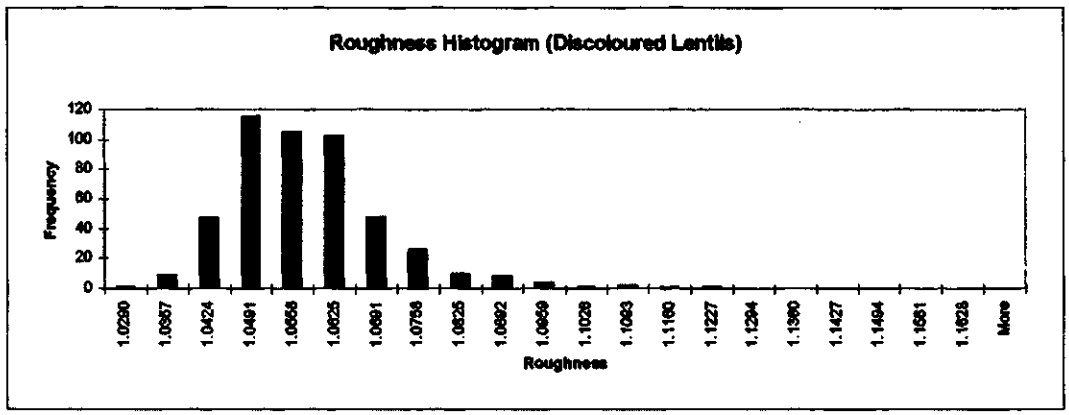
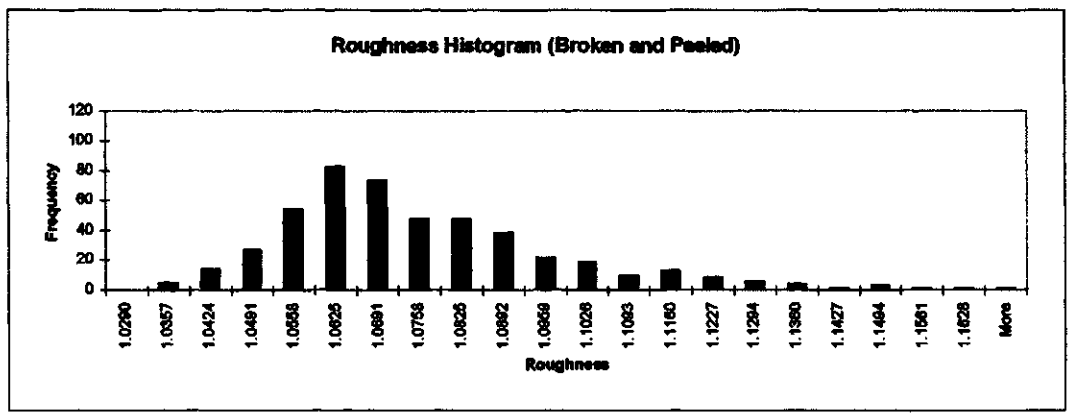


Figure B4 Histogram and normal curves for Compactness.







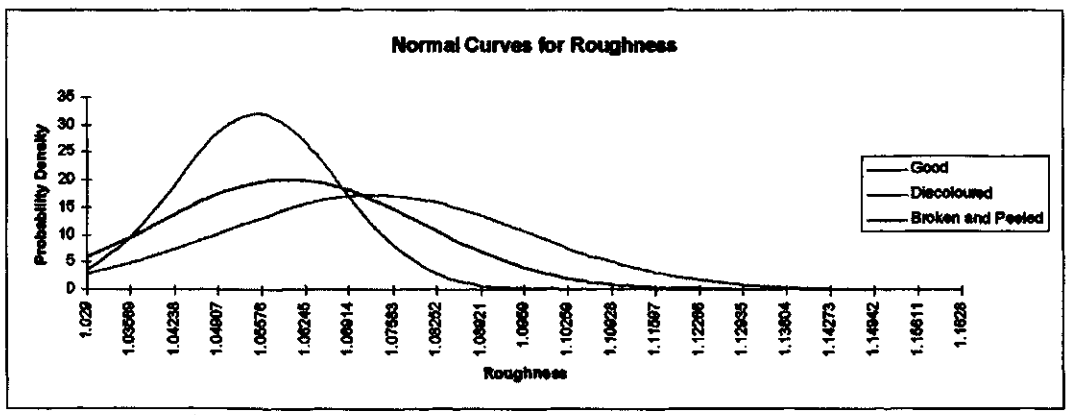
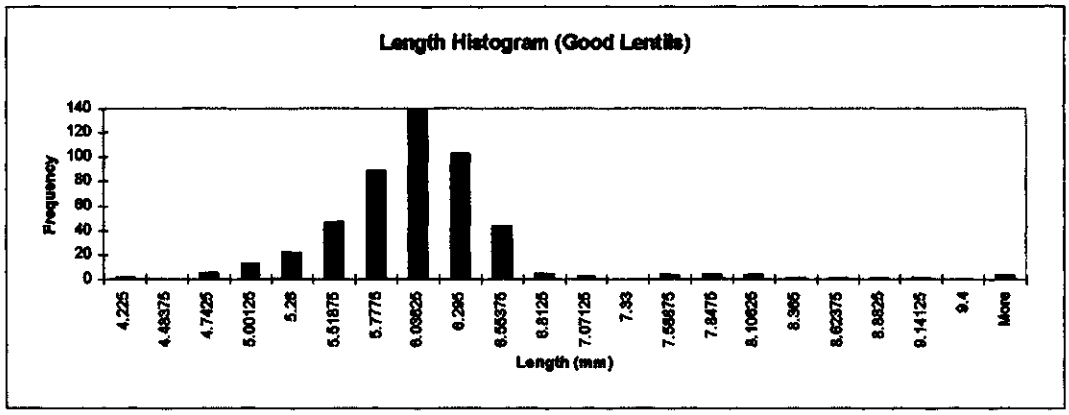
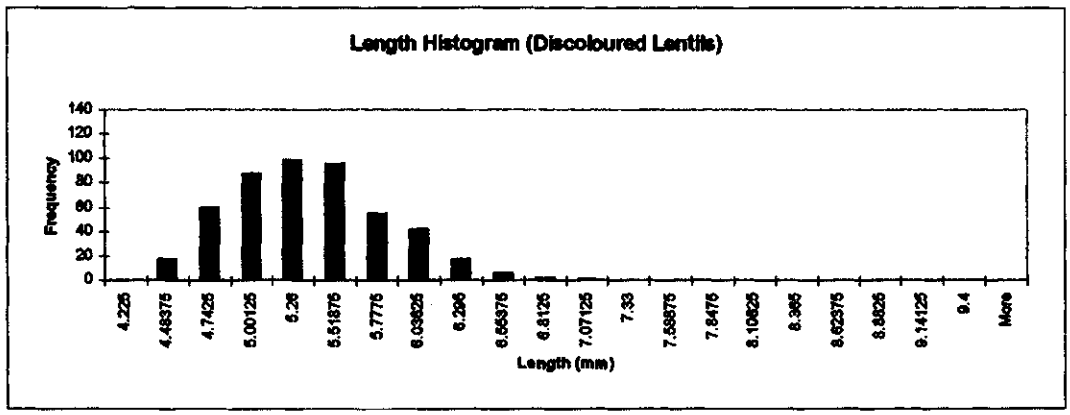
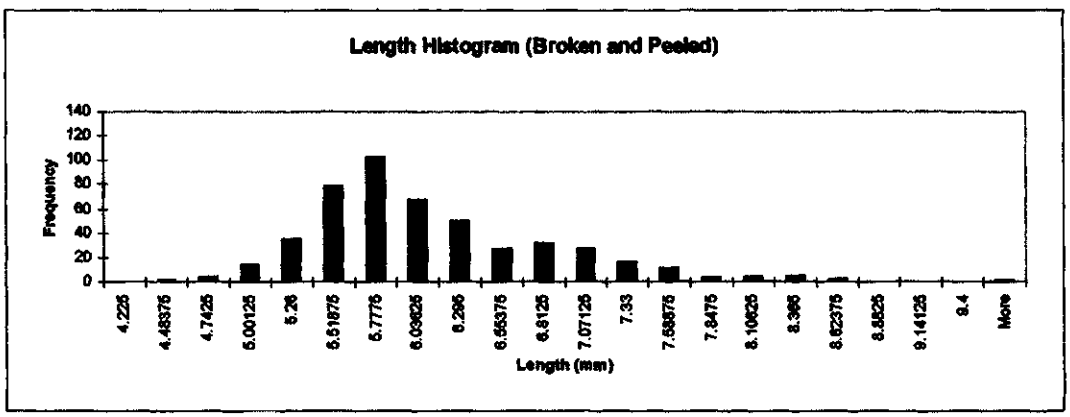


Figure B5 Histogram and normal curves for Roughness.







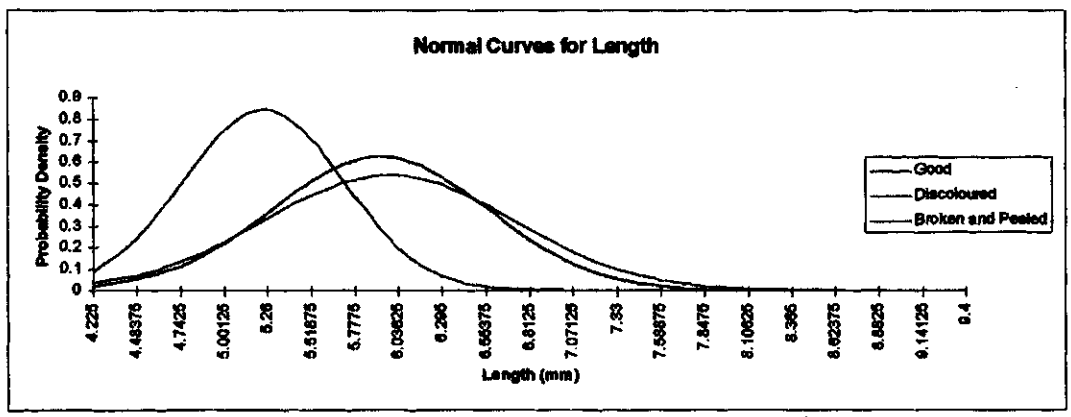


Figure B6 Histogram and normal curves for Length.

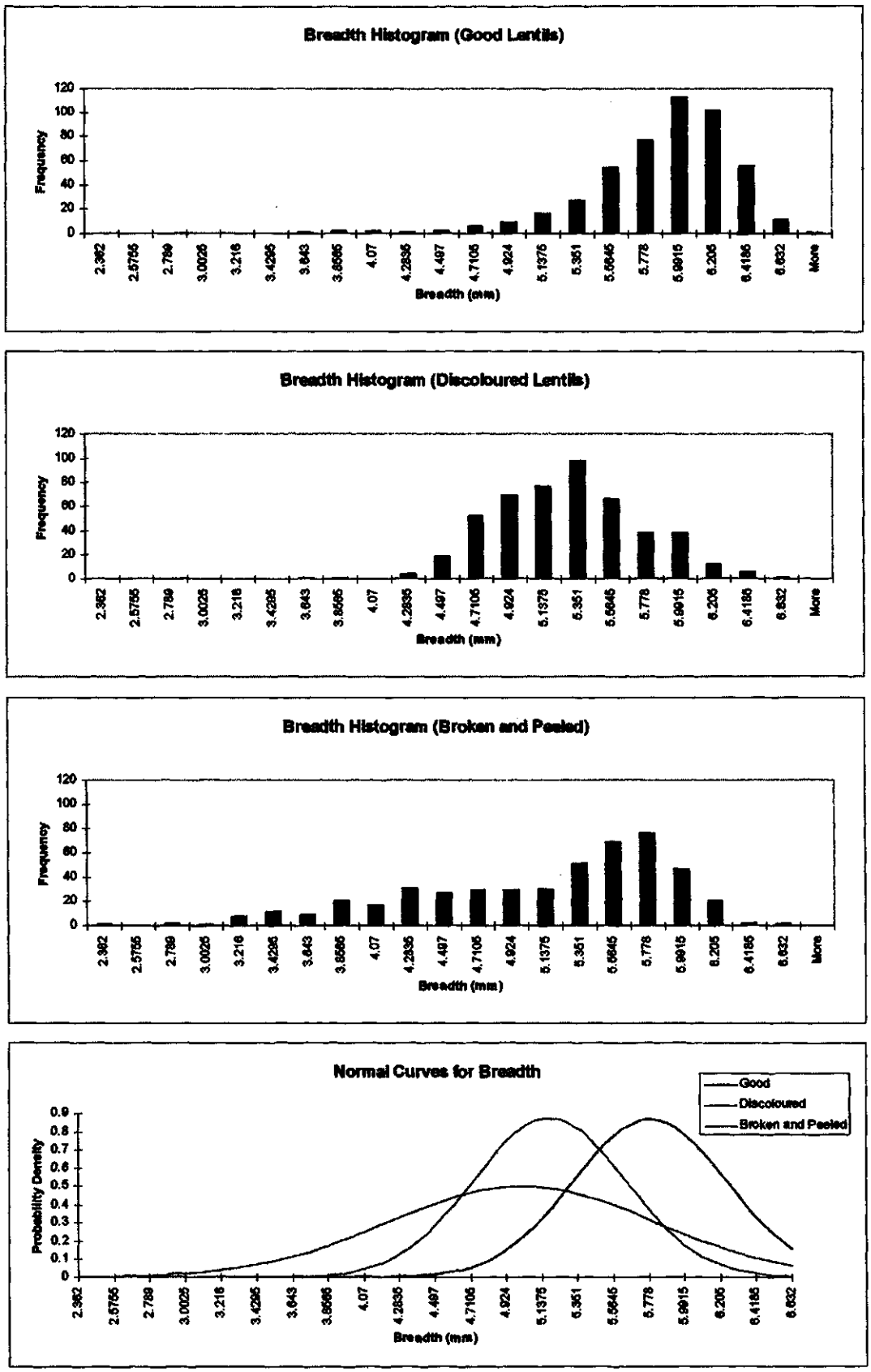


Figure B7 Histogram and normal curves for Breadth.

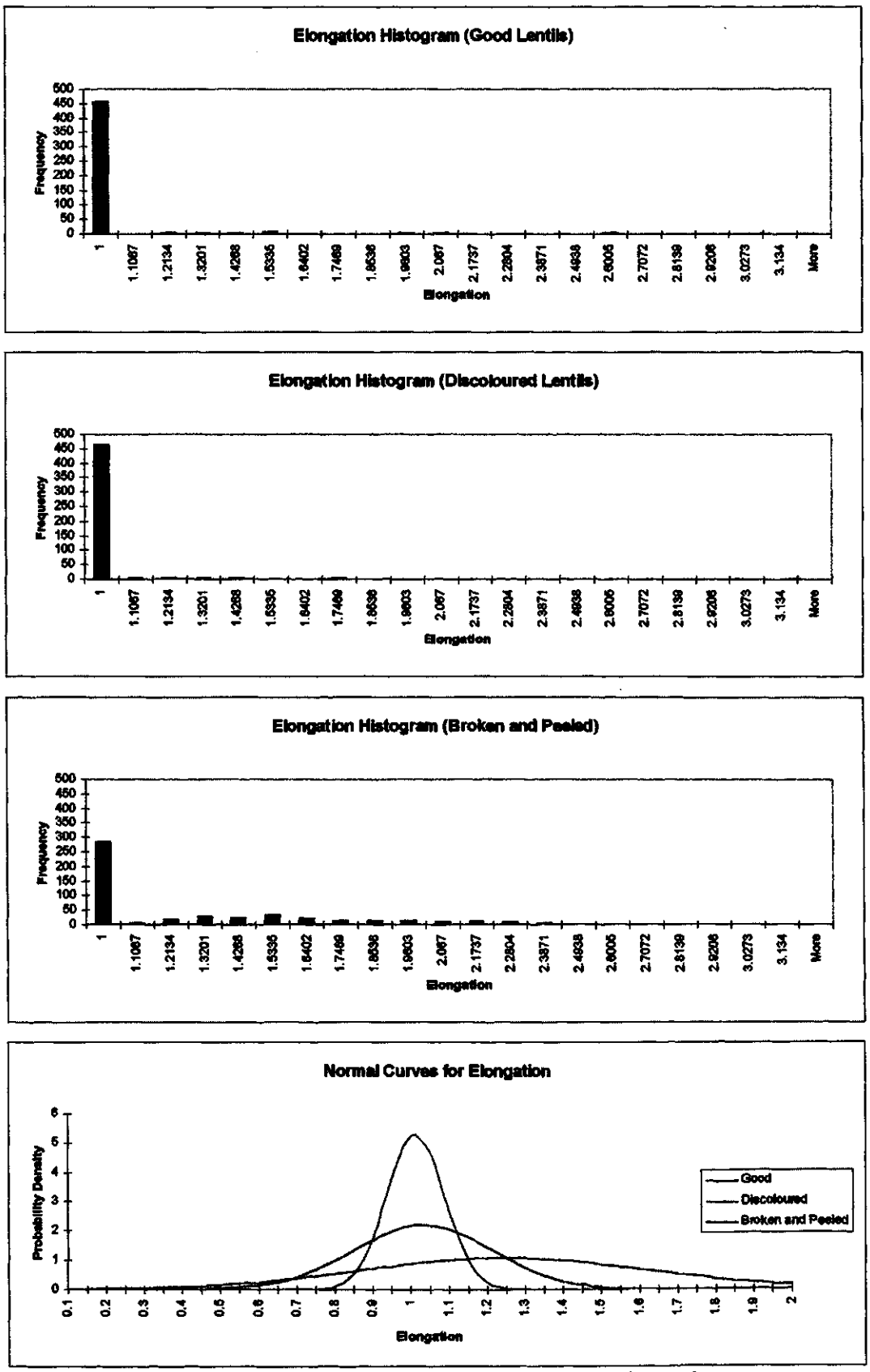


Figure B8 Histogram and normal curves for Elongation.

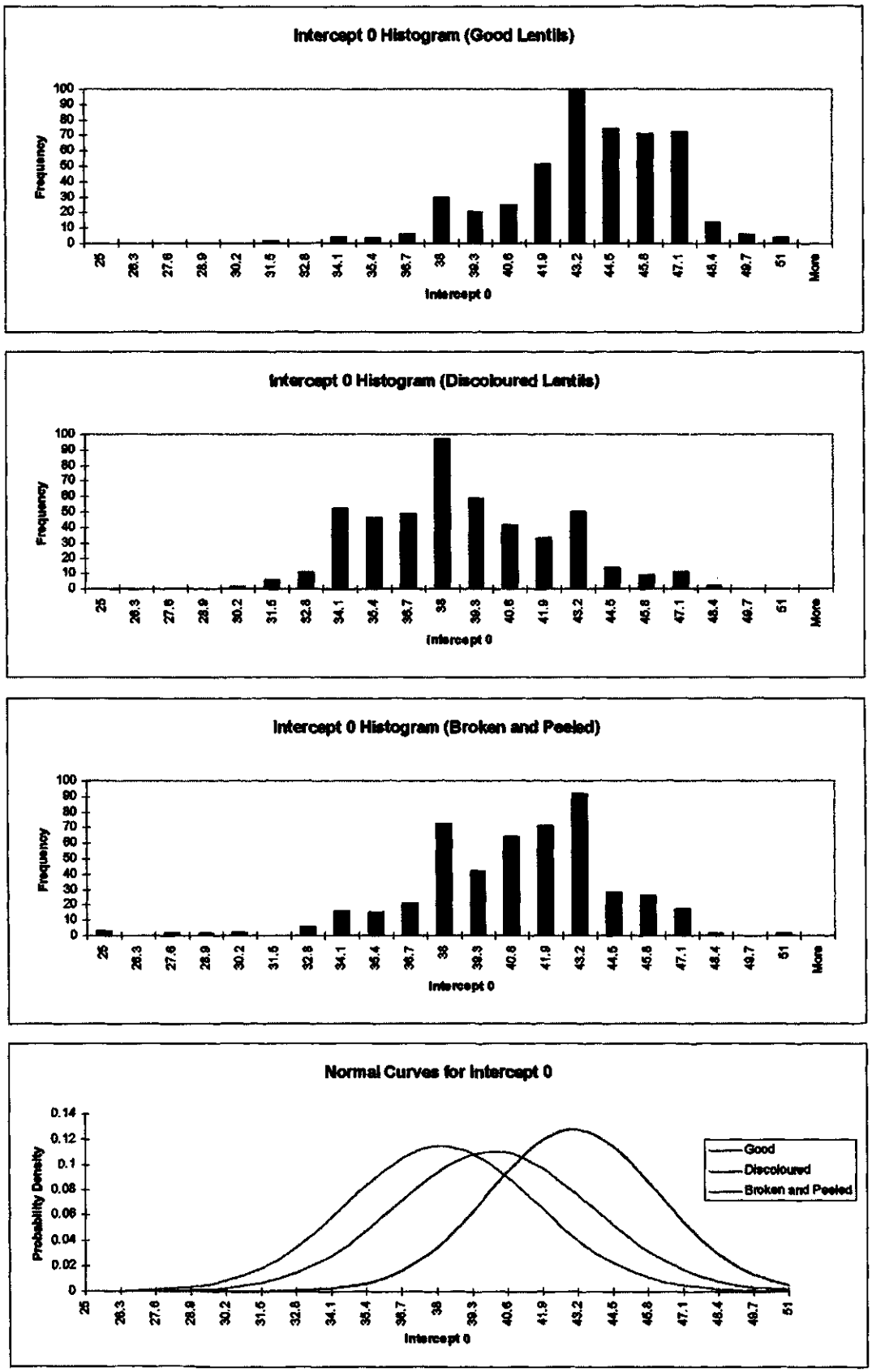


Figure B9 Histogram and normal curves for Intercept 0.

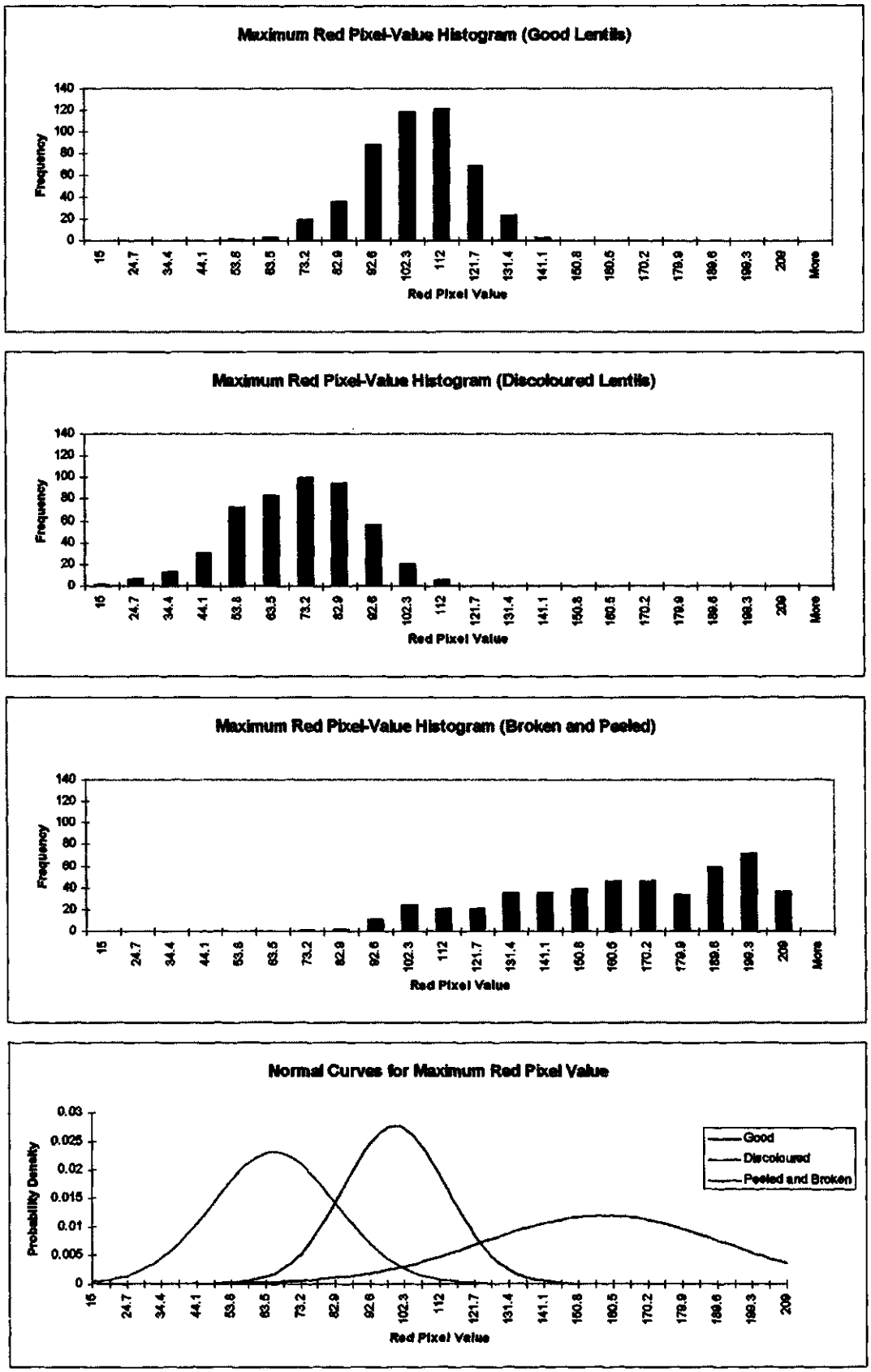
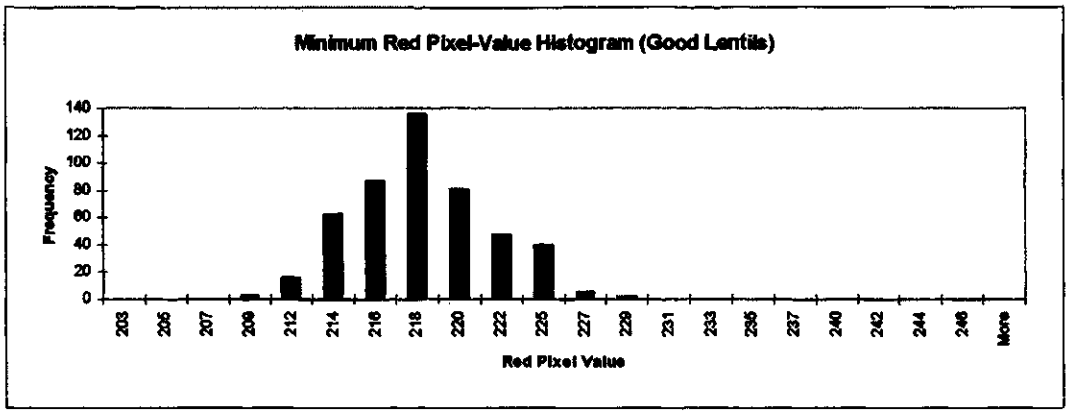
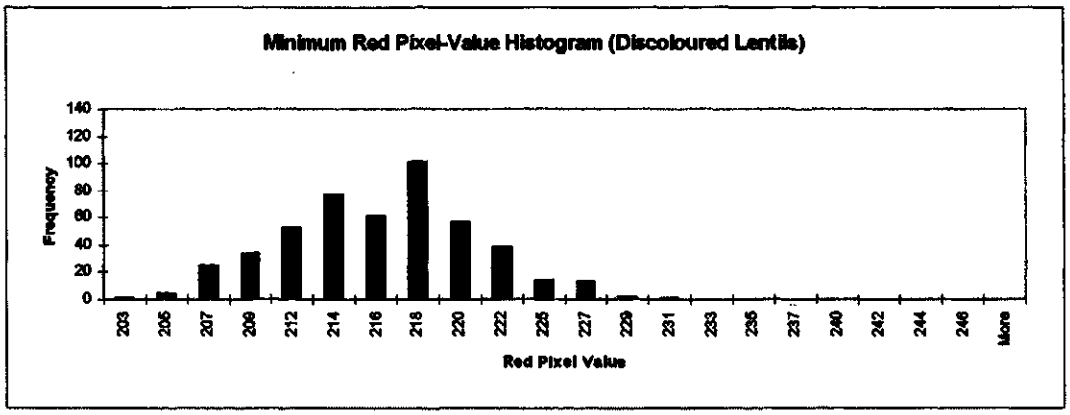
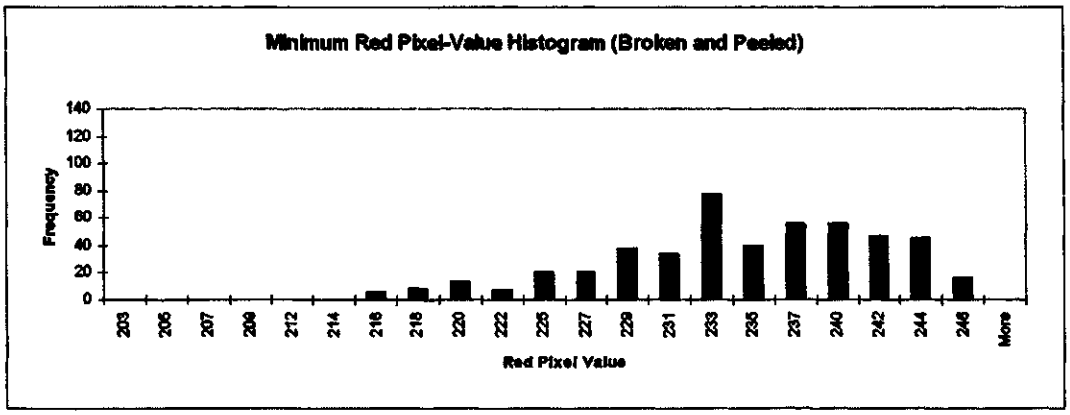


Figure B10 Histogram and normal curves for Maximum Red Pixel.







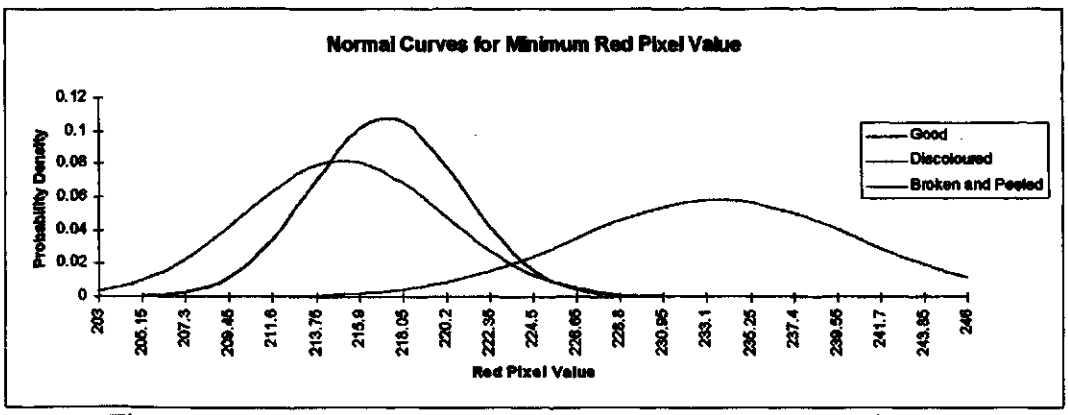


Figure B11 Histogram and normal curves for Minimum Red Pixel.

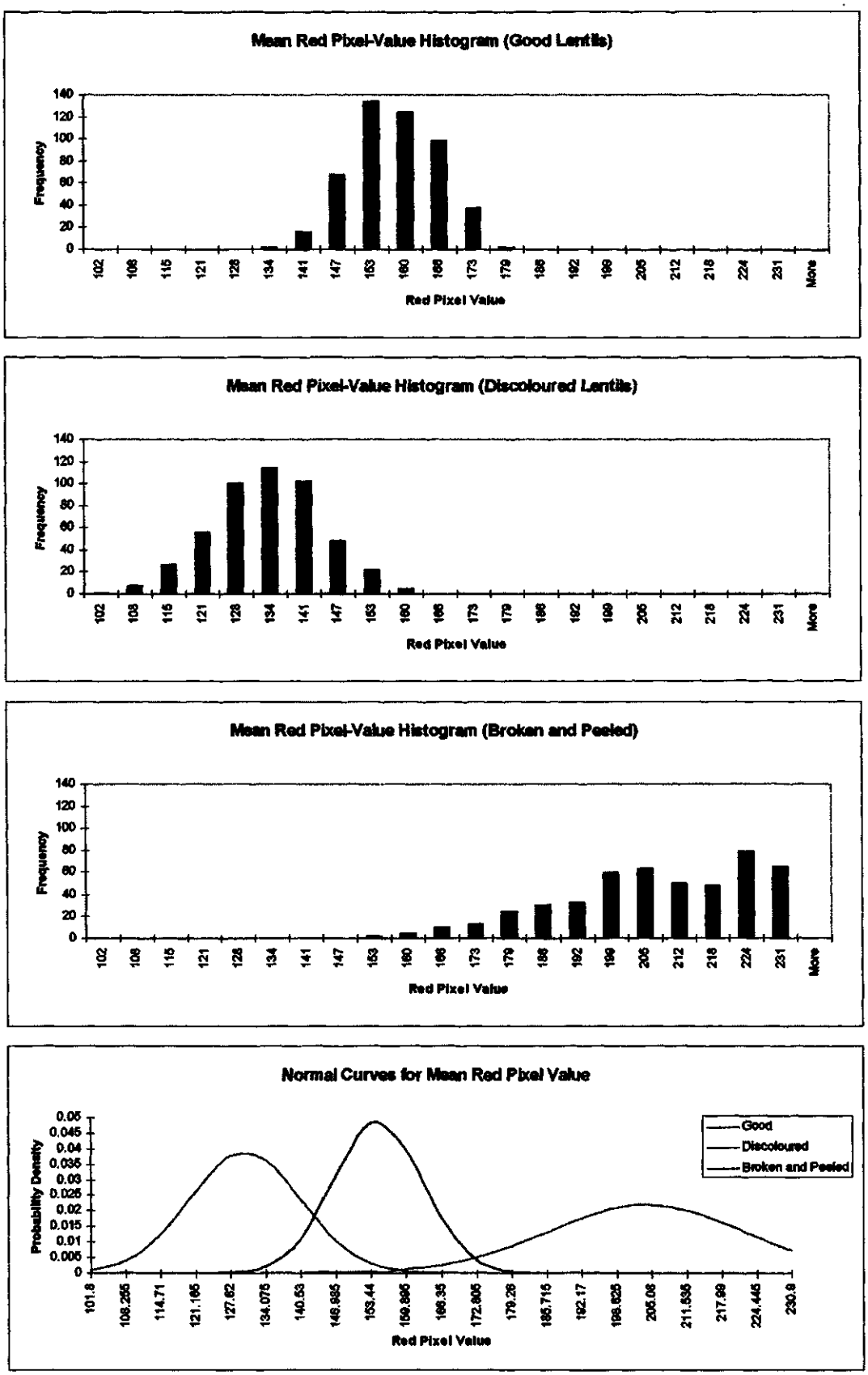
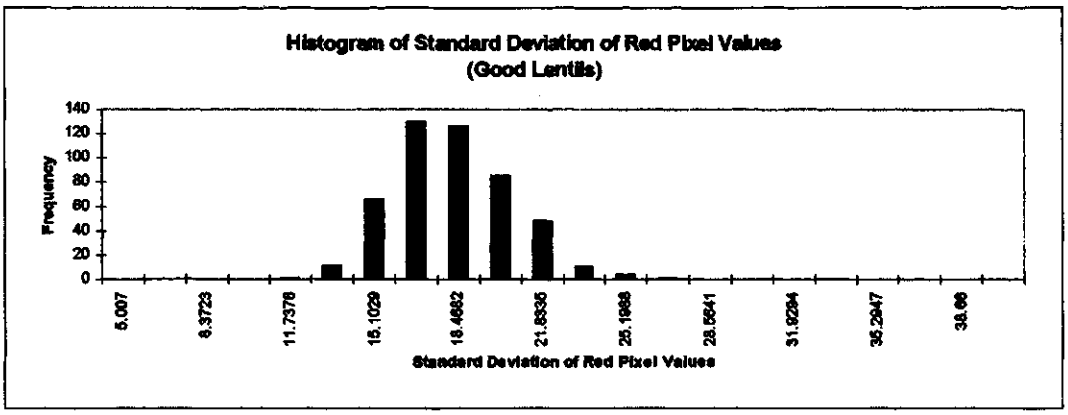
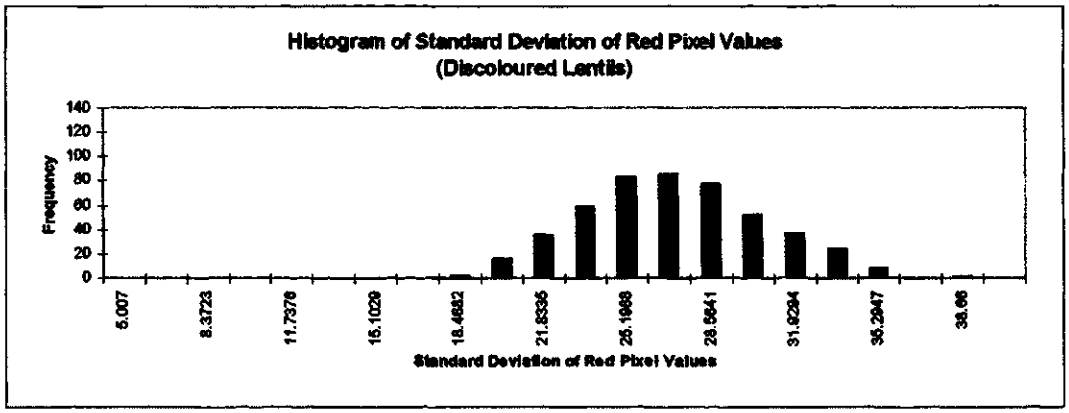
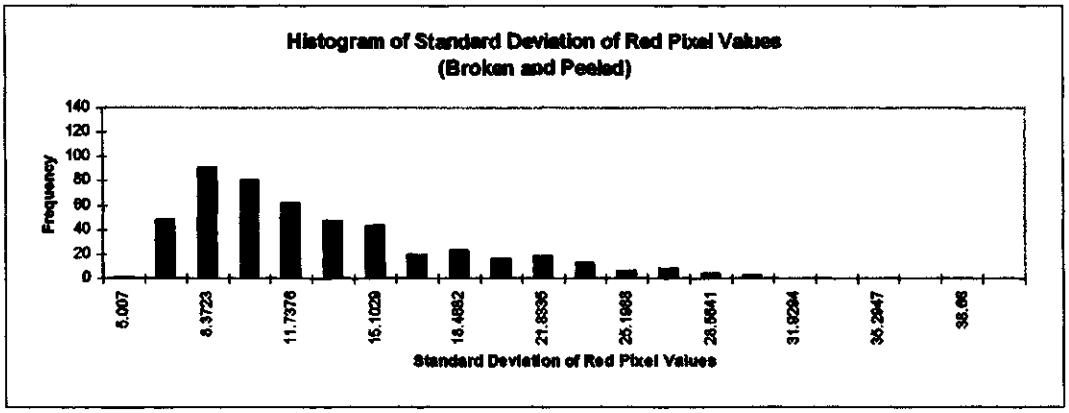


Figure B12 Histogram and normal curves for Mean Red Pixel.







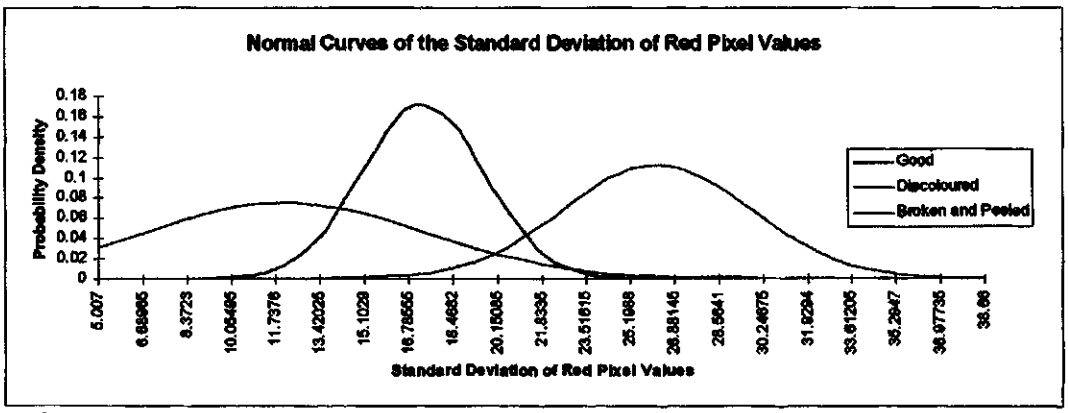


Figure B13 Histogram and normal curves for Standard Deviation of Red Pixels.

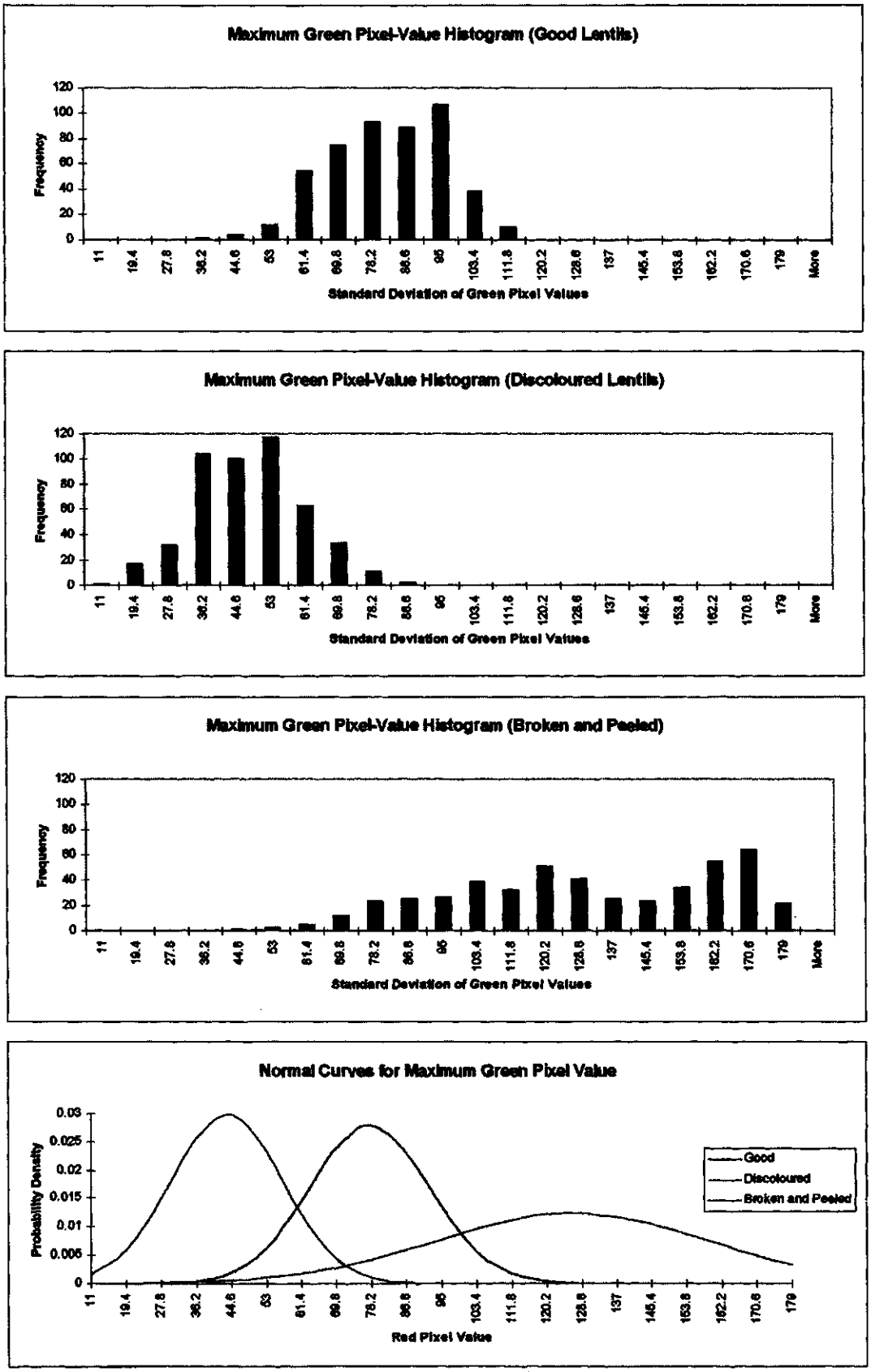


Figure B14 Histogram and normal curves for Maximum Green Pixel.

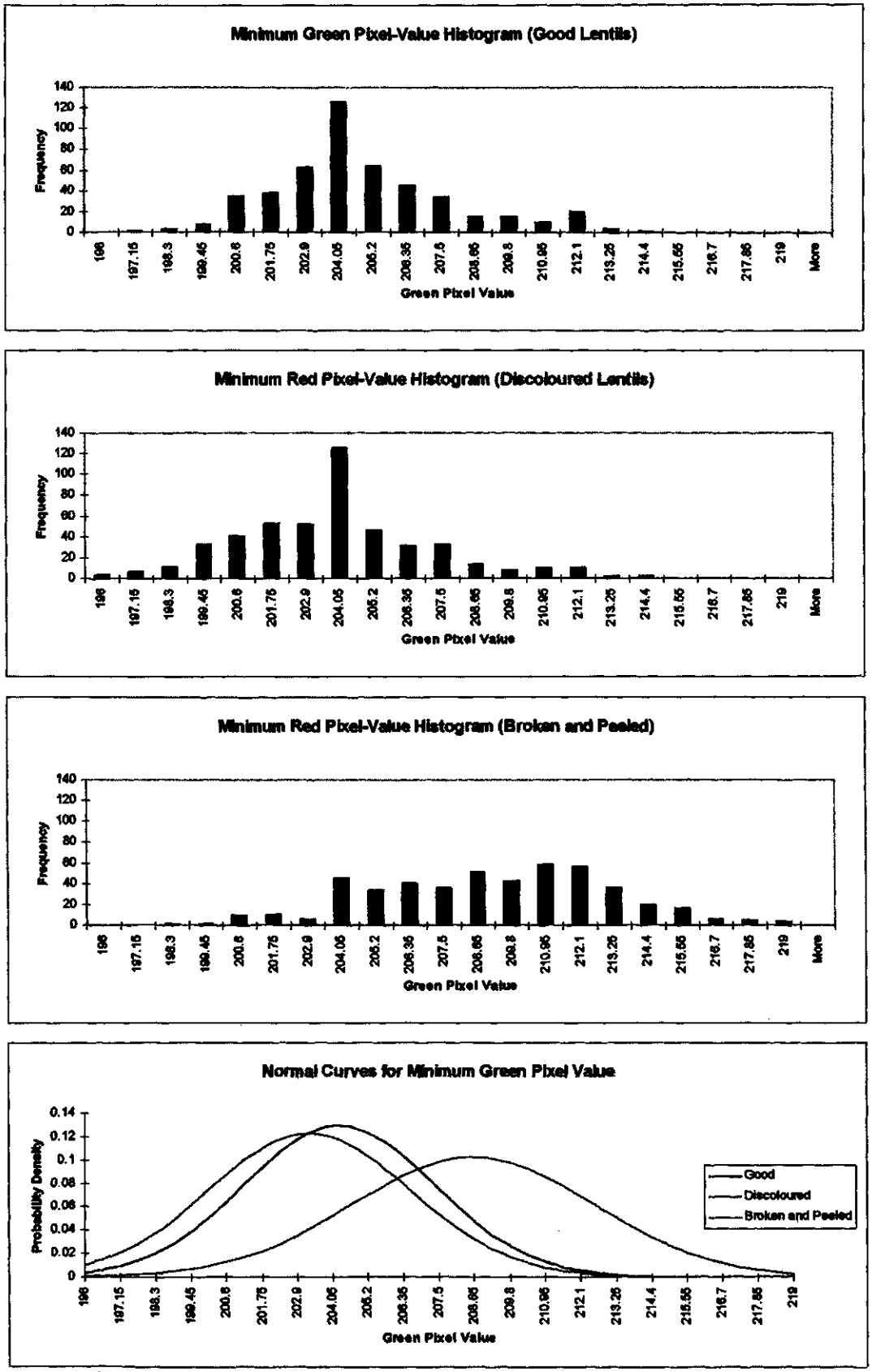


Figure B15 Histogram and normal curves for Minimum Green Pixel.

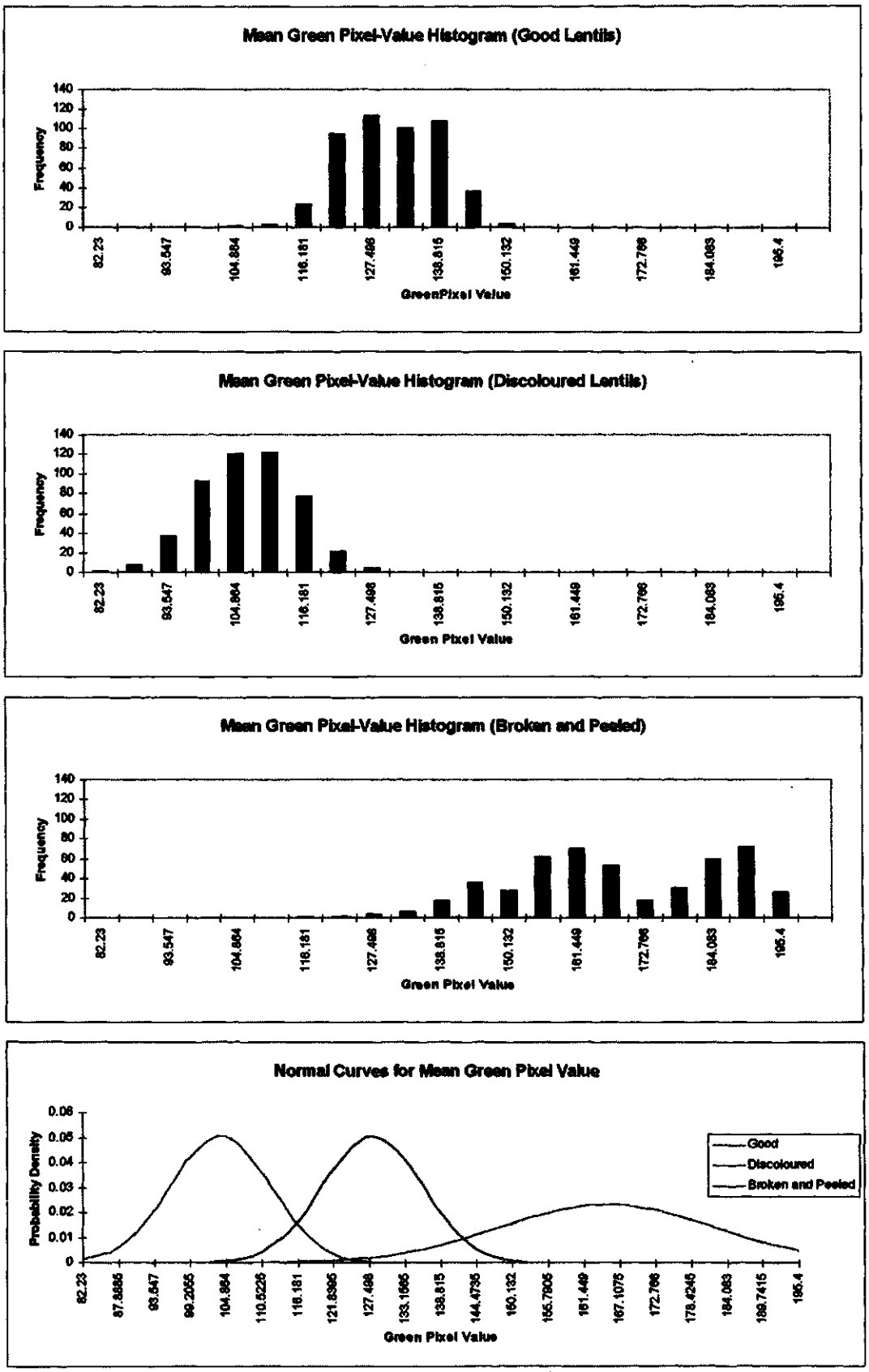


Figure B16 Histogram and normal curves for Mean Green Pixel.

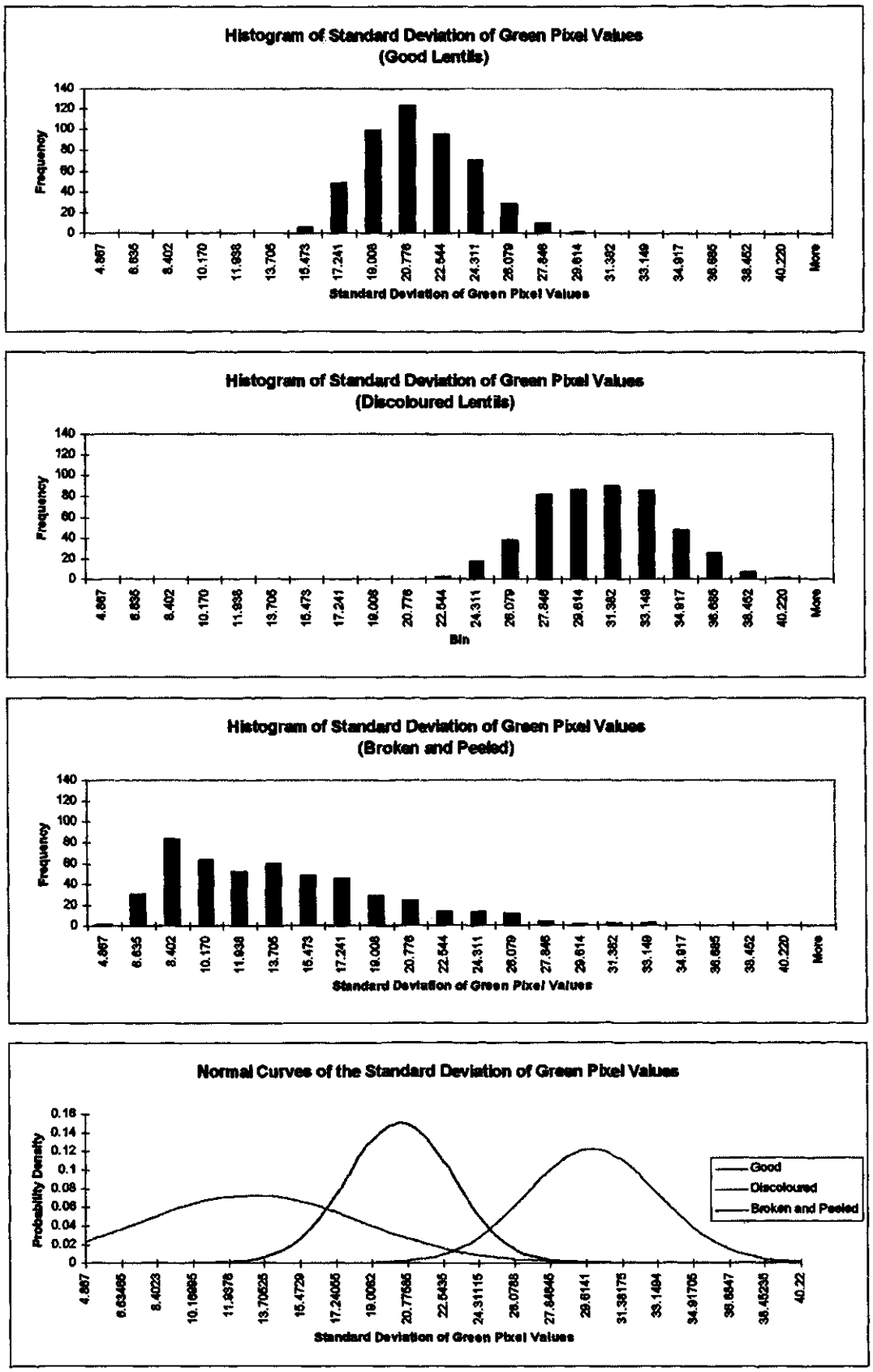
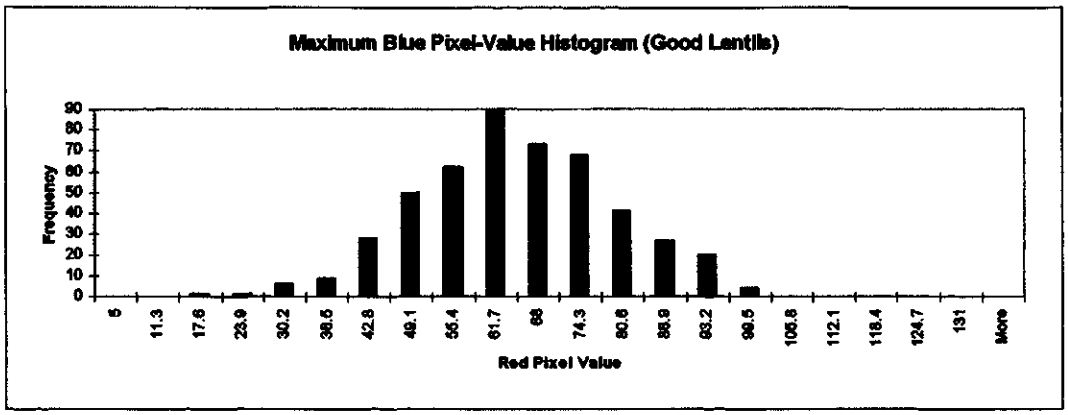
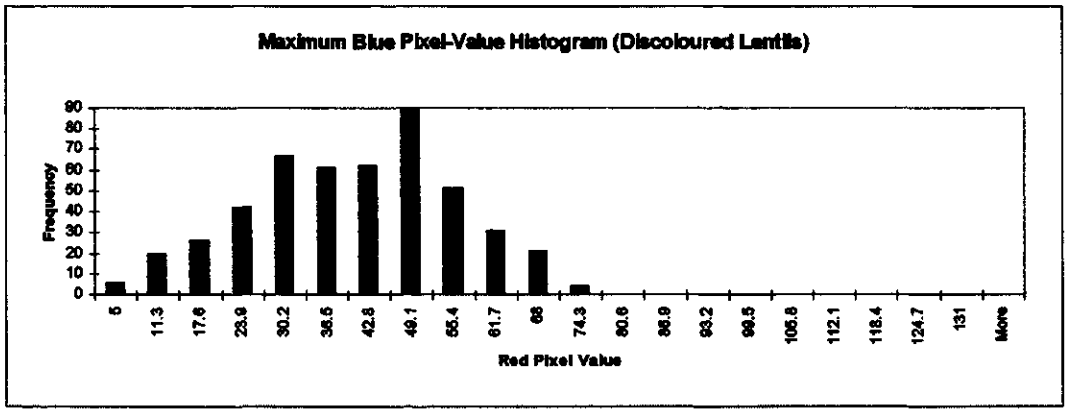
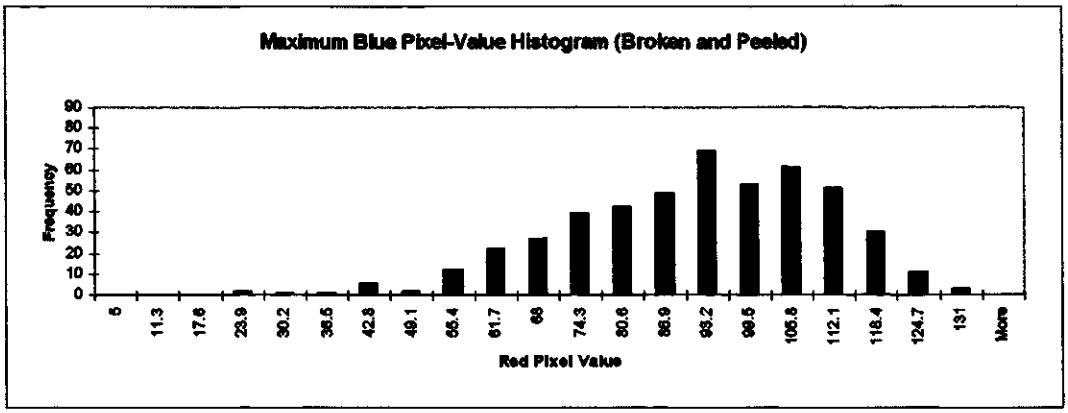


Figure B17 Histogram and normal curves for Standard Deviation of Green Pixels.







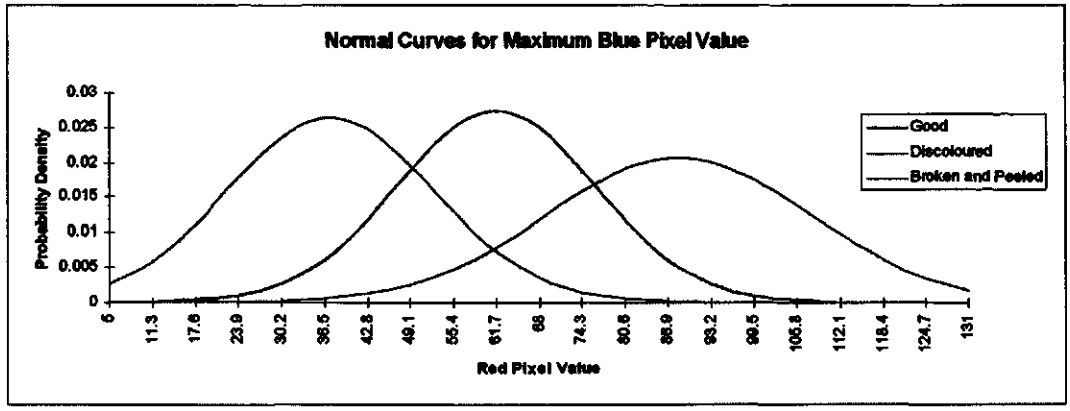
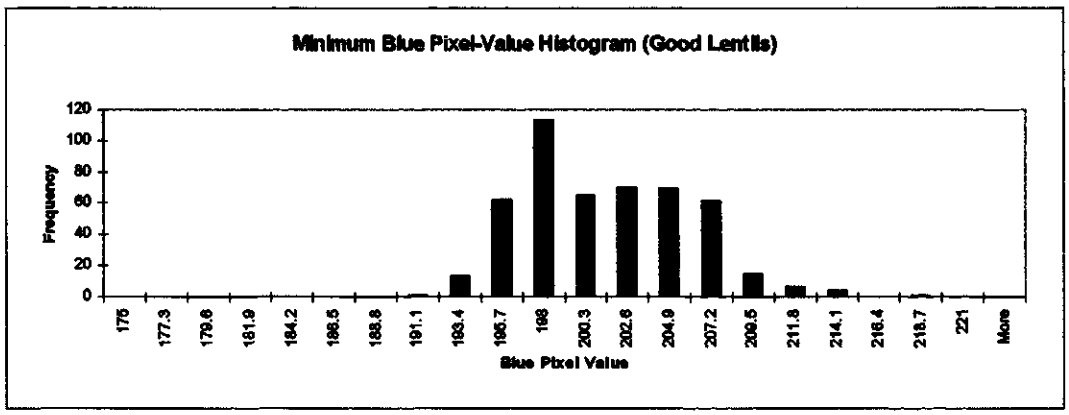
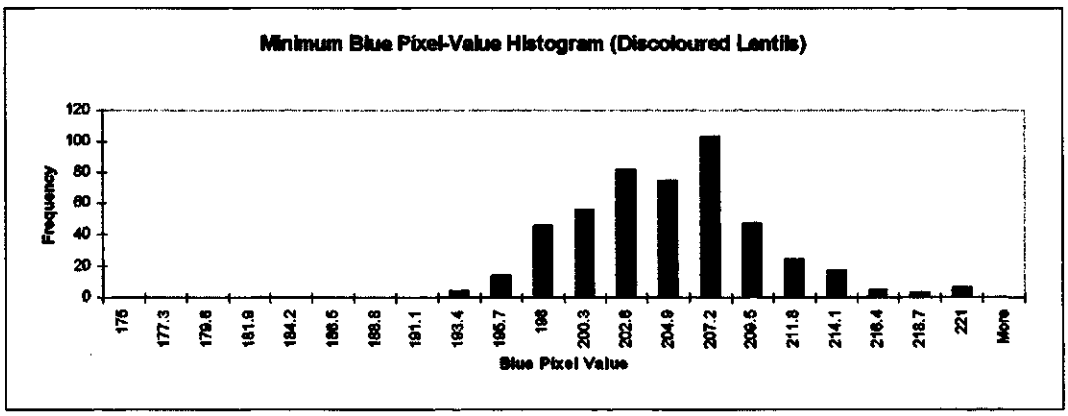
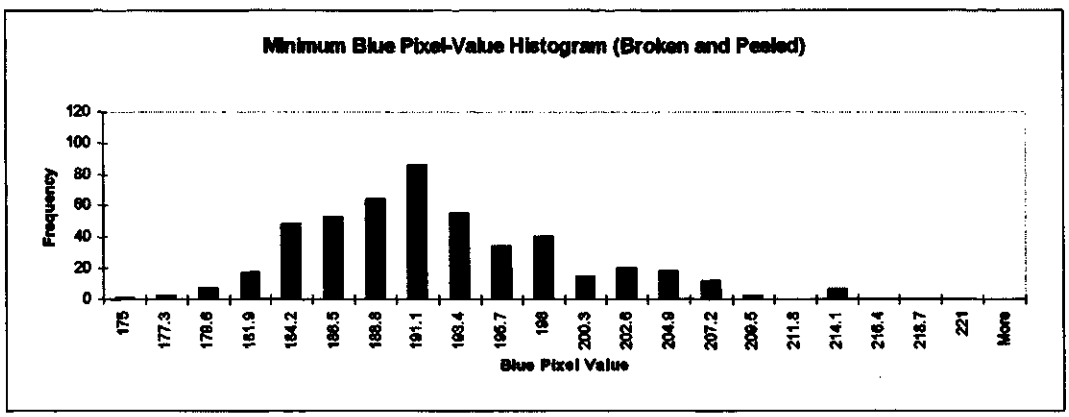


Figure B18 Histogram and normal curves for Maximum Blue Pixel.







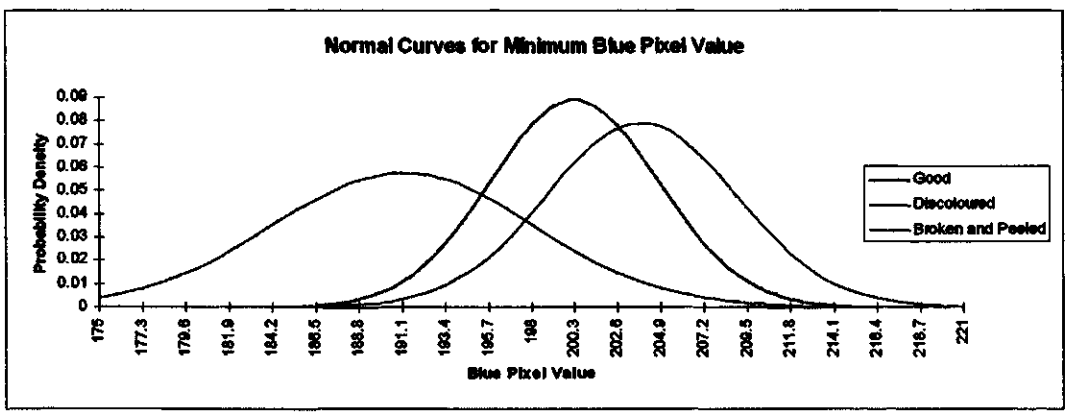
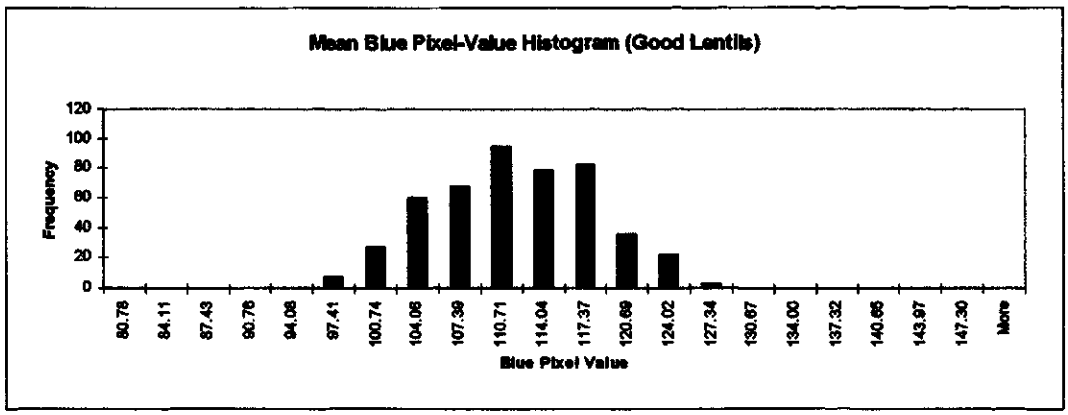
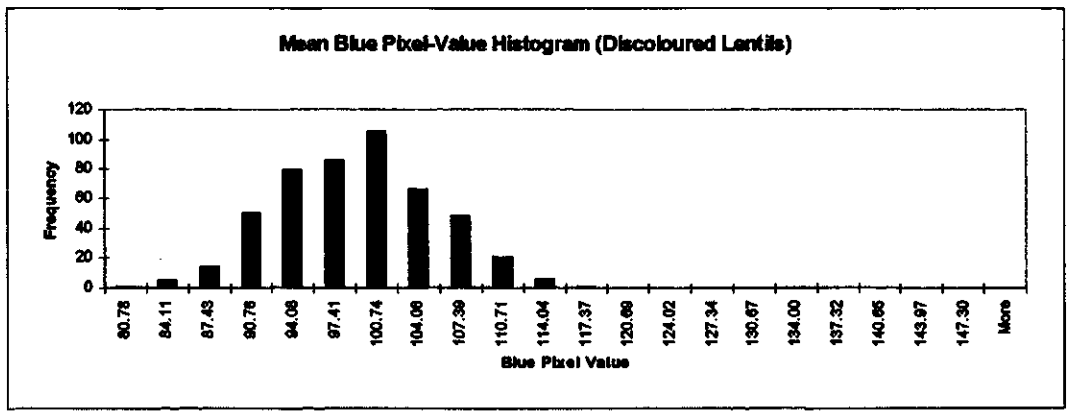
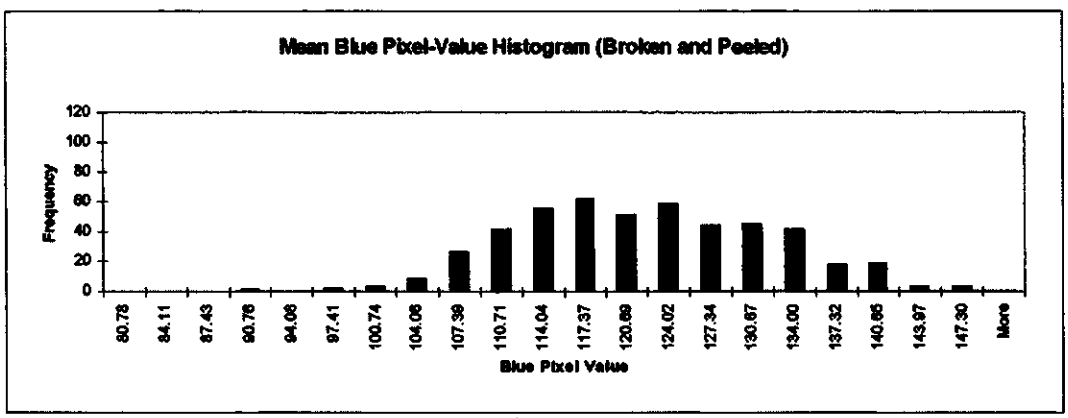


Figure B19 Histogram and normal curves for Minimum Blue Pixel.







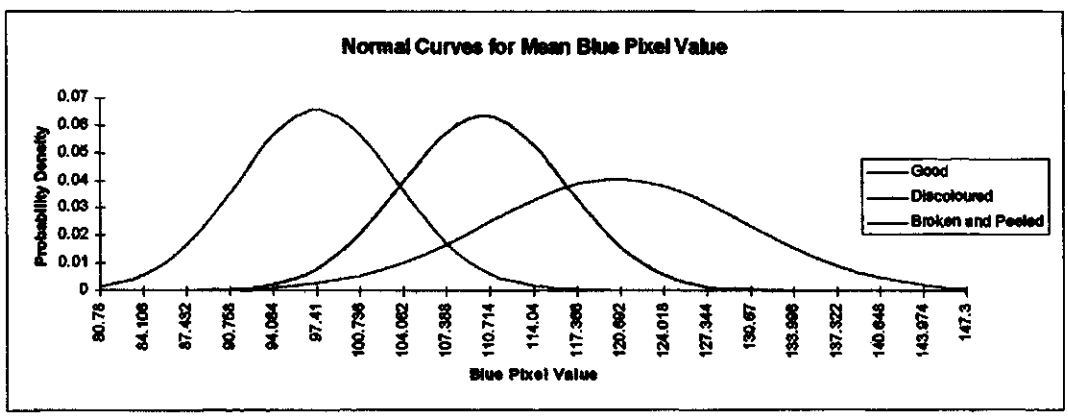
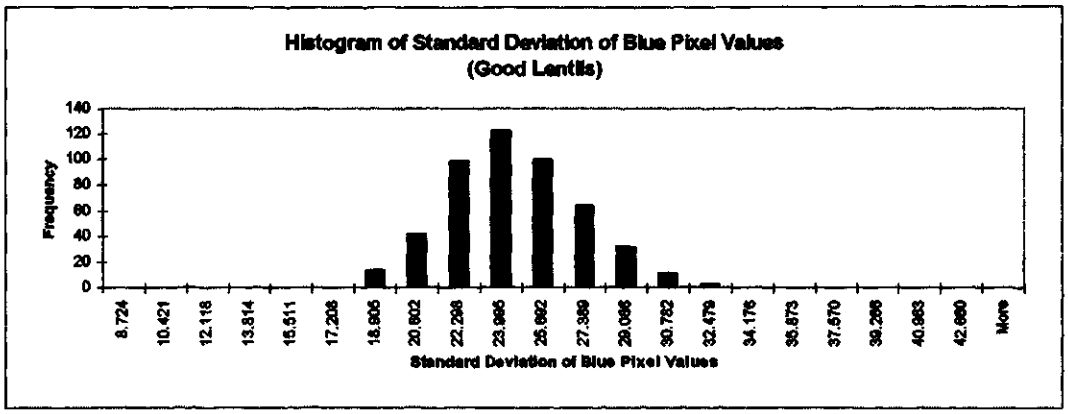
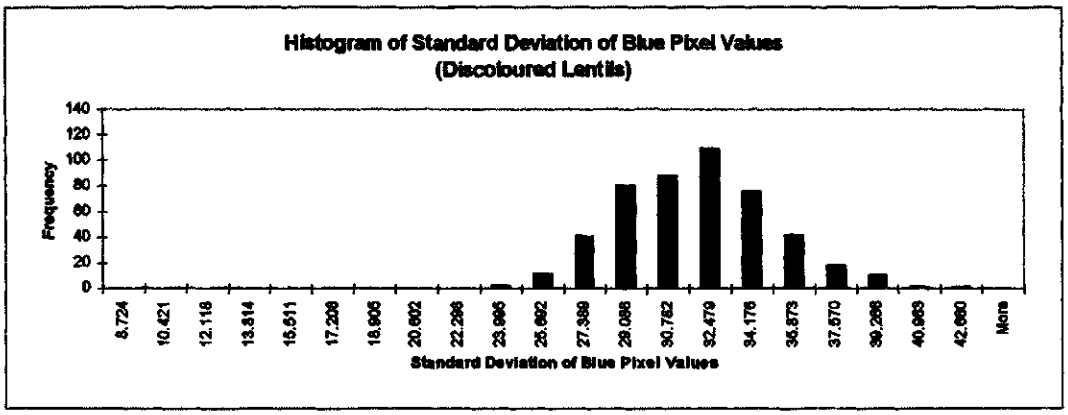
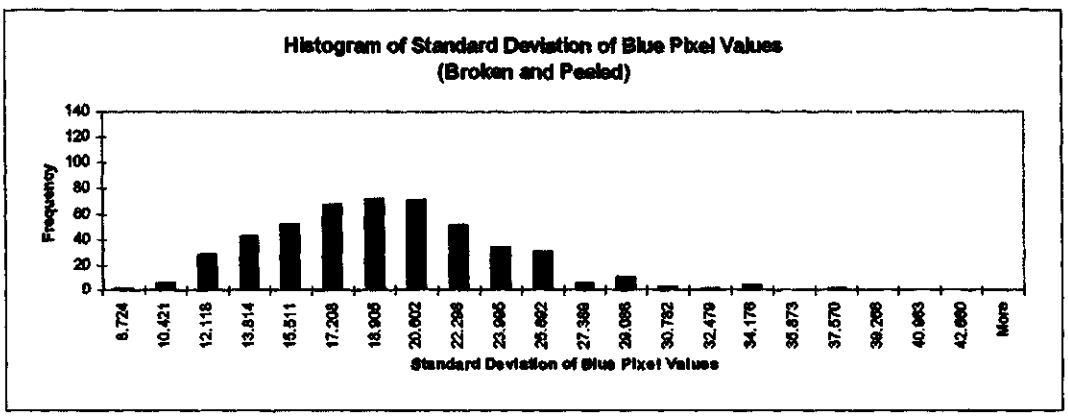


Figure B20 Histogram and normal curves for Mean Blue Pixel.







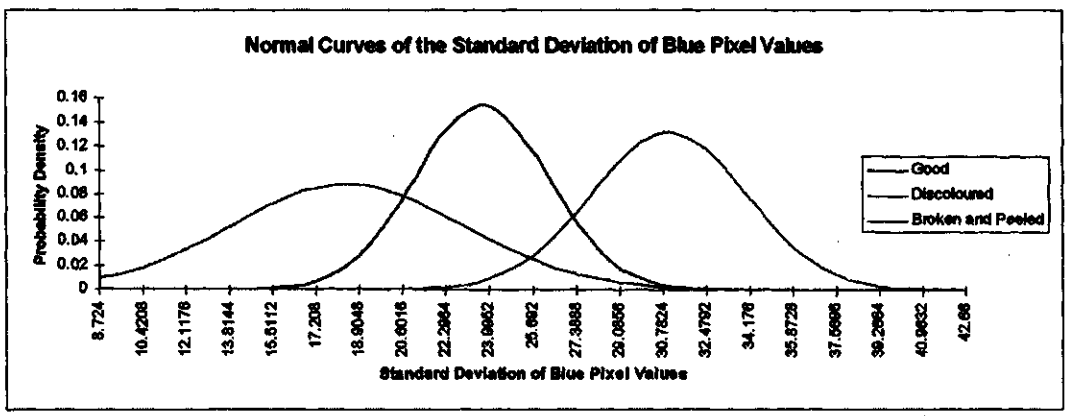


Figure B21 Histogram and normal curves for Standard Deviation of Blue Pixels.

APPENDIX C

Derivation of the Back-Propagation Equations

C-1 Update of Output Weights

The output of the neural network is given by the equations:

$$net^{o}_{pk} = \sum_{j=1}^{L} w^{o}_{kj} i_{pj} + b^{o}_{k} \qquad (A.1)$$

and

$$o_{pk} = f^{o}_{k}(net^{o}_{pk}) \qquad (A.2)$$

where net^{o}_{pk} is the sum of the weighted inputs of the k^{th} output neuron due to the p^{th} network input,

w^{o}_{kj} is the j^{th} weight on the k^{th} output node,

i_{pj} is the output of the j^{th} neuron of the previous layer due to the p^{th} input,

b^{o}_{k} is the bias weight on the k^{th} output node,

f^{o}_{k} is the activation function of the k^{th} output node, and

o_{pk} is the output of the k^{th} output node due to the p^{th} input.

In the back-propagation learning algorithm, a given input must have a known network output associated with it. The error of the network is then:

$$\delta_{pk} = (y_{pk} - o_{pk}), \tag{A.3}$$

where δ_{pk} is the error of the k^{th} output neuron due to the p^{th} input, and

y_{pk} is the desired output of the k^{th} output neuron due to the p^{th} input,

and the sum of squares of the errors E_p of all L output nodes to be:

$$E_p = \frac{1}{2} \sum_{k=1}^{L} \delta_{pk}^2 . \tag{A.4}$$

The factor of $\frac{1}{2}$ in Equation 2.15 is an arbitrary factor to cancel out a factor of 2 from differentiating later. The error value E_p can be thought of as a (N+1) dimensional surface, where N is the number of input vectors. To update the weights, the direction of the steepest descent on the error surface must be found. This is done by calculating the negative gradient of E_p with respect to the weights w^o_{kj}.

$$\Delta w_{kj} = -\frac{\partial E_p}{\partial w^o_{kj}} \tag{A.5}$$

where Δw_{kj} is the calculated change in the weight. Expanding this we get:

$$\Delta w_{kj} = -\frac{\partial E_P}{\partial o_{pk}} \cdot \frac{\partial o_{pk}}{\partial net_{pk}} \cdot \frac{\partial net_{pk}}{\partial w_{kj}} \tag{A.6}$$

The first term in Equation A.6 is:

$$\frac{\partial E_P}{\partial o_{pk}} = \frac{1}{2}\frac{\partial}{\partial o_{pk}}(\Sigma(y_{pk} - o_{pk})^2)$$

$$= y_{pk} - o_{pk} \tag{A.7}$$

We can temporarily represent the middle term as:

$$\frac{\partial f^o_k}{\partial(net^o_{pk})} = f^{o'}_k \tag{A.8}$$

If we look at the last term of Equation A.6 and substitute in Equation A.1 we get:

$$\frac{\partial(\text{net}^{o}_{pk})}{\partial w^{o}_{kj}} = \frac{\partial}{\partial w^{o}_{kj}}\left(\sum_{j=1}^{L} w^{o}_{kj} i_{pj} + b^{o}_{k}\right) = i_{pj} \qquad \text{(A.9)}$$

Substituting this back into Equation A.6 we obtain:

$$\Delta w_{kj} = (ypk - opk) f^{o\prime}_{k} i_{pj} \qquad \text{(A.10)}$$

We can now introduce a constant η, called the learning rate, which can control how fast the weights change. Therefore, the new weights are calculated by:

$$w^{o}_{kj}(t+1) = w^{o}_{kj}(t) + \eta \Delta w^{o}_{kj}(t)$$

$$= w^{o}_{kj}(t) + \eta (y_{pk} - o_{pk}) f^{o\prime}_{k} i_{pj} \qquad \text{(A.11)}$$

where t represents the iteration number.

If we take the activation function to be the sigmoid function, then the derivative is:

$$f^{o\prime}_{k} = \frac{1}{2}(1 - o^{2}_{pk}) \qquad \text{(A.12)}$$

Therefore equation A.11 becomes:

$$w^{o}_{kj}(t+1) = w^{o}_{kj}(t) + \eta(y_{pk} - o_{pk})\frac{1}{2}(1 - o^{2}_{pk})i_{pj} \qquad \text{(A.13)}$$

Alternatively, we can make the update equation independent of the output function by defining a factor:

$$\delta^{o}_{pk} = (y_{pk} - o_{pk})f^{o'}_{k}$$

$$= \delta_{pk} f^{o'}_{k} \qquad \text{(A.14)}$$

$$\therefore w^{o}_{kj}(t+1) = w^{o}_{kj}(t) + \eta\delta^{o}_{pk} i_{pj} \qquad \text{(A.15)}$$

which is the equation to update the weights in the output layer.

C-2 Update of Hidden Layer Weights

Since the outputs of the hidden neurons are not known, the weights of the hidden neurons must be updated as follows:

$$\Delta w^{h}_{ji} = -\eta \frac{\partial E_{p}}{\partial w^{h}_{ji}} \qquad \text{(A.16)}$$

where the superscript h indicates hidden layer quantities, and η is the learning rate.

$$\Delta w^{h}_{kj} = -\eta \cdot \frac{\partial E_p}{\partial i_{pk}} \cdot \frac{\partial i_{pk}}{\partial net^{h}_{pk}} \cdot \frac{\partial net^{h}_{pk}}{\partial w^{h}_{kj}} \qquad \text{(A.17)}$$

The first term in Equation A.17 can be expanded as:

$$\frac{\partial E_p}{\partial i_{pk}} = \frac{\partial}{\partial i_{pk}}\left(\frac{1}{2}\sum_k (y_{pk} - o_{pk})^2\right)$$

$$= \frac{\partial}{\partial i_{pk}}\left(\frac{1}{2}\sum_k (y_{pk} - f^{o}_{k}(\sum_j w_{kj} i_{pk}))^2\right)$$

$$= \sum_k (y_{pk} - o_{pk}) f^{o}_{pk}{}'(net^{o}_{pk}) w^{o}_{kj}$$

$$= \sum_k \delta^{o}_{pk} w_{kj} \qquad \text{(A.18)}$$

From Equation A.17, term 2 becomes:

$$\frac{\partial i_{pk}}{\partial net^{h}_{pk}} = \frac{\partial f^{h}_{k}(net^{h}_{pk})}{\partial net^{h}_{pk}} = f^{h}_{k}{}' \qquad \text{(A.19)}$$

Term 3 of Equation A.17 becomes:

$$\frac{\partial(net^{h}_{pk})}{\partial w^{h}_{kj}} = \frac{\partial}{\partial w^{h}_{kj}}\left(\sum_{j=1}^{L} w^{h}_{kj} x_{pj} + b^{o}_{k}\right) = x_{pj} \qquad \text{(A.20)}$$

where x_{pj} is the input to the hidden neurons from the previous layer.

Therefore, substituting back into Equation A.17 we get:

$$\begin{aligned}
\Delta w^{h}_{kj} &= \eta\left(\sum_{k} \delta^{o}_{pk} w_{kj}\right) f^{h'}_{k} x_{pj} \\
&= \eta\left(\sum_{k} \delta^{o}_{pk} w_{kj}\right)\left(\frac{1}{2}(1 - i^{2}_{pk})\right) x_{pj} \\
&= \eta \delta^{h}_{pk} x_{pj} \qquad \text{(A.21)}
\end{aligned}$$

$$\therefore w^{h}_{kj}(t+1) = w^{h}_{kj}(t) + \eta \delta^{h}_{pj} x_{pj} \qquad \text{(A.22)}$$

which is the equation to update weights in the hidden layers.