COLOR IMAGE ANALYSIS FOR CEREAL GRAIN CLASSIFICATION

A Thesis
Submitted to the Faculty of Graduate Studies
The University of Manitoba
in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

by

Xiang Yang Luo

Department of Biosystems Engineering
University of Manitoba
Winnipeg, Canada
R3T 5V6

0-612-23630-7

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION

COLOR IMAGE ANALYSIS FOR CEREAL GRAIN CLASSIFICATION

by

XIANG YANG LUO

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Dedicated

to my beloved parents

ABSTRACT

Images of individual kernels and bulk-grain samples for five grain types (Canada Western Red Spring (CWRS) wheat, Canada Western Amber Durum (CWAD) wheat, barley, rye, and oats) from 20 different growing regions across western Canada were acquired. Images of individual CWRS wheat kernels were also acquired for six damage types (broken, mildewed, grass-green/green-frosted, black-point/smudged, heated, and bin/fire-burnt). Morphological and color features were extracted to identify different grain types and damage types (for CWRS wheat only) using statistical and neural network classification methods with different selected feature models (morphological, color, and combined).

For the classification of different types of individual kernels, combining morphological and color features in the feature model improved the classification accuracies over using morphological or color features alone. A non-parametric (k-nearest neighbor) statistical classifier with a feature set of 15 morphological and 13 color features selected using SAS STEPDISC and DISCRIM procedures gave the best results. The average classification accuracies were 98.2, 96.9, 99.0, 98.2, and 99.0% for CWRS wheat, CWAD wheat, barley, rye, and oats, respectively, when using three different training and testing data sets. Similar classification accuracies were achieved using a neural network classifier with the same features.

For the classification of damaged CWRS wheat kernels, color features were more efficient than morphological features, while combining morphological features with color features improved the classification accuracies over using color features alone. A non-

parametric (k-nearest neighbor) statistical classifier with a selected feature set of 24 color and 4 morphological features gave the classification accuracies of 92.5(healthy), 90.3(broken), 98.6(mildewed), 99.0(grass-green/green-frosted), 99.1(black-point/smudged), 97.5(heated), and 100.0 (bin/fire-burnt)%, when using three different training and testing data sets. Similar classification results were obtained using a neural network classifier with the same features.

For the classification of bulk-grain samples, a selected feature set of 8 color features was used with parametric and non-parametric statistical classifiers, and a neural network classifier. When tested on three different training and testing data sets, set1, set2, and set3, all the tested bulk sample images were correctly classified by the non-parametric classifier, while 5 out of 21 bulk images of CWAD wheat in set 2 were mis-classified as CWRS wheat by the parametric classifier and 3 out of 21 images of CWAD wheat in set 1 were mis-classified as barley by the neural network classifier.

For the classification of bulk CWRS wheat samples from three grades (grade 1, 2, and 3), a selected feature set of 20 color features was used with parametric and non-parametric statistical classifiers, and a neural network classifier. When tested on three different training and testing data sets, the neural network classifier gave the best results with 81.0, 67.7, and 82.5% average classification accuracies for bulk CWRS wheat samples of grade 1, 2, and 3, respectively. However, the classification accuracies varied significantly (23.8% for grade 1, 36.5% for grade 2, and 47.6% for grade 3) with different training and testing data sets, indicating that the color features extracted from bulk-wheat images did not carry sufficient information for differentiating different wheat grades.

ACKNOWLEDGMENTS

I express my appreciation to Dr. D.S. Jayas for his advice, support, and encouragement along the way of my study.

Many thanks to Dr. S.J. Symons (Canadian Grain Commission, Winnipeg) for his help in collecting the grain samples and his role as my advisory committee member.

Thanks also to other members of my advisory committee, Dr. H.D. Sapirstein (Dept. of Food Science), Dr. N.R. Bulley (Dept. of Biosystems Engineering) and Dr. E. Shwedyk (Dept. of Electrical Engineering) for their valuable comments and suggestions to my research.

I acknowledge Prince Rupert Grain Ltd., Natural Sciences and Engineering Research Council of Canada, Agriculture and Agri-Food Canada, and University of Manitoba Graduate Fellowship Committee for financial support of this study.

Appreciations to Dr. P.C. Williams and Mr. P. Morris (Canadian Grain Commission, Winnipeg) for assistance in collection of grain samples and to Mr. Dan Goberdhan (Canadian Grain Commission, Winnipeg) for identifying and grading damaged wheat samples used in this research.

Special thanks to D. Bourns, Messrs. Jack G. Putnam, and M. McDonald for their technical assistance in fabricating the illumination chamber and setting up the imaging system.

I also thank J.L. Hehn, M. Nair, P. Shatadal, and S. Majumdar for their help and cooperation. Special thanks to M. Koutis for acquiring grain images.

Finally, I wish to thank my wife, Bei Jiang, for her love, support, and encouragement throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

B-P	Back-propagation
CCD	Coupled charge device
CM	Center of mass
CWRS	Canada Western Red Spring
CWAD	Canada Western Amber Durum
FOV	Field of view
HSI	Hue-saturation-intensity
MA	Minor axis
MNN	Multilayer neural network
MER	Minimum enclosing rectangle
NN	Neural network
NTSC	National Television System Committee
PA	Principal axis
RGB	Red-green-blue
SMER	Standard minimum enclosing rectangle

I INTRODUCTION

Canada, as one of the major grain growing countries in the world, produced annually an average of 55 Mt (million tonnes) of grains and oilseeds worth about $ 6 billion during the years from 1983 to 1992 (Canada Grains Council (CGC), 1994). These grains were collected, stored, and distributed for domestic consumption (30%) or exported (70%). In the current Canadian grain handling system, grain is delivered from farms by truck to primary elevators, transferred to terminal elevators by train for distribution and sale. At elevators, grain is bulked by type and shipped according to CGC set grades meeting customers' specifications. During transport from farm to customer, information on grain quality is needed at different handling stages to allow blending of the type and grade to maintain grain quality and direct proper grain handling operations (receiving, cleaning, binning, and shipping). Currently, visual inspection is used to assess quality rapidly , with protein, oil, and moisture determined objectively by near infrared reflectance (NIR). The visual process is by nature subjective and tedious. An objective, rapid and reliable automatic grain inspection and grading system would be beneficial to the grain industry.

In conjunction with imaging, image storage, and pattern recognition techniques, image analysis is capable of extracting various image features (shape, size, color, texture, and brightness) of objects, and performing task-relevant analysis and interpretation with precision, objectivity, and speed. It offers an attractive potential tool for the automation of the grain inspection and grading processes in the grain industry.

Although substantial efforts have been made on applying image analysis for automatic information acquisition of the content and quality of grain samples in the last decade (see **Chapter II**), many of the special needs and problems involved in the commercial application are still unsolved and commercial computer vision systems for grain inspection and grading are not yet available (AgroVision AB (S-223 70 Lund, Sweden) has developed a computer vision system to classify wheat, barley, oats, rye, and triticale, but to the best of the author's knowledge, its performance is not reported in the literature).

At the current stage of development, an image analysis system is more realistic for automated control of grain handling systems rather than for automated grain grading. For instance, an image analysis system could be installed in a terminal at point of receipt to identify the grain type during a rail car unloading for directing the machinery to transfer the grain into a bin of like class and grade. The requirement for the image analysis system is to rapidly identify the major grain types with 100% accuracy. Most of the previous work dealing with identifying different grains was based on the analysis of individual kernel features, which requires kernels to be presented to the camera in a scattered or non-touching manner or one kernel at a time. This kernel positioning process was mainly performed manually. Although some sample presentation devices have been developed (Keefe and Draper 1988; Casady and Paulsen 1989; Murray 1993), and an algorithm for separating contiguous grain-kernel image-regions has been proposed (Shatadal et al. 1995a), they were not always effective for different grain types. For example, the device developed by Murray (1993) was built specifically for canola and the algorithm of Shatadal et al. (1995a) gave higher failures in separation for oats than for other grain types. In addition, the kernel separation process and the single-kernel-feature based classification algorithms are usually too slow (e.g. the disconnect algorithm of Shatadal et al. (1995a) took 20 min for a typical image of 25-50 kernels) for practical use. In the grain industry, a railcar containing 80-100 tonnes of grain is unloaded in less than 6 min. If the content identification can be done using features of bulk grain samples, the processing speed should be much faster. To date, there are no reports on the use of color features of bulk grain images for identifying different grain types.

Another potential application of image analysis is in grain cleaning section at terminal locations. An image analysis system can be used to monitor cleaner performance and provide information for adjustment of the cleaning machines for optimal cleaning of grain. To determine the cleaning performance, the constituents of the grain samples (different types of grains, dockage and other foreign materials) before and after the cleaning have to be identified correctly. Most of the early studies in classifying different grains using image analysis used small size and carefully cleaned samples. High (>95%) classification accuracies among cereal grains have been reported using morphological and reflectance features (Sapirstein et al. 1987, Sapirstein and Bushuk 1989). It was hypothesized that the classification accuracy might be reduced if tested on large commercial samples collected from different growing regions.

The application of image analysis for grain grading is a greater challenge. In Canada's current grading system, grain is graded based on the five principal grading factors established by the Canadian Grain Commission: test weight, varietal purity, soundness, vitreousness, and maximum limit of foreign material. Of these, test weight, as the only objectively determined factor, cannot be determined by image analysis, while the other four factors, visually determined by trained personnel, are difficult to specify precisely in quantitative image features. However, the research on relating various grain visual features (size and shape, color, and texture) with grain species, classes, varieties, damage status, and impurities would be beneficial in the development of an objective and quantitative method for grain grading. While considerable studies have been done on using image analysis to discriminate wheat classes and varieties (see **Chapter II**), few work has been reported on using image analysis to identify different types of damaged grain kernels.

Color is an important visual attribute of grains used in grain inspection and grading. Different grains and their varieties are commonly characterized according to grain color, and certain degrading factors like grass-green, bin-burnt, mildewed, and fungal-damaged are expressed as discoloration. The use of color increases the information content for grain image analysis. However, most of the previous research has been focussed on using morphological (size and shape) features to characterize different grains and their varieties. The work dealing with the use of color features for cereal grain image analysis was only reported by Neuman et al. (1989a, 1989b) for classifying wheat classes and their varieties using a limited set of color features. The main reason behind this lack in research perhaps is that color information extracted from images is usually variable and unreliable due to the illumination variations existed in common light sources. A consistent illumination system is essential for color grain image analysis. So far no work has been reported on designing and calibrating illumination systems for color grain image analysis.

The objectives of this research were to evaluate color as a component for grain classification by

1. testing the hypothesis that surface color features of bulk grain samples can be used for rapid identification of different cereal grains (i.e., Canada Western Red Spring (CWRS) wheat, Canada Western Amber Durum (CWAD) wheat, barley, oats, and rye);
2. testing the hypothesis that surface color features of individual grain kernels can be used to improve the classification accuracy obtained using the morphological features alone;
3. testing the hypothesis that color features of individual grain kernels can be used for identification of healthy and some types of damaged wheat kernels (e.g., broken, grass-green/green-frosted, bin-/fire-burnt, black-point/smudged, heated, and mildewed);
4. testing the hypothesis that a neural network classifier is more efficient and adaptable than statistical classifiers in classifying different types of cereal grains using combined features (morphological and color features).

II REVIEW OF LITERATURE

2.1 Overview

Although established about 30 year ago, image analysis did not become a practical technique widely used in industries until the early 80's, when substantial advances had been made in the related technique areas especially in computer and imaging techniques. Its applications now can be found in broad areas of produce inspection, process guidance (robotic vision), and scientific research. Commercial image analysis systems are already common in many industries, such as automotive, electronics, and manufacturing (Ballard and Brown 1982; Gonzalez and Safabakhsh 1982; Haralick and Shapiro 1992).

Applying image processing technique to the agri-food industry is challenging. Unlike other industrial objects of defined size, shape, color, and texture, the objects or work-pieces involved in the agri-food industry usually demonstrate a natural variability, which requires that image processing systems must be sufficiently flexible and robust to cope with this variability (Kranzler 1985; Sarkar 1986; Tillet 1991). In addition, processing speed demands in most food processing applications are very high, needing specialized image processing hardware and software.

Image analysis systems began to appear in the agri-food industry in significant numbers in the early 80's (Shaw 1990). The initial use was limited to some simple sorting and inspection tasks. This application was expanded very rapidly in the next few years. In 1989, 12% of the installed image analysis systems were being installed by food processors

(Novini 1990). Not only the number of installed image analysis systems increased, the scope of application was also expanded to almost every aspect in food processing inspection, from raw material grading to final product and packaging inspection. The application of image analysis systems has eliminated the tedious and inefficient manual inspection tasks in the agri-food industry (Novini 1990). To food processors, the application of image analysis systems is no longer a luxury but a necessity to keep and increase the competition abilities of their products in the market. Currently, the need for image processing systems in the agri-food industry is still high. According to the estimation by Nello Zuech (Vision Systems International, 3 Milton Drive, Yardley, PA 19067), the total market for image analysis systems in the agri-food industry is about US$581 million, but only about 465 units valued at US$57 million have been installed.

From a technical point of view, the application of image processing in the agri-food industry is still at its early stage. "Specified purpose" and "lack of color" might be the main limitations in the applications. Most of the installed image analysis systems are 2-D monochromatic or black and white systems with a resolution of 128 x 128 or 512 x 512 pixels. They are based on PCs (personal computers) with a 80286 or 80386 processor. Limited by the computing speed of the PCs, most of the systems use very simple image processing techniques (Tillet 1991), and many systems use specialized hardware or chips to increase the inspection speed. These systems are successful only under constrained conditions for specific applications. As high speed microcomputers (80486, Pentium) with reasonable and continuous lower prices have become commercially available, image analysis algorithms can be implemented in software rather than custom hardware, giving more flexible and adaptable applications. Color image processing systems began to emerge in the agri-food industry in the early 90's, due to the advance in solid-state color imaging sensors as well as the increased computing speed of microcomputers. Although research in these areas has grown rapidly and substantially in the recent years, the adoption of both generic and color image processing systems to food processing are very few. There are still a lot of generic problems to be overcome (Tillet 1991). Most of the developments are still being studied under laboratory conditions.

The application of image analysis for grain inspection and grading has not as yet reached the commercial stage. The main obstacle is the difficulty in quantifying the major grading features used in the current inspection and grading system in terms of various image features (size, shape, brightness, color, and texture). In the last decade, however, considerable efforts have been made on using image analysis for automatic information acquisition on the content and quality of grain samples. The following two sections review the previous work specifically for identification and classification of cereal grains (hereafter grains refers to cereal grains) (**Section 2.2**) and some applications of color image analysis in the agri-food industry (**Section 2.3**).

2.2 Identification of Cereal Grains Using Image Analysis

The major studies in this area can be found in a review by Sapirstein (1995). Most of the published research has been focused on using morphological features to identify different cereal grains and their varieties, while very limited work has been reported on using color features.

Morphological features were found effective in distinguishing different cereal grains by several researchers. Brogan and Edison (1974) successfully classified wheat, barley, oats, rye, soybeans, and corn with an overall accuracy of 98%, using a recursive learning algorithm. Sapirstein et al. (1987) extracted a set of morphological features including kernel length, width, area, aspect and thinness ratios, contour length and normalized central moments to classify among wheat, oats, barley, and rye kernels, using a linear discriminant model. For a sample size of 1160 kernels (half for training and half for testing), the classification accuracies were 100.0, 99.3, 100.0, and 96.5%, for HRS wheat, barley, oats, and rye, respectively. Similar classification results were also obtained when using an optimal feature set of four, selected by step-wise discriminant analysis. In a later study, Sapirstein and Bushuk (1989) tested the similar features on a larger and randomly selected sample of 2766 kernels (1366 for training and 1400 for testing). The classification accuracies were 98.4, 93.7, 78.3, and 98.0%, for HRS wheat, barley, oats, and rye, respectively, with a significant drop for oats. The results suggest that a large and representative sample set is critical for deriving a robust and reliable classification model. In the same study, they demonstrated that by incorporating the mean reflectance of kernels into the feature set, the classification accuracies were significantly improved to 99.2, 95.7, 95.3, and 98.3%, for HRS wheat, barley, oats, and rye, respectively. In testing an algorithm developed for disconnecting touching grain kernels, Shatadal et al. (1995b) reported classification accuracies of 98.5, 94.5, 92.6, 90.7, and 95.2%, for HRS wheat, CWAD wheat, barley, oats, and rye, respectively, using small sound grain samples and a set of morphological features, similar to the one used by Sapirstein and Bushuk (1989).

Using image analysis to discriminate wheat classes and varieties, Keefe and Draper (1986) tried to identify 5 U.K. wheat cultivars using size and shape features. The classification accuracies were not reported in the literature. In a later study, Keefe (1992) reported a semi-automatic image analysis system for wheat grading. When tested for identifying twenty U.K. wheat varieties using the 33 measured and 36 derived morphological features, the classification errors were between 32.9 to 65.8%. Similarly, Zayas et al. (1985, 1986) extracted morphological kernel features to differentiate among different American wheat classes and varieties. Using pair-wise discrimination methods, they achieved the average classification accuracies of 77% and 85%, respectively in discriminating among wheat classes and among varieties in a same wheat class. These early studies, however, had a major limitation that grain kernels had to be placed manually in a specific orientation for imaging and a single kernel per image was required. This drawback was overcome in the work conducted by Neuman et al. (1987) and the later studies by other researchers. Neuman et al. (1987) computed plan-form spatial shape features and Fourier descriptors of kernel perimeters from silhouette wheat kernel images to discriminate Canadian wheat classes and cultivars within classes. Using a pedigreed sample size of 576 kernels from 14 wheat cultivars of 6 wheat classes, they found that CWRS and CWAD wheat kernels were the most easily differentiated classes, while considerable confusion existed among CWRW (Canada Western Red Winter), CWSWS (Canada Western Soft White Spring), CPS (Canada Prairie Spring), and CU (Canada Utility) wheat classes. Discriminant analysis of varieties within classes gave inclusive results with classification accuracies ranging from 15 to 96%. Similar studies were also reported by Symons and Fulcher (1988a, 1988b) on determination of

Eastern Canadian wheat kernel morphological variation by digital image analysis and Barker et al. (1992a; 1992b, 1992c, 1992d) on use of different morphological features for the discrimination of Australian wheat varieties. Despite different morphological features and different classification methods being used by different researchers in the different studies, unsatisfactory results having a large range of classification errors were usually obtained, indicating the incapability of morphological features in differentiating among different wheat classes and varieties.

In an attempt to increase the information content, Chen et al. (1989) used a laser range finder to acquire a cross-section profile of kernels. The inclusion of the features extracted from the cross-section profile to the plan-form morphological features, extracted from the 2-D images acquired by a camera, improved the classification rates. They reported mis-classifications of 8-12% among different wheat classes and 20-26% among different wheat varieties within the same class. However, the high cost and the complexity in manipulating the system made the method less attractive.

The use of color image analysis for identifying different wheat grain classes and varieties was reported by Neuman et al. (1989a, 1989b). The mean red (R), green (G), and blue (B) pixel reflectance features of individual wheat kernels were evaluated for identification of kernels as to one of six wheat classes grown in Western Canada. In general, the red, white, and amber colored wheat types were well separated, while some confusion existed between certain red kernel types. On average, the pair-wise trials gave 88% correct varietal classification. Correct classification rates for individual varieties varied from 34 to 90%. They concluded that color features could assist or facilitate discrimination and identification of contrasting wheat classes.

Multivariate discriminant analysis was used to distinguish between wheat and non-wheat, and between weed seeds and stones in the non-wheat part of a sample (Zayas et al. 1989). With success in identifying wheat and weed seeds, unsatisfactory results were found for identifying stones in the samples.

Work on identifying damaged kernels in wheat samples was reported by Thomson and Pomeranz (1991). They modified the laser scanning system developed by Chen et al. (1989) to acquire 3-D images of wheat kernels. Using the extracted morphological features, they correctly identified 89% of the sprouted and 83% of the un-sprouted wheat kernels. In the same study, they also used the system to classify two American wheat varieties with 92 - 94% correct scores.

2.3 Applications of Color Image Analysis in the Agri-food Industry

The applications of color image analysis in the agri-food industry have been focused mainly on sorting or grading agricultural products and identifying or distinguishing plants and plant parts.

Wigger et al. (1988) applied color image analysis to detect and classify fungal-damaged soybeans. Individual soybeans were correctly classified into one of five categories - healthy, with 98% accuracy, and those showing symptoms of infection due to *Phomopsis sp.*, *Alternaria sp.*, *Fusarium sp.*, and *Cercospora kikuchii* with 77 to 91% accuracies. Intensity and ratios of red to blue, red to green, and green to blue were used as features for discrimination. Shyy and Misra (1989) used the color information combined with other derived features to evaluate the quality of soybeans. Damaged soybeans were correctly classified with an accuracy of 85%. Casady et al. (1992) developed a trainable algorithm on a color image analysis system for inspection of soybean seed quality. The algorithm correctly classified asymptomatic soybean seeds, seeds infected by *C. kikuchii*, seeds that belong to a group used by the Federal Grain Inspection Service called "seeds of other colors", and "materially damaged seeds" with 94, 97, 85, and 96% accuracy, respectively. The variables used for classification were color chromaticity coordinates and seed sphericity.

Miller and Delwiche (1989) developed a color machine vision system to inspect and grade fresh market peaches. They used diffuse lighting and normalized luminance to reduce the red, green, and blue inputs to two-dimensional chromaticity coordinates. Peach color was compared to standard peach maturity colors. Machine maturity classification agreed with manual maturity classification in 54% of the test samples, and was within one color standard in 88% of the tests. Shearer and Payne (1990) used a color machine vision system to sort bell peppers according to color and damage. Red-green-blue pixel intensity values were mapped to one of eight possible hues and the relative hue distributions of pixel in six orthogonal views were calculated and used as color quantitative variables. An accuracy of up to 96% was achieved for grading bell peppers by color.

Precètti and Krutz (1993a; 1993b; 1993c) developed a PC-based real-time color classification system to perform corn husk deduction measurements. They segmented color images of corn cobs with husks into five color classes (e.g. background, dried husk, green husk, red cob, and yellow kernels) and calculated the husk to corn surface ratio which was linearly related to the husk mass to corn mass ratio with a correlation coefficient of 0.95. The machine vision system gave measurements with 1% variation, while manual measurements yielded a variation of approximately 4%.

Slaughter and Harrell (1987) analyzed images for chromaticity and intensity as a means of distinguishing between oranges on a tree and background foliage. An NTSC color decoder was used to transform the original composite video signal recorded on a video tape into RGB video signals. A color look-up table was constructed to specify the RGB color space into 32,768 possible colors, and used to segment an image by assigning each pixel in the image a binary status denoting whether the pixel fell within derived hue and saturation thresholds. When tested with three images of natural orange grove scenes, 93, 45, and 85% orange pixels were correctly classified, respectively.

Thomas et al. (1988) applied color image processing technique to 35-mm color slides of canopy-soil combinations for distinguishing plants from their natural background. They transformed the color images into grey-scale images of each primary color by viewing each slide through separate red, green, and blue colored filters mounted on a video camera. By subtracting the red image from the green or blue image, a grey-scale image resulted with perceptually brighter leaf pixel and darker soil pixel. Compared to the human visual inspection procedure, the image processing procedure gave better results. However, the application was limited by the slow processing speed and the system cost.

Shearer and Holmes (1990) identified plants by color-texture characterization of canopy sections. Three color co-occurrence matrices were derived from image matrices for each color attribute: intensity, saturation, and hue. Eleven texture features were calculated from each of the co-occurrence matrices and used in a discriminant analysis model to identify plants. Overall classification accuracy of 91% was achieved when this model was used to identify seven common cultivars of nursery stock.

Humphries and Simonton (1993) used color as well as geometric features to identify geranium cutting features such as petioles, main stem, leaf blades, and growing tip. Correct classifications for leaf, petiole, and main stem material were 97, 95, and 93%, respectively.

Woebbecke et al. (1994) analyzed color slide images of weeds among various soils and residues for the chromatic coordinates r, g, and b (Gonzalez and Woods 1992). Indices of r-g, g-b, (g-b)/|r-g|, and 2g-r-b and a modified hue were derived and tested for identifying weeds from soils and residues. It was reported that the modified hue, 2g-r-b, and green chromatic coordinate distinguished weeds from a non-plant background (0.05 level of significance) better than other indices.

Other applications of color image analysis found in the literature were characterizing germplasm properties (Panigrahi and Misra 1989), inspecting apples, mushrooms, and potatoes (Morrow et al. 1990), and sorting wood into color groups (Haney et al. 1994).

III IMAGE ACQUISITION

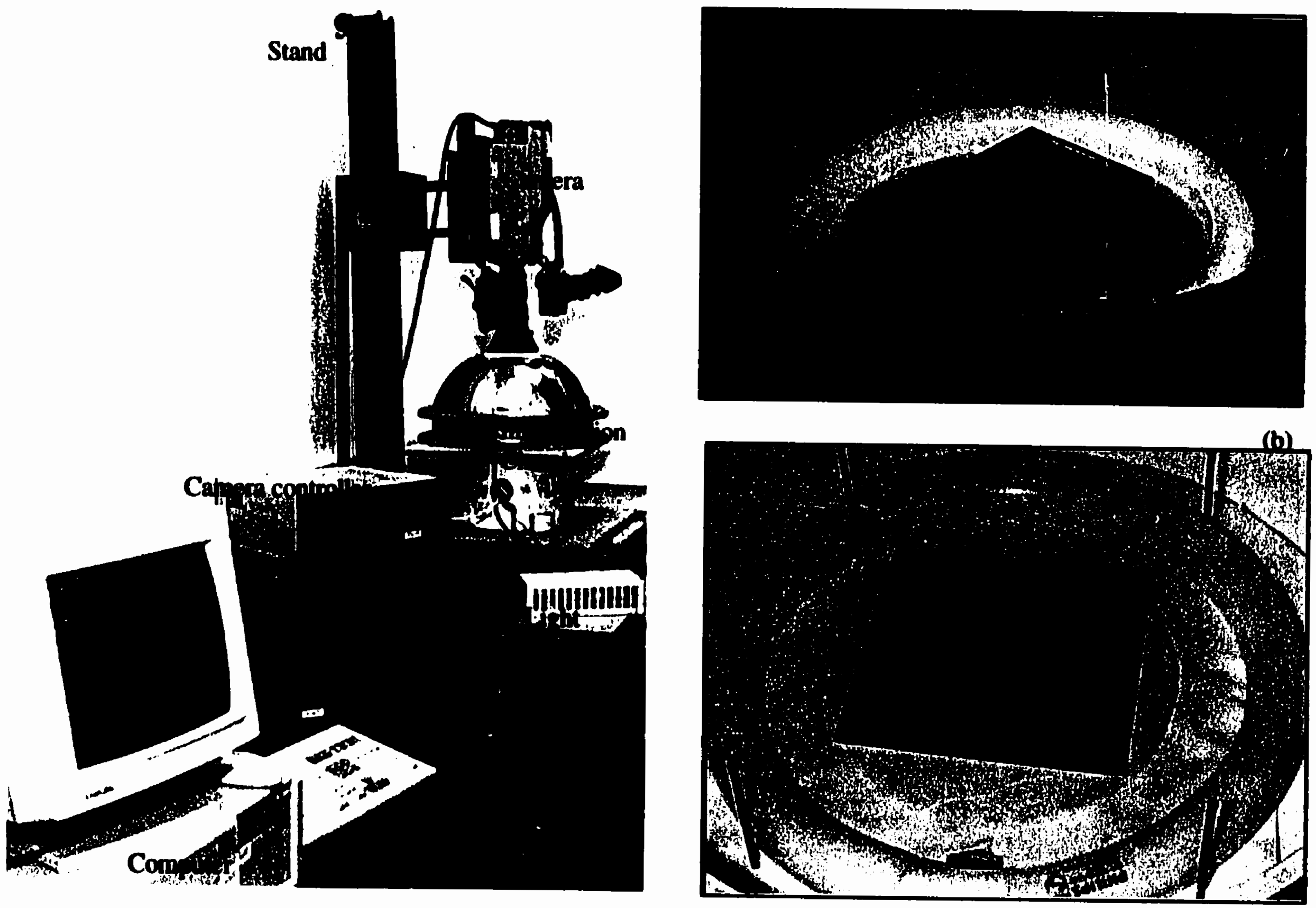
Image acquisition is the first and probably the most important step in image analysis applications. Proper integration and calibration of an imaging system are essential for high quality image acquisition. Selecting representative grain samples is crucial for the generality of the analysis results. This chapter addresses the imaging system used in the research in **Section 3.1**, the illumination design in **Section 3.2**, the system calibration in **Section 3.3**, and the grain sample collecting and sampling technique in **Section 3.4**.

3.1 Imaging System

A typical image analysis system basically consists of a video camera for acquiring images of the objects of interest, a light source for providing proper illumination for the imaging, a frame-grabber for digitizing the acquired images, and a computer with proper software for storing, analysing, and understanding the digitized images. **Fig 3.1(a)** shows the image analysis system used in this research.

3.1.1 Hardware

The hardware consisted of a 3-chip CCD (coupled charge device) color camera (DXC-3000A, Sony, Japan) with a zoom lens of 10-120 mm focal length (VCL-1012BY), a camera control unit (CCU-M3, Sony, Japan), a color monitor (PVM-1342, Sony, Japan), a personal computer (PC) (386/20MHz, UNISYS), a color frame grabbing and processing board (DT2871 & DT 2858, DATA Translation, Marlboro, MA), an optical disk drive (SMC-S502, Sony, Japan), and a diffuse illumination chamber.



(a) (b) (c)

Fig 3.1 The image analysis system.
(a) System set up. (b) Bulk sample imaging. (c) Individual kernel imaging.

Mounted over the illumination chamber on a stand which provided easy vertical movement, the camera captured images of objects in the illumination chamber. The NTSC (National Television System Committee) composite color signal from the camera was converted by the camera control unit at a speed of 30 frames per second into three parallel analog video signals, namely red (R), green (G), and blue (B), corresponding to the three NTSC color primaries, and a sync signal. The camera control unit also enabled selectable manual/automatic iris and video signal gain control and white/black balance of the camera (manual iris control was used in this research to adjust the illumination level, see **Section 3.3.5**). The frame grabber installed in the PC digitized the RGB analog video signals from the camera control unit into three 8-bit 512 x 480 digital images and stored them in three on-board buffers. The digital images were then sent to the color monitor for on-line display and transferred to the networked optical disk for storage.

3.1.2 Software

The image acquisition software was developed on the PC in C language using the supporting subroutine library (Aurora, Data Translation, Marlboro, MA) for the frame grabber DT2871. It included an illumination standardizing program (**Appendix A: litadj.c**), an image acquiring program (**Appendix A: xvsave.c**), and two of the imaging system tuning programs. The illumination standardizing program monitored the illumination level inside the illumination chamber by continuously calculating the average RGB grey levels over a small central region (50 x 50 pixels) of the camera's field of view (FOV) and graphically displaying them on the computer's screen. By adjusting the iris control knob and performing a black/white balance, the RGB grey levels were brought to pre-determined values. The image acquiring program enabled the saving of an image to a computer file with the selections of image size, image part (window), and color mode (Black/white, or RGB, or HSI (hue, saturation, and intensity)). The image analysis software was developed on another (Pentium/166 MHZ) in C language under the DOS environment. It was independent of the frame grabber. The detailed functions of the analysis software will be given in **Chapter IV** and **V**.

3.1.3 System model

The following mathematical model is commonly used to describe a color image analysis system (Ballard and Brown 1982):

$$C_i(x,y) = \int I(\lambda)\ O(\lambda,x,y)\ S_i(\lambda)\ d\lambda \tag{3.1}$$

where:

i	= index spanning the three color channels (red, green, and blue),
x, y	= space coordinates,
λ	= light wavelength,
$C_i()$	= output signal of color channel i,
$I()$	= light energy incident upon object surface,
$O()$	= spectral reflectance of object surface, and
$S_i()$	= spectral response of camera sensor for color channel i.

This is an ideal model under the assumptions: (1) the illumination is uniform over the FOV and constant with time, (2) the lens system does not introduce any distortion over the FOV and the transmittance is constant with light wavelength, (3) the spectral responses of the sensors are uniform over the sensor's array, (4) the image digitization does not introduce any error. These assumptions, however, are usually not true in reality, due to less than perfect optical and electrical components in an image analysis system. For example, illumination is usually non-uniform over the FOV and variable with time in both intensity and color, due to changes in supply voltage, lamp deterioration, and ambient temperature. There is also always a sensitivity variation among sensing cells of an imaging sensor array. Therefore an object image, as captured by an image analysis system, is not only a function of the spectral properties of the object surface (which are of interest), but also is a function of the illumination spectral distribution and the camera spectral response (determined by the lens transmittance and the sensor's spectral response), as described in **Equation 3.2**.

$$C_i(x,y,t,v) = \int I(\lambda,x,y,t,v)\ O(\lambda,x,y)\ S_i(\lambda,x,y)\ L(\lambda,x,y)\ d\lambda \tag{3.2}$$

where:

t = time variable,
v = power voltage of light source, and
$L()$ = lens transmittance.

As a result, images of an object taken at different times or at different locations of the FOV may appear differently in either size and shape or color and brightness. This makes comparison and analysis of object images difficult, especially when color or reflectance information is involved. The imperfect factors in a practical image analysis system may not be totally eliminated by any means, however, they can be minimized by proper system integration and tuning or if necessary by software correction. Their effects on imaging accuracy should be closely examined before taking any images.

3.2 Illumination Design

Illumination plays an important role in image acquisition. To acquire an object image carrying accurate information of the spectral properties of the object surface, the illumination upon the object must be uniform over the FOV, consistent with time, and shadow free (diffused). Uniform diffused illumination can be achieved by proper arrangement of light sources. However, illumination usually varies in intensity and color with time due to changes in power voltage, ambient temperature, and lamp deterioration. This inconsistency in illumination may be eliminated by adjusting the illumination each time an image is acquired, but this may become impractical in industrial applications of image analysis. Consistent illumination over an 8 h working shift is usually desired.

To select an acceptable light source for the image analysis system, three types of commonly used light sources: incandescent, halogen, and fluorescent lamps were evaluated in the following aspects: (1) sensitivity to lamp voltage variations, (2) stability with time, and (3) uniformity over FOV.

3.2.1 Light sources

The incandescent light sources were eight 40-W bulbs (Soft White, GE Lighting Canada, Mississauga, ON) with a rated voltage of 120 V. The halogen light sources were eight 46-W bulbs (Power Par 20, Duro-Test Co., Fairfield, NJ) with a rated voltage of 122 V. The fluorescent light source was a 30.5-cm diameter, 32-W circular lamp (FC12T9/CW, Philips, Singapore) with a rated voltage of 120 V.

3.2.2 Illumination chamber and power supply

An illumination chamber was designed and developed to provide uniform diffuse illumination over the FOV. For testing the incandescent and halogen light sources, the eight bulbs were oriented vertically in a ring around a round object plane of 150 mm in diameter in the centre of the illumination chamber. For testing the fluorescent light source, the lamp was placed around and just below the surface of the object plane (**Fig 3.1(b)** and **(c)**). As a light diffuser, a steel bowl of approximately 400-mm diameter, painted white and smoked with magnesium oxide on the inside was inverted and covered the light bulbs and the object plane such that the object plane was only exposed to the diffused light. The steel bowl had a 125-mm diameter opening at its top (in the inverted position) through which the camera acquired images.

A voltage regulator (Sola Canada Inc., Toronto, ON) supplied stable AC power (±0.1 V) to the light sources and the voltage to the lamps was adjusted by a variac. The fluorescent lamp was also tested with a light controller (FX0648-2/120, Mercron, Richardson, TX) incorporated in its power supply. The light controller automatically detected the illumination level in the illumination chamber using a photodiode light sensor and adjusted the AC frequency to the lamp to maintain a stable level of illumination under varying conditions. The frequency of the AC power output of the controller varied between 140 kHz at the minimum light levels to 60 kHz at full power.

3.2.3 Test I: sensitivity to lamp voltage variations

The lamps were turned on and the illumination was standardized (see **Section 3.3.5**) at the rated lamp voltage V_R after a 3 h warm-up time. Then the lamp voltage was gradually changed from (V_R - 1.0 V) to (V_R +1.0 V) with a step of 0.1 V by adjusting the variac. At each of the 21 steps, the digital image of a Kodak white card (E152-7795, Eastman Kodak Co., Rochester, NY) was acquired immediately following the voltage adjustment, and the mean R, G, and B grey-level values over a small central area (50 x 50 pixels) were calculated and recorded. The mean R, G, and B values at the different lamp voltages were then divided by the mean R, G, and B values at the rated lamp voltage V_R and defined as the voltagedependent relative intensities, Rv, Gv, and Bv, respectively. The same test was repeated five times for each type of light source and the average Rv, Gv, and Bv of the five tests were plotted versus the lamp voltage.

3.2.4 Test II: stability with time

The illumination was standardized (see **Section 3.3.5**) immediately after switching on the lamps (t_0). The image of the Kodak white card was captured repeatedly, and the mean R, G, and B values over a small central area (50 x 50 pixels) were computed and recorded every 10 min for 8 h. The lamp voltage was maintained at the rated value V_R all the time. The mean R, G, and B values at the different times were then divided by the mean R, G, and B values at t_0 and defined as the time-dependent relative intensities, Rt, Gt, and Bt, respectively. The same test was repeated five times for each type of light source and the average Rt, Gt, and Bt of the five tests were plotted versus time.

3.2.5 Test III: uniformity over FOV

It was impossible to separate the illumination evenness from the effects of the variation due to lens transmittance and the responses among the sensor arrays. Therefore the uniformity over the FOV was examined as a composite result of illumination distribution determined by the configuration of light sources, the lens transmittance, and the sensor responses.

Again, the illumination was standardized (see **Section 3.3.5**) and the image of the Kodak white card was captured. Mean R, G, and B values were calculated for each row (down the image) and each column (across the image) in the image. The row means of R, G, and B signals were then divided by the overall mean R, G, and B values and defined as the row-dependent relative intensities, Rr, Gr, and Br, respectively. Similarly, the column means of R, G, and B signals were divided by the overall mean R, G, and B values and defined as the column-dependent relative intensities, Rc, Gc, and Bc, respectively. For each light-source type, ten images of the same white card with different orientations and viewing regions were acquired and analysed. Average Rr, Gr, and Br and average Rc, Gc, and Bc of the ten tests were plotted versus the row and column numbers, respectively.

3.3 System Calibration

3.3.1 Aspect-ratio

The DT2871 frame grabber installed in the PC converts analog video images into digital images using rectangular pixels, as a result of the horizontal re-sampling in the digitization process. A digitized image is actually a 512 x 480 data matrix. Each element of this matrix corresponds to a rectangular portion of the original analog video image. In other words, a rectangular-pixel digital image has different vertical and horizontal pixel resolutions. The resolutions of the images acquired by the imaging system shown in **Fig 3.1** were 0.20 mm/pixel in horizontal and 0.16 mm/pixel in vertical directions.

The relationship between the vertical and horizontal spacing is described by the aspect-ratio, a ratio of the length to width of the rectangular area in the original analog image represented by a pixel in the digitized image. The knowledge of the aspect-ratio is essential for interpreting image size and shape information in real world dimensions. However, there are no published data of the aspect-ratio value, because it is determined not only by the frame-grabber (digitization), but also by other camera parameters (such as magnification, lens distortion, and etc.). Several methods have been proposed to practically evaluate the aspect-ratio (Toscani and Faugeras 1987; Lenz and Tsai 1987; Ganapathy 1984).

In this research, a Canadian quarter coin was used to determine the aspect-ratio. With the same camera setting (magnification) as used in the grain imaging, four rectangular-pixel digital images of a Canadian quarter coin were acquired with the coin located in the centre of the camera's FOV at 4 different orientations. In each image, the coin region was separated from the background using the segmentation method described in **Chapter IV**, and the numbers of pixel rows and columns, Nr and Nc, required to traverse the coin were calculated. The aspect-ratio is the average pixel row number divided by the average column number.

To investigate the effect of the magnification on the aspect ratio, another group of 4 rectangular-pixel image of the same quarter coin were acquired in the similar way but using a magnification of about 1.13 times larger than the previous. Similarly, the numbers of pixel rows and columns required to traverse the coin were calculated and the aspect-ratio was determined.

For the convenience of image analysis, the rectangular-pixel digital images were transformed to the square-pixel digital images with the knowledge of the aspect ratio, using an algorithm called *pixel filling algorithm* (Castleman 1979):

$$g(x, y) = (1-\alpha)f(x', y) + \alpha f(x'+1, y) \tag{3.3}$$

$$x' = \mathrm{Int}[x/k], \quad \alpha = x/k - x'$$

where: k = aspect ratio,
Int[] = function truncating to integer,
f() = original rectangular-pixel image, and
g() = square-pixel image.

3.3.2 Spatial resolution of square-pixel images

The spatial resolution is needed to relate the pixel dimensions computed from digital images to the real world dimensions in the size feature measurements. Again a Canadian quarter coin was used to get the resolution information of the imaging system. The image of the coin located in the centre of the camera's FOV was acquired and transformed to the square-pixel image using the aspect ratio determined in **Section 3.3.1**. The diameter of the coin was measured to the nearest 0.001 mm using a micrometer. The mean diameter of 23.869 mm, was calculated by averaging four readings at 45 degree intervals around the coin. The spatial resolution was then determined by dividing the coin diameter by the mean value of pixel columns and rows required to traverse the coin. To accommodate the possible changes in the camera's magnification, an image of a same Canadian quarter coin was acquired and saved for the future use of the spatial calibration prior to each imaging session.

3.3.3 Image distortion

Image distortion is a composite result of the imperfect factors in an imaging system, such as camera misalignment (the camera is not vertical to the object plane), lens distortion, and image digitization. The transformation from rectangular pixels to square pixels may also introduce further image distortion. A direct consequence of image distortion is that the size and shape measurements of an object become variant to the location and orientation of the object in the camera's FOV.

To examine the image distortion introduced by the camera misalignment, the lens distortion, and the image digitization, twenty rectangular-pixel images of a Canadian quarter coin were acquired. Four images were acquired with the coin located in each of the upper and lower corners and the centre of the camera's FOV, and the numbers of the pixel rows and columns to traverse the diameter of the coin in the images, Nr and Nc, were calculated as in the case of determining the aspect ratio.

To investigate the effect of the transformation from rectangular pixels to square pixels on the image distortion, the twenty rectangular-pixel coin images were transformed to the square images using the aspect ratio computed from the 4 central images, and the number of the pixel rows and columns to traverse the diameter of the coin in the square images, Nr' and Nc', were calculated as in the case of the rectangular images.

3.3.4 Gamma correction

Gamma correction is universally done on commercial video cameras for the purpose of correct reproduction of light intensity on display devices. The light intensity generated by an image displaying device is usually not a linear function of the applied signal. A conventional CRT (cathode-ray tube) has a power-law response to applied voltage: light intensity produced at the face of the screen is approximately the applied voltage raised to some (typically 2.5) power. The numerical value of the exponent of this power function is colloquially known as gamma. To achieve correct reproduction of light intensity on the display device, the applied signal must be modified by a nonlinear transformation, called gamma correction which is effectively the inverse of the response of the display device. In an NTSC-RGB video camera, the gamma correction is performed by applying the following transfer functions to the tristimulus RGB signals:

$$R' = R_o^{1/\gamma}, \quad G' = G_o^{1/\gamma}, \quad B' = B_o^{1/\gamma}, \quad \gamma = 2.2 \tag{3.4}$$

where:

R_o, G_o, B_o = tristimulus RGB signals normalized in the range of [0, 1],

R', G', B' = gamma-corrected video outputs normalized in the range of [0, 1], and
γ = gamma exponent.

The interest of this research was in the physical color difference in grains instead of displayed images on the display device and linear relationships between the system outputs and the object reflectance were desired, so the gamma correction imposed by the camera should be "removed" or "re-corrected". For some video cameras, this can be done by just simply disabling the gamma correction function. Since the gamma correction was integrated within the camera used in this research, the removal of gamma correction was done in software by applying the following transformations to the digitized gamma-corrected RGB images from the frame grabber:

$$r(x, y) = [R(x, y)/R_{ref}]^{\gamma}$$

$$g(x, y) = [G(x, y)/G_{ref}]^{\gamma}$$

$$b(x, y) = [B(x, y)/B_{ref}]^{\gamma}, \qquad \gamma = 2.2 \qquad \textbf{(3.5)}$$

where:

R(), G(), B() = digitized gamma-corrected RGB signals at (x, y),
R_{ref}, G_{ref}, B_{ref} = digitized gamma-corrected RGB values of white reference,
r(), g(), b() = normalized RGB signals at (x, y), and

The system linearity was examined using a 20-step paper gray scale (Cat 152-7762, Eastman Kodak Co., Rochester, NY) which has a varying reflection density ranging from 0.05 at step 1 to 1.95 at step 20, with an equal difference of 0.10 between two adjacent steps. The illumination was standardized (see **Section 3.3.5**) first. An image of each of the 20 steps in the scale was captured by presenting the corresponding step in the centre of the FOV, and the mean R, G, and B grey-level values over a small central area (20 x 20 pixels) were computed and recorded. These mean R, G, and B values of the different steps were plotted versus step number.

3.3.5 Illumination standardization

A color image of an object, as captured by an image analysis system, is actually a function of the spectral properties of the object surface as well as the illumination spectral distribution and the camera spectral response. The color data extracted from the captured image are therefore device-dependant. When analysing color data, especially when comparing color data taken under different conditions (illumination and cameras), it is necessary to calibrate images to accommodate variations in illumination and camera sensor response. Many color calibration methods has been proposed to map device-dependant color data onto an absolute (device-independent) color system (Hetzroni and Miles 1994; Lee 1988; Gershon and Jepson 1989; Green and Ismail 1990; Tominaga 1992; Brainard and Wandell 1990; Levine 1985; Ballard and Brown 1982), using either the predetermined spectral response of the camera sensors and illumination distribution or the calibration matrices developed by using test color standards with known absolute color coordinates.

The absolute color data were not of interest for this research, since the same camera was used to take all grain images using a fixed camera setting and a standard (consistent and uniform) illumination source. It was assumed that the color data extracted from images taken at different times or from different portions of an image were comparable.

A Kodak white card with 90% reflectance (E152-7795, Eastman Kodak Co., Rochester, NY) was used as the white reference to standardize the illumination level. The lamp voltage was set to the rated value V_R. Then the color image of the white card was acquired, and the mean R, G, and B grey-level values over a small central area (50 x 50 pixels) were computed and used as the illumination-level indicators. By manually adjusting the iris control (the lens aperture) and performing white-balance with the camera control unit, all three values were adjusted to 250 ± 1 ($R_{ref} = G_{ref} = B_{ref} = 250$).

3.4 Grain Samples

3.4.1 Sample sources

The cereal grain samples used in this study were obtained from the Industry Services Division of the Canadian Grain Commission, Winnipeg, Manitoba. For the 1994 growing year, unclean commercial samples of five grain types (**Fig 3.2**) were collected from different growing regions distributed across Western Canada. The grain samples were grouped into seven categories: CWRS1; CWRS2; CWRS3 (wheat grade 1, 2, and 3, respectively); CWAD wheat (grade 1, 2, 3, and 4); barley (grade EX1 and 1); rye (grade 1), and oats (grade unknown). For each category, a composite sample (1000 - 1500 g) was made for each of the growing regions by mixing and sampling the available samples from different farms (stations) within a growing region, using a Boerner Divider. Based on the sample availabilities, twenty composite samples from twenty growing regions (**Appendix B**) were selected for each grain category to represent the climatic and regional variabilities over the Canadian Prairies.

For the identification of damaged CWRS wheat kernels, six types of damaged kernels (**Fig 3.3**) were collected. Samples of three types of damaged CWRS wheat kernels, namely ***broken***, ***grass-green/green-frosted***, and ***black-point/smudge***, were manually picked from the

Fig 3.2 A sample image of five grain types:
CWRS wheat (upper left), durum wheat (upper right), barley (lower left), rye (lower right), and oats (center)

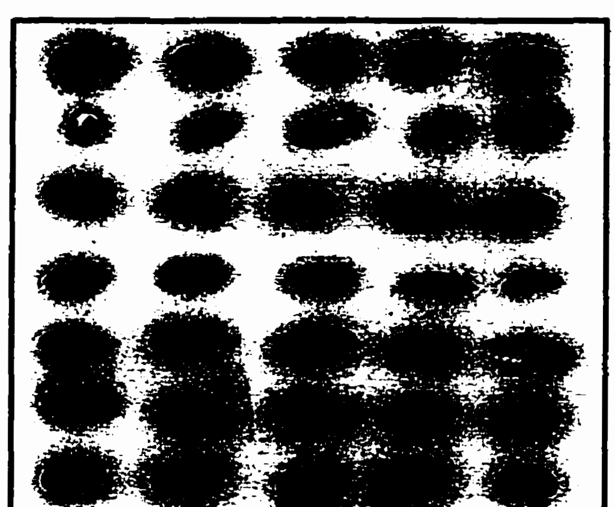


Fig 3.3 A sample image of healthy and six types of damaged CWRS wheat kernels. (From top to bottom rows: healthy, broken, black-point/smudged, grass-green/green-frosted, mildewed, heated, and bin-/fire-burnt.)

unclean commercial samples of CWRS grade 3. Samples of the other three types of damaged CWRS wheat kernels, *mildewed, heated,* and *bin-/fire-burnt*, were created in the laboratory. All the damaged samples were verified by Mr. Dan Goberdhan (Assistant Operations Supervisor, Prairie Region, Industry Services, Canadian Grain Commission, Winnipeg, Manitoba) as being typical of naturally occurring damaged kernels.

The mildewed kernels were prepared by keeping sound CWRS kernels (grade 1), conditioned to 20 ~ 25% moisture content (wet basis), in a sealed plastic bag at room temperature (23 ~ 25°C) for a period ranging from 7 to 21 d, until the mildew damage occurred to the desired extent. The bag was shaken regularly to ensure an uniform development of mildew in the different parts of the bag. The heated kernels were created by keeping sound CWRS kernels (grade 1) in an oven at a temperature of 150°C for a period ranging from 2 to 20 h until the kernels were heated to the desired extent. The bin-/fire-burnt kernels were created by keeping sound CWRS kernels (grade 1) in an oven at a temperature of 200°C for a period ranging from 100 to 120 h until the kernels were heated to the desired extent.

3.4.2 Sampling technique and sample size

For the identification of grain types, 600 - 900 kernels (approximately 225 g in mass) were sampled from each of the 7 x 20 composite samples (20 growing regions for each of the 7 grain categories). The composite grain sample (1000 - 1500 g) was poured into a plastic bag and mixed thoroughly. A sub-sample of about 75 g was then withdrawn by randomly taking grain kernels from different parts of the bag using a scoop. Similarly the second and third sub-samples were obtained from the remaining grain which were re-mixed after the previous withdrawal. The three sub-samples were mixed again by passing them through the Boerner Divider 4 times to give a sample of approximately 225 g.

The sample of 225 g was first split into three replicate samples for bulk sample imaging. Each of the replicate samples was put into a bulk sample container and presented to the camera (see **Section 3.4.3**). The container held the kernels in 2 - 3 kernel deep layers and only the kernels in the top layer appeared in the image. There were 100 - 150 kernels covered in each bulk sample image.

After the bulk sample imaging, the three replicate samples were re-mixed to give a sample of 225 g. From each of the 7 x 20 samples, 300 kernels were randomly picked and imaged in 12 images (25 kernels per image). In total 42 000 grain kernels were imaged in 1680 (12 x 7 x 20) images.

For the identification of individual damaged CWRS wheat kernels, 1000 kernels were collected for each of the 6 damage types as well as the healthy kernels (CWRS grade 1) and imaged in 25 images (40 kernels per image). Totally 7000 kernels were imaged in 175 images. The 1000 *mildewed* kernels consisted of 600 (60%), 280 (28%), and 120 (12%) kernels from three laboratory-conditioned samples graded as grade 2, 3, and feed because of the mildew damage, respectively. The 1000 *heated* kernels consisted of 500 (40%), 300 (30%), and 200 (20%) kernels from three laboratory-conditioned samples graded as grade 3, feed, and sample because of the heat damage, respectively. The 1000 *bin-/fire-burnt* kernels consisted of 500 (50%) and 500 (50%) kernels from two laboratory-conditioned samples identified as bin-burnt and fire-burnt, respectively. The 1000 healthy kernels were randomly picked from a sample of CWRS grade 1. For each of the other three damage types, *broken*, *grass-green/green-frosted*, and *black-point/smudge*, 1000 kernels were picked from samples of CWRS grade 3.

3.4.3 Sample imaging

An operation guide (**Appendix C**) for grain imaging using the image analysis system described in **Section 3.1** was developed and followed in each of the grain imaging sessions to ensure the image quality. It specified the settings of the imaging system and the procedures for illumination standardization and spatial calibration.

For imaging bulk grain samples, each sample (75 g in mass) was poured into a rectangular container made of transparent epoxy fibreglass with inner dimensions of 135 x 100 x 10 mm (**Fig 3.1(b)**). A fibreglass board with dimensions of 135 x 100 mm was used to press the sample in the container so that the sample in the container was held in approximately two-three layers and the sample surface was levelled. Then the container with the sample was placed on the object plane in the illumination chamber in such a position that almost all the surface grain kernels were covered in a full size (512 x 480) image. The color image of the sample was saved in a file and transferred to an optical disk for storage.

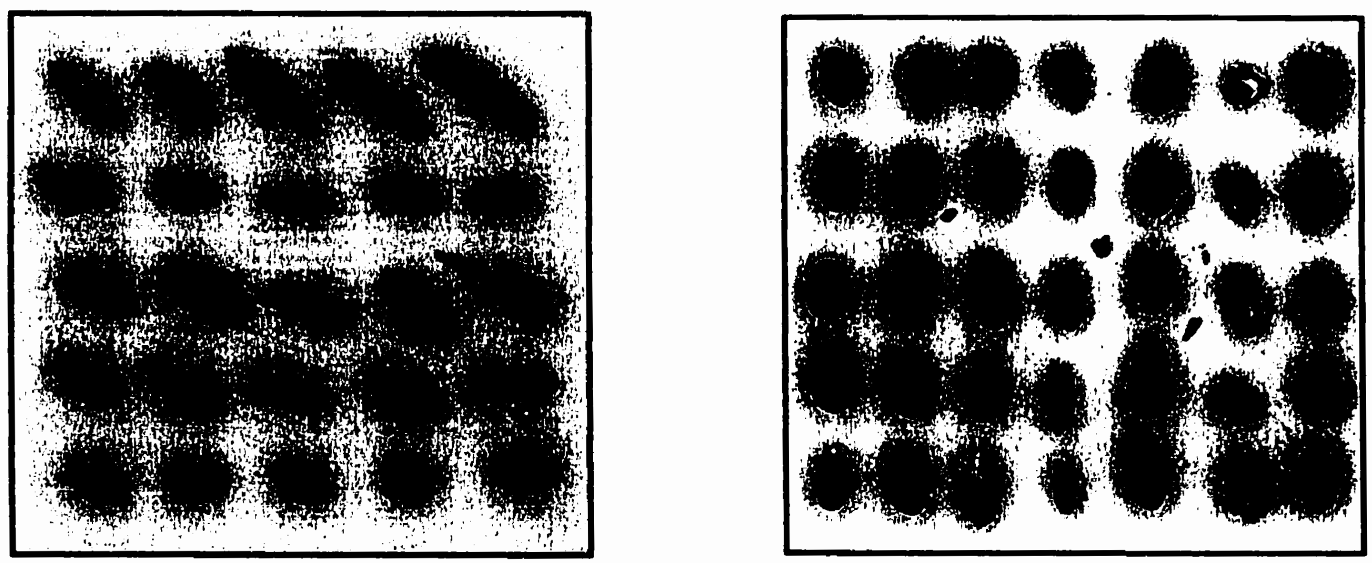
For imaging individual kernels for grain type identification analysis, 25 individual kernels were randomly placed on a black background board in a separated (non-touching) manner and presented to the camera's FOV for imaging (**Fig 3.1(c)**). The position of the kernels was adjusted by moving the background board to being around the centre of the camera's FOV. The image saving program allowed using a mouse to select a proper image window and size to cover all the 25 kernels. A similar procedure was applied to the imaging of damaged CWRS wheat kernels, except that a white background was used and 40 kernels were imaged in one image.

IV IMAGE SEGMENTATION

The purpose of image segmentation is to isolate each individual grain kernel in an image from the background and from each other so that the morphological and color features can be extracted from each of the individual kernels in the image. This chapter illustrates the segmentation algorithms developed and used in this research using the two grain sample images in **Fig 4.1** as examples.

4.1 Thresholding

The first step of image segmentation is to separate objects in a color image from the background by converting the color image into a bi-level image which has only two pixel values: white (255) for the background and black (0) for the objects, or vice versa. This process is called image thresholding and is performed by examining each image pixel and deciding using some criteria whether it belongs to objects or to the background. To threshold a grey level image, the decision is generally made by comparing each pixel value against a fixed number called a threshold. If a pixel value is less than the threshold, the pixel is set to zero; otherwise set to 255. Because a color image consists of three grey-level images, namely red, green, and blue band images, it is quite natural to consider using one of the three color bands for thresholding. Thus, the problem to be solved is to select a proper threshold value and choose a right band.



(a) (b)

Fig 4.1 Test images for segmentation. (a) Kernels of different grain types. (b) Healthy and damaged CWRS wheat kernels and dusts.

4.1.1 Selecting a threshold

Selecting a good threshold is the key for successful image thresholding. Although it is difficult to give a precise definition, a good threshold generally means a threshold value by which the thresholded image has black regions that generally agree with the areas of the objects and white regions that correspond to the background of the image. This definition assumes that the objects are darker than the background and the pixels in the original image are set to zero if their values are less than the threshold and 255 if their values are larger than or equal to the threshold.

Threshold selecting can be done either manually by visually comparing the thresholding results for different threshold values, or automatically using an threshold selecting algorithm. In practice, especially industrial applications, it is usually impossible to manually select a threshold for each image, and a predetermined threshold for all images may not accommodate the intensity variations among images due to the possible changes in illumination. A threshold has to be extracted from each individual image automatically.

Many methods have been developed for automatic threshold selecting (Parker 1994, Gonzalez and Woods 1992), based on the problem being investigated. In this study, an algorithm called *iterative selection* (Parker 1994) was used to select a threshold for a grey level grain image. The algorithm is a recurring search process. Initially, the overall mean grey level of an image is computed as the initial threshold estimate T. The next step calculates T_b and T_o as the average grey levels of the background (pixels with grey level larger than or equal to T) and the objects (pixels with grey level less than T), assuming that the objects are darker than the background, and uses their average as the new threshold estimate: T = (T_b +

$T_o)/2$. The process is repeated until the same value T is obtained on two consecutive iterations, at which point T is considered to be a good threshold for the image; or the number of the iterations is larger than a predetermined value (40), at which point it is considered that there is no region (object) in the image and T is set to 127 (half of the maximum grey level). The C language code of the algorithm is given in Function **thresh_is ()** in **Appendix A**.

4.1.2 Single-band thresholding

A preliminary test was conducted to investigate the suitability of red, green, and blue bands for color grain image thresholding. Sixty five individual grain images (5 images for each of the five grain types, 5 images for each of the six damage types of CWRS wheat , 5 images of mixed grain-type kernels, and 5 images of mixed-damage type CWRS wheat kernels) were tested for each of the three color bands. The thresholded images were visually examined and compared. For the images containing single type grain kernels, there were no significant differences among the thresholded images of the red, green, and blue bands, except that the thresholded images of the red band usually enclosed some shadows as the object areas. However for images containing grain kernels from different grain types or damaged kernels from different damage types, there were significant differences among the thresholded images of the three color bands. The blue band was the best and the green band was usually better than the red band for thresholding.

Fig 4.1 shows two extreme cases: **Fig 4.1(a)** contains kernels from each of the five grain types being investigated, and **Fig 4.1 (b)** contains healthy and damaged CWRS wheat kernels with different extent from each of the six damage types being investigated. **Fig 4.2 (a)**, **(b)**, and **(c)** are the thresholded results of the image in **Fig 4.1(a)**, using the red, green,

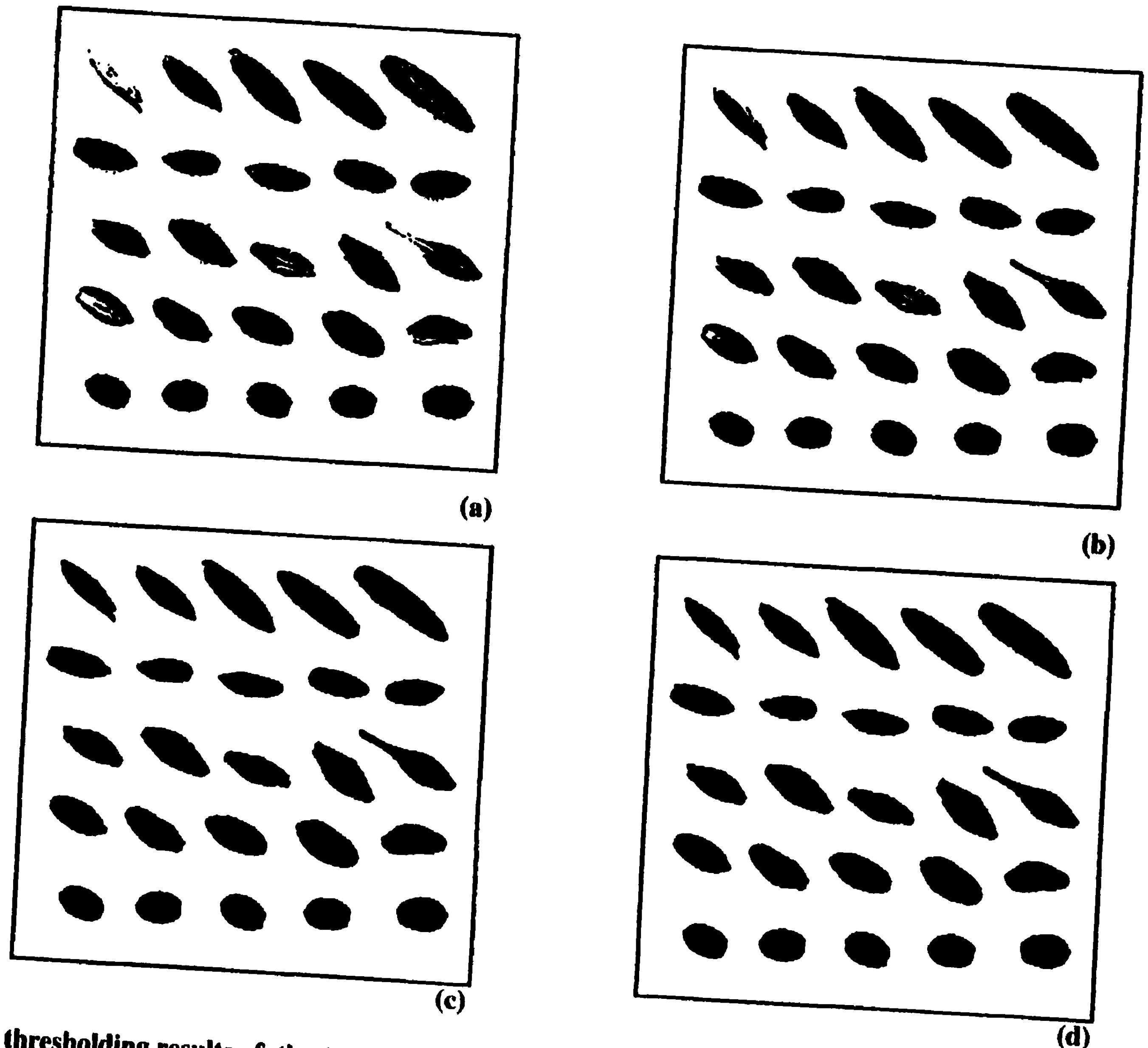
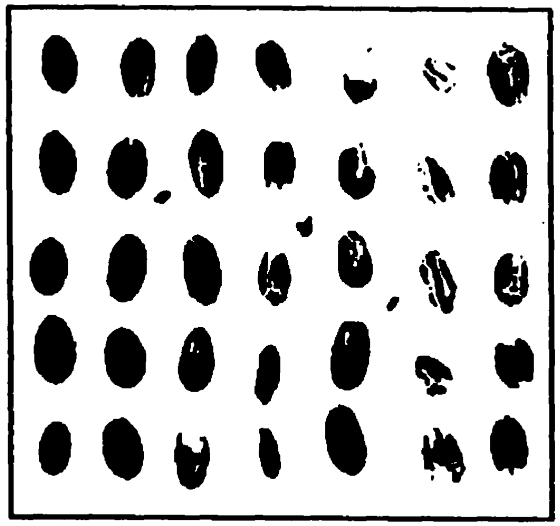
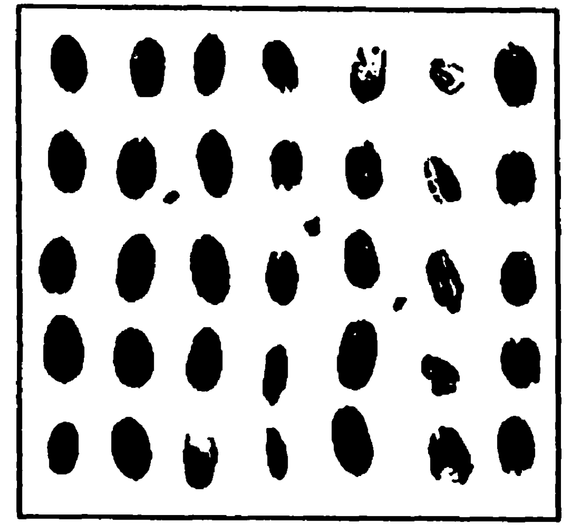


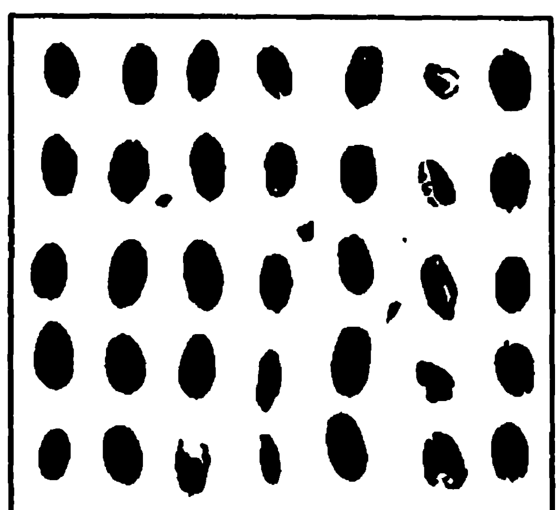
Fig 4.2 The thresholding results of the (a) red, (b) green, (c) blue, and (d) multi-bands of the test image in Fig 4.1(a).



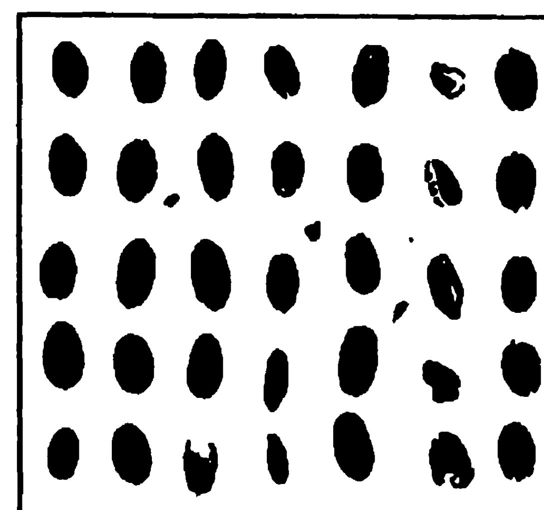
(a)



(b)



(c)



(d)

Fig 4.3 The thresholding results of the (a) red, (b) green, (c) blue, and (d) multi-bands of the test image in Fig 4.1(b).

and blue band, respectively. **Fig 4.3 (a)**, **(b)**, and **(c)** are the thresholded results of the image in **Fig 4.1(b)**, using the red, green, and biue band, respectively.

4.1.3 Multi-band thresholding

Although the preliminary tests showed that satisfactory results could be achieved by thresholding the blue band, it is still arguable that the use of the single band always produces good thresholding results. In practical applications, it is possible that a color grain image can not be satisfactorily thresholded using any of the three single color bands. A multi-band thresholding method was proposed to take advantage of color information in a color grain image. First single band thresholding was performed on each of the three color bands of a color image, resulting in three bi-level images, $f_R(x, y)$, $f_G(x, y)$, and $f_B(x, y)$ corresponding to the red, green, and blue bands, respectively. Then a thresholded image, $f(x, y)$, of the color image was produced by taking the following logical operation:

$$f(x, y) = f_R(x, y) \otimes f_G(x, y) \oplus f_R(x, y) \otimes f_B(x, y) \oplus f_G(x, y) \otimes f_B(x, y) \qquad \textbf{(4.1)}$$

where:

$\otimes$ = logical "and" and
$\oplus$ = logical "or".

The multi-band method was also tested with the sixty five individual grain images previously used for testing of the single band thresholding. The thresholding results of the image in **Fig 4.1(a)** and **(b)** using the multi-band method are shown in **Fig 4.2(d)** and **Fig 4.3 (d)**, respectively. The test results showed that using multi-band method was at least as good as using the blue band for thresholding the test images. The C-language code of the algorithm is given in Function **auto_thresh ()** in **Appendix A.**

4.2 Labeling

After thresholding, a bi-level image is obtained with the object areas having one grey-level value and the background having the other. The labeling process is to further distinguish the objects from each other by assigning a unique label to each of the separated regions (considered as an object area) in the bi-level image.

The labeling process is based on the relationship between individual pixels. Consider a small area of 3 x 3 pixels, centered about a pixel called P_0 at row x and column y of an image:

⊛ P_8 (x-1, y-1)	⊛ P_1 (x-1, y)	⊛ P_2 (x-1, y+1)
⊛ P_7 (x, y-1)	**⊛ $\mathbf{P_0}$ (x, y)**	⊛ P_3 (x, y+1)
⊛ P_6 (x+1, y-1)	⊛ P_5 (x+1, y)	⊛ P_4 (x+1, y+1)

The pixel P_0 has 8 neighbors: P_3 and P_7 in horizontal; P_1 and P_5 in vertical; P_2, P_4, P_6, and P_8 in diagonal directions. They are called *8-adjacent* neighbors of the pixel P_0, and the pixels P_1, P_3, P_5, and P_7 are called *4-adjacent* neighbors of the pixel P_0. Based on the neighboring relationship of two pixels, two major rules are defined to decide whether the two pixels are connected to each other: (1) two pixels are 4-connected if they are 4-adjacent and have the same pixel values; (2) two pixels are 8-connected if they are 8-adjacent and have the same pixel values. Two 4-connected pixels are 8-connected, while two 8-connected pixels may not be 4-connected. **Fig 4.4** shows the possible combinations of two 4-connected pixels. **Fig 4.5** shows the possible combinations of two 8-connected pixels.

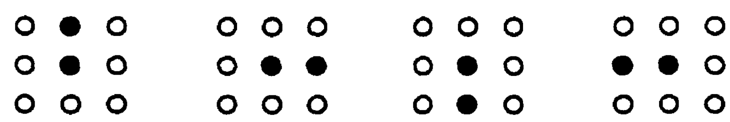


Fig 4.4 The possible combinations of two 4-connected pixels.

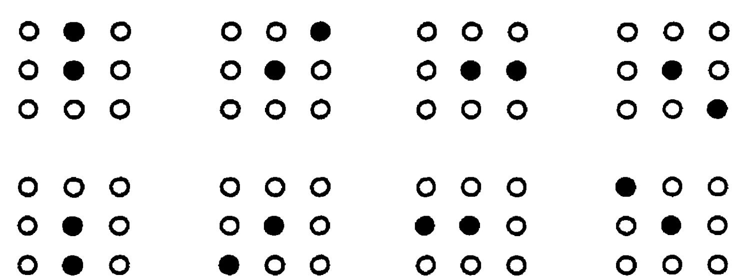


Fig 4.5 The possible combinations of two 8-connected pixels.

In a thresholded image, an object area (separated region) is actually a group of pixels which are connected to each other, in either the 4-connected or 8-connected sense. If a region consists of 4-connected pixels, it is called a 4-connected region. Similarly, if a region consists of 8-connected pixels, it is called an 8-connected region. By tracing the connectivity relationship (4-connected or 8-connected) between pixels, connected regions (object areas) in an image can be labeled (located and differentiated from each other).

Use of different connectivity relationships to locate regions in the same image may result in different divisions of the regions. **Fig 4.6** shows an image containing one region in the 8-connected sense while three regions in the 4-connected sense. The choice of the 4-connected or 8-connected neighbor relationships for region labeling depends on the specific application. Since the grain kernels were imaged in a separated manner, there is no preference to one over the other. The 4-connected relationship was chosen in the algorithm development.

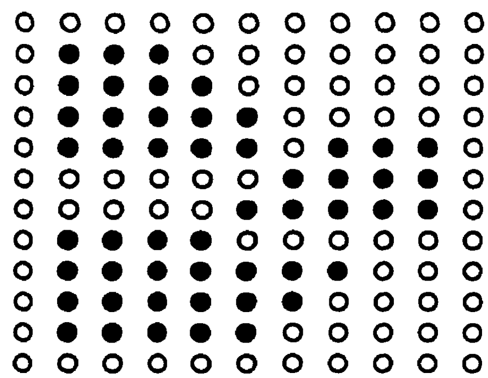


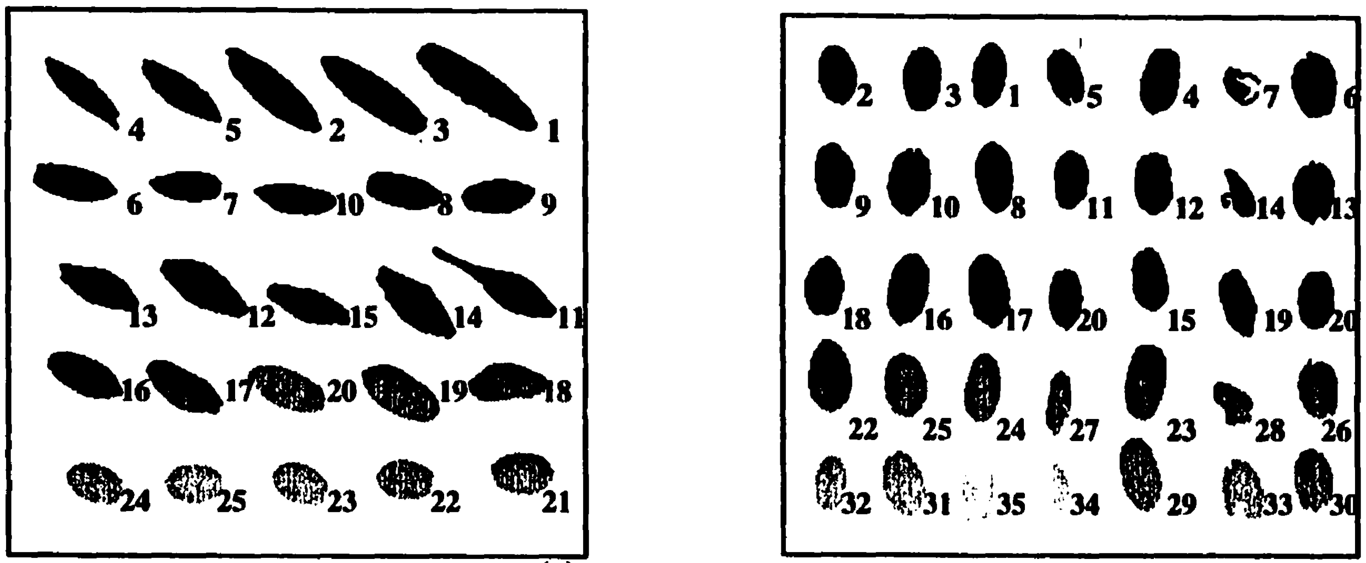
Fig 4.6 An image containing an 8-connected or three 4-connected regions.

The region labeling algorithm consisted of two functional phases: seed pixel searching and region growing. It was assumed that the regions to be identified are black (0) on a white (255) background. Starting from the left-top pixel, the algorithm scanned the bi-level image row by row until a black pixel (with 0 grey value) was found. This pixel called seed pixel was then assigned a grey value of 1 and used to "grow" a region. From the seed pixel, the algorithm grew a region by setting all the 4-adjacent black pixels of the seed pixel to a grey level value of 1. These pixels then became sub-seed pixels and their 4-connected black pixels were traced and set to the grey level value of 1. The region growing phase continued until all the black pixels, 4-connected with the seed pixel, were found and set to the grey level value of 1. At this point, the first region was labeled with a grey level value of 1 and the image had three grey levels: 0 - the unlabeled regions, 1 - the first region, and 255 - the background. The same procedure was repeated to label the second region with a grey level value of 2, the third with 3, and so on, until no black pixels remained. The searching of the seed pixel in each region labeling process started from the seed pixel of the last labeling process. Since a bi-level image uses only 2 out of 256 possible grey levels, at maximum 254 regions can be labeled in a bi-level image using the remaining 1 - 254 grey levels. For an image containing more than 254 regions (objects), numbers larger than 255, although they do not represent grey levels, are be used for the labeling. The C language code of the 4-connected region labeling algorithm is given in Function **region_4** () and the C language code of the 8-connected region labeling algorithm is given in Function **region_8** () in **Appendix A**.

4.3 Hole-filling and False-region-deleting

In a thresholded image there could be some groups of pixels with the background grey level value (255) enclosed in object regions (0) (as seen in **Fig 4.2(d)** and **Fig 4.3(d)**) due to the bright spots on object surfaces (as seen in **Fig 4.1**). These pixel groups, called "holes", have to be set to the object grey value for the accurate measurement of the object features. In practical applications, dusts, dirty background spots, or small pieces of grain shell may appear in a sample image (as seen in **Fig 4.1(b)**), resulting in small false regions in the thresholded image (as seen in **Fig 4.3(d)**). It is necessary to eliminate these false regions to avoid further feature measurements on these regions.

The hole-filling and small-region-deleting were performed right after the labeling. The hole-filling program (Function **fill_holes** () in **Appendix A**) is based on the fact that in a labeled image the background pixels are 4-connected to each other, while the hole pixels are enclosed in object regions although they have the same grey level value as the background pixels. Using any of the background pixels (usually the top left pixel) as the seed pixel, the background was labeled with a grey level value not used in region labeling (usually 254). Then only the hole pixels were of the white grey level value (255) in the image. The next step was to change the grey level value of the hole pixels to the values of the enclosing regions. The grey level value of the background was finally set back to the white (255). The false-region-deleting subroutine (Function **del_reg ()** in **Appendix A**) simply calculated the area of each region in pixels and changed the values of the pixel in those regions which contained 60 or less pixels (an area of $< 2.4\ (mm)^2$) to the background grey value (255). **Fig 4.7(a)** and **(b)** shows the final labeled images of the images in **Fig 4.1(a)** and **(b)** after hole-filling and false-region-deleting.



(a) (b)

✓ 4.2

Fig 4.7 The labeled test images (see Fig 4.1). (a) Kernels of different grain types. (b) Healthy and damaged wheat kernels and dusts. (The number at the lower right of each region is the grey level value of that region).

V FEATURE MEASUREMENTS

The objective of this research is to use an image analysis system to identify different types or different types of damaged grain kernels which are presented in the form of color images. A human observer can make identifications by simply looking at the images, but a computer has to make decisions by analyzing a set of quantitative data extracted from the images. These quantitative data, called image features, may represent the objects (grains in this case) in an image in different aspects. The most commonly used image features can be grouped under three categories: morphological, color, and texture. The morphological features are the measurements of the size and shape of the object. The color features describe the spectral characteristics of the object surface in terms of the three color band values. The texture features represent the texture content, such as smoothness, coarseness, and regularity of the object surface (Gonzalez and Woods 1992). This chapter describes all of the morphological and color measurements made on grain images. From these measurements, various features were selected for specific classification analyses (**Section 6.2**).

5.1 Measurements on Individual Grain Kernels

An image of individual grains contains spatially separated image regions. The segmentation process distinguishes each individual kernel from the background and from each other in the labeled multiple grey level image, with the white (255) grey level representing the background and each of the remaining grey levels representing a grain kernel. To take measurements on each individual kernel, a bi-level image of each kernel is "cut out" from the labeled image and a color image of the kernel is "cut out" from the original color image. The C language code of the computer program for doing this is given as Function **extract_obj ()** in **Appendix A. Fig 5.1(a)** shows the bi-level image of the upper right kernel in the image shown in **Fig 4.7(a)**. In the following sections, this kernel image will be used as an example to illustrate the extraction of various measurements.

5.1.1 Morphological measurements

The following concepts or definitions are essential in describing the morphological measurement extractions:

Center of mass (CM) The concept of the center of mass of an image object is borrowed from the physical concept of the center of mass which refers to a point in an object that has the same amount of substance around it in any direction. For a grey level image f(x, y) containing a single object of N pixels, the center of mass, (cm_x, cm_y), of the object can be defined as:

$$cm_x = \frac{1}{N}\sum_{\Omega} x\ f(x,y) \qquad\qquad cm_y = \frac{1}{N}\sum_{\Omega} y\ f(x,y) \tag{5.1}$$

where:

Ω = object region in the x-y plane.

In the case of a bi-level image of an object consisting of N pixels at locations (x_i, y_i), i =1, ...N, the CM of the object can be computed as:

$$cm_x = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad\qquad cm_y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{5.2}$$

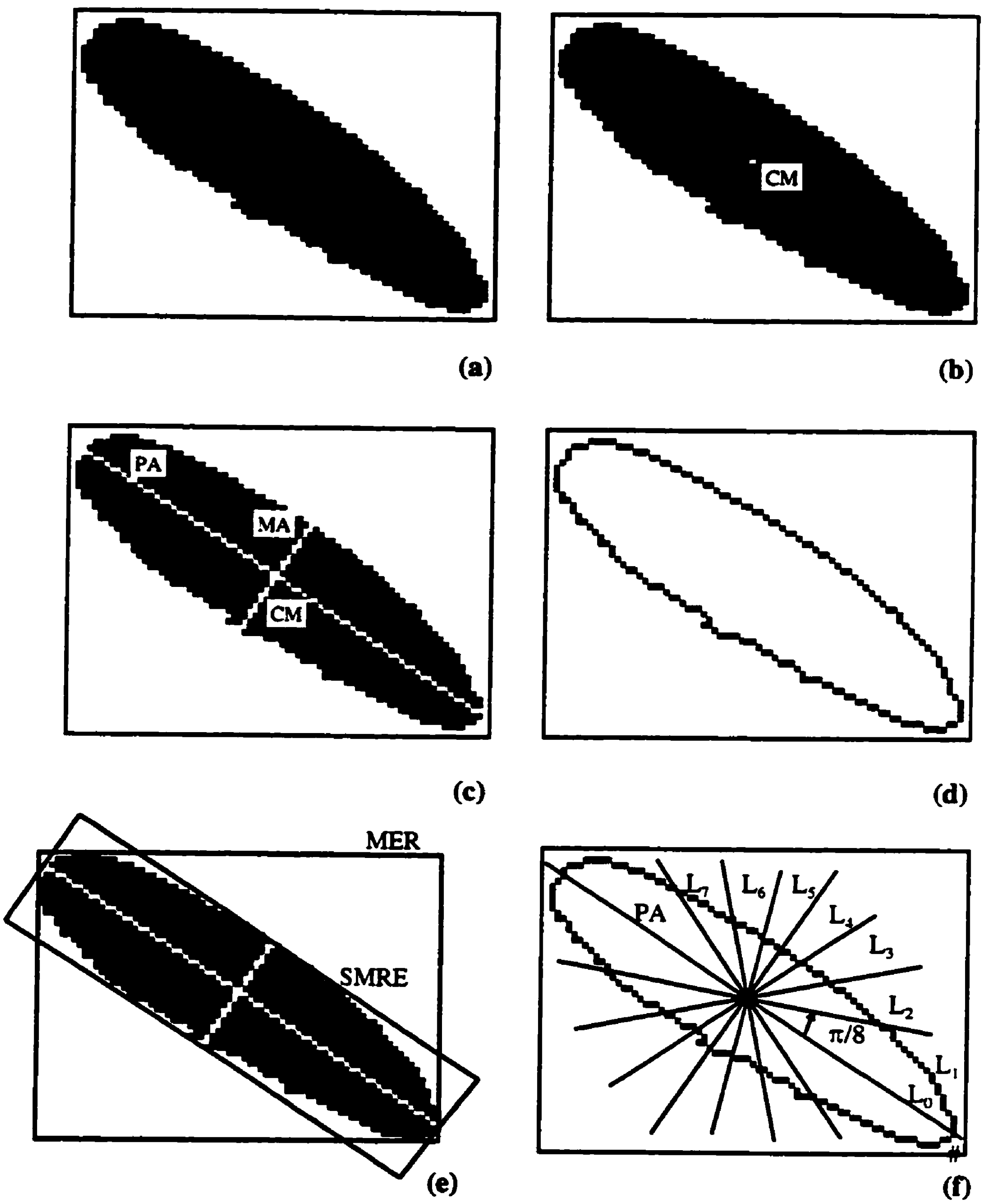


Fig 5.1 The (a) area, (b) center of mass (CM), (c) principal axis (PA) and minor axis (MA), (d) boundary, (e) minimum enclosing rectangle (MER) and standard minimum enclosing rectangle (SMER), and (f) signatures of the upper right kernel in the image Fig 4.1(a).

It can be viewed as a reference point or origin of the object. The C language code for computing the CM of an object in a bi-level image is given as Function **centre_of_mass ()** in **Appendix A**. **Fig 5.1(b)** shows the CM of the grain kernel in **Fig 5.1(a)**.

Distance between two pixels The distance between pixel P_1 at coordinates (x_1, y_1) and pixel P_2 at coordinates (x_2, y_2) is defined as the Euclidian distance:

$$d = [(x_1 - x_2)^2 + (y_1 - y_2)^2]^{1/2} \tag{5.3}$$

Distance from a pixel to a line The distance from a pixel P to a line L is defined as the minimum of the distances between the pixel P and any pixels on the line L.

Principal axis (PA) and minor axis (MA) The principal axis of an object in a bi-level image is defined as a pixel line passing through the object's CM and having a minimum total pixel distance from all pixels belonging to the object (Parker 1994). The minor axis is the pixel line passing through the CM in a direction perpendicular to the PA. Function **principal_axis ()** in **Appendix A** determines the PA of an object in a bi-level image by giving the coordinates of two pixels on the PA. **Fig 5.1(c)** shows the PA and MA of the grain kernel in **Fig 5.1(a)**.

Boundary The boundary of an object in a bi-level image is defined as the pixels belonging to the object and having at least one neighbor that belongs to the background. As discussed in **Section 4.2**, pixel neighboring relationship could be in 4-adjacent sense or 8-adjacent sense. Consequently, the boundary of a bi-level object could be a 4-adjacent boundary or an 8-adjacent boundary. The 4-adjacent boundary are the pixels that belong to the object and that are 4-connected to each other while

8-adjacent to the background. The 8-adjacent boundary are the pixels that belong to the object and that are 8-connected to each other while 4-adjacent to the background. Since there is no preference to one over the other, the 8-adjacent boundary was used in the measurement extraction. (Hereafter the boundary of an object always refers to the 8-adjacent boundary). **Fig 5.1(d)** shows the boundary of the grain kernel in **Fig 5.1(a)**.

Standard minimum enclosing rectangle (SMER) The standard minimum enclosing rectangle (SMER) of an image object is the minimum enclosing rectangle (MER) oriented along the PA of the object. **Fig 5.1(e)** shows the MER oriented along the x axis and SMER of the grain kernel in **Fig 5.1(a)**.

5.1.1.1 Size measurements

Area The area of an object was computed by counting the total number of pixels belonging to the object in the bi-level image.

Perimeter The perimeter of an object was computed as the length of the object boundary. Since the boundary was expressed in a bi-level image as a group of pixels which belong to the object and are connected to each other, the perimeter can be roughly estimated as the total number of the boundary pixels. However, a pixel represents a small square area, not a linear distance. The distance enclosed in a pixel depends on the way in which the physical object boundary passes through the small square area represented by the pixel. For example, assuming that the pixel represents an area of 1 unit by 1 unit, then the pixel represents 1 unit distance if the boundary passes the pixel area vertically or horizontally, or 1.4142 unit distance if the boundary runs diagonally across the pixel area. Although from a digitized object image there is no way to precisely determine how the boundary passes a boundary pixel, it is possible to make a guess by looking at the neighborhood of the pixel. **Fig 5.2** shows the neighborhood templates used for estimating the distance represented by a boundary pixel. A boundary pixel was assigned a pixel distance of d_i, if the boundary pixel and its 8-neighbors matched any of the templates in group i (i = a, b, c). The perimeter of an object was then obtained by summing up all the pixel distances represented by each boundary pixel. The program for computing the perimeter of an object from its bi-level boundary image is given as Function **perimeter ()** in **Appendix A**.

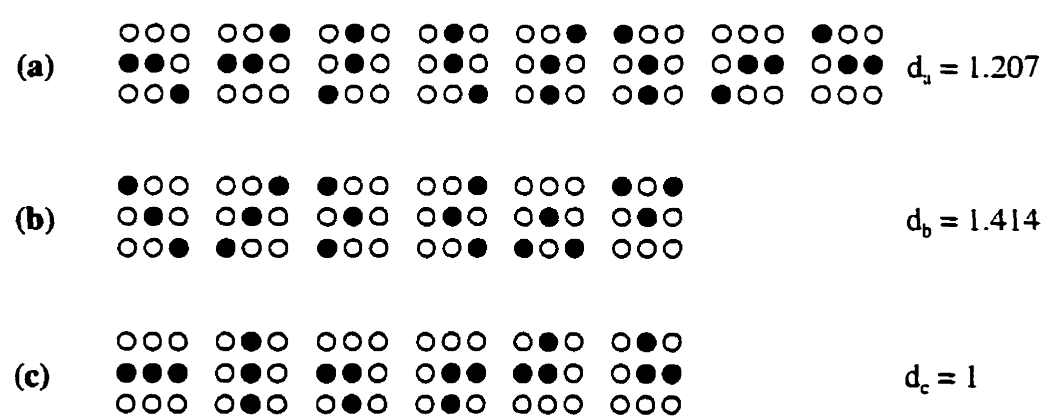


Fig 5.2 The templates used for estimating distances represented by boundary pixels.

Length and width The length and width of an object were defined as the length and width of the SMER of the object.

Length of principal axis (PA) The length of principal axis was calculated as the distance between the two intersection pixels of the PA and the boundary of the object.

Length of minor axis (MA) The length of the minor axis was calculated as the distance between the two intersection pixels of the MA and the boundary of the object.

Minimum and maximum radii and mean and variance of radii The distances between each pixel on the boundary and the CM were computed and their minimum, maximum, mean, and variance values were calculated as the minimum and maximum radii, and the mean and variance of radii, respectively.

All of the size measurements were first computed in pixel units and then converted to physical units (mm^2 for the area and mm for the others) using the pixel resolution (mm/pixel) (**Section 3.3.2**).

5.1.1.2 Shape measurements

Derived measurements The following measurements were derived from the size measurements to characterize the shape of individual grain kernels:

$$\text{Rectangular ratio} = \text{Length} / \text{Width} \tag{5.4}$$

$$\text{Aspect ratio} = \text{Length of PA} / \text{Length of MA} \tag{5.5}$$

$$\text{Area ratio} = (\text{Length} \times \text{Width}) / \text{Area} \tag{5.6}$$

$$\text{Radius ratio} = \text{Maximum Radius} / \text{Minimum Radius} \tag{5.7}$$

$$\text{Thinness ratio} = \text{Perimeter}^2 / \text{Area} \tag{5.8}$$

$$\text{Haralick ratio} = \text{Mean Radius} / \text{Standard Deviation of Radii} \tag{5.9}$$

Moments For an object image f(x, y), the central moment of order (p + q) of the object, denoted as μ_{pq}, is defined as (Gonzalez and Woods 1992):

$$\mu_{pq} = \sum_{\Omega} (x - cm_x)^p \ (y - cm_y)^q \ f(x,y) \tag{5.10}$$

The normalized central moments are calculated as:

$$\eta_{pq} = \mu_{pq} / \mu_{00}^{\gamma}, \qquad \gamma = (p + q)/2 + 1 \tag{5.11}$$

From the second and third normalized central moments, a set of measurements that are invariant to translation, rotation, and scaling of the object (Gonzalez and Woods 1992) can be derived as follows:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{5.12}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\,\eta_{11}^{\ 2} \tag{5.13}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{21} - \eta_{03})^2 \tag{5.14}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{5.15}$$

$$\begin{aligned} \phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3\,(\eta_{21} + \eta_{03})^2] + \\ & (\eta_{03} - 3\eta_{21})(\eta_{03} + \eta_{21})[(\eta_{03} + \eta_{21})^2 - 3\,(\eta_{12} + \eta_{30})^2] \end{aligned} \tag{5.16}$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}\,(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{5.17}$$

$$\begin{aligned} \phi_7 = {} & (\eta_{30} - 3\eta_{12})(\eta_{03} + \eta_{21})[(\eta_{03} + \eta_{21})^2 - 3\,(\eta_{12} + \eta_{30})^2] - \\ & (\eta_{30} - 3\eta_{21})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3\,(\eta_{21} + \eta_{03})^2] \end{aligned} \tag{5.18}$$

The above invariant moments were computed for each of the individual grain kernels from their bi-level images ($f(x, y) = 1$ for grain kernel regions), and the first four of them were used as shape measurements (the last three were found too small for most of the grain kernels).

Signatures A signature represents a 2-dimensional object shape by a set of 1-dimensional data. There are different ways to extract a signature from an object image. For example, the radius as a function of the angle between the radius line and the PA is a signature (Gonzalez and Woods 1992). In this study, three types of signatures were extracted from the bi-level grain images.

Consider an object with the CM at the coordinates (cm_x, cm_y) and the PA defined by line L_0: $a[0]\ x + b[0]\ y + c[0] = 0$. A line that intersects the PA at point (cm_x, cm_y) with an angle of θ can be determined by $a[\theta]\ x + b[\theta]\ y + c[\theta] = 0$, where:

$$a[\theta] = a[0] - b[0]\ tg(\theta), \tag{5.19}$$

$$b[\theta] = b[0] + a[0]\ tg(\theta), \tag{5.20}$$

$$c[\theta] = -a[\theta]\ cm_x - b[\theta]\ cm_y. \tag{5.21}$$

Using **Equation 5.19 to 5.21**, lines that intersect the PA at point (cm_x, cm_y) with an angle of $\pi/8$, $2\pi/8$, $3\pi/8$, $4\pi/8$, $5\pi/8$, $6\pi/8$, and $7\pi/8$, respectively, were determined as L_i: $a[i] * x + b[i] * y + c[i] = 0$, $i = 1, 2, \ldots 7$. The seven lines, together with the PA, divided the object area into 16 fan-shape subregions. The subregions were numbered as subregion i ($i = 0, 1, \ldots 15$) in such a way that subregion 0 was the one enclosed by lines L_0 and L_1 and adjacent to the intersection point of the boundary and the PA farthest from the CM and subregion 8 was the other one enclosed by the lines L_0 and L_1. The subregion i was the one enclosed by the lines L_i and $L_{Module8(i+1)}$ and next to subregion i-1 and subregion i+8 was the other one enclosed by the lines L_i and $L_{Module8(i+1)}$ ($i = 1, \ldots 7$), where $ModuleM(i) = i$, if $i < M$ or $i - M$, if $i \geq M$ (**Fig 5.1(f)**).

The sub-area A_i, length of perimeter segment P_i, and mean radius R_i, were calculated as for each of the subregion numbers i (i = 0, 1, ...15). These three sequences can be viewed as three types of signatures of the object shape: area, perimeter, and radius signatures. They are invariant to transformation and orientation. However from the definition they depend on scaling. To achieve scaling invariance, A_i, P_i, and R_i were normalized by the area A, the perimeter P, and the maximum radius R_{max} of the object, respectively:.

$$a(i) = A_i/A, \quad p(i) = P_i/P, \quad r(i) = R_i/R_{max} \tag{5.22}$$

The orientation invariance of a(i), p(i), and r(i) is based on the assumption that the same starting point (the intersection point of the boundary and the PA farthest from the CM) can be located on an object regardless of the object's orientation. Unfortunately, it was found that the starting point could be located on either the germ or the other ends of a grain kernel, depending on the kernel's orientation and location in the FOV. This may be due to the non-uniform magnification over the FOV introduced by the lens distortion and image digitization. As a result, subregion i could be numbered as subregion Module16(i+8) if the orientation or location of the kernel was changed. Consequently, the first halves of sequences a(i), p(i), and r(i) could become the last halves. In other words, the sequences could be shifted circularly by half. To cope with this problem, the magnitudes of the Fourier transforms of the normalized sequences were computed as the final area, perimeter, and radius signatures of the object by **Equations 5.23**, **5.24**, and **5.25** (N = 16). They are invariant to the starting point (therefore orientation) because the magnitude of the Fourier transform of a sequence is invariant to the circular-shift of the sequence.

$$A(k)=\left|\frac{1}{N}\sum_{i=0}^{N-1} a(i)\exp[-j2\pi ik/N]\right|=\frac{1}{N}\sqrt{[\sum_{i=0}^{N-1} a(i)\cos(2\pi ik/N)]^2+[\sum_{i=0}^{N-1} a(i)\cos(2\pi ik/N)]^2} \quad \textbf{(5.23)}$$

$$P(k)=\left|\frac{1}{N}\sum_{i=0}^{N-1} p(i)\exp[-j2\pi ik/N]\right|=\frac{1}{N}\sqrt{[\sum_{i=0}^{N-1} p(i)\cos(2\pi ik/N)]^2+[\sum_{i=0}^{N-1} p(i)\cos(2\pi ik/N)]^2} \quad \textbf{(5.24)}$$

$$R(k)=\left|\frac{1}{N}\sum_{i=0}^{N-1} r(i)\exp[-j2\pi ik/N]\right|=\frac{1}{N}\sqrt{[\sum_{i=0}^{N-1} r(i)\cos(2\pi ik/N)]^2+[\sum_{i=0}^{N-1} r(i)\cos(2\pi ik/N)]^2} \quad \textbf{(5.25)}$$

A list of the total 68 morphological measurements extracted from an individual grain kernel is given in **Table 5.1**. The C language code for extracting the morphological measurements is given as Function **size_shape_features()** in **Appendix A**.

5.1.2 Color measurements

5.1.2.1 Measurements derived from normalized RGB signals

The normalized RGB signals, r(x, y), g(x, y), and b(x, y) were computed for each image from its three color band signals, R(x, y), G(x, y), and B(x, y), respectively, using the **Equation 3.5**. The following measurements were derived from the normalized RGB signals of a kernel region Ω which consisting of N pixels.

Mean normalized RGB signals

$$\bar{r}=\frac{1}{N}\sum_{\Omega} r(x,y) \qquad \bar{g}=\frac{1}{N}\sum_{\Omega} g(x,y) \qquad \bar{b}=\frac{1}{N}\sum_{\Omega} b(x,y) \quad \textbf{(5.26)}$$

Variances of normalized RGB signals

$$\sigma_r^2=\frac{1}{N-1}(\sum_{\Omega} r^2(x,y)-N\bar{r}^2) \qquad \sigma_g^2=\frac{1}{N-1}(\sum_{\Omega} g^2(x,y)-N\bar{g}^2) \qquad \sigma_b^2=\frac{1}{N-1}(\sum_{\Omega} b^2(x,y)-N\bar{b}^2) \quad \textbf{(5.27)}$$

Table 5.1 Morphological measurements on individual grain kernels

Number	Measurement	Code
Size measurements		
1	Area	A
2	Perimeter	P
3	Length	L
4	Width	W
5	Length of PA	LPA
6	Length of MA	LMA
7	Minimum radius	R_{min}
8	Maximum radius	R_{max}
9	Mean radius	R_{mean}
10	Variance of radii	Var_R
Shape measurements		
11	Aspect ratio	aspR
12	Rectangular ratio	rctR
13	Radius ratio	radR
14	Thinness ratio	thnR
15	Area ratio	areaR
16	Haralick ratio	hraR
17 - 20	First four invariant moments	mnt1 - mnt4
21 - 36	Area signatures	AS1 - AS16
37 - 52	Perimeter signatures	PS1 - PS16
53 - 68	Radius signatures	RS1 - RS16

Ranges of normalized RGB signals

$$\Delta r = r_{max} - r_{min} = \max_{\Omega}[r(x,y)] - \min_{\Omega}[r(x,y)]$$

$$\Delta g = g_{max} - g_{min} = \max_{\Omega}[g(x,y)] - \min_{\Omega}[g(x,y)]$$

$$\Delta b = b_{max} - b_{min} = \max_{\Omega}[b(x,y)] - \min_{\Omega}[b(x,y)] \qquad \textbf{(5.28)}$$

5.1.2.2 Measurements derived from HSI signals

The HSI color model is another commonly used color model. In the HSI color model, color is described by three attributes: hue, saturation, and intensity. Hue is an attribute associated with the dominant pure color (such as pure yellow, pure red, etc.); saturation refers to relative purity or the amount of white light mixed with a hue; and intensity is defined as a measure of the brightness of achromatic light.

The HSI color model owes its usefulness in image processing to two principal facts. First, the intensity attribute I is decoupled from the color information. Second, the hue attribute H and the saturation attribute S, together called chromaticity, are intimately related to the way in which human beings perceive color (Nevatia 1982). These features make the HSI color model an ideal tool for developing an image algorithm based on some of the color sensing properties of the human visual system (Gonzalez and Woods 1992).

The attributes H, S, and I can be derived from the normalized RGB values r, g, and b by (Gonzalez and Woods 1992):

$$H = \cos^{-1}\left\{ \frac{0.5\,[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(r+g+b)}\,[\min(r, g, b)]$$

$$I = \frac{1}{3}(r+g+b) \tag{5.29}$$

The HSI signals, H(x, y), S(x, y), and I(x, y) were computed for each image from its three color band signals, R(x, y), G(x, y), and B(x, y), respectively, for each image using the **Equations 3.5** and **5.29**. The following measurements were derived from the HSI signals of a kernel region Ω of N pixels.:

Mean HSI signals

$$\bar{H} = \frac{1}{N}\sum_{\Omega} H(x,y) \qquad \bar{S} = \frac{1}{N}\sum_{\Omega} S(x,y) \qquad \bar{I} = \frac{1}{N}\sum_{\Omega} I(x,y) \tag{5.30}$$

Variances of HSI signals

$$\sigma_H^2 = \frac{1}{N-1}\left(\sum_{\Omega} H^2(x,y) - N\bar{H}^2\right) \qquad \sigma_S^2 = \frac{1}{N-1}\left(\sum_{\Omega} S^2(x,y) - N\bar{S}^2\right) \qquad \sigma_I^2 = \frac{1}{N-1}\left(\sum_{\Omega} I^2(x,y) - N\bar{I}^2\right) \tag{5.31}$$

Ranges of HSI signals

$$\Delta H = H_{max} - H_{min} = \max_{\Omega}[H(x,y)] - \min_{\Omega}[H(x,y)]$$

$$\Delta S = S_{max} - S_{min} = \max_{\Omega}[S(x,y)] - \min_{\Omega}[S(x,y)]$$

$$\Delta I = I_{max} - I_{min} = \max_{\Omega}[I(x,y)] - \min_{\Omega}[I(x,y)] \tag{5.32}$$

5.1.2.3 Color moments

In **Section 5.1.1.1** the invariant moments defined by **Equations 5.12** to **5.18** were computed on the bi-level grain images (f(x, y) = 1 for grain kernel regions) as shape measurements ϕ_i (i = 1, ...7). This time, the invariant moments, called color moments, were computed on each of the three normalized color bands, namely r(x, y), g(x, y), and b(x, y), for each individual grain kernel as color measurements ϕR_i, ϕG_i, and ϕB_i (i = 1, ... 7), respectively. The f(x, y) was set equal to r(x, y), g(x, y), and b(x, y), respectively instead of 1, if the pixel at (x, y) belongs to a kernel region.

5.1.2.4 RGB histograms

An M-band histogram of an object in a digital image with grey levels in the range [0, L-1] is defined as a discrete function $H(k) = n_k/N$, k = 0, ..., M-1 ($1 \le M \le L$); where k is the band number, n_k is the number of pixels in the object region with grey levels in the kth band range [k*L/M, (k+1)*L/M-1], and N is the total number of pixels in the object region. Because a color image consists of three grey level images, namely R, G, and B images, correspondingly three M-band histograms, $H_R(k)$, $H_G(k)$, and $H_B(k)$, of an object in a color image can be obtained from the three grey level images. These histograms provide a global description of the object's color appearance. The selection of the number of bands, M, depends on specific applications. Generally, the larger M is the more precisely do the histograms describe the color appearance. However, when the histograms are used as color features to represent color differences between different objects, this statement is not always true. In addition, a larger M means a larger number of measurements (the three histograms give 3 x M measurements in total). A preliminary test was conducted to compare the histograms with M = 8, 16, and 32, by examining the significance of the corresponding measurements to the classification of the different types of cereal grains. It was found that 16-band histograms gave the best measurements. The 48 measurements from the three 16-

band histograms, $H_R(k)$, $H_G(k)$, and $H_B(k)$, were finally used as color measurements.

A list of the total 78 color measurements extracted from an individual grain kernel is given in **Table 5.2**. The C language code for extracting the color measurements is given as Function **color_features()** in **Appendix A**.

5.2 Measurements on Bulk Grain Images

For the bulk grain image analysis, all the color measurements used for the individual grain image analysis except the color moments were extracted from the color bulk grain images. They were computed over the whole image instead of individual kernel regions. The histograms of R, G, and B were computed as 32-band instead of 16-band. A preliminary study showed that the 32-band histograms were better than the 16-band histograms in discriminating the bulk images of different grain types. A list of the total 114 color measurements extracted from a bulk grain image is given in **Table 5.3**. The C language code for extracting the color measurements from a bulk grain image is given as **Function bulk_features()** in **Appendix A**.

Table 5.2 Color measurements on individual grain kernels

Number	Measurement	Code
1	Mean of r	r_{mean}
2	Mean of g	g_{mean}
3	Mean of b	b_{mean}
4	Variance of r	Var_r
5	Variance of g	Var_g
6	Variance of b	Var_b
7	Range of r	Δr
8	Range of g	Δg
9	Range of b	Δb
10	Mean of H	H_{mean}
11	Mean of S	S_{mean}
12	Mean of I	I_{mean}
13	Variance of H	Var_H
14	Variance of S	Var_S
15	Variance of I	Var_I
16	Range of H	ΔH
17	Range of S	ΔS
18	Range of I	ΔI
19 - 34	16-band histograms of R	hstR1 - hstR16
35 - 50	16-band histograms of G	hstG1 - hstG16
51 - 66	16-band histograms of B	hstB1 - hstB16
67 - 70	First four invariant moments of r	mntr1 - mntr4
71 -74	First four invariant moments of g	mntg1 - mntg4
75 - 78	First four invariant moments of b	mntb1 - mntb4

Table 5.3 Color measurements on bulk grain images

Number	Measurement	Code
1	Mean of r	r_{mean}
2	Mean of g	g_{mean}
3	Mean of b	b_{mean}
4	Variance of r	Var_r
5	Variance of g	Var_g
6	Variance of b	Var_b
7	Range of r	Δr
8	Range of g	Δg
9	Range of b	Δb
10	Mean of H	H_{mean}
11	Mean of S	S_{mean}
12	Mean of I	I_{mean}
13	Variance of H	Var_H
14	Variance of S	Var_S
15	Variance of I	Var_I
16	Range of H	ΔH
17	Range of S	ΔS
18	Range of I	ΔI
19 - 50	32-band histograms of R	hstR1 - hstR32
51 - 82	32-band histograms of G	hstG1 - hstG32
83 - 114	32-band histograms of B	hstB1 - hstB32

VI CLASSIFICATION ANALYSIS

6.1 Classification Criteria (Classifiers)

Classification analysis needs the use of a decision rule, called *a classification criterion,* to classify objects into two or more known groups, called *classes*, on the basis of the quantitative features extracted from the objects. A set of features extracted from an object is called an observation of the object. The classification criterion is usually derived from the observations of known classes, called *the training or designing data*. The derived classification criterion then can be applied to classify new observations, called *the test data*.

A classification criterion partitions an observation or feature hyper-space, Ω, into hyper-regions Ω_i, $i = 1, ..., N$, where N is the number of classes. An object is classified as coming from class ω_i if its corresponding feature vector or observation $\mathbf{x}$, a point in the hyper-space Ω, belongs to the region Ω_i. There are many methods for developing a classification criterion from a training data set.

6.1.1 Statistical methods

The statistical methods are based on the *Bayes minimum error rule* (Duda and Hart 1973):

$$\mathbf{x} \in \Omega_k \quad \text{if } P(w_k|\mathbf{x}) > P(w_j|\mathbf{x}) \qquad \forall\, j \neq k \tag{6.1}$$

where:

$P(w_i|\mathbf{x})=$ the *posterior probability*, by which an object with a feature vector $\mathbf{x}$ belongs to class w_i,

$\in$ = "belongs to" , and

$\forall$ = "for all".

The rule states that to minimize the average probability of error, an object should be classified as belonging to a class w_i that maximizes the posterior probability $P(w_i|\mathbf{x})$.

By applying the Bayes' theorem:

$$P(w_i|\mathbf{x}) = P(w_i)\, p(\mathbf{x}|w_i)/p(\mathbf{x}) \tag{6.2}$$

a more practical formulation of the rule can be obtained as

$$\mathbf{x} \in \Omega_k \text{ if } P(w_k)\, p(\mathbf{x}|w_k) > P(w_j)\, p(\mathbf{x}|w_j) \qquad \forall\, j \neq k \tag{6.3}$$

where:

$P(w_i)$ = the prior probability by which an object comes from class w_i,
$p(\mathbf{x})$ = the probability density function for $\mathbf{x}$, and
$p(\mathbf{x}|w_i)$ = the class-conditional probability density function for $\mathbf{x}$.

In practical applications, it is rare that the posterior probabilities or the class-conditional probability density functions are known. They usually need to be estimated from the training data set. There are two fundamental approaches to do this.

6.1.1.1 Parametric approach

The parametric approach is based on the assumption that the class-conditional probability density function for $\mathbf{x}$, $p(\mathbf{x}|w_i)$, has a form of multivariate normal distribution:

$$p(\mathbf{x}\,|w_i) = (2\pi)^{-d/2}\, |\Sigma_i|^{-1/2} \exp[\,-0.5\,(\mathbf{x} - \boldsymbol{\mu}_i)'\, \boldsymbol{\Sigma}_i^{-1}\, (\mathbf{x} - \boldsymbol{\mu}_i)] \tag{6.4}$$

where:

d = the dimension of the feature vector,
$\boldsymbol{\mu}_i$ = the d-dimensional vector containing feature means in class w_i,
$\boldsymbol{\Sigma}_i$ = the covariance matrix, and
$'$ means transfer.

So to estimate the probability density one needs to estimate the parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$. The parameters, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, can be estimated from the training data set using different parameter estimation methods (Hand 1981, Chapter 3). The prior probability $P(w_i)$ can also be estimated from the training data set. Then the classification criterion, **Equation (6.1)** or **(6.3)**, can be determined in an analytical form.

6.1.1.2 Non-parametric approach

The non-parametric approach calculates the posterior probability $P(w_i|\mathbf{x})$ directly from the training data set without any assumption of the underlying probability density. There are several methods for estimating $P(w_i|\mathbf{x})$ such as the histogram, the kernel method, the k-nearest-neighbor method, and the series method (Hand 1981, Chapter 2). The k-nearest-neighbor method was used in this study.

The idea of the k-nearest-neighbor method is quite straightforward. Let n_i be the number of the training set points in class w_i, $i = 1, ..., N$, and n be the total number of the training set points (so that $n = \sum_i n_i$). For a new observation $\mathbf{x}$, the method calculates the distances from $\mathbf{x}$ to each of the training set points and finds out the k points that are the nearest to $\mathbf{x}$. Suppose that amongst these k points there are k_i from class w_i. Then the class-conditional probability density function at $\mathbf{x}$ is estimated as:

$$p(\mathbf{x}|w_i) = k_i / [n_i \, V_k(\mathbf{x})] \qquad \textbf{(6.5)}$$

The prior probability by which an object comes from class w_i is estimated as

$$P(w_i) = n_i / n \qquad \textbf{(6.6)}$$

The probability density function for $\mathbf{x}$ is estimated as:

$$p(\mathbf{x}) = k / [n \, V_k(\mathbf{x})] \qquad \textbf{(6.7)}$$

where $V_k(\mathbf{x})$ is the volume of the hyper-sphere which centers at $\mathbf{x}$ and just encloses the k nearest points of the training set. The posterior probability is given by the Bayes' theorem (**Equation 6.2**) as:

$$P(w_i|\mathbf{x}) = P(w_i)\, p(\mathbf{x}|w_i)/p(\mathbf{x})$$
$$= (n_i / n)\{k_i /[n_i\, V_k(\mathbf{x})]\} / \{k / [n\, V_k(\mathbf{x})]\}$$
$$= k_i / k \tag{6.8}$$

By the Bayes minimum error rule (**Equation 6.1**), this results in the classification criterion: classify an object with a feature vector $\mathbf{x}$ as belonging to class w_i, if $k_i = \max_j (k_j)$.

The parametric approach has the advantage that the derived classification criterion is of an analytical form which can be easily transferred into a computer classification program. However, the assumption of the multivariate normal distribution, made for the class-conditional probability density function in deriving the classification criterion, could be incorrect or insufficient in many applications and may lead to a large classification errors. The k-nearest-neighbor approach avoids the subjective assumption by directly estimating the posterior probability $P(w_i|\mathbf{x})$ from the training data set. A disadvantage of this approach is that the derived classification criterion cannot be expressed analytically. All of the training data must be retained - the distance from a new observation $\mathbf{x}$ to each of the training set points must be determined to choose the k nearest points. This means a large amount of computer memory and a slow classification process. In addition, the estimation of the posterior probability is biased (Rosenblatt 1956) towards larger values.

6.1.1.3 SAS Procedure DISCRIM

A statistical classification analysis procedure, DISCRIM, is available in SAS (SAS 1990). The procedure can derive a classification criterion from a training data set using either parametric or non-parametric approaches and apply the derived classification criterion to classify a new (test) data set during the same execution of the procedure. If a parametric approach is used, the derived classification criterion is given in an output data set.

The DISCRIM can evaluate the derived classification criterion in three methods. The first, called *re-substitution classification*, is to apply the classification criterion derived from a training data set to the same data set and then count the number of mis-classification observations called error-count estimate in each class. This error-count estimate has an optimistic bias. The second method, called *cross-validation classification*, is to apply the classification criterion derived from the N-1 out of the N observations of the training data set to the one observation left-out. The process is repeated for each of the N training observations and then the mis-classification rate for each class is calculated as the proportion of observations in the class that are mis-classified. The estimation is nearly unbiased but with a relatively large variance. The last method, called *hold-out classification*, calculates the error-count estimate by applying the classification criterion derived from a training data set to a test data set and then count the number of mis-classified observations in the testing set.

6.1.2 Neural network method

6.1.2.1 Neural networks

A neural network (NN) is a computing network of numerous simple, highly interconnected processing elements called neurons or nodes. A neuron has many continuous-valued input signals $\mathbf{x} = [x_i]$, i=1, 2, ..., N, which represent the activity at the input or the momentary frequency of neural impulses delivered by other neurons to this input (Kohonen 1988), and an output y which represents the response of the neuron to the input signals. The relationship between the inputs and the output of a neuron is described by the neuron's transfer function, $y = f[\mathbf{x}]$. In the simplest model of a neuron, the output value or the frequency of the neuron, y, is often approximated by:

$$y = f[\mathbf{x}] = K \, \phi \, (\sum_{i=1}^{N} w_i \, x_i - \theta) \tag{6.9}$$

where K is a constant and ϕ is a nonlinear function which takes the value +1 for positive arguments and -1 (or 0) for negative arguments. The w_i is called "synaptic efficacy" (Kohonen 1988), or weight, and θ is a threshold.

For some years now, many neural network models, dating as far back as the 1960's (Rosenblatt 1962), have been developed with different neuronal transfer functions, network structures, and training methods. Most of them have had limited real-world applications. However, the *multilayer neural network* with the *generalized delta rule for learning by back-propagation* has been used successfully in various practical problems, especially in pattern recognition.

6.1.2.2 MNN and B-P algorithm

A multilayer neural network with the generalized delta rule for learning by back-propagation learning algorithm (Rumelhart et al. 1986) is an effective system for learning discriminants for classes from a set of examples (Sejnowski and Rosenberg 1987, Tesauro and Sejnowski 1989). In general such a network is made up of sets of neurons (nodes) arranged in several layers (**Fig 6.1**). There are three distinct types of layers: the input layer, the hidden layer(s), and the output layer. The connections between the neurons of adjacent layers relay the output signals from one layer to the next. The input layer receives the input information and distributes the information to the next processing layer (the first hidden layer). The number of the neurons in the input layer equals to the dimension of the input vector **x** (the number of the features). The hidden and output layers process the incoming signals by amplifying or attenuating or inhibiting the signals through weighting factors. Except for the input layer neurons, the network input to each neuron is the sum of the weighted outputs of the neurons in the previous layer. The number of the neurons in the output layer is determined by the number of the classes under investigation. The number of hidden layers and the number of neurons in each hidden layer depend on specific applications.

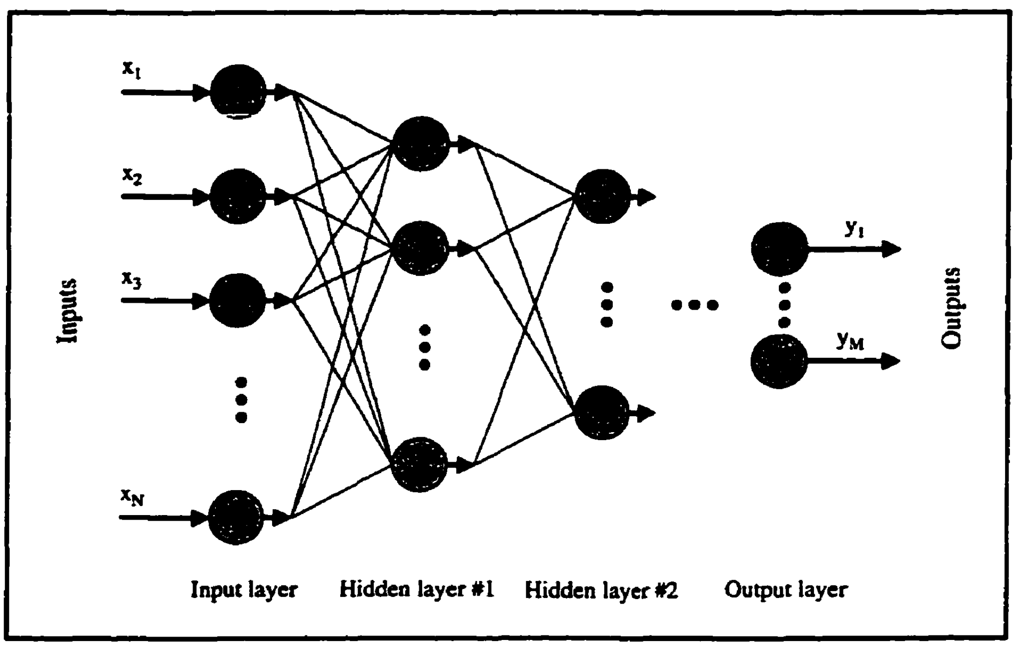


Fig 6.1 A schematic depiction of a multilayer neural network.

The application of the B-P algorithm involves two phases. During the first phase the inputs **x** are presented and propagated forward through the network to compute the outputs $y_k(n)$ in presentation n for each unit k, i.e.:

$$y_k(n) = f_k[\ net_k(n)\ ] \quad \textbf{(6.10)}$$

where:

$$net_k(n) = \sum_j w_{kj}(n)\ y_j(n) \quad \textbf{(6.11)}$$

$w_{kj}(n)$ is the weight of the connection from neuron j in the previous layer to neuron k in the current layer in presentation n, and $f_k[]$ is the transfer function at unit k which is differentiable and non-decreasing. A widely used choice for a transfer function is the sigmoid function:

$$f_k[net_k(n)] = 1\ /\ (1 + e^{-[net_k(n) - \theta_k]}) \quad \textbf{(6.12)}$$

where θ_k is the threshold for unit k.

The second phase involves a backward pass through the network (analogous to the initial forward pass), during which, the difference between the actual output and desired output generates an error signal $\delta_k(n)$ and this error signal is passed to each unit in the network and the appropriate weight changes are made according to:

$$w_{jk}(n+1) = w_{jk}(n) + \epsilon\ \delta_k(n)\ y_j(n) + \alpha\ [w_{jk}(n)-w_{jk}(n-1)] \quad \textbf{(6.13)}$$

where ϵ is the learning rate which is a scalar referring to learning speed, α is the learning momentum which is a scalar determining the effects of past weights on the convergence of the network in the weight space. This second, backward pass allows the recursive computation of $\delta_j(n)$ (Rumelhart et al. 1986). Once $\delta_j(n)$ arrives at the desired error, the network will have found a set of weights that produce the correct output for every input, in other words, the MNN will have stored the class knowledge in its weights and be ready to classify new input data.

When working as a classifier, an MNN operates as a black box which receives an input vector **x** (a set of observations) and produces responses y_j from its output units j (j = 1, 2, ..., M, where M depends on the number of classes). Generally, $y_j = 1$ if neuron j is active for the current input vector **x**, and $y_j = -1$ (or 0) if it is inactive. That means that for a specific input vector **x**, the outputs give the binary representation of its class number.

Like a k-nearest-neighbor classifier, an MNN classifier learns the class knowledge directly from the training data set. Therefore, it is unnecessary to make any assumptions regarding the underlying probability density functions. An advantage of the MNN classifier over a k-nearest-neighbor classifier is that it takes less computer memory and less time in the classification process. After training (learning), the MNN classifier is specified by a set of processing elements which are arranged in a certain topological structure and interconnected with fixed connections (weights). It can be easily transferred into a computer classification program. There is no need for retaining all the training data and no extensive computation is involved in the classification process. However, a problem in designing the MNN classifier is that there is no theoretical method available to optimally determine the network structure, the number of the hidden layers, and the node numbers in each hidden layer, which control the MNN's learning and classifying ability. Although, it has been shown that an MNN with two hidden layers can form any discriminant surface (Pao 1989), MNNs with three or more hidden layers are also used for their efficiency and speed in learning (training). An MNN with a small and simple hidden layer structure may not grasp sufficient class knowledge for classification, while an MNN with a large and complex hidden layer structure may tend to memorize the specific patterns in the training data set rather than learn the general class information. The best way for the structure design is to start with small number of hidden layers and processing nodes. The network complexity can be gradually increased until sufficient training degree is obtained. The time required for training an MNN strongly depends on the complexity of the network, the size of the training data set, and the computer speed. For a complex MNN and a large size training data set, the training process may take several days. For example, it took approximately 48 h to train an MNN of a 24-6-4-5 structure with a training data set of 29 400 samples, where 24-6-4-5 represents a network consisting of an input layer with 24 nodes, two hidden layers with 6 nodes in the first and 4 nodes in the second, and an output layer with 5 nodes.

6.1.2.3 Qnet

A commercial software package, **Qnet** (Qnet V2: 32-bit Neural Modeling for Windows, Vesta Service, Inc., 1001 Green Bay Rd., Box 196, Winnetka, IL) was used for the MNN modeling in this research. Qnet provides graphical tools under Windows for creating, training, and testing (recalling) an MNN. For creating an MNN, Qnet allows specifying the number of input nodes, the number of hidden layers and the number of nodes in each hidden layer, the number of output nodes, the connections between layers, and the transfer functions used in each layer. Qnet uses an optimized B-P algorithm for training an MNN. The training parameters, learning rate ϵ, learning moment α, and maximum number of iterations can be specified at the beginning of the training and automatically or manually adjusted during the training according to the training situation. The training process can be monitored through the real-time training analysis tools, such as the training error history plot, the testing error history plot, the learning rate history plot, the targets/output plots, the divergence check, and so on. Qnet can automatically save the training results (trained networks) at a rate or interval specified by the user, which allows unattended training (recovering from overtraining situations and training divergence). The trained MNN can be recalled in Qnet for testing with new observation data or output in a file (*.net) which can be incorporated into C/C++ application programs.

6.2 Feature Selection

For a given classification problem, there could be a large number of measurements which can be extracted from the objects to be classified. In the present case, there are 146 measurements extracted from each individual grain kernel image and 114 measurements extracted from each bulk grain image. Some of them may be redundant or highly correlated. It is, therefore, necessary to select an effective feature set from the extracted measurements which leads to satisfactory classification results.

The feature selection was done in two steps. First a SAS procedure STEPDISC (SAS 1990) was used to select a group of feature models of different sizes (feature numbers), according to the feature's contributions to the discriminatory power of the corresponding model. Then the feature models suggested by STEPDISC were further evaluated using SAS DISCRIM and an optimal feature model was then selected for the final classification analysis.

6.2.1 Stepwise discriminant analysis

The SAS procedure STEPDISC selects a set of features step by step, using forward selection, backward elimination, or stepwise selection methods. Two criteria can be used to choose measurements to enter or leave the selected feature set: the significance level and the squared partial correlation. The stepwise selection method and the significance level criterion were used by default in the feature discriminant analysis. A minimum significant level of 0.15 was specified for a measurement to enter and stay in the selected feature set. The stepwise selection method starts with no measurements in the selected feature set. At each step, a covariance analysis is performed with the measurements already in the selected feature set serving as covariates and the measurements not in the set being the dependents. If the measurement in the set that contributes least to the discriminatory power fails to meet the criterion to stay, then the measurement is removed from the set. Otherwise, the measurement not in the set that contributes most to the discriminatory power is entered. The feature selection process continues until all measurements in the selected set meet the criterion to stay and none of the other measurements meets the criterion to enter. The stepwise discriminant analyses were carried out using the measurement data (observations) from all the available grain samples.

6.2.2 Evaluation of feature models

To select an optimal feature model, the discriminating abilities of the different size feature models suggested by STEPDISC were evaluated using SAS DISCRIM. The evaluation started with the feature model of the first 4 features suggested by STEPDISC, and gradually incorporated more features from the feature set suggested by STEPDISC. Each time the next 4 features on the feature list were added in. For each feature model, both the parametric (quadratic) and the non-parametric (k-nearest neighbor) classification criteria were derived from all the available observations (grain samples) and the cross-validation method was used to evaluate the discriminating abilities of the feature model under the parametric and non-parametric classification criteria. The mean of the classification accuracies (MCA) for each class was computed for each of the two classification criteria and used as a measure of the discriminating ability of the corresponding feature model. The feature model with the highest mean classification accuracy was chosen as the feature model for the final classification analysis.

6.3 Classification Analysis

6.3.1 Grain type identification of individual grain kernels

Both of parametric and non-parametric statistical classifiers were used with three types of feature models: namely morphological, color, and combined (morphological and color). For each type of the feature models, an optimal set of features was selected using the feature selection method described in **Section 6.2**. The data set consisted of 42 000 observations of grain kernels collected from five grain types and seven grain categories: CWRS wheat grade 1, 2, and 3; CWAD wheat; barley; rye; and oats. Each category contained 6000 observations (kernels) from 20 growing regions (300 kernels per region). Each grain type was considered as a *class* (CWRS wheat grade 1, 2, and 3 were treated as a single class). The hold-out method was used for the 5-class classification analysis. The data set was split into three subsets according to the growing regions. The first subset contained the observations of the grain kernels from 7 growing regions (14 700 grain kernels), the second subset contained the observations of the grain kernels from another different 7 growing regions (14 700 grain kernels), and the third subset contained the observations of the grain kernels from the remaining 6 growing regions (12 600 grain kernels). Using any two of the three data subsets as the training data to derive the classification criterion (classifier) and the remaining one as the testing data to test the derived classifier, three training and testing data set pairs (Sets 1, 2, and 3) were available. Correspondingly, three classification results were obtained for each of the two statistical classifiers (parametric and non-parametric) with each of the three feature models. The average of the three classification results was computed and considered as the classification result for the classifier with the feature model. The means of the classification accuracies for each of the five classes were calculated and used as a measure of the discrimination ability of that classifier with that feature model. For comparison, a neural network classifier was applied with the feature model which resulted in the highest mean classification accuracy when used with either the parametric or non-parametric statistical classifier.

6.3.2 Identification of damaged CWRS wheat kernels

Both the parametric and non-parametric statistical classifiers were used with three types of feature models: namely morphological, color, and combined (morphological and color). For each type of the feature models, an optimal set of features was selected using the feature selection method described in **Section 6.2**. The data set consisted of 7000 observations of CWRS wheat kernels in seven categories: undamaged, broken, black-point/smudged, grass-green/green-frosted, mildewed, heated, and bin-/fire-burnt. Each category, containing 1000 observations (kernels), was considered as a *class*. The hold-out method was used for the 7-class classification analysis. The data set was split into three subsets. The first subset contained the observations of the 300 randomly selected kernels for each of the 7 classes (2100 kernels in total), the second subset contained the observations of another 300 randomly selected kernels for each class (2100 kernels in total), and the third subset contained the observations of the remaining 400 kernels in each class (2800 kernels in total). Using any two of the three data subsets as the training data to derive the classification criterion (classifier) and the remaining one as the testing data to test the derived classifier, three training and testing data set pairs (Sets 1, 2, and 3) were available. Correspondingly, three classification results were obtained for each of the two statistical classifiers (parametric and non-parametric) with each of the three feature models. The average of the three classification results was computed and considered as the classification result for the classifier with the feature model. The means of the classification accuracies for each of the seven classes were calculated and used as a measure of the discrimination ability of that classifier with that feature model. For comparison, a neural network classifier was applied with the feature model which resulted in the highest mean classification accuracy when used with either the parametric or non-parametric statistical classifier.

6.3.3 Grain type identification of bulk grain samples

An optimal set of features was selected from the 114 extracted color features using the method described in **Section 6.2**. Both the parametric and non-parametric statistical classifiers were used with selected features. The data set consisted of 420 observations of bulk grain samples of five grain types and seven grain categories: CWRS wheat grade 1, 2, and 3; CWAD wheat; barley; rye; and oats. Each category contained 60 observations (bulk samples) from 20 growing regions (3 samples per region). Each grain type was considered as a *class* (CWRS wheat grade 1, 2, and 3 were treated as a single class). The hold-out method was used for the 5-class classification analysis. The data set was split into three subsets according to the growing regions. The first subset contained the observations of the grain kernels from 7 growing regions (147 samples), the second subset contained the observations of the grain kernels from another different 7 growing regions (147 samples), and the third subset contained the observations of the grain samples from the remaining 6 growing regions (126 samples).Using any two of the three data subsets as the training data to derive the classification criterion (classifier) and the remaining one as the testing data to test the derived classifier, three training and testing data set pairs (Set 1, 2, and 3) were available. Correspondingly, three classification results were obtained for each of the two statistical classifiers (parametric and non-parametric) with the selected feature model. The average of the three classification results was computed and considered as the classification result for the classifier with the feature model. The means of the classification accuracies for each of the five classes were calculated and used as a measure of the discrimination ability of that classifier with that feature model. For comparison, a neural network classifier was applied with the selected feature model.

6.3.4 Grade identification of bulk wheat samples

An optimal set of features was selected from the 114 extracted color features using the method described in **Section 6.2**. Both the parametric and non-parametric statistical classifiers were used with the selected features. The data set consisted of 180 observations of bulk CWRS wheat samples collected in three categories: CWRS wheat grade 1, 2, and 3. Each category had 60 observations (bulk samples) from 20 growing regions (3 samples per region). Each grade was considered as a *class*. The hold-out method was used for the 3-class classification analysis. The data set was split into three subsets according to the growing regions. The first subset contained the observations of the grain samples from 7 growing regions (21 samples), the second subset contained the observations of the grain kernels from another different 7 growing regions (21 samples), and the third subset contained the observations of the grain samples from the remaining 6 growing regions (18 samples).Using any two of the three data subsets as the training data to derive the classification criterion (classifier) and the remaining one as the testing data to test the derived classifier, three training and testing data set pairs (Set 1, 2, and 3) were available. Correspondingly, three classification results were obtained for each of the two statistical classifiers (parametric and non-parametric) with the selected feature model. The average of the three classification results was computed and considered as the classification result for the classifier with the feature model. The means of the classification accuracies for each of the three classes were calculated and used as a measure of the discrimination ability of that classifier with that feature model. For comparison, a neural network classifier was applied with the selected feature model.

VII RESULTS AND DISCUSSIONS

7.1 Illumination Design

7.1.1 Test I: sensitivity to lamp voltage variations

Fig 7.1 shows the average Rv, Gv, and Bv of the five replicate tests (see **Section 3.2.3**) with the lamp voltages in the range of V_R - 1.0 V to V_R +1.0 V for the different light sources. Data for each curve were consistent with standard deviations for the R, G, and B color bands and for the 21 voltage levels being less than 0.0016, 0.0016, 0.0025, and 0.0020 for the incandescent, halogen, fluorescent, and controlled fluorescent lamps, respectively.

The results showed that the output intensities of the R, G, and B signals varied linearly with the lamp voltage for all of the light sources. Given a 1 V change from the rated supply voltage, the maximum changes among the three color signals occurred in the blue (1.8%), blue (1.3%), green (0.5%), and green(0.5%) bands for the incandescent, halogen, fluorescent, and controlled fluorescent lamps, respectively.

The Rv, Gv, and Bv values were affected differently by a voltage change for a given lamp type. Slopes were different among the colors for the incandescent and halogen bulbs (**Figs 7.1(a)** and **(b)**) but were nearly identical for the fluorescent lamp (**Fig 7.1(c)**). These results indicated that lamp voltage changes from the rated supply voltage caused slight color shifts in the light outputs of the incandescent and halogen bulbs and little change in the spectral output from the fluorescent lamp.

Incorporating the light controller in the power supply for the fluorescent lamp (**Fig**

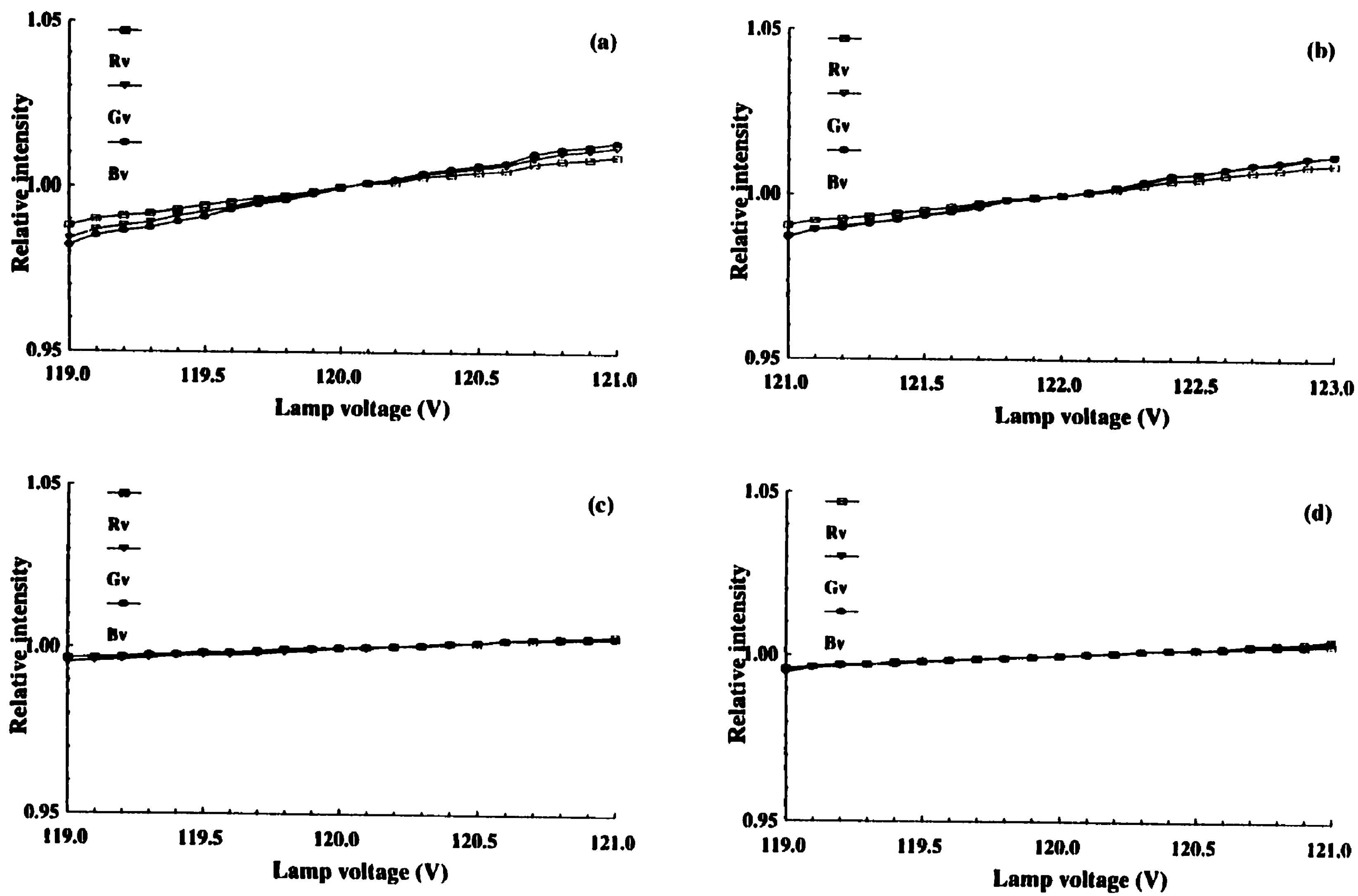


Fig 7.1 Light sensitivities to lamp voltage variations for incandescent (a), halogen (b), fluorescent (c), and controlled fluorescent (d) lamps.

7.1(d)) did not show any improvement, because the changes in the light output caused by the changes in the lamp voltage were within the control accuracy of the light controller.

7.1.2 Test II: stability with time

Fig 7.2 shows the average Rt, Gt, and Bt of the five replicate tests (see **Section 3.2.4**)over a duration of 8 h for the different light sources. Data for each curve were consistent with standard deviations for the three curves and for the 48 time intervals being less than 0.007, 0.011, 0.020, and 0.011 for the incandescent, halogen, fluorescent, and controlled fluorescent lamps, respectively.

The results (**Figs 7.2(a)**, **(b)**, and **(c)**) showed that there were significant changes in the outputs from the three light sources over 8 h. The three color signals of the light changed differently with maximum differences of 0.6, 0.9, and 5.6% in the red, 4.7, 5.0, and 7.7% in the green, and 2.5, 3.0, and 6.4% in the blue, for the incandescent, halogen, and fluorescent lamps, respectively. This indicated that not only the light intensities but also the light colors changed.

The general trends of the curves showed that the major variations occurred within the first 3 h. This may have been due to the ambient temperature changes in the illumination chamber, which increased after the light sources were switched on.

The light levels of the incandescent and halogen bulbs varied in a similar way such that the G signals varied the most, followed by B then R. The R signals were actually quite stable with less than 0.1% variations over 8 h. All of the three color signals of the fluorescent tube dropped significantly over the 8 h. As with the incandescent and halogen lamps, the G signal dropped the most, followed by the B and R signals.

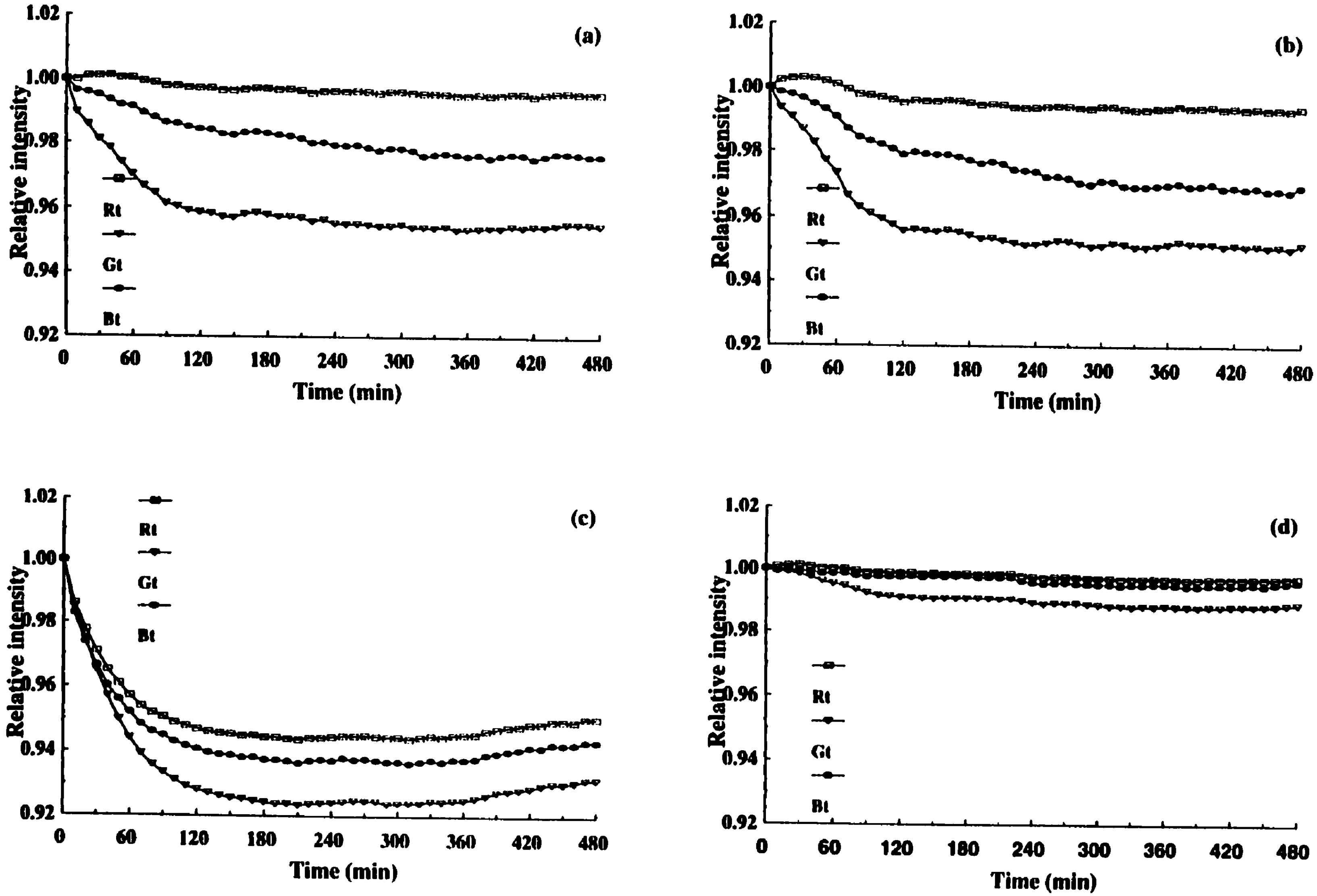


Fig 7.2 Light stabilities with time for incandescent (a), halogen (b), fluorescent (c), and controlled fluorescent (d) lamps.

Incorporating the light controller in the power supply for the fluorescent lamp (**Fig 7.2(d)**) showed a significant improvement in the light stability. The intensity variations in the R, G, and B signals were reduced to 0.5, 1.2, and 0.5% respectively. Again the G signal decreased the most, followed by B then R.

7.1.3 Test III: uniformity over FOV

Fig 7.3 shows the average Rc, Gc, Bc, Rr, Gr, and Br of the ten images (see **Section 3.2.5**)of the Kodak white card under the three light source types. Across the width of the FOV (column number), the maximum intensity variations among the three color signals were 2.1, 2.1, and 3.1% of the overall image intensity means, for the incandescent, halogen, and fluorescent lamps, respectively. Down the depth of the FOV (row number), the maximum intensity variations among the three color signals were 1.0, 1.2, and 1.5% of the overall image intensity means, for the incandescent, halogen, and fluorescent lamps, respectively.

With a similar configuration, the incandescent and halogen bulbs produced an almost identical light distribution over the FOV (**Figs 7.3(a), (a'), (b), and (b')**). The fluorescent and controlled fluorescent lamps produced an identical light distribution (**Figs 7.3(c) and (c')**) with lower intensities at the edge and slightly higher intensities near the center of the FOV. The obvious drop in light intensities at the right edge of the FOV (high column numbers) was due to the power lead junction of the fluorescent lamp.

In spite of the different light sources and position and orientation of the white card, the Rc, Gc, and Bc curves (**Figs 7.3(a), (b), and (c)**) have similar patterns. The Rr, Gr, and Br curves showing variations down the depth of the FOV (**Figs 7.3(a'), (b'), and (c')**) also have common trends. This indicated that there were response variations in each direction of

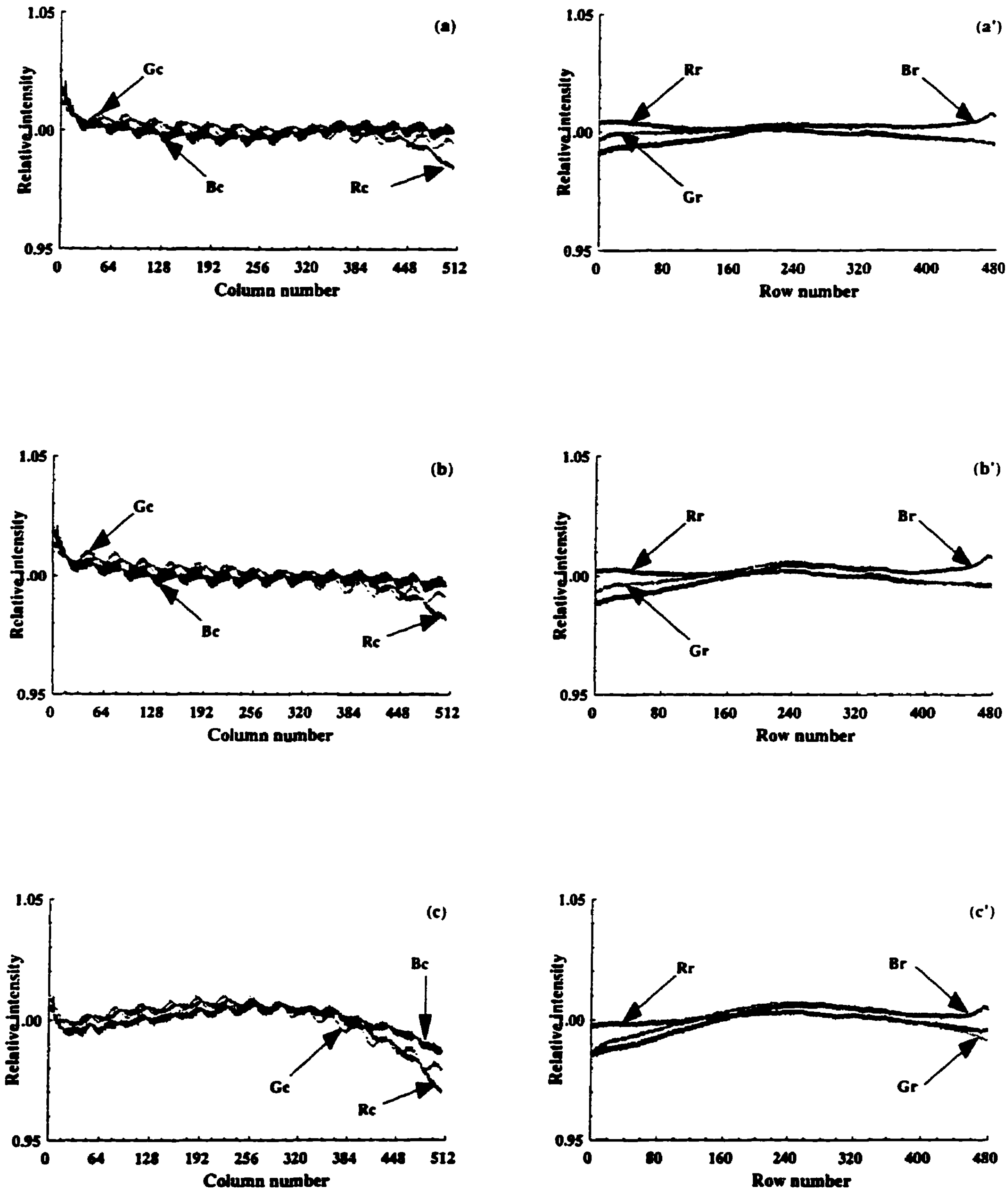


Fig 7.3 Illumination uniformities across (column) and down (row) the field of view for incandescent [(a) and (a')], halogen [(b) and (b')], and fluorescent [(c) and (c')] lamps.

the three color sensor arrays or transmittance variations over the lens.

The test results showed that the greatest intensity variations always occurred at the edges of the FOV because of the variations in the camera's responses as well as the configuration of the light sources. This suggests that when taking images, objects should be placed as close to the center of the FOV as possible. If a 64-pixel wide strip near each of the edges of the FOV is neglected, the intensities varied by less than 1.0% of the mean for all three light source types.

Overall, the controlled fluorescent lamp was the best light source, with the least variations with voltage changes and time and the acceptable uniformity over the FOV. It was chosen as the light source for the image analysis system.

7.2 System Calibration

7.2.1 Aspect-ratio

Table 7.1 lists the pixel numbers of rows and columns required to traverse a Canadian quarter coin in four rectangular-pixel images of the same coin located in the center of the FOV with four different orientations (see **Section 3.3.1**). An aspect-ratio of 1.275 was obtained by dividing the average pixel row number by the average column number.

Table 7.2 lists the pixel numbers of rows and columns required to traverse a Canadian quarter coin in another four rectangular-pixel images of the same coin located in the center of the FOV with four different orientations (see **Section 3.3.1**). The magnification of the camera was about 1.13 times larger than the previous. An aspect ratio of 1.273 was obtained by dividing the average pixel row number by the average column number.

Table 7.1 Pixel numbers of rows and columns required to traverse a Canadian quarter coin in four rectangular-pixel images of the same coin located in the center of the FOV with four different orientations (Resolutions: 0.20 H x 0.16 V mm/pixel).

Image*	Nr (No. of rows)	Nc (No. of columns)
CTR1.XV	151	118
CTR2.XV	150	118
CTR3.XV	151	118
CTR4.XV	150	118
Average	150.5	118

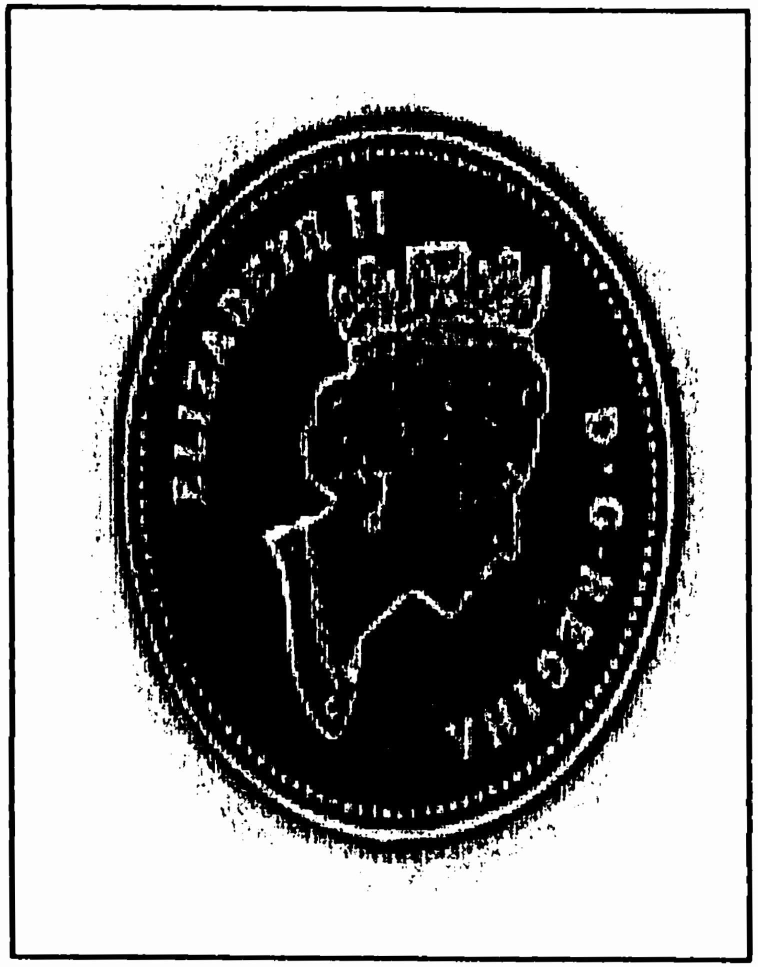
* CTR = center, Arabic numbers represent replicate number, and affix .xv represents viff format image

Table 7.2 Pixel numbers of rows and columns required to traverse a Canadian quarter coin in four rectangular-pixel images of the same coin located in the center of the FOV with four different orientations (Resolutions: 0.18 H x 0.14 V mm/pixel).

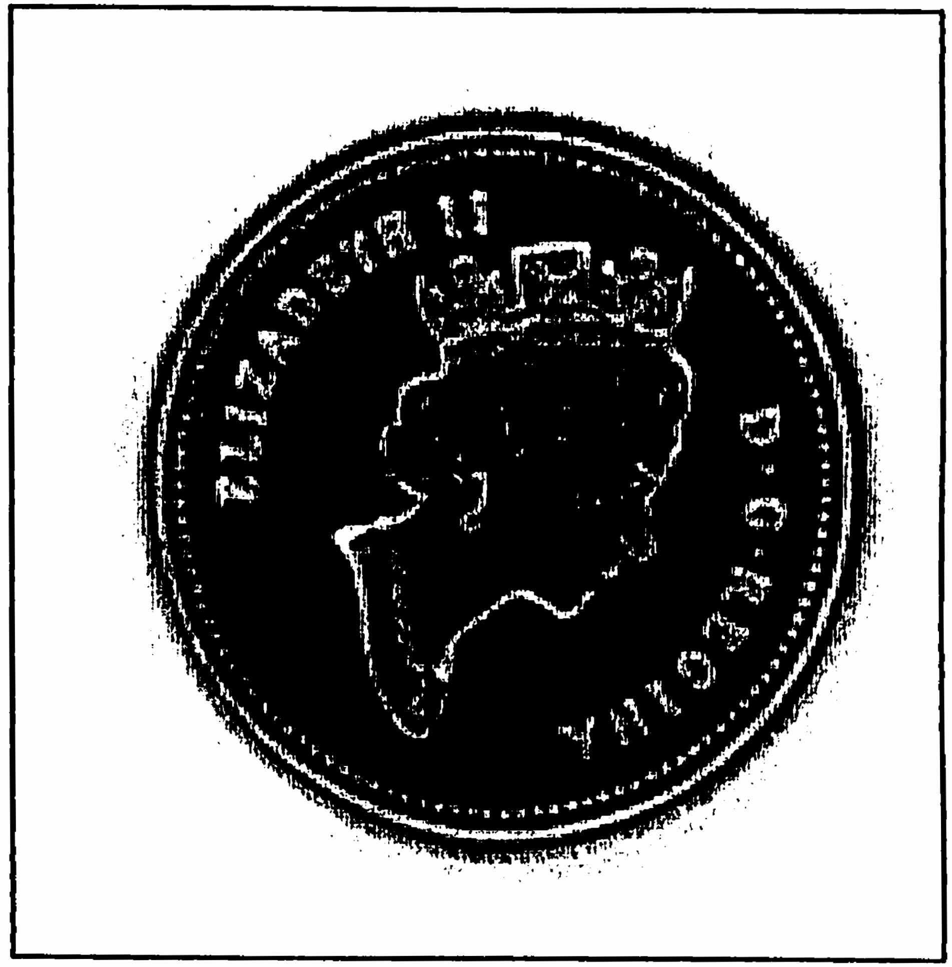
Image	Nr (No. of rows)	Nc (No. of columns)
CTR1'.XV	171	134
CTR2'.XV	170	134
CTR3'.XV	170	133
CTR4'.XV	169	133
Average	170	133.5

The results indicated that the aspect ratio did not change significantly with a slight change (13.0%) in camera's magnification. An aspect of 1.275 was used in the transformation of rectangular- to square-pixel images.

Fig 7.4(a) shows a rectangular-pixel image of a Canadian quarter coin as displayed as a square-pixel (same resolution in vertical and horizontal). The coin image was distorted into an ellipse. **Fig 7.4(b)** shows the square-pixel image transformed from the coin image in **Fig 7.4(a)** using **Equation 3.3**.



(a) Rectangular pixels (152 x 192)



(b) Square pixels (193 x 192)

Fig 7.4 A grey-level image of a Canadian quarter coin illustrating the transformation from rectangular to square pixels.

7.2.2 Image distortion

Table 7.3 lists the numbers of the pixel rows and columns, Nr and Nc, required to traverse a Canadian quarter coin in the twenty rectangular-pixel images of the same coin located in each of the upper and lower corners and the centre of the camera's FOV with different orientations. The numbers of the pixel rows and columns, Nr' and Nc', required to traverse a Canadian quarter coin in the twenty corresponding square-pixel images are also listed in **Table 7.3 (see Section 3.3.3)**.

The results show that Nr and Nc were consistent with a 1 pixel variation over the 4 corners in the camera's FOV, while 2.3 pixel on average larger in the central images. This indicated that the camera was well aligned, but the lens system has a magnification symmetrically decreasing from the center to the edge along the radii. The 1 pixel variation among the row and column numbers in the 4 images at each location was due to the digitization and segmentation processes. After the transformation from rectangular- to square-pixel, the numbers of the rows and columns of the coin, Nr' and Nc', were equal with 1 pixel difference in all images, while the maximum difference in the column number between the central images and the corner images was 2 pixels. This increase was due to the use of the aspect ratio calculated from the central images in the transformation.

In summary, the camera misalignment and the rectangular-to-square pixel transformation did not introduce significant image distortion, compared to the 1 pixel inherent error caused by the digitization and segmentation processes. The lens distortion contributed the most to the image distortion.

Table 7.3 Pixel numbers of rows and columns required to traverse a Canadian quarter coin in the rectangular-pixel and square-pixel images of the same coin located in different portions of the FOV with different orientations.

Image*	Rectangular-pixel		Square-pixels	
	Nr	Nc	Nr'	Nc'
CTR1.XV	151	118	151	151
CTR2.XV	150	118	150	150
CTR3.XV	151	118	151	151
CTR4.XV	150	118	150	151
UL1.XV	149	116	149	148
UL2.XV	148	116	148	148
UL3.XV	148	116	148	148
UL4.XV	148	115	148	147
LL1.XV	148	116	148	148
LL2.XV	148	116	148	148
LL3.XV	148	115	148	148
LL4.XV	148	116	148	148
UR1.XV	149	116	149	148
UR2.XV	149	116	149	148
UR3.XV	149	116	149	148
UR4.XV	149	115	149	148
LR1.XV	148	115	148	147
LR2.XV	148	115	148	148
LR3.XV	148	116	148	147
LR4.XV	148	116	148	148

* UL = upper left, LL = low left, UR = upper right, and LR = low right

Image distortion resulted in a non-uniform spatial resolution over the camera's FOV, which certainly would degrade the system's precision of the size and shape measurements. It is difficult to quantify the degradation accurately, however, a rough estimation can be made for the size measurement under certain assumptions. As the lens distortion was the major contributor to the image distortion, it was simply assumed that the difference between the size measurements of an object made with the object located at two fixed different locations in the camera's FOV was proportional to the real size of the object, and the size measurement was invariant to the object orientation. The maximum difference between the column measurements of the quarter coin (which can roughly be viewed as the measurements of the coin diameter) made with the coin located at the center and the corners of the FOV was 4 pixels as showed in **Table 7.3**. Then a maximum difference of $4*(\alpha/23.689)$ pixels (where 23.689 is the diameter in mm of the coin) would be expected in the measurements for an object with a length of α (mm), if it was measured in the similar way as the coin. For a typical CWRS wheat kernel with a length of 5.7 mm, the maximum difference comes to $4*(5.7/23.689) \approx 0.96$ pixel. This measurement error caused by the image distortion is comparable to the inherent measurement error of 1 pixel caused by the digitization and segmentation processes, which is irrespective to the size of the object being measured. Based on the above estimation, it was assumed that the image distortion does not significantly affect the precision of the size and shape measurements of cereal grain kernels.

7.2.3 Gamma correction

Fig 7.5(a) shows the system outputs (in mean R, G, and B grey-level values) for each reflectance step of the Kodak paper gray scale. Non-linear relationships were observed between the system outputs and the object reflectance. **Fig 7.5(b)** shows the results of removing the gamma correction using **Equation 3.5** with $\gamma = 2.2$.

The maximum range of the r, g, and b values of grain kernels were from 0.41 to 0.97,

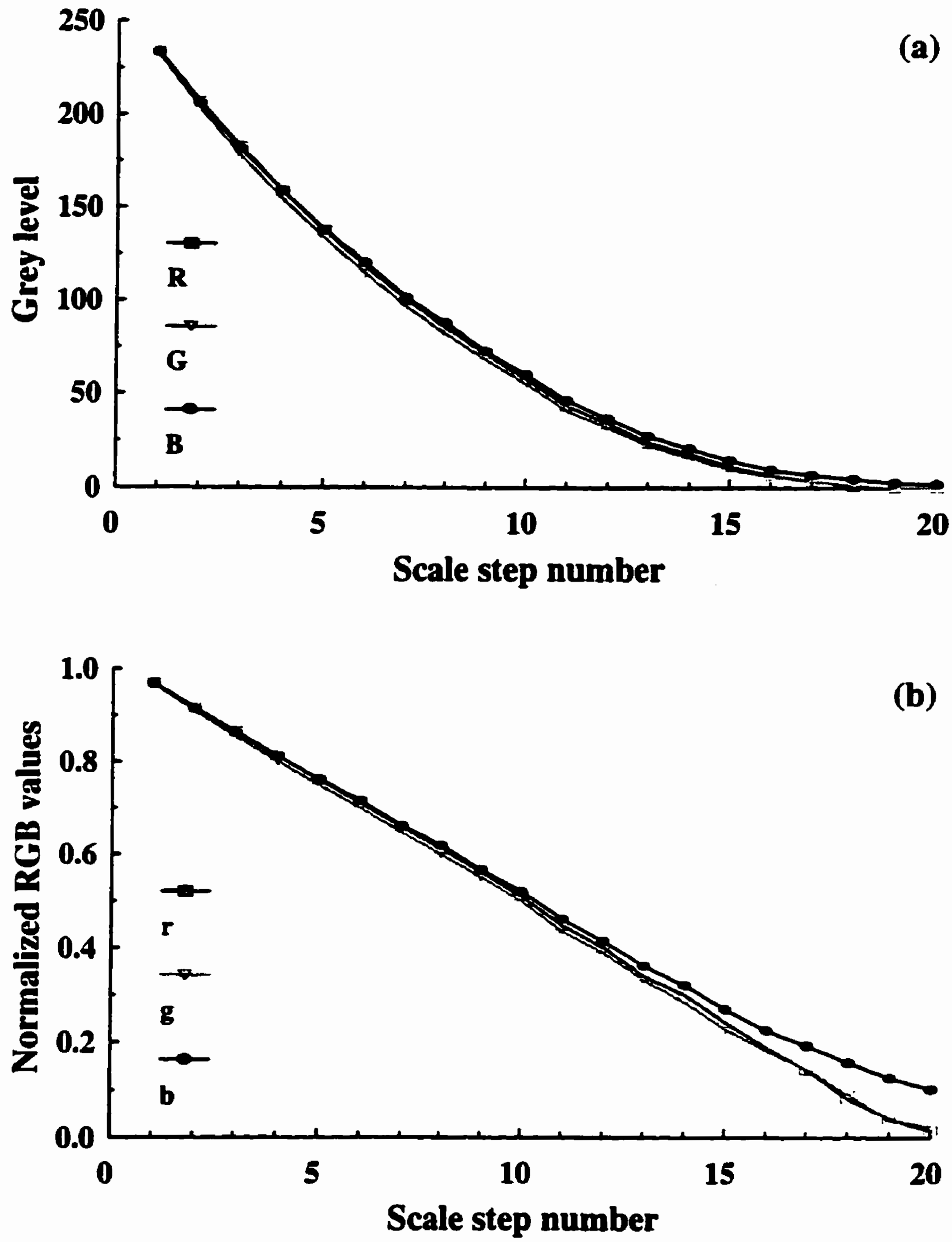


Fig 7.5 System linearity before (a) and after (b) removal of gamma correction.

as measured by the image analysis system in 18000 CWRS, 6000 CWAD wheat, 6000 barley, 6000 rye, and 6000 oats kernels. Within this range, the relationships between r, g, and b and the object reflectance can be viewed as linear.

7.3 Grain Type Identification of Individual Grain Kernels

7.3.1 Morphological feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 65 features from the 68 extracted morphological features and ranked them according to their contributions to the discriminatory powers of the feature corresponding model (**Appendix D-1**). **Table 7.4** lists the first 28 steps for selecting up to 28 best morphological features. The discriminating abilities of the feature models Im4 (the best 4 morphological features), Im8 (the best 8 morphological features), Im12 (the best 12 morphological features), ..., and Im28 (the best 28 morphological features) were evaluated using SAS DISCRIM (**Appendix E-1**). For both the parametric (quadratic) and non-parametric (k-nearest neighbor) classifiers, the mean classification accuracies increased to a certain extent and then remained relatively constant as the number of features increased (**Fig 7.6(a)**). For all examined morphological models, the mean classification accuracies were higher with the non-parametric classifiers than with the parametric classifiers, suggesting that the assumption of multivariate normal distribution did not hold firmly for the extracted morphological feature data of individual grain kernels. The highest mean classification accuracy (94.8%) was obtained with the feature model Im24 using the non-parametric classifier. So the model Im24 was chosen as the morphological feature model for the hold-out grain-type classification analysis of

Table 7.4 The first 28 steps for selecting up to 28 best morphological features by SAS STEPDISC for grain type identification analysis of individual grain kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	L[†]		1	0.8886	83785.35	0.0001	0.1114	0.0001	0.2222	0.0001
2	AS13		2	0.5433	12490.61	0.0001	0.0509	0.0001	0.3379	0.0001
3	Var_R		3	0.5807	14541.24	0.0001	0.0213	0.0001	0.4737	0.0001
4	areaR		4	0.3231	5010.18	0.0001	0.0144	0.0001	0.5330	0.0001
5	R_{max}		5	0.1830	2352.15	0.0001	0.0118	0.0001	0.5556	0.0001
6	R_{min}		6	0.1255	1506.01	0.0001	0.0103	0.0001	0.5708	0.0001
7	hraR		7	0.1564	1946.56	0.0001	0.0087	0.0001	0.5891	0.0001
8	R_{mean}		8	0.0899	1036.94	0.0001	0.0079	0.0001	0.5974	0.0001
9	RS1		9	0.0707	798.38	0.0001	0.0074	0.0001	0.6057	0.0001
10	PS13		10	0.0543	603.07	0.0001	0.0070	0.0001	0.6111	0.0001
11	mnt2		11	0.0484	533.49	0.0001	0.0066	0.0001	0.6157	0.0001
12	W		12	0.0520	575.17	0.0001	0.0063	0.0001	0.6209	0.0001
13	AS4		13	0.0442	484.91	0.0001	0.0060	0.0001	0.6245	0.0001
14	AS7		14	0.0364	396.25	0.0001	0.0058	0.0001	0.6273	0.0001
15	P		15	0.0306	330.74	0.0001	0.0056	0.0001	0.6310	0.0001
16	thnR		16	0.0325	352.51	0.0001	0.0054	0.0001	0.6340	0.0001
17	mnt1		17	0.0506	559.06	0.0001	0.0051	0.0001	0.6395	0.0001
18	RS16		18	0.0289	312.57	0.0001	0.0050	0.0001	0.6419	0.0001
19	AS15		19	0.0241	258.89	0.0001	0.0049	0.0001	0.6435	0.0001
20	RS14		20	0.0235	252.91	0.0001	0.0048	0.0001	0.6472	0.0001
21	PS4		21	0.0197	210.44	0.0001	0.0047	0.0001	0.6488	0.0001
22	AS6		22	0.0179	190.90	0.0001	0.0046	0.0001	0.6498	0.0001
23	rectR		23	0.0159	170.05	0.0001	0.0045	0.0001	0.6508	0.0001
24	mnt4		24	0.0143	152.21	0.0001	0.0044	0.0001	0.6533	0.0001
25	mnt3		25	0.0244	262.00	0.0001	0.0043	0.0001	0.6557	0.0001
26	RS2		26	0.0152	161.41	0.0001	0.0043	0.0001	0.6569	0.0001
27	AS11		27	0.0128	135.58	0.0001	0.0042	0.0001	0.6582	0.0001
28	PS10		28	0.0130	138.07	0.0001	0.0042	0.0001	0.6594	0

* Probability. [§]Average squared canonical correlation. [†] See **Table 5.1** for definitions.

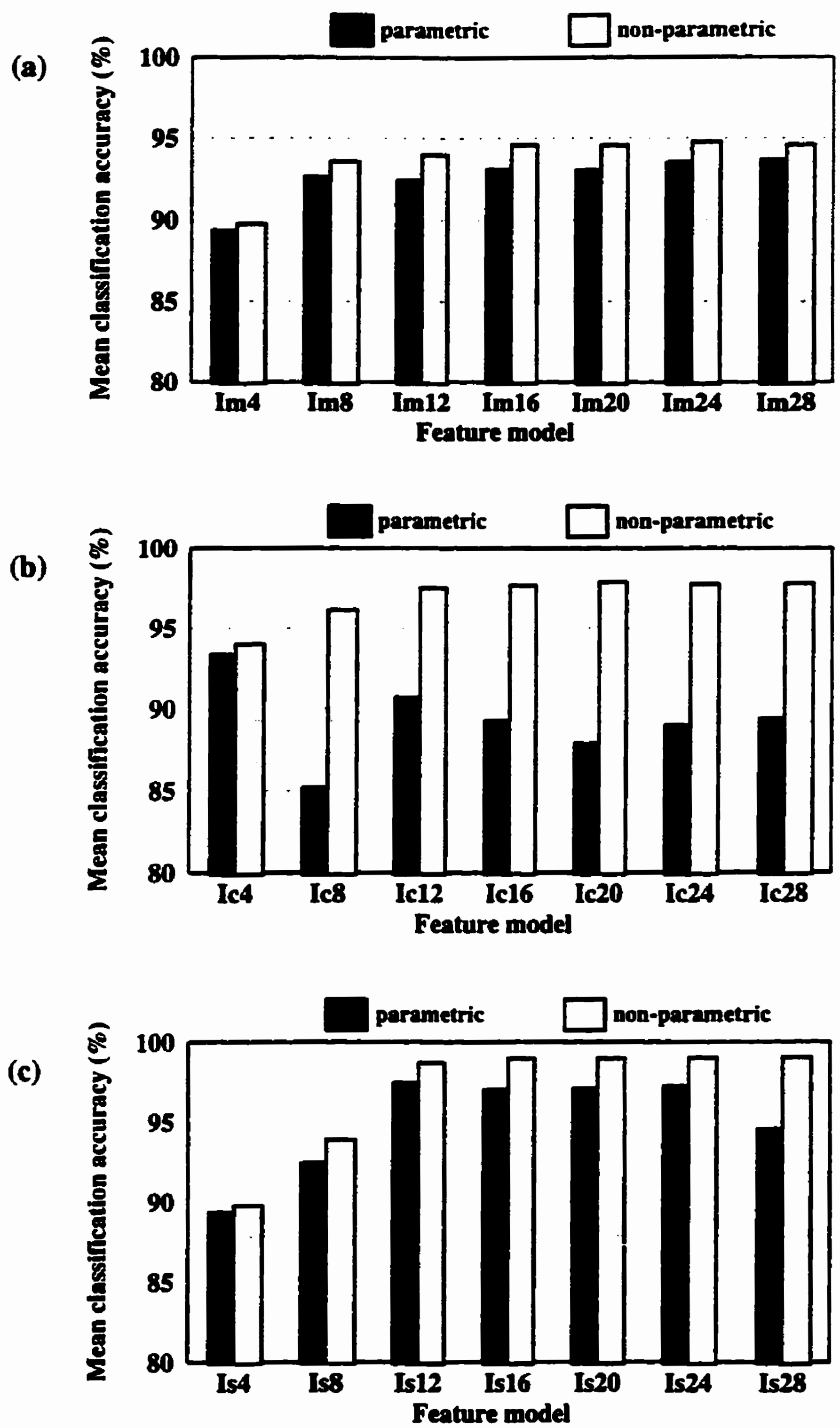


Fig 7.6 Evaluation of morphological (a), color (b), and combined (c) feature models for grain type identification analysis of individual kernels using SAS DISCRIM.

individual kernels.

The hold-out grain-type classification analysis of individual kernels was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-1**) are summarized in **Table 7.5(a)** for the parametric classifier and in **Table 7.5(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 93.6, 84.3, 96.0, 93.5, and 97.3% for CWRS, CWAD, barley, rye, and oats, respectively. For the non-parametric classifier, the average classification accuracies of the three training and testing data sets were 96.3, 87.8, 97.5, 88.4, and 98.0% for CWRS, CWAD, barley, rye, and oats, respectively. The mean classification accuracy for all five types of grains was 93.6% with the non-parametric classifier, which was statistically higher than 92.9% with the parametric classifier. As for individual grain types, the average classification accuracies of the three training and testing data sets were statistically higher with the non-parametric classifier than with the parametric classifier for CWRS, CWAD, barley, and oats, while lower for rye.

The classification accuracies (of the non-parametric classifier) using the different training and testing data sets were generally consistent (with the variations of 3.0, 8.1, 0.7, 0.9, and 0.9% for CWRS, CWAD, barley, rye, and oats, respectively). This suggested that there was no significant difference in the morphological characteristics among grain kernels from different growing regions and the classifier developed based on the selected morphological features was robust.

The major mis-classifications occurred among CWRS, CWAD and rye (**Table 7.5**).

Table 7.5(a) Grain type classification of individual grain kernels by a parametric statistical classifier (quadratic discriminating function) using 24 selected morphological features

Class to ⇒ from ⇓	CWRS No.	CWRS %	Durum No.	Durum %	Barley No.	Barley %	Rye No.	Rye %	Oats No.	Oats %	MCA* %
CWRS											
Set1(6300♀)	5926	**94.1**	301	4.8	6	0.1	66	1.1	1	0.0	
Set2(6300)	5887	**93.4**	370	5.9	3	0.1	39	0.6	1	0.0	
Set3(5400)	5030	**93.2**	286	5.3	6	0.1	75	1.4	3	0.1	
average		**93.6**		5.3		0.1		1.0		0.0	
Durum											
Set1(2100)	29	1.4	1760	**83.8**	3	0.1	308	14.7	0	0.0	
Set2(2100)	152	7.2	1820	**86.7**	8	0.4	120	5.7	0	0.0	
Set3(1800)	43	2.4	1483	**82.4**	9	0.5	265	14.7	0	0.0	
average		3.7		**84.3**		0.3		11.7		0.0	
Barley											
Set1(2100)	1	0.1	3	0.1	1999	**95.2**	38	1.8	59	2.8	
Set2(2100)	0	0.0	16	0.8	2049	**97.6**	16	0.8	19	0.9	
Set3(1800)	6	0.3	32	1.8	1716	**95.3**	24	1.3	22	1.2	
average		0.1		0.9		**96.0**		1.3		1.6	
Rye											
Set1(2100)	5	0.2	89	4.2	13	0.6	1989	**94.7**	4	0.2	
Set2(2100)	20	1.0	163	7.8	18	0.9	1891	**90.1**	8	0.4	
Set3(1800)	1	0.1	59	3.3	14	0.8	1721	**95.6**	5	0.3	
average		0.4		5.1		0.8		**93.5**		0.3	
Oats											
Set1(2100)	0	0.0	3	0.1	51	2.4	18	0.9	2028	**96.6**	
Set2(2100)	0	0.0	0	0.0	22	1.1	17	0.8	2061	**98.1**	
Set3(1800)	0	0.0	1	0.1	36	2.0	13	0.7	1750	**97.2**	
average		0.0		0.1		1.8		0.8		**97.3**	**92.9**

* Mean classification accuracy ♀ Testing data size

Table 7.5(b) Grain type classification of individual grain kernels by a non-parametric statistical (k-nearest neighbour) classifier using 24 selected morphological features

Class to ⇒	CWRS		Durum		Barley		Rye		Oats		Unknown		MCA*
from ⇓	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS													
Set1(6300♀)	6041	**95.9**	206	3.3	9	0.1	29	0.5	1	0.0	14	0.2	
Set2(6300)	6173	**98.0**	94	1.5	2	0.0	12	0.2	1	0.0	18	0.3	
Set3(5400)	5131	**95.0**	188	3.5	6	0.1	30	0.6	2	0.0	43	0.8	
average		**96.3**		2.8		0.1		0.4		0.0		0.4	
Durum													
Set1(2100)	64	3.1	1883	**89.7**	0	0.0	137	6.5	0	0.0	16	0.8	
Set2(2100)	221	10.5	1740	**82.9**	4	0.2	109	5.2	0	0.0	26	1.2	
Set3(1800)	46	2.6	1637	**90.9**	0	0.0	106	5.9	0	0.0	11	0.6	
average		5.4		**87.8**		0.1		5.9		0.0		0.9	
Barley													
Set1(2100)	0	0.0	7	0.3	2040	**97.1**	24	1.1	24	1.1	5	0.2	
Set2(2100)	1	0.1	14	0.7	2046	**97.4**	18	0.9	13	0.6	8	0.4	
Set3(1800)	1	0.1	8	0.4	1761	**97.8**	14	0.8	10	0.6	6	0.3	
average		0.0		0.5		**97.5**		0.9		0.8		0.3	
Rye													
Set1(2100)	6	0.3	195	9.3	12	0.6	1849	**88.1**	4	0.2	34	1.6	
Set2(2100)	27	1.3	167	8.0	10	0.5	1868	**89.0**	1	0.1	27	1.3	
Set3(1800)	5	0.3	165	9.2	6	0.3	1586	**88.1**	2	0.1	36	2.0	
average		0.6		8.8		0.5		**88.4**		0.1		1.6	
Oats													
Set1(2100)	0	0.0	9	0.4	21	1.0	14	0.7	2050	**97.6**	6	0.3	
Set2(2100)	0	0.0	5	0.2	11	0.5	14	0.7	2068	**98.5**	2	0.1	
Set3(1800)	0	0.0	3	0.2	16	0.9	11	0.6	1764	**98.0**	6	0.3	
average		0.0		0.3		0.8		0.7		**98.0**		0.2	**93.6**

* Mean classification accuracy ♀ Testing data size

This happened because the CWAD kernels are more similar in morphology to CWRS and rye kernels than to kernels of other grain types. This result is different from the result reported by Sapirstein and Bushuk (1989) that oats, with the lowest classification score (78.3%), were mainly mis-classified as rye (20.0%), when morphological features were used to differentiate CWRS, barley, oats, and rye. The difference in the result is partially due to the difference in the grain samples used. In their research, a small grain sample from limited sources was used and CWAD was not included. It was hypothesized that inclusion of color features would improve the classification accuracies of these grain types because of their differences in color.

7.3.2 Color feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 65 features from the 78 extracted color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-1**). **Table 7.6** lists the first 28 steps for selecting up to 28 best color features. The discriminating abilities of the feature models Ic4 (the best 4 color features), Ic8 (the best 8 color features), Ic12 (the best 12 color features), ..., and Ic28 (the best 28 color features) were evaluated using SAS DISCRIM (**Appendix E-1**). The mean classification accuracies were statistically significantly higher with the non-parametric (k-nearest neighbor) classifiers than with the parametric (quadratic) classifiers (**Fig 7.6(b)**), indicating that the extracted color feature data did not follow the multivariate normal distributions. For the non-parametric classifiers, as with the morphological models, the mean classification accuracy increased to a certain extent and then remained relatively constant as the number of features increased.

Table 7.6 The first 28 steps for selecting up to 28 best color features by SAS STEPDISC for grain type identification analysis of individual grain kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	mntr2[†]		1	0.7064	25258.34	0.0001	0.2936	0.0001	0.1766	0.0001
2	r_{mean}		2	0.4900	10086.42	0.0001	0.1497	0.0001	0.2646	0.0001
3	mntg2		3	0.3540	5753.33	0.0001	0.0967	0.0001	0.3293	0.0001
4	I_{mean}		4	0.5376	12203.04	0.0001	0.0447	0.0001	0.4486	0.0001
5	hstR1		5	0.1908	2475.88	0.0001	0.0362	0.0001	0.4728	0.0001
6	hstB1		6	0.2426	3362.43	0.0001	0.0274	0.0001	0.5072	0.0001
7	Δb		7	0.1761	2243.97	0.0001	0.0226	0.0001	0.5391	0.0001
8	mntr1		8	0.1172	1394.08	0.0001	0.0199	0.0001	0.5520	0.0001
9	g_{mean}		9	0.1068	1255.05	0.0001	0.0178	0.0001	0.5640	0.0001
10	mntb2		10	0.0843	965.99	0.0001	0.0163	0.0001	0.5728	0.0001
11	mntg1		11	0.0735	832.42	0.0001	0.0151	0.0001	0.5776	0.0001
12	hstG9		12	0.0706	797.13	0.0001	0.0140	0.0001	0.5833	0.0001
13	hstR9		13	0.0518	573.06	0.0001	0.0133	0.0001	0.5905	0.0001
14	Var_r		14	0.0426	466.89	0.0001	0.0127	0.0001	0.5957	0.0001
15	S_{mean}		15	0.0376	409.70	0.0001	0.0123	0.0001	0.5990	0.0001
16	hstB2		16	0.0491	541.83	0.0001	0.0117	0.0001	0.6033	0.0001
17	Var_I		17	0.0413	452.36	0.0001	0.0112	0.0001	0.6069	0.0001
18	Var_S		18	0.0538	596.82	0.0001	0.0106	0.0001	0.6104	0.0001
19	Var_g		19	0.0519	575.04	0.0001	0.0100	0.0001	0.6162	0.0001
20	mntb4		20	0.0379	413.62	0.0001	0.0097	0.0001	0.6205	0.0001
21	hstR10		21	0.0327	354.23	0.0001	0.0093	0.0001	0.6238	0.0001
22	hstG8		22	0.0399	436.18	0.0001	0.0090	0.0001	0.6274	0.0001
23	hstB6		23	0.0425	465.76	0.0001	0.0086	0.0001	0.6334	0.0001
24	hstG7		24	0.0665	747.10	0.0001	0.0080	0.0001	0.6417	0.0001
25	hstR5		25	0.0351	381.37	0.0001	0.0077	0.0001	0.6454	0.0001
26	hstG3		26	0.0232	249.65	0.0001	0.0076	0.0001	0.6473	0.0001
27	hstR11		27	0.0223	239.07	0.0001	0.0074	0.0001	0.6492	0.0001
28	hstG10		28	0.0610	681.82	0.0001	0.0069	0.0001	0.6547	0

* Probability. [§]Average squared canonical correlation. [†] See **Table 5.2** for definitions.

For the parametric classifiers, the mean classification accuracy varied considerably with the feature model. Since the highest mean classification accuracy (97.9%) was obtained using the non-parametric classifier with the feature model Ic20, this model was chosen as the color feature model for the hold-out grain-type classification analysis of individual kernels.

The hold-out grain-type classification analysis of individual kernels was carried out using the three pairs of training and testing data sets using both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-1**) are summarized in **Table 7.7(a)** for the parametric classifier and in **Table 7.7(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 73.7, 84.6, 92.7, 98.9, and 99.2% for CWRS, CWAD, barley, rye, and oats, respectively. For the non-parametric classifier, the average classification accuracies of the three training and testing data sets were 96.7, 95.4, 94.8, 97.3, and 97.9% for CWRS, CWAD, barley, rye, and oats, respectively. The mean classification accuracy for all five types of grains was 96.4% with the non-parametric classifier, which was statistically higher than 89.8% with the parametric classifier. As for individual grain types, the average classification accuracies of the three training and testing data sets were statistically higher with the non-parametric classifier than with the parametric classifier for CWRS, CWAD, and barley, while lower for rye and oats.

Compared to the classification results using the morphological feature model, larger variations (6.3, 6.3, 9.5, 4.0, and 5.4% for CWRS, CWAD, barley, rye, and oats, respectively) existed in the classification accuracies (non-parametric classifier) using the different training and testing data sets, suggesting that larger differences existed in the color

Table 7.7(a) Grain type classification of individual grain kernels by a parametric statistical classifier (quadratic discriminating function) using 20 selected color features

Class to →	CWRS		CWAD		Barley		Rye		Oats		MCA*
from ↓	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS											
Set1(6300♀)	4975	**79.0**	1080	17.1	101	1.6	144	2.3	0	0.0	
Set2(6300)	2782	**51.5**	2569	47.6	23	0.4	26	0.5	0	0.0	
Set3(5400)	4897	**90.7**	385	7.1	33	0.6	85	1.6	0	0.0	
average		**73.7**		24.0		0.9		1.5		0.0	
CWAD											
Set1(2100)	17	0.8	1652	**78.7**	329	15.7	102	4.9	0	0.0	
Set2(2100)	3	0.2	1784	**99.1**	4	0.2	9	0.5	0	0.0	
Set3(1800)	15	0.8	1370	**76.1**	221	12.3	194	10.8	0	0.0	
average		0.6		**84.6**		9.4		5.4		0.0	
Barley											
Set1(2100)	40	1.9	12	0.6	1909	**90.9**	58	2.8	81	3.9	
Set2(2100)	7	0.4	174	9.7	1611	**89.5**	5	0.3	3	0.2	
Set3(1800)	6	0.3	29	1.6	1757	**97.6**	3	0.2	5	0.3	
average		0.9		4.0		**92.7**		1.1		1.4	
Rye											
Set1(2100)	10	0.5	5	0.2	11	0.5	2074	**98.8**	0	0.0	
Set2(2100)	3	0.2	9	0.5	6	0.3	1782	**99.0**	0	0.0	
Set3(1800)	0	0.0	2	0.1	11	0.6	1783	**99.1**	4	0.2	
average		0.2		0.3		0.5		**98.9**		0.1	
Oats											
Set1(2100)	4	0.2	0	0.0	25	1.2	5	0.2	2066	**98.4**	
Set2(2100)	0	0.0	0	0.0	8	0.4	0	0.0	1792	**99.6**	
Set3(1800)	0	0.0	0	0.0	6	0.3	0	0.0	1794	**99.7**	
average		0.1		0.0		0.7		0.1		**99.2**	**89.8**

* Mean classification accuracy ♀ Testing data size

Table 7.7(b) Grain type classification of individual grain kernels by a non-parametric statistical (k-nearest neighbour) classifier using 20 selected color features

Class to ⇒ from ⇓	CWRS No.	CWRS %	CWAD No.	CWAD %	Barley No.	Barley %	Rye No.	Rye %	Oats No.	Oats %	Unknown No.	Unknown %	MCA* %
CWRS													
Set1(6300♀)	6190	**98.3**	35	0.6	1	0.0	70	1.1	0	0.0	4	0.1	
Set2(6300)	6241	**99.1**	41	0.7	2	0.0	11	0.2	0	0.0	5	0.1	
Set3(5400)	5008	**92.7**	351	6.5	23	0.4	10	0.2	0	0.0	8	0.2	
average		**96.7**		2.6		0.2		0.5		0.0		0.1	
CWAD													
Set1(2100)	23	1.1	2033	**96.8**	33	1.6	8	0.4	1	0.1	2	0.1	
Set2(2100)	74	3.5	1922	**91.5**	81	3.9	9	0.4	1	0.1	13	0.6	
Set3(1800)	8	0.4	1761	**97.8**	20	1.1	5	0.3	0	0.0	6	0.3	
average		1.7		**95.4**		2.2		0.4		0.0		0.4	
Barley													
Set1(2100)	6	0.3	111	5.3	1858	**88.5**	16	0.8	98	4.7	11	0.5	
Set2(2100)	1	0.1	20	1.0	2057	**98.0**	5	0.2	13	0.6	4	0.2	
Set3(1800)	0	0.0	29	1.6	1763	**97.9**	0	0.0	6	0.3	2	0.1	
average		0.1		2.6		**94.8**		0.3		1.9		0.3	
Rye													
Set1(2100)	17	0.8	74	3.5	5	0.2	1990	**94.8**	0	0.0	14	0.7	
Set2(2100)	5	0.2	13	0.6	2	0.1	2068	**98.5**	0	0.0	12	0.6	
Set3(1800)	0	0.0	10	0.6	8	0.4	1778	**98.8**	1	0.1	3	0.2	
average		0.4		1.6		0.3		**97.3**		0.0		0.5	
Oats													
Set1(2100)	0	0.0	2	0.1	11	0.5	0	0.0	2085	**99.3**	2	0.1	
Set2(2100)	0	0.0	0	0.0	108	5.1	3	0.1	1984	**94.5**	5	0.2	
Set3(1800)	0	0.0	0	0.0	3	0.2	0	0.0	1797	**99.8**	0	0.0	
average		0.0		0.0		1.9		0.1		**97.9**		0.1	**96.4**

* Mean classification accuracy ♀ Testing data size

than in the morphological characteristics of the grain kernels from the different growing regions.

For all types of grains, except for barley, the average classification accuracies using the color feature model (non-parametric classifier) were comparable to or higher than the classification results using the morphological feature model (**Fig 7.7**). In particularly, substantial improvements in the classification accuracies of CWAD and rye demonstrated the significant advantage of the color features over the morphological features in differentiating the different types of grains. With the lowest classification accuracy, barley kernels were mis-classified as CWAD wheat kernels (2.6%) or oats kernels (1.9%), and vice versa (**Table 7.7(b)**).

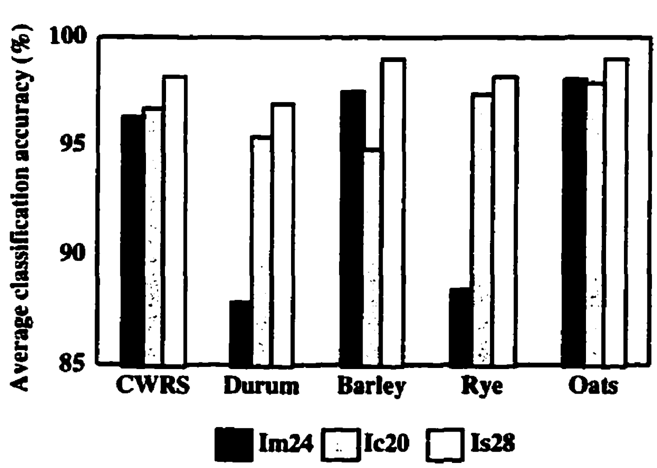


Fig 7.7 A comparison of morphological, color, and combined feature models for grain type identification of individual kernels using non-parametric(k-nearest neighbor) classifiers. (Im24: 24 morphological features; Ic20: 20 color features; Is28: 28 morphological and color features)

Sapirstein and Bushuk (1989) also reported that the greatest degree of mis-classification (approximately 4%) occurred between bright barley and oats kernels when including reflectance in addition to morphological features to discriminate CWRS wheat, barley, oats, and rye kernels (CWAD wheat kernels were not included). However, these three types of grains, CWAD wheat, barley, and oats, were very well differentiated using the morphological features (**Table 7.5(b)**). It was hypothesized that higher classification rates could be obtained by using a combination model of morphological and color features.

7.3.3 Combined feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 129 features from the 146 extracted morphological and color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-1**). **Table 7.8** lists the first 28 steps for selecting up to 28 best combined features. The discriminating abilities of the feature models Is4 (the best 4 combined features, same as Im4), Is8 (the best 8 combined features, including 7 features in Im8 and a color feature hstr12), ..., Is28 (the best 28 combined features, including 15 morphological and 13 color features) were evaluated using SAS DISCRIM (**Appendix E-1**). The mean classification accuracies were higher with the non-parametric (k-nearest neighbor) classifiers than with the parametric (quadratic) classifiers (**Fig 7.6(c)**). For the non-parametric classifiers, as with the morphological models, the mean classification accuracy increased to a certain extent and then remained relatively constant as the number of features increased, while for the parametric classifiers, the mean classification accuracy increased to a maximum as the number of features increased from 4 to 12, then decreased as the number of features increased. Since the highest mean classification accuracy (99.1%) was obtained using the non-parametric classifier with the feature model Is28 (higher mean classification accuracy

Table 7.8 The first 28 steps for selecting up to 28 best combined features by SAS STEPDISC for grain type identification analysis of individual grain kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	L[†]		1	0.8886	83785.35	0.0001	0.1114	0.0001	0.2222	0.0001
2	AS13		2	0.5433	12490.61	0.0001	0.0509	0.0001	0.3379	0.0001
3	Var_R		3	0.5807	14541.24	0.0001	0.0213	0.0001	0.4737	0.0001
4	areaR		4	0.3231	5010.18	0.0001	0.0144	0.0001	0.5330	0.0001
5	hstR12		5	0.2128	2838.64	0.0001	0.0114	0.0001	0.5505	0.0001
6	R_{max}		6	0.1692	2138.46	0.0001	0.0094	0.0001	0.5693	0.0001
7	R_{min}		7	0.1253	1504.24	0.0001	0.0083	0.0001	0.5844	0.0001
8	hraR		8	0.1436	1760.78	0.0001	0.0071	0.0001	0.6012	0.0001
9	Δb		9	0.1016	1187.02	0.0001	0.0064	0.0001	0.6104	0.0001
10	r_{mean}		10	0.0906	1045.20	0.0001	0.0058	0.0001	0.6190	0.0001
11	g_{mean}		11	0.2591	3670.42	0.0001	0.0043	0.0001	0.6558	0.0001
12	b_{mean}		12	0.1292	1557.19	0.0001	0.0037	0.0001	0.6618	0.0001
13	hstB1		13	0.1326	1604.07	0.0001	0.0032	0.0001	0.6770	0.0001
14	S_{mean}		14	0.2376	3271.16	0.0001	0.0025	0.0001	0.7033	0.0001
15	R_{mean}		15	0.0853	978.11	0.0001	0.0023	0.0001	0.7083	0.0001
16	PS13		16	0.0528	584.68	0.0001	0.0021	0.0001	0.7139	0.0001
17	hstG6		17	0.0423	463.27	0.0001	0.0020	0.0001	0.7178	0.0001
18	RS1		18	0.0407	445.79	0.0001	0.0020	0.0001	0.7206	0.0001
19	rctR		19	0.0391	427.08	0.0001	0.0019	0.0001	0.7241	0.0001
20	mnt1		20	0.0393	429.62	0.0001	0.0018	0.0001	0.7263	0.0001
21	Δr		21	0.0344	374.04	0.0001	0.0017	0.0001	0.7300	0.0001
22	AS4		22	0.0325	352.83	0.0001	0.0017	0.0001	0.7321	0.0001
23	hstB6		23	0.0317	343.56	0.0001	0.0016	0.0001	0.7342	0.0001
24	AS15		24	0.0308	333.31	0.0001	0.0016	0.0001	0.7362	0.0001
25	RS16		25	0.0283	305.10	0.0001	0.0015	0.0001	0.7378	0.0001
26	hstG5		26	0.0216	231.19	0.0001	0.0015	0.0001	0.7404	0.0001
27	hstR14		27	0.0245	263.51	0.0001	0.0015	0.0001	0.7428	0.0001
28	hstG13		28	0.0261	281.44	0.0001	0.0014	0.0001	0.7439	0

* Probability. [§]Average squared canonical correlation. [†] See **Tables 5.1 and 5.2** for definitions.

may be obtained using more features, however, it was concluded from the trend (**Fig 7.6(c)**) that the improvement was negligible), this model was chosen as the combined feature model for the hold-out grain-type classification analysis of individual kernels.

The hold-out grain-type classification analysis of individual kernels was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-1**) are summarized in **Table 7.9(a)** for the parametric classifier and in **Table 7.9(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 97.2, 82.0, 97.5, 98.1, and 98.8% for CWRS, CWAD, barley, rye, and oats, respectively. For the non-parametric classifier, the average classification accuracies of the three training and testing data sets were 98.2, 96.9, 99.0, 98.2, and 99.0% for CWRS, CWAD, barley, rye, and oats, respectively. The mean classification accuracy for all five types of grains was 98.3% with the non-parametric classifier, which was statistically significantly higher than 94.7% with the parametric classifier. As for individual grain types, the average classification accuracies of the three training and testing data sets were higher with the non-parametric classifier than with the parametric classifier for all types of grains.

Compared to the classification results using the morphological or the color feature model alone, the variations (3.5, 4.6, 1.6, 2.4, and 1.8% for CWRS, CWAD, barley, rye, and oats, respectively) in the classification accuracies (of the non-parametric classifier) using the different training and testing data sets were generally less than using the color feature model, but larger than using the morphological model. It still could be considered that there was no

Table 7.9(a) Grain type classification of individual grain kernels by a parametric statistical classifier (quadratic discriminating function) using 28 selected combined features

Class to ⇒	CWRS		CWAD		Barley		Rye		Oats		MCA*
from ⇓	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS											
Set1(6300♀)	6083	**96.6**	99	1.6	6	0.1	112	1.8	0	0.0	
Set2(6300)	6171	**98.0**	51	0.8	5	0.1	73	1.2	0	0.0	
Set3(5400)	5241	**97.1**	84	1.6	0	0.0	75	1.4	0	0.0	
average		**97.2**		1.3		0.1		1.4		0.0	
CWAD											
Set1(2100)	21	1.0	1815	**86.4**	17	0.8	119	5.7	128	6.1	
Set2(2100)	140	6.7	1801	**85.8**	12	0.6	147	7.0	0	0.0	
Set3(1800)	31	1.7	1327	**73.7**	24	1.3	418	23.2	0	0.0	
average		3.1		**82.0**		0.9		12.0		2.0	
Barley											
Set1(2100)	20	1.0	2	0.1	2006	**95.5**	5	0.2	67	3.2	
Set2(2100)	1	0.1	3	0.1	2072	**98.7**	6	0.3	18	0.9	
Set3(1800)	1	0.1	22	1.2	1769	**98.3**	3	0.2	5	0.3	
average		0.4		0.5		**97.5**		0.2		1.4	
Rye											
Set1(2100)	5	0.2	22	1.1	18	0.9	2054	**97.8**	1	0.1	
Set2(2100)	17	0.8	17	0.8	15	0.7	2051	**97.7**	0	0.0	
Set3(1800)	6	0.3	7	0.4	11	0.6	1776	**98.7**	0	0.0	
average		0.5		0.8		0.7		**98.1**		0.0	
Oats											
Set1(2100)	0	0.0	4	0.2	21	1.0	0	0.0	2075	**98.8**	
Set2(2100)	0	0.0	1	0.1	47	2.2	0	0.0	2052	**97.7**	
Set3(1800)	0	0.0	1	0.1	4	0.2	0	0.0	1795	**99.7**	
average		0.0		0.1		1.2		0.0		**98.8**	**94.7**

* Mean classification accuracy ♀ Testing data size

Table 7.9(b) Grain type classification of individual grain kernels by a non-parametric statistical (k-nearest neighbour) classifier using 28 selected combined features

Class to ⇒	CWRS		CWAD		Barley		Rye		Oats		Unknown		MCA*
from ⇓	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS													
Set1(6300♀)	6212	**98.6**	48	0.8	0	0.0	31	0.5	1	0.0	8	0.1	
Set2(6300)	6280	**99.7**	9	0.1	0	0.0	9	0.1	0	0.0	2	0.0	
Set3(5400)	5196	**96.2**	191	3.5	1	0.0	5	0.1	0	0.0	7	0.1	
average		**98.2**		1.5		0.0		0.2		0.0		0.1	
CWAD													
Set1(2100)	9	0.4	2023	**96.3**	0	0.0	6	0.3	60	2.9	2	0.1	
Set2(2100)	86	4.1	2002	**95.3**	1	0.1	7	0.3	0	0.0	4	0.2	
Set3(1800)	2	0.1	1784	**99.1**	2	0.1	7	0.4	0	0.0	5	0.3	
average		1.6		**96.9**		0.1		0.3		1.0		0.2	
Barley													
Set1(2100)	0	0.0	11	0.5	2056	**97.9**	1	0.1	27	1.3	5	0.2	
Set2(2100)	1	0.1	4	0.2	2091	**99.6**	0	0.0	4	0.2	0	0.0	
Set3(1800)	0	0.0	8	0.4	1791	**99.5**	0	0.0	1	0.1	0	0.0	
average		0.0		0.4		**99.0**		0.0		0.5		0.1	
Rye													
Set1(2100)	6	0.3	55	2.6	2	0.1	2030	**96.7**	1	0.1	6	0.3	
Set2(2100)	9	0.4	15	0.7	0	0.0	2072	**98.7**	0	0.0	4	0.2	
Set3(1800)	0	0.0	13	0.7	0	0.0	1784	**99.1**	0	0.0	3	0.2	
average		0.2		1.4		0.0		**98.2**		0.0		0.2	
Oats													
Set1(2100)	0	0.0	7	0.3	4	0.2	0	0.0	2083	**99.2**	6	0.3	
Set2(2100)	0	0.0	3	0.1	33	1.6	2	0.1	2058	**98.0**	4	0.2	
Set3(1800)	0	0.0	0	0.0	2	0.1	1	0.1	1797	**99.8**	0	0.0	
average		0.0		0.2		0.6		0.1		**99.0**		0.2	**98.3**

* Mean classification accuracy ♀ Testing data size

Table 7.9(c) Grain type classification of individual grain kernels by a neural network classifier (28-6-4-5) using 28 selected combined features

Class to ⇒ from ⇓	CWRS No.	CWRS %	CWAD No.	CWAD %	Barley No.	Barley %	Rye No.	Rye %	Oats No.	Oats %	MCA* %
CWRS											
Set1(6300♀)	6197	**98.4**	60	1.0	0	0.0	41	0.7	12	0.2	
Set2(6300)	6274	**99.6**	17	0.3	1	0.0	8	0.1	10	0.2	
Set3(5400)	5252	**97.3**	135	2.5	2	0.0	11	0.2	0	0.0	
average		**98.4**		1.2		0.0		0.3		0.1	
CWAD											
Set1(2100)	17	0.8	2064	**98.3**	8	0.4	8	0.4	3	0.1	
Set2(2100)	149	7.1	1935	**92.1**	5	0.2	11	0.5	0	0.0	
Set3(1800)	11	0.6	1768	**98.2**	11	0.6	10	0.6	0	0.0	
average		2.8		**96.2**		0.4		0.5		0.1	
Barley											
Set1(2100)	0	0.0	17	0.8	2045	**97.4**	1	0.1	37	1.8	
Set2(2100)	1	0.1	12	0.6	2078	**99.0**	0	0.0	9	0.4	
Set3(1800)	0	0.0	19	1.1	1777	**98.7**	0	0.0	4	0.2	
average		0.0		0.8		**98.4**		0.0		0.8	
Rye											
Set1(2100)	14	0.7	50	2.4	4	0.2	2030	**96.7**	2	0.1	
Set2(2100)	43	2.1	20	1.0	1	0.1	2035	**96.9**	1	0.1	
Set3(1800)	19	1.1	5	0.3	0	0.0	1776	**98.7**	0	0.0	
average		1.3		1.2		0.1		**97.4**		0.1	
Oats											
Set1(2100)	0	0.0	5	0.2	16	0.8	0	0.0	2079	**99.0**	
Set2(2100)	0	0.0	3	0.1	59	2.8	0	0.0	2038	**97.1**	
Set3(1800)	0	0.0	4	0.2	6	0.3	0	0.0	1790	**99.4**	
average		0.0		0.2		1.3		0.0		**98.5**	97.8

* Mean classification accuracy ♀ Testing data size

significant difference in the overall characteristics among grain kernels from different growing regions and the grain samples were representative. It was noted that for each feature model, the largest variation in the classification accuracies always occurred to the grain type with the lowest average classification accuracy (**Table 7.5(b)**, **Table 7.7(b)**, and **Table 7.9(b)**).

For all types of grains, except for barley, the average classification accuracies using the combined feature model (non-parametric classifier) were statistically higher than the classification results using the morphological or the color feature model alone (**Fig 7.7**). The major mis-classifications occurred among CWRS, CWAD and rye when using the morphological features alone. The CWAD wheat kernels, with the lowest average classification accuracy (96.9%), were mis-classified as CWRS wheat kernels (1.6%); the rye kernel, with the second lowest average classification accuracy (98.2%), were mis-classified as CWAD wheat kernels (1.4%); and the CWRS wheat kernels, with the next lowest average classification accuracy (98.2%), were mis-classified as CWAD kernels (1.5%) (**Table 7.9(b)**). Overall, using the combined features significantly improved the classification accuracies obtained using the morphological or color features alone in identifying the different type grain kernels.

As a comparison to the statistical classifiers, a MNN classifier with a structure of 28-6-4-5 (four layers with 28 nodes in the input, 6 nodes in the first hidden, 4 nodes in the second hidden, and 5 nodes in the output layer) was used with the combined feature model Is28. The results are summarized in **Table 7.9(c)**. The average classification accuracies were 98.4, 96.2, 98.4, 97.4, and 98.5% for CWRS, CWAD, barley, rye, and oats, respectively, which were slightly lower than using the non-parametric classifier but statistically significantly higher than using the parametric classifier.

From the classification results, it can be concluded that the MNN classifier is better than the parametric (quadratic) classifier, but it cannot be concluded that the non-parametric (k-nearest neighbor) classifier is better than the MNN classifiers. The performance of a MNN classifier strongly depends on the structure of the network, specifically the number of hidden layers and the numbers of nodes in each hidden layer. Since, so far there is no theoretical method for the optimal design of MNN structures, the structure of a MNN classifier can only be determined by experience and experiments for the specific classification problem. Limited by time (the training time required by a MNN classification is usually very long, especially when a large number of features is used with a large size of training data set, as in the case of this study; it took approximately 50 h to train the MNN classifier with a training data set of 27 300 observations of 28 features), only three MNN classifiers with different structures were tested for the classification task, and the one reported was chosen due to its superior performance. Although the classification results (with a mean classification accuracy of 97.8%) were slightly lower than the classification results using the non-parametric statistical classifier, the differences were small. Considering the advantages of neural networks over k-nearest neighbor classifiers in required computer memory and executing (classifying) time, a MNN classifier is still recommended as the first choice for the classification task.

7.4 Identification of Damaged CWRS Wheat Kernels

7.4.1 Morphological feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 57 features from the 68 extracted morphological features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-2**). **Table 7.10** lists the first 28 steps for selecting up to 28 best morphological features. The discriminating abilities of the feature models Dm4 (the best 4 morphological features), Dm8 (the best 8 morphological features), Dm12 (the best 12 morphological features), ..., and Dm28 (the best 28 morphological features) were evaluated using SAS DISCRIM (**Appendix E-2**). The morphological features were not sufficient for distinguishing the healthy kernels from the six types of damaged CWRS wheat kernels (**Fig 7.8(a)**). The highest mean classification accuracy (only 63.4%) was obtained with the feature model Dm28 using the non-parametric (k-nearest neighbor) classifier. The reason behind this incapability of morphological features in differentiating the healthy and different damage types of CWRS wheat kernels is quite obvious because most of the damage types are very similar in morphology to the healthy kernels and to each other, except for the broken and grass-green/green-frosted types. Despite the poor performance in the discriminant analysis, the model Dm28 was still tried as the morphological feature model for the hold-out classification analysis of damaged CWRS wheat kernels.

The hold-out classification analysis of damaged CWRS wheat kernels was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-2**) are

Table 7.10 The first 28 steps for selecting up to 28 best morphological features by SAS STEPDISC for identification analysis of damaged CWRS wheat kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	R_{min}[†]		1	0.4326	888.62	0.0001	0.5674	0.0001	0.0721	0.0001
2	RS1		2	0.2955	488.68	0.0001	0.3998	0.0001	0.1211	0.0001
3	AS7		3	0.2736	438.90	0.0001	0.2904	0.0001	0.1624	0.0001
4	mnt3		4	0.1416	192.14	0.0001	0.2493	0.0001	0.1821	0.0001
5	LPA		5	0.0851	108.40	0.0001	0.2280	0.0001	0.1921	0.0001
6	RS16		6	0.0758	95.48	0.0001	0.2108	0.0001	0.2027	0.0001
7	PS7		7	0.0620	76.95	0.0001	0.1977	0.0001	0.2097	0.0001
8	RS14		8	0.0565	69.70	0.0001	0.1865	0.0001	0.2160	0.0001
9	RS5		9	0.0429	52.22	0.0001	0.1785	0.0001	0.2219	0.0001
10	AS13		10	0.0521	63.92	0.0001	0.1692	0.0001	0.2300	0.0001
11	hraR		11	0.0426	51.80	0.0001	0.1620	0.0001	0.2348	0.0001
12	AS13		12	0.0419	50.83	0.0001	0.1552	0.0001	0.2390	0.0001
13	areaR		13	0.0356	43.01	0.0001	0.1497	0.0001	0.2431	0.0001
14	A		14	0.0349	42.06	0.0001	0.1445	0.0001	0.2466	0.0001
15	R_{mean}		15	0.0740	92.99	0.0001	0.1338	0.0001	0.2536	0.0001
16	L		16	0.0452	55.12	0.0001	0.1277	0.0001	0.2594	0.0001
17	P		17	0.0372	44.90	0.0001	0.1230	0.0001	0.2633	0.0001
18	rctR		18	0.0344	41.42	0.0001	0.1188	0.0001	0.2676	0.0001
19	R_{max}		19	0.0450	54.72	0.0001	0.1134	0.0001	0.2725	0.0001
20	RS6		20	0.0287	34.38	0.0001	0.1102	0.0001	0.2750	0.0001
21	thnR		21	0.0270	32.24	0.0001	0.1072	0.0001	0.2781	0.0001
22	Var_R		22	0.0286	34.27	0.0001	0.1041	0.0001	0.2806	0.0001
23	mnt1		23	0.0425	51.55	0.0001	0.0997	0.0001	0.2853	0.0001
24	mnt2		24	0.0303	36.25	0.0001	0.0967	0.0001	0.2888	0.0001
25	mnt4		25	0.0293	35.06	0.0001	0.0938	0.0001	0.2921	0.0001
26	W		26	0.0255	30.43	0.0001	0.0914	0.0001	0.2948	0.0001
27	radR		27	0.0209	24.75	0.0001	0.0895	0.0001	0.2970	0.0001
28	AS5		28	0.0184	21.76	0.0001	0.0879	0.0001	0.2986	0

* Probability. [§]Average squared canonical correlation. [†] See **Table 5.1** for definitions.

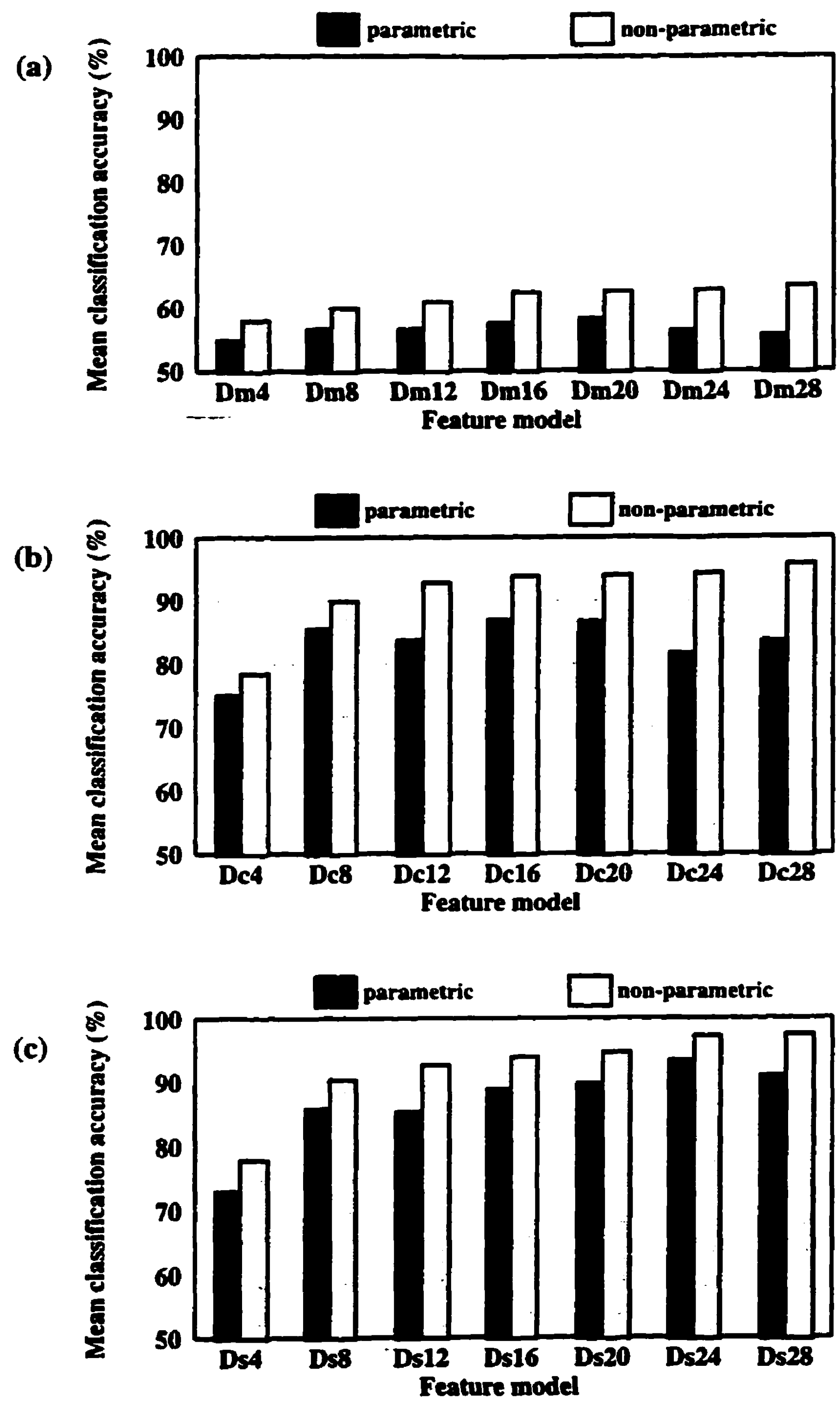


Fig 7.8 Evaluation of morphological (a), color (b), and combined (c) feature models for damage identification analysis of CWRS wheat kernels using SAS DISCRIM.

summarized in **Table 7.11(a)** for the parametric classifier and in **Table 7.11(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 57.2(healthy), 59.4(broken), 58.6(mildewed), 76.5(grass-green/green-frosted), 76.6(black-point/smudged), 25.4(heated), and 71.8(bin-/fire-burnt)%. For the non-parametric classifier, the average classification accuracies of the three training and testing data sets were 43.5(healthy), 57.4(broken), 44.7(mildewed), 64.2(grass-green/green-frosted), 67.8(black-point/smudged), 24.6(heated), and 55.4(bin-/fire-burnt)%. The mean classification accuracy for the healthy and all six type damaged CWRS wheat kernels was 51.1% with the non-parametric classifier, lower than 60.8% with the parametric classifier.

The low differentiating rates for healthy, mildewed, black-point/smudged, heated, and bin-/fire-burnt kernels were expected because of their similarities in kernel morphology. However, the rates for the broken and grass-green/green-frosted kernels were also quite low, despite their morphological differences (smaller sizes and irregular shapes for broken and smaller sizes for grass-green/green-frosted kernels) from the kernels of other damage types. The explanation could be that the features used were selected based on the overall performance in distinguishing the healthy and six types of damaged kernels rather than the performance in distinguishing the kernels of these two damage types from the others. By and large, the morphological features were inadequate in differentiating the healthy and damaged CWRS wheat kernels.

7.4.2 Color feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected

Table 7.11(a) Classification of damaged CWRS wheat kernels by a parametric statistical classifier (quadratic discriminating function) using 28 selected morphological features

Class to → from ↓	Healthy No.	Healthy %	Broken No.	Broken %	Mildewed No.	Mildewed %	Grass-green No.	Grass-green %	Black-point No.	Black-point %	Heated No.	Heated %	Bin/fire burnt No.	Bin/fire burnt %	MCA* %
Healthy															
Set 1(300♀)	175	**58.3**	1	0.3	81	27.0	3	1.0	12	4.0	15	5.0	13	4.3	
Set 2(300)	181	**60.3**	2	0.7	62	20.7	4	1.3	19	6.3	14	4.7	18	6.0	
Set 3(400)	212	**53.0**	3	0.8	146	36.5	2	0.5	10	2.5	15	3.8	12	3.0	
average		**57.2**		0.6		28.1		0.9		4.3		4.5		4.4	
Broken															
Set 1(300)	30	10.0	160	**53.3**	46	15.3	39	13.0	5	1.7	18	6.0	2	0.7	
Set 2(300)	25	8.3	183	**61.0**	39	13.0	34	11.3	2	0.7	14	4.7	3	1.0	
Set 3(400)	29	7.3	255	**63.8**	52	13.0	50	12.5	1	0.3	8	2.0	5	1.3	
average		8.5		**59.4**		13.8		12.3		0.9		4.2		1.0	
Mildewed															
Set 1(300)	64	21.3	3	1.0	175	**58.3**	7	2.3	16	5.3	28	9.3	7	2.3	
Set 2(300)	94	31.3	1	0.3	164	**54.7**	5	1.7	17	5.7	7	2.3	12	4.0	
Set 3(400)	77	19.3	2	0.5	251	**62.8**	15	3.8	11	2.8	17	4.3	27	6.8	
average		24.0		0.6		**58.6**		2.6		4.6		5.3		4.4	
Grass-green															
Set 1(300)	0	0.0	5	1.7	14	4.7	224	**74.7**	3	1.0	26	8.7	28	9.3	
Set 2(300)	1	0.3	11	3.7	24	8.0	217	**72.3**	1	0.3	24	8.0	22	7.3	
Set 3(400)	1	0.3	7	1.8	18	4.5	330	**82.5**	1	0.3	21	5.3	22	5.5	
average		0.2		2.4		5.7		**76.5**		0.5		7.3		7.4	
Black-point															
Set 1(300)	4	1.3	5	1.7	14	4.7	0	0.0	227	**75.7**	16	5.3	34	11.3	
Set 2(300)	5	1.7	4	1.3	8	2.7	1	0.3	257	**85.7**	7	2.3	18	6.0	
Set 3(400)	11	2.8	3	0.8	22	5.5	3	0.8	274	**68.5**	21	5.3	66	16.5	
average		1.9		1.3		4.3		0.4		**76.6**		4.3		11.3	
Heated															
Set 1(300)	16	5.3	6	2.0	21	7.0	11	3.7	66	22.0	103	**34.3**	77	25.7	
Set 2(300)	31	10.3	1	0.3	40	13.3	9	3.0	64	21.3	48	**16.0**	107	35.7	
Set 3(400)	26	6.5	1	0.3	45	11.3	24	6.0	64	16.0	103	**25.8**	137	34.3	
average		7.4		0.9		10.5		4.2		19.8		**25.4**		31.9	
Bin/fire-burnt															
Set 1(300)	6	2.0	2	0.7	15	5.0	7	2.3	22	7.3	38	12.7	210	**70.0**	
Set 2(300)	7	2.3	0	0.0	22	7.3	9	3.0	39	13.0	13	4.3	210	**70.0**	
Set 3(400)	4	1.0	2	0.5	18	4.5	26	6.5	24	6.0	25	6.3	301	**75.3**	
average		1.8		0.4		5.6		3.9		8.8		7.8		**71.8**	**60.8**

* Mean classification accuracy ♀ Testing data size

Table 7.11(b) Classification of damaged CWRS wheat kernels by a non-parametric statistical classifier (k-nearest neighbor) using 28 selected morphological features

Class to →	Healthy		Broken		Mildewed		Grass-green		Black-point		Heated		Bin/fire burnt		Unknown		MCA*
from ↓	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	%
Healthy																	
Set 1(300♀)	132	**44.0**	1	0.3	68	22.7	0	0.0	15	5.0	17	5.7	14	4.7	53	17.7	
Set 2(300)	123	**41.0**	5	1.7	62	20.7	1	0.3	21	7.0	18	6.0	15	5.0	55	18.3	
Set 3(400)	182	**45.5**	4	1.0	106	26.5	1	0.3	17	4.3	16	4.0	7	1.8	67	16.8	
average		**43.5**		1.0		23.3		0.2		5.4		5.2		3.8		17.6	
Broken																	
Set 1(300)	15	5.0	164	**54.7**	27	9.0	13	4.3	6	2.0	25	8.3	9	3.0	41	13.7	
Set 2(300)	15	5.0	183	**61.0**	25	8.3	20	6.7	2	0.7	7	2.3	5	1.7	43	14.3	
Set 3(400)	33	8.3	226	**56.5**	37	9.3	33	8.3	1	0.3	18	4.5	11	2.8	41	10.3	
average		6.1		**57.4**		8.9		6.4		1.0		5.1		2.5		12.8	
Mildewed																	
Set 1(300)	48	16.0	0	0.0	126	**42.0**	4	1.3	19	6.3	30	10.0	16	5.3	57	19.0	
Set 2(300)	53	17.7	5	1.7	138	**46.0**	5	1.7	11	3.7	15	5.0	17	5.7	56	18.7	
Set 3(400)	71	17.8	3	0.8	184	**46.0**	18	4.5	10	2.5	21	5.3	18	4.5	75	18.8	
average		17.1		0.8		**44.7**		2.5		4.2		6.8		5.2		18.8	
Grass-green																	
Set 1(300)	0	0.0	1	0.3	3	1.0	178	**59.3**	7	2.3	19	6.3	36	12.0	56	18.7	
Set 2(300)	4	1.3	1	0.3	17	5.7	195	**65.0**	2	0.7	17	5.7	24	8.0	40	13.3	
Set 3(400)	2	0.5	4	1.0	15	3.8	273	**68.3**	0	0.0	16	4.0	36	9.0	54	13.5	
average		0.6		0.6		3.5		**64.2**		1.0		5.3		9.7		15.2	
Black-point																	
Set 1(300)	1	0.3	0	0.0	12	4.0	0	0.0	210	**70.0**	16	5.3	27	9.0	34	11.3	
Set 2(300)	2	0.7	0	0.0	6	2.0	6	2.0	202	**67.3**	19	6.3	23	7.7	42	14.0	
Set 3(400)	3	0.8	1	0.3	14	3.5	0	0.0	264	**66.0**	30	7.5	36	9.0	52	13.0	
average		0.6		0.1		3.2		0.7		**67.8**		6.4		8.6		12.8	
Heated																	
Set 1(300)	16	5.3	1	0.3	10	3.3	12	4.0	65	21.7	73	**24.3**	62	20.7	61	20.3	
Set 2(300)	19	6.3	0	0.0	18	6.0	9	3.0	39	13.0	68	**22.7**	76	25.3	71	23.7	
Set 3(400)	15	3.8	0	0.0	19	4.8	17	4.3	70	17.5	107	**26.8**	78	19.5	94	23.5	
average		5.1		0.1		4.7		3.8		17.4		**24.6**		21.8		22.5	
Bin/fire-burnt																	
Set 1(300)	4	1.3	1	0.3	16	5.3	4	1.3	29	9.7	31	10.3	153	**51.0**	62	20.7	
Set 2(300)	2	0.7	0	0.0	3	1.0	9	3.0	23	7.7	25	8.3	186	**62.0**	52	17.3	
Set 3(400)	4	1.0	0	0.0	10	2.5	18	4.5	29	7.3	36	9.0	213	**53.3**	90	22.5	
average		1.0		0.1		2.9		2.9		8.2		9.2		**55.4**		20.2	**51.1**

* Mean classification accuracy ♀ Testing data size

69 features from the 78 extracted color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-2**). **Table 7.12** lists the first 28 steps for selecting up to 28 best color features. The mean hue value over a kernel (H_{mean}) was ranked as the most significant color feature for distinguishing the healthy and the different damaged CWRS wheat kernels. The discriminating abilities of the feature models Dc4 (the best 4 color features), Dc8 (the best 8 color features), Dc12 (the best 12 color features), ..., and Dc28 (the best 28 color features) were evaluated using SAS DISCRIM (**Appendix E-2**). The color features were quite powerful in discriminating the healthy and different damaged kernels (**Fig 7.8(b)**). For all examined color models, the mean classification accuracies were higher with the non-parametric (k-nearest neighbor) classifiers than with the parametric (quadratic) classifiers, indicating that the extracted color feature data did not follow the multivariate normal distribution very well. For the non-parametric classifiers, the mean classification accuracy increased to a certain extent and then remained relatively constant as the number of features increased, while for the parametric classifiers, the mean classification accuracy varied non-monotonously with the feature size. Since the highest mean classification accuracy (95.8%) was obtained using the non-parametric classifier with the feature model Dc28, this model was chosen as the color feature model for the hold-out classification analysis of damaged CWRS wheat kernels.

The hold-out classification analysis of damaged CWRS kernels was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-2**) are

Table 7.12 The first 28 steps for selecting up to 28 best color features by SAS STEPDISC for identification analysis of damaged CWRS wheat kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	H_{mean}[†]		1	0.9722	40792.33	0.0001	0.0278	0.0001	0.1620	0.0001
2	hstG10		2	0.6223	1920.39	0.0001	0.0105	0.0001	0.2647	0.0001
3	Δr		3	0.4137	822.06	0.0001	0.0062	0.0001	0.3213	0.0001
4	hstR6		4	0.3531	635.79	0.0001	0.0040	0.0001	0.3793	0.0001
5	hstB10		5	0.3061	513.79	0.0001	0.0028	0.0001	0.4111	0.0001
6	mntb3		6	0.2781	448.63	0.0001	0.0020	0.0001	0.4524	0.0001
7	hstB11		7	0.2276	343.05	0.0001	0.0015	0.0001	0.4782	0.0001
8	Var_I		8	0.2514	390.98	0.0001	0.0012	0.0001	0.4909	0.0001
9	r_{mean}		9	0.1993	289.79	0.0001	0.0009	0.0001	0.4955	0.0001
10	Var_r		10	0.3700	683.59	0.0001	0.0006	0.0001	0.5066	0.0001
11	hstB13		11	0.2172	322.90	0.0001	0.0005	0.0001	0.5096	0.0001
12	g_{mean}		12	0.2015	293.71	0.0001	0.0004	0.0001	0.5293	0.0001
13	mntb1		13	0.1590	219.98	0.0001	0.0003	0.0001	0.5452	0.0001
14	mntb4		14	0.1683	235.40	0.0001	0.0003	0.0001	0.5643	0.0001
15	hstG4		15	0.1382	186.59	0.0001	0.0002	0.0001	0.5655	0.0001
16	hstG3		16	0.2058	301.31	0.0001	0.0002	0.0001	0.5674	0.0001
17	hstB1		17	0.1771	250.26	0.0001	0.0001	0.0001	0.5686	0.0001
18	hstR5		18	0.1599	221.33	0.0001	0.0001	0.0001	0.5721	0.0001
19	hstG2		19	0.1275	169.94	0.0001	0.0001	0.0001	0.5726	0.0001
20	hstR12		20	0.1033	133.86	0.0001	0.0001	0.0001	0.5826	0.0001
21	hstB8		21	0.1005	129.84	0.0001	0.0001	0.0001	0.5897	0.0001
22	hstR16		22	0.0964	123.91	0.0001	0.0001	0.0001	0.5906	0.0001
23	S_{mean}		23	0.0973	125.17	0.0001	0.0001	0.0001	0.5976	0.0001
24	hstR15		24	0.1069	139.01	0.0001	0.0001	0.0001	0.6050	0.0001
25	hstR13		25	0.0853	108.36	0.0001	0.0001	0.0001	0.6095	0.0001
26	mntr1		26	0.0854	108.47	0.0001	0.0001	0.0001	0.6156	0.0001
27	b_{mean}		27	0.2579	403.47	0.0001	0.0000	0.0001	0.6312	0.0001
28	mntg1		28	0.0792	99.90	0.0001	0.0000	0.0001	0.6357	0

* Probability. [§]Average squared canonical correlation. [†] See **Table 5.2** for definitions.

summarized in **Table 7.13(a)** for the parametric classifier and in **Table 7.13(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 70.5(healthy), 51.8(broken), 97.2(mildewed), 96.3(grass-green/green-frosted), 95.6(black-point/smudged), 91.8(heated) and 100.0(bin-/fire-burnt)%. For the non-parametric classifier, the average classification accuracies of the three training and testing data sets were 87.3(healthy), 84.9(broken), 97.4(mildewed), 97.0(grass-green/green-frosted), 99.0(black-point/smudged), 97.1(heated) and 100.0(bin-/fire-burnt)%. The mean classification accuracy for the healthy and all six types of damaged CWRS wheat kernels was 94.7% with the non-parametric classifier, which was much higher than 86.2% with the parametric classifier. As for individual grain types, the average classification accuracies of the three training and testing data sets were higher with the non-parametric classifier than with the parametric classifier for the healthy and all damage types of CWRS wheat kernels, except for the bin-/fire-burnt kernels that were 100.0% correctly identified using either the parametric or non-parametric classifiers. Compared to the classification results using the morphological feature model, the average classification accuracies (non-parametric classifier) using the color feature model were much higher for each class (**Fig 7.9**).

It was not surprising that very high classification accuracies were achieved for the bin-/fire-burnt and black-point/smudged kernels. The bin-/fire-burnt kernels were totally black and the germ ends of the black-point/smudged kernels had unique black spots. The major mis-classifications were found in two groups of damage types (**Table 7.13(b)**). The first group includes healthy, broken, and mildewed damage types. The broken kernels, with

Table 7.13(a) Classification of damaged CWRS wheat kernels by a parametric statistical classifier (quadratic discriminating function) using 28 selected color features

Class to → from ↓	Healthy No.	Healthy %	Broken No.	Broken %	Mildewed No.	Mildewed %	Grass-green No.	Grass-green %	Black-point No.	Black-point %	Heated No.	Heated %	Bin/fire burnt No.	Bin/fire burnt %	MCA* %
Healthy															
Set 1(300♀)	200	**66.7**	3	1.0	72	24.0	0	0.0	10	3.3	15	5.0	0	0.00	
Set 2(300)	206	**68.7**	4	1.3	80	26.7	0	0.0	3	1.0	7	2.3	0	0.00	
Set 3(400)	304	**76.0**	4	1.0	84	21.0	0	0.0	4	1.0	4	1.0	0	0.00	
average		**70.5**		1.1		23.9		0.0		1.8		2.8		0.00	
Broken															
Set 1(300)	51	17.0	151	**50.3**	38	12.7	12	4.0	18	6.0	30	10.0	0	0.00	
Set 2(300)	57	19.0	175	**58.3**	33	11.0	14	4.7	8	2.7	13	4.3	0	0.00	
Set 3(400)	108	27.0	187	**46.8**	55	13.8	28	7.0	6	1.5	16	4.0	0	0.00	
average		21.0		**51.8**		12.5		5.2		3.4		6.1		0.00	
Mildewed															
Set 1(300)	7	2.3	0	0.0	293	**97.7**	0	0.0	0	0.0	0	0.0	0	0.00	
Set 2(300)	6	2.0	1	0.3	293	**97.7**	0	0.0	0	0.0	0	0.0	0	0.00	
Set 3(400)	14	3.5	1	0.3	385	**96.3**	0	0.0	0	0.0	0	0.0	0	0.00	
average		2.6		0.2		**97.2**		0.0		0.0		0.0		0.00	
Grass-green															
Set 1(300)	0	0.0	1	0.3	0	0.0	287	**95.7**	9	3.0	3	1.0	0	0.00	
Set 2(300)	0	0.0	1	0.3	0	0.0	286	**95.3**	1	0.3	12	4.0	0	0.00	
Set 3(400)	0	0.0	1	0.3	0	0.0	391	**97.8**	0	0.0	8	2.0	0	0.00	
average		0.0		0.3		0.0		**96.3**		1.1		2.3		0.00	
Black-point															
Set 1(300)	0	0.0	1	0.3	0	0.0	0	0.0	298	**99.3**	1	0.3	0	0.00	
Set 2(300)	0	0.0	0	0.0	0	0.0	3	1.0	282	**94.0**	15	5.0	0	0.00	
Set 3(400)	0	0.0	0	0.0	0	0.0	10	2.5	374	**93.5**	16	4.0	0	0.00	
average		0.0		0.1		0.0		1.2		**95.6**		3.1		0.00	
Heated															
Set 1(300)	1	0.3	0	0.0	0	0.0	18	6.0	4	1.3	277	**92.3**	0	0.00	
Set 2(300)	0	0.0	0	0.0	0	0.0	23	7.7	0	0.0	277	**92.3**	0	0.00	
Set 3(400)	0	0.0	0	0.0	0	0.0	35	8.8	2	0.5	363	**90.8**	0	0.00	
average		0.1		0.0		0.0		7.5		0.6		**91.8**		0.00	
Bin/fire-burnt															
Set 1(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.00**	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.00**	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	400	**100.00**	
average		0.0		0.0		0.0		0.0		0.0		0.0		**100.00**	**86.2**

* Mean classification accuracy ♀ Testing data size

Table 7.13(b) Classification of damaged CWRS wheat kernels by a non-parametric statistical classifier (k-nearest neighbor) using 28 selected color features

Class to →	Healthy		Broken		Mildewed		Grass-green		Black-point		Heated		Bin/fire burnt		Unknown		MCA*
from ↓	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	%
Healthy																	
Set 1(300♀)	269	**89.7**	3	1.0	14	4.7	0	0.0	1	0.3	2	0.7	0	0.0	11	3.7	
Set 2(300)	264	**88.0**	4	1.3	21	7.0	0	0.0	1	0.3	0	0.0	0	0.0	10	3.3	
Set 3(400)	337	**84.3**	13	3.3	35	8.8	0	0.0	1	0.3	0	0.0	0	0.0	14	3.5	
average		**87.3**		1.9		6.8		0.0		0.3		0.2		0.0		3.5	
Broken																	
Set 1(300)	18	6.0	250	**83.3**	13	4.3	3	1.0	1	0.3	0	0.0	0	0.0	15	5.0	
Set 2(300)	21	7.0	260	**86.7**	10	3.3	0	0.0	0	0.0	1	0.3	0	0.0	8	2.7	
Set 3(400)	17	4.3	339	**84.8**	21	5.3	1	0.3	2	0.5	1	0.3	0	0.0	19	4.8	
average		5.8		**84.9**		4.3		0.4		0.3		0.2		0.0		4.1	
Mildewed																	
Set 1(300)	5	1.7	0	0.0	294	**98.0**	0	0.0	0	0.0	0	0.0	0	0.0	1	0.3	
Set 2(300)	4	1.3	1	0.3	291	**97.0**	0	0.0	0	0.0	0	0.0	0	0.0	4	1.3	
Set 3(400)	10	2.5	0	0.0	389	**97.3**	0	0.0	0	0.0	0	0.0	0	0.0	1	0.3	
average		1.8		0.1		**97.4**		0.0		0.0		0.0		0.0		0.6	
Grass-green																	
Set 1(300)	0	0.0	0	0.0	0	0.0	287	**95.7**	5	1.7	7	2.3	0	0.0	1	0.3	
Set 2(300)	0	0.0	2	0.7	0	0.0	293	**97.7**	1	0.3	2	0.7	0	0.0	2	0.7	
Set 3(400)	0	0.0	1	0.3	0	0.0	391	**97.8**	0	0.0	6	1.5	0	0.0	2	0.5	
average		0.0		0.3		0.0		**97.0**		0.7		1.5		0.0		0.5	
Black-point																	
Set 1(300)	0	0.0	2	0.7	0	0.0	1	0.3	297	**99.0**	0	0.0	0	0.0	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	297	**99.0**	3	1.0	0	0.0	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	396	**99.0**	4	1.0	0	0.0	0	0.0	
average		0.0		0.2		0.0		0.1		**99.0**		0.7		0.0		0.0	
Heated																	
Set 1(300)	1	0.3	0	0.0	0	0.0	9	3.0	3	1.0	285	**95.0**	0	0.0	2	0.7	
Set 2(300)	0	0.0	0	0.0	0	0.0	2	0.7	1	0.3	297	**99.0**	0	0.0	0	0.0	
Set 3(400)	1	0.3	1	0.3	0	0.0	1	0.3	2	0.5	389	**97.3**	0	0.0	6	1.5	
average		0.2		0.1		0.0		1.3		0.6		**97.1**		0.0		0.7	
Bin/fire-burnt																	
Set 1(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	400	**100.0**	0	0.0	
average		0.0		0.0		0.0		0.0		0.0		0.0		**100.0**		0.0	94.7

* Mean classification accuracy ♀ Testing data size

the lowest average classification accuracy, were mis-classified as healthy (5.8%) or mildewed (4.3%) or unknown (4.1%) kernels. The healthy kernels, with the second lowest average classification accuracy, were mis-classified as mildewed (6.8%) or broken (1.9%) or unknown (3.5%) kernels. The mildewed kernels were mis-classified as healthy kernels (1.8%). The second group includes the grass-green/green-frosted and heated damage types where 1.5% grass-green/green-frosted kernels were mis-classified as heated and 1.3% heated kernels were mis-classified as grass-green/green-frosted kernels. It was hypothesized that higher classification rates could be achieved by including morphological features with the color features, because of the morphological differences of the broken from the healthy and mildewed kernels and the morphological differences of the grass-green/green-frosted from heated kernels.

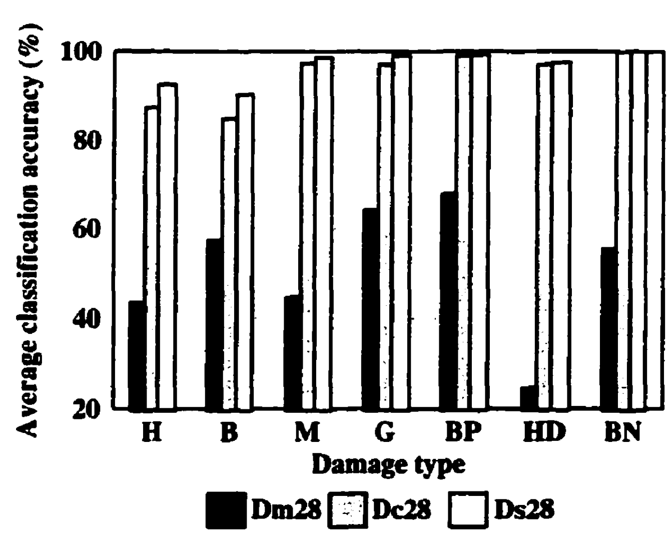


Fig 7.9 A comparison of morphological, color, and combined feature models for identification of damaged CWRS wheat kernels using non-parametric(k-nearest neighbor) classifiers. (Dm28: 28 morphological features; Dc28: 28 color features; Ds28: 28 morphological and color features; H: healthy; B: broken; M: mildewed; G: grass-green/green-frosted; BP: black-point/smudged; HD: heated; BN: bin-/fire-burnt)

7.4.3 Combined feature model

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 113 features from the 146 extracted morphological and color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-2**). **Table 7.14** lists the first 28 steps for selecting up to 28 best combined features including 24 color and four morphological features. The mean hue (H_{mean}) was still ranked as the most significant feature, while the kernel area (A) was ranked at the third place. The discriminating abilities of the feature models Ds4 (the best 4 combined features), Ds8 (the best 8 combined features), ..., Ds28 (the best 28 combined features) were evaluated using SAS DISCRIM (**Appendix E-2**). For all examined combined feature models, the mean classification accuracies were higher with the non-parametric (k-nearest neighbor) classifiers than with the parametric (quadratic) classifiers (**Fig 7.8(c)**). For the non-parametric classifiers, as with the color models, the mean classification accuracy increased to a certain extent and then remained relatively constant as the number of features increased, while for the parametric classifiers, the mean classification accuracy varied non-monotonously with the feature size. Since the highest mean classification accuracy (97.4%) was obtained using the non-parametric classifier with the feature model Ds28 (higher mean classification accuracy may be obtained using more features, however, it was concluded from the trend (**Fig 7.8(c)**) that the improvement was negligible), this model was chosen as the combined feature model for the hold-out classification analysis of damaged CWRS wheat kernels.

The hold-out classification analysis of damaged kernels was carried out using the

Table 7.14 The first 28 steps for selecting up to 28 best combined features by SAS STEPDISC for identification analysis of damaged CWRS wheat kernels

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	H_{mean}[†]		1	0.9722	40792.33	0.0001	0.0278	0.0001	0.1620	0.0001
2	hstG10		2	0.6223	1920.39	0.0001	0.0105	0.0001	0.2647	0.0001
3	A		3	0.4230	854.34	0.0001	0.0061	0.0001	0.3336	0.0001
4	Δr		4	0.3477	620.86	0.0001	0.0039	0.0001	0.3701	0.0001
5	hstR6		5	0.3519	632.44	0.0001	0.0026	0.0001	0.4251	0.0001
6	hstB10		6	0.2663	422.70	0.0001	0.0019	0.0001	0.4466	0.0001
7	hstB11		7	0.2697	430.13	0.0001	0.0014	0.0001	0.4831	0.0001
8	Var_I		8	0.2409	369.50	0.0001	0.0010	0.0001	0.4960	0.0001
9	RS14		9	0.2073	304.46	0.0001	0.0008	0.0001	0.5181	0.0001
10	Var_r		10	0.2049	300.02	0.0001	0.0007	0.0001	0.5275	0.0001
11	r_{mean}		11	0.3645	667.63	0.0001	0.0004	0.0001	0.5310	0.0001
12	hstB13		12	0.2090	307.48	0.0001	0.0003	0.0001	0.5352	0.0001
13	g_{mean}		13	0.1719	241.59	0.0001	0.0003	0.0001	0.5502	0.0001
14	mntb4		14	0.1450	197.33	0.0001	0.0002	0.0001	0.5674	0.0001
15	mntb1		15	0.1645	229.05	0.0001	0.0002	0.0001	0.5831	0.0001
16	hstG4		16	0.1390	187.81	0.0001	0.0002	0.0001	0.5844	0.0001
17	hstG3		17	0.2103	309.68	0.0001	0.0001	0.0001	0.5860	0.0001
18	hstB1		18	0.1686	235.82	0.0001	0.0001	0.0001	0.5869	0.0001
19	hstR5		19	0.1632	226.74	0.0001	0.0001	0.0001	0.5904	0.0001
20	areaR		20	0.1311	175.33	0.0001	0.0001	0.0001	0.6026	0.0001
21	mnt1		21	0.1273	169.59	0.0001	0.0001	0.0001	0.6123	0.0001
22	mntr1		22	0.4624	999.58	0.0001	0.0000	0.0001	0.6539	0.0001
23	b_{mean}		23	0.2075	304.13	0.0001	0.0000	0.0001	0.6612	0.0001
24	S_{mean}		24	0.1678	234.30	0.0001	0.0000	0.0001	0.6693	0.0001
25	hstG2		25	0.1650	229.45	0.0001	0.0000	0.0001	0.6739	0.0001
26	hstG15		26	0.1388	187.12	0.0001	0.0000	0.0001	0.6806	0.0001
27	Var_g		27	0.1324	177.19	0.0001	0.0000	0.0001	0.6840	0.0001
28	hstR12		28	0.1145	150.10	0.0001	0.0000	0.0001	0.6918	0

* Probability. [§]Average squared canonical correlation. [†] See **Tables 5.1 and 5.2** for definitions.

three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-2**) are summarized in **Table 7.15(a)** for the parametric classifier and in **Table 7.15(b)** for the non-parametric classifier. For the parametric classifier, the average classification accuracies of the three training and testing data sets were 86.0(healthy), 73.0(broken), 96.9(mildewed), 97.5(grass-green/green-frosted), 97.9(black-point/smudged), 93.9(heated) and 100.0(bin-/fire-burnt)%. For the non-parametric classifier, the average classification accuracies were 92.5(healthy), 90.3(broken), 98.6(mildewed), 99.0(grass-green/green-frosted), 99.1(black-point/smudged), 97.5(heated) and 100.0(bin-/fire-burnt)%. The mean classification accuracy for the healthy and all damage types was 96.7% with the non-parametric classifier, which was statistically higher than 92.2% with the parametric classifier. As for individual grain types, the average classification accuracies of the three training and testing data sets were higher with the non-parametric classifier than with the parametric classifier for the healthy and all the damage types.

Compared to the classification results using the color features alone, higher average classification accuracies were achieved by the inclusion of morphological features in the feature model for all types of damaged CWRS kernels, especially the broken and healthy kernels (**Fig 7.9**). The major mis-classifications were found in the same two groups of damage types (**Table 7.15(b)**). The broken kernels, with the lowest average classification accuracy, were mis-classified as healthy (6.1%), or mildewed (1.4%) kernels, or unknown (1.1%) kernels. The healthy kernels, with the second lowest average classification accuracy, were mis-classified as mildewed (5.0%), or broken (1.1%), or unknown (1.9%) kernels. The

Table 7.15(a) Classification of damaged CWRS wheat kernels by a parametric statistical classifier (quadratic discriminating function) using 28 selected combined features

Class to → from ↓	Healthy No.	Healthy %	Broken No.	Broken %	Mildewed No.	Mildewed %	Grass-green No.	Grass-green %	Black-point No.	Black-point %	Heated No.	Heated %	Bin/fire burnt No.	Bin/fire burnt %	MCA* %
Healthy															
Set 1(300[♀])	272	**90.7**	0	0.0	26	8.7	0	0.0	2	0.7	0	0.0	0	0.0	
Set 2(300)	256	**85.3**	3	1.0	39	13.0	0	0.0	2	0.7	0	0.0	0	0.0	
Set 3(400)	328	**82.0**	3	0.8	66	16.5	0	0.0	2	0.5	1	0.3	0	0.0	
average		**86.0**		0.6		12.7		0.0		0.6		0.1		0.0	
Broken															
Set 1(300)	62	20.7	211	**70.3**	15	5.0	0	0.0	11	3.7	1	0.3	0	0.0	
Set 2(300)	62	20.7	228	**76.0**	9	3.0	0	0.0	1	0.3	0	0.0	0	0.0	
Set 3(400)	68	17.0	291	**72.8**	33	8.3	4	1.0	3	0.8	1	0.3	0	0.0	
average		19.5		**73.0**		5.4		0.3		1.6		0.2		0.0	
Mildewed															
Set 1(300)	8	2.7	0	0.0	292	**97.3**	0	0.0	0	0.0	0	0.0	0	0.0	
Set 2(300)	10	3.3	1	0.3	289	**96.3**	0	0.0	0	0.0	0	0.0	0	0.0	
Set 3(400)	11	2.8	1	0.3	388	**97.0**	0	0.0	0	0.0	0	0.0	0	0.0	
average		2.9		0.2		**96.9**		0.0		0.0		0.0		0.0	
Grass-green															
Set 1(300)	0	0.0	1	0.3	0	0.0	294	**98.0**	2	0.7	3	1.0	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	288	**96.0**	1	0.3	11	3.7	0	0.0	
Set 3(400)	0	0.0	1	0.3	0	0.0	394	**98.5**	0	0.0	5	1.3	0	0.0	
average		0.0		0.2		0.0		**97.5**		0.3		2.0		0.0	
Black-point															
Set 1(300)	0	0.0	2	0.7	0	0.0	1	0.3	296	**98.7**	1	0.3	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	2	0.7	291	**97.0**	7	2.3	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	3	0.8	392	**98.0**	5	1.3	0	0.0	
average		0.0		0.2		0.0		0.6		**97.9**		1.3		0.0	
Heated															
Set 1(300)	0	0.0	0	0.0	0	0.0	8	2.7	4	1.3	288	**96.0**	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	17	5.7	1	0.3	282	**94.0**	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	31	7.8	2	0.5	367	**91.8**	0	0.0	
average		0.0		0.0		0.0		5.4		0.7		**93.9**		0.0	
Bin/fire-burnt															
Set 1(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	400	**100.0**	
average		0.0		0.0		0.0		0.0		0.0		0.0		**100.0**	**92.2**

* Mean classification accuracy ♀ Testing data size

Table 7.15(b) Classification of damaged CWRS wheat kernels by a non-parametric statistical classifier (k-nearest neighbor) using 28 selected combined features

Class to →	Healthy		Broken		Mildewed		Grass-green		Black-point		Heated		Bin/fire burnt		Unknown		MCA*
from ↓	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	No.	%	%
Healthy																	
Set 1(300♀)	278	**92.7**	2	0.7	16	5.3	0	0.0	0	0.0	0	0.0	0	0.0	4	1.3	
Set 2(300)	283	**94.3**	5	1.7	10	3.3	0	0.0	1	0.3	0	0.0	0	0.0	1	0.3	
Set 3(400)	362	**90.5**	4	1.0	25	6.3	0	0.0	2	0.5	0	0.0	0	0.0	7	1.8	
average		**92.5**		1.1		5.0		0.0		0.3		0.0		0.0		1.1	
Broken																	
Set 1(300)	16	5.3	271	**90.3**	3	1.0	0	0.0	1	0.3	0	0.0	0	0.0	9	3.0	
Set 2(300)	19	6.3	272	**90.7**	5	1.7	1	0.3	0	0.0	0	0.0	0	0.0	3	1.0	
Set 3(400)	27	6.8	359	**89.8**	6	1.5	0	0.0	1	0.3	0	0.0	0	0.0	7	1.8	
average		6.1		**90.3**		1.4		0.1		0.2		0.0		0.0		1.9	
Mildewed																	
Set 1(300)	2	0.7	0	0.0	297	**99.0**	0	0.0	0	0.0	0	0.0	0	0.0	1	0.3	
Set 2(300)	4	1.3	1	0.3	295	**98.3**	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	
Set 3(400)	6	1.5	0	0.0	394	**98.5**	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	
average		1.2		0.1		**98.6**		0.0		0.0		0.0		0.0		0.1	
Grass-green																	
Set 1(300)	0	0.0	0	0.0	0	0.0	297	**99.0**	1	0.3	2	0.7	0	0.0	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	295	**98.3**	0	0.0	5	1.7	0	0.0	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	399	**99.8**	0	0.0	1	0.3	0	0.0	0	0.0	
average		0.0		0.0		0.0		**99.0**		0.1		0.9		0.0		0.0	
Black-point																	
Set 1(300)	2	0.7	0	0.0	0	0.0	0	0.0	296	**98.7**	1	0.3	0	0.0	1	0.3	
Set 2(300)	1	0.3	0	0.0	0	0.0	0	0.0	298	**99.3**	1	0.3	0	0.0	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	397	**99.3**	2	0.5	0	0.0	1	0.3	
average		0.3		0.0		0.0		0.0		**99.1**		0.4		0.0		0.2	
Heated																	
Set 1(300)	0	0.0	0	0.0	0	0.0	6	2.0	3	1.0	289	**96.3**	0	0.0	2	0.7	
Set 2(300)	0	0.0	0	0.0	0	0.0	5	1.7	3	1.0	292	**97.3**	0	0.0	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	1	0.3	2	0.5	395	**98.8**	0	0.0	2	0.5	
average		0.0		0.0		0.0		1.3		0.8		**97.5**		0.0		0.4	
Bin/fire burnt																	
Set 1(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	0	0.0	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	400	**100.0**	0	0.0	
average		0.0		0.0		0.0		0.0		0.0		0.0		**100.0**		0.0	**96.7**

* Mean classification accuracy ♀ Testing data size

Table 7.15(c) Classification of damaged CWRS wheat kernels by a neural network classifier (28-13-7) using 28 selected combined features

Class to → from ↓	Healthy No.	Healthy %	Broken No.	Broken %	Mildewed No.	Mildewed %	Grass-green No.	Grass-green %	Black-point No.	Black-point %	Heated No.	Heated %	Bin/fire burnt No.	Bin/fire burnt %	MCA* %
Healthy															
Set 1(300♀)	281	**93.7**	2	0.7	16	5.3	0	0.0	1	0.3	0	0.0	0	0.0	
Set 2(300)	271	**90.3**	7	2.3	12	4.0	1	0.3	6	2.0	3	1.0	0	0.0	
Set 3(400)	361	**90.3**	13	3.3	22	5.5	0	0.0	2	0.5	2	0.5	0	0.0	
average		**91.4**		2.1		4.9		0.1		0.9		0.5		0.0	
Broken															
Set 1(300)	20	6.7	272	**90.7**	3	1.0	1	0.3	3	1.0	1	0.3	0	0.0	
Set 2(300)	14	4.7	278	**92.7**	3	1.0	2	0.7	1	0.3	2	0.7	0	0.0	
Set 3(400)	19	4.8	366	**91.5**	11	2.8	2	0.5	0	0.0	2	0.5	0	0.0	
average		5.4		**91.6**		1.6		0.5		0.4		0.5		0.0	
Mildewed															
Set 1(300)	6	2.0	1	0.3	293	**97.7**	0	0.0	0	0.0	0	0.0	0	0.0	
Set 2(300)	6	2.0	3	1.0	291	**97.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set 3(400)	13	3.3	1	0.3	386	**96.5**	0	0.0	0	0.0	0	0.0	0	0.0	
average		2.4		0.5		**97.1**		0.0		0.0		0.0		0.0	
Grass-green															
Set 1(300)	0	0.0	3	1.0	0	0.0	294	**98.0**	0	0.0	3	1.0	0	0.0	
Set 2(300)	0	0.0	3	1.0	0	0.0	291	**97.0**	0	0.0	6	2.0	0	0.0	
Set 3(400)	0	0.0	3	0.8	0	0.0	393	**98.3**	0	0.0	4	1.0	0	0.0	
average		0.0		0.9		0.0		**97.8**		0.0		1.3		0.0	
Black-point															
Set 1(300)	1	0.3	3	1.0	0	0.0	1	0.3	294	**98.0**	1	0.3	0	0.0	
Set 2(300)	1	0.3	0	0.0	0	0.0	1	0.3	297	**99.0**	1	0.3	0	0.0	
Set 3(400)	5	1.3	1	0.3	0	0.0	0	0.0	390	**97.5**	4	1.0	0	0.0	
average		0.6		0.4		0.0		0.2		**98.2**		0.6		0.0	
Heated															
Set 1(300)	3	1.0	1	0.3	0	0.0	7	2.3	2	0.7	287	**95.7**	0	0.0	
Set 2(300)	0	0.0	0	0.0	0	0.0	9	3.0	2	0.7	289	**96.3**	0	0.0	
Set 3(400)	2	0.5	0	0.0	0	0.0	8	2.0	2	0.5	388	**97.0**	0	0.0	
average		0.5		0.1		0.0		2.4		0.6		**96.3**		0.0	
Bin/fire-burnt															
Set 1(300)	0	0.0	0	0.0	1	0.3	0	0.0	0	0.0	0	0.0	299	**99.7**	
Set 2(300)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	300	**100.0**	
Set 3(400)	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	0	0.0	400	**100.0**	
average		0.0		0.0		0.1		0.0		0.0		0.0		**99.9**	**96.0**

* Mean classification accuracy ♀ Testing data size

mildewed kernels were mis-classified as healthy kernels (1.2%). In the second group, 0.9% grass-green/green-frosted kernels were mis-classified as heated and 1.3% heated kernels were mis-classified as grass-green/green-frosted kernels. Overall, using the combined features significantly improved the classification accuracies obtained using the morphological or color features alone in identifying the different types of damaged grain kernels.

As a comparison to the statistical classifiers, a MNN classifier with a structure of 28-13-7 (three layers with 28 nodes in the input, 13 nodes in the hidden, and 7 nodes in the output layer) was used with the combined feature model Ds28. The results are summarized in **Table 7.15(c)**. The average classification accuracies were 91.4(healthy), 91.6(broken), 97.1(mildewed), 97.8(grass-green/green-frosted), 98.2(black-point/smudged), 96.3(heated) and 99.9(bin-/fire-burnt)%, slightly lower than using the non-parametric classifier. Because the structure of the MNN, therefore the performance of the MNN classifier was not optimized, it cannot be concluded that the non-parametric (k-nearest neighbor) classifier is better than the MNN classifier. The reported MNN classifier was chosen from three tested MNN classifiers with different structures, due to its superior performance. For the damage-type identification problem, the MNN classifier can be considered as good as the non-parametric (k-nearest neighbor) classifier.

7.5 Grain Type Identification of Bulk Grain Samples

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected 55 features from the 114 extracted color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix**

D-3). **Table 7.16** lists the first 32 steps for selecting up to 28 best color features. The discriminating abilities of the feature models Bc4 (the best 4 features), Bc8 (the best 8 features), Bc12 (the best 12 features), ..., Bc28 (the best 28 features) were evaluated using SAS DISCRIM (**Appendix E-3**). For both the parametric (quadratic) and non-parametric (k-nearest neighbor) classifiers, a high classification rate of 99.9% was achieved with the best 4 feature model, and a 100.0% classification rate was achieved with the best 8 and 12 feature models (**Fig 7.10**). After that the classification rate decreased for the parametric classifier while remained constant for the non-parametric classifier as the size of the feature model increased. The model Bc8 was chosen as the color feature model for the hold-out grain-type classification analysis of bulk grain samples. In the model Bc8, 4 out of the 8 features were directly extracted from the red band of the color images, compared to 2 from the green band and 1 from the blue band. The remaining feature was mean saturation. This agrees with the results reported by Neuman et al. (1989b) and Hawk et al. (1970) that the reflectance properties of bulk samples of cereal grains were more distinct in the red color band than in other color bands of the visual spectrum.

The hold-out grain-type classification analysis was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-3**) are summarized in **Table 7.17(a)** for the parametric classifier and in **Table 7.17(b)** for the non-parametric classifier. For the parametric classifier, 100.0% classification accuracies were obtained for each of the five grain types with each of the three training and testing data sets, except for CWAD wheat with the training and testing data set 1 where 5 out of 21 CWAD wheat images in the testing

Table 7.16 The first 32 steps for selecting up to 28 best color features by SAS STEPDISC for grain type identification analysis of bulk grain samples

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[§]	Prob >ASCC
1	Var_r[†]		1	0.9453	1791.900	0.0001	0.0547	0.0001	0.2363	0.0001
2	Var_g		2	0.8393	540.358	0.0001	0.0088	0.0001	0.4455	0.0001
3	hstR4		3	0.8019	417.842	0.0001	0.0017	0.0001	0.6295	0.0001
4	S_{mean}		4	0.6635	203.128	0.0001	0.0006	0.0001	0.6770	0.0001
5	hstR17		5	0.6439	185.824	0.0001	0.0002	0.0001	0.8282	0.0001
6	hstR21		6	0.4462	82.592	0.0001	0.0001	0.0001	0.8443	0.0001
7	hstB14		7	0.4763	92.998	0.0001	0.0001	0.0001	0.8764	0.0001
8	hstG8		8	0.3607	57.548	0.0001	0.0000	0.0001	0.8947	0.0001
9	hstR2		9	0.2816	39.879	0.0001	0.0000	0.0001	0.9000	0.0001
10	H_{mean}		10	0.2286	30.086	0.0001	0.0000	0.0001	0.9053	0.0001
11	hstR1		11	0.2035	25.861	0.0001	0.0000	0.0001	0.9136	0.0001
12		hstR4	10	0.0091	0.930	0.4462	0.0000	0.0001	0.9133	0.0001
13	Var_b		11	0.1659	20.144	0.0001	0.0000	0.0001	0.9185	0.0001
14	hstB25		12	0.1405	16.510	0.0001	0.0000	0.0001	0.9207	0.0001
15	hstB9		13	0.1444	17.009	0.0001	0.0000	0.0001	0.9244	0.0001
16	hstG19		14	0.1117	12.640	0.0001	0.0000	0.0001	0.9266	0.0001
17	hstB15		15	0.1755	21.342	0.0001	0.0000	0.0001	0.9279	0.0001
18	hstR14		16	0.1090	12.239	0.0001	0.0000	0.0001	0.9302	0.0001
19	Var_I		17	0.2066	25.975	0.0001	0.0000	0.0001	0.9341	0.0001
20	hstG32		18	0.3866	62.715	0.0001	0.0000	0.0001	0.9378	0.0001
21	hstG20		19	0.1102	12.291	0.0001	0.0000	0.0001	0.9401	0.0001
22	hstR23		20	0.1961	24.153	0.0001	0.0000	0.0001	0.9442	0.0001
23	hstR3		21	0.0817	8.780	0.0001	0.0000	0.0001	0.9449	0.0001
24	hstG1		22	0.1053	11.597	0.0001	0.0000	0.0001	0.9460	0.0001
25	hstB17		23	0.0719	7.617	0.0001	0.0000	0.0001	0.9471	0.0001
26	hstR13		24	0.0647	6.781	0.0001	0.0000	0.0001	0.9480	0.0001
27	Var_H		25	0.0617	6.427	0.0001	0.0000	0.0001	0.9485	0.0001
28	hstG18		26	0.0556	5.740	0.0002	0.0000	0.0001	0.9490	0.0001
29		hstG19	25	0.0108	1.061	0.3754	0.0000	0.0001	0.9489	0.0001
30	hstR20		26	0.0639	6.654	0.0001	0.0000	0.0001	0.9497	0.0001
31	hstR19		27	0.0708	7.414	0.0001	0.0000	0.0001	0.9504	0.0001
32	hstG24		28	0.0557	5.724	0.0002	0.0000	0.0001	0.9507	0.0001

* Probability. [§] Average squared canonical correlation. [†] See **Table 5.3** for definitions.

Table 7.17(a) Grain type classification of bulk grain samples by a parametric statistical classifier (quadratic discriminating function) using 8 selected color features

Class to ⇒	CWRS		CWAD		Barley		Rye		Oats		MCA*
from ⇓	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS											
Set1(63♀)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set2(63)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set3(54)	54	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
average		**100.0**		0.0		0.0		0.0		0.0	
CWAD											
Set1(21)	5	23.8	16	**76.2**	0	0.0	0	0.0	0	0.0	
Set2(21)	0	0.0	21	**100.0**	0	0.0	0	0.0	0	0.0	
Set3(18)	0	0.0	18	**100.0**	0	0.0	0	0.0	0	0.0	
average		7.9		**92.1**		0.0		0.0		0.0	
Barley											
Set1(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set2(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set3(18)	0	0.0	0	0.0	18	**100.0**	0	0.0	0	0.0	
average		0.0		0.0		**100.0**		0.0		0.0	
Rye											
Set1(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set2(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set3(18)	0	0.0	0	0.0	0	0.0	18	**100.0**	0	0.0	
average		0.0		0.0		0.0		**100.0**		0.0	
Oats											
Set1(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set2(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set3(18)	0	0.0	0	0.0	0	0.0	0	0.0	18	**100.0**	
average		0.0		0.0		0.0		0.0		**100.0**	**98.4**

* Mean classification accuracy ♀ Testing data size

Table 7.17(b) Grain type classification of bulk grain samples by a non-parametric statistical (k-nearest neighbour) classifier using 8 selected color features

Class to ⇒	CWRS		CWAD		Barley		Rye		Oats		MCA*
from ⬇	No.	%	No.	%	No.	%	No.	%	No.	%	%
CWRS											
Set1(63♀)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set2(63)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set3(54)	54	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
average		**100.0**		0.0		0.0		0.0		0.0	
CWAD											
Set1(21)	0	0.0	21	**100.0**	0	0.0	0	0.0	0	0.0	
Set2(21)	0	0.0	21	**100.0**	0	0.0	0	0.0	0	0.0	
Set3(18)	0	0.0	18	**100.0**	0	0.0	0	0.0	0	0.0	
average		0.0		**100.0**		0.0		0.0		0.0	
Barley											
Set1(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set2(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set3(18)	0	0.0	0	0.0	18	**100.0**	0	0.0	0	0.0	
average		0.0		0.0		**100.0**		0.0		0.0	
Rye											
Set1(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set2(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set3(18)	0	0.0	0	0.0	0	0.0	18	**100.0**	0	0.0	
average		0.0		0.0		0.0		**100.0**		0.0	
Oats											
Set1(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set2(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set3(18)	0	0.0	0	0.0	0	0.0	0	0.0	18	**100.0**	
average		0.0		0.0		0.0		0.0		**100.0**	**100.0**

* Mean classification accuracy ♀ Testing data size

Table 7.17(c) Grain type classification of bulk grain samples by a neural network classifier (8-6-4-5) using 8 selected color features

Class to ⇒ from ⇓	CWRS No.	CWRS %	CWAD No.	CWAD %	Barley No.	Barley %	Rye No.	Rye %	Oats No.	Oats %	MCA* %
CWRS											
Set1(63♀)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set2(63)	63	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
Set3(54)	54	**100.0**	0	0.0	0	0.0	0	0.0	0	0.0	
average		**100.0**		0.0		0.0		0.0		0.0	
CWAD											
Set1(21)	0	0.0	21	**100.0**	0	0.0	0	0.0	0	0.0	
Set2(21)	0	0.0	18	**85.7**	3	14.3	0	0.0	0	0.0	
Set3(18)	0	0.0	18	**100.0**	0	0.0	0	0.0	0	0.0	
average		0.0		**95.2**		4.8		0.0		0.0	
Barley											
Set1(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set2(21)	0	0.0	0	0.0	21	**100.0**	0	0.0	0	0.0	
Set3(18)	0	0.0	0	0.0	18	**100.0**	0	0.0	0	0.0	
average		0.0		0.0		**100.0**		0.0		0.0	
Rye											
Set1(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set2(21)	0	0.0	0	0.0	0	0.0	21	**100.0**	0	0.0	
Set3(18)	0	0.0	0	0.0	0	0.0	18	**100.0**	0	0.0	
average		0.0		0.0		0.0		**100.0**		0.0	
Oats											
Set1(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set2(21)	0	0.0	0	0.0	0	0.0	0	0.0	21	**100.0**	
Set3(18)	0	0.0	0	0.0	0	0.0	0	0.0	18	**100.0**	
average		0.0		0.0		0.0		0.0		**100.0**	**99.1**

* Mean classification accuracy ♀ Testing data size

data set were mis-classified as CWRS wheat. For the non-parametric classifier, 100.0% classification accuracies were obtained for each of the five grain types with each of the three training and testing data sets.

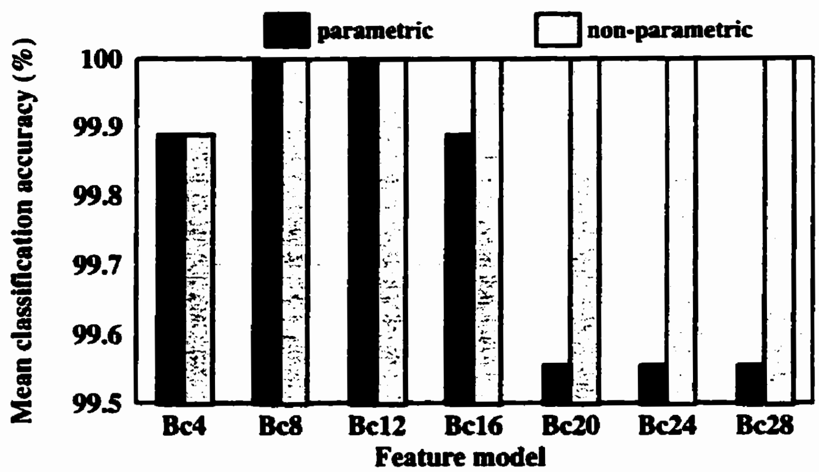


Fig 7.10 Evaluation of color feature models for grain type identification analysis of bulk grain samples using SAS DISCRIM.

As a comparison to the statistical classifiers, a MNN classifier with a structure of 8-6-4-5 (four layers with 8 nodes in the input, 6 nodes in the first hidden, 4 nodes in the second hidden, and 5 nodes in the output layer) was used with the feature model Bc8. The results are summarized in **Table 7.17(c)**. The 100.0% classification accuracies were obtained for each of the five grain types with each of the three training and testing data sets, except for CWAD wheat. With the training and testing data set 2, 3 out of 21 CWAD wheat images in the testing data set were mis-classified as barley.

7.6 Grade Identification of Bulk CWRS Wheat Samples

With a minimum significant level of 0.15, the SAS procedure STEPDISC selected only 20 features from the 114 extracted color features and ranked them according to their contributions to the discriminatory powers of the corresponding feature model (**Appendix D-4**). **Table 7.18** lists the 32 steps for selecting these 20 color features. The discriminating abilities of the feature models Hc4 (the best 4 features), Hc8 (the best 8 features), Hc12 (the best 12 features), ..., Hc20 (the best 20 features) were evaluated using SAS DISCRIM (**Appendix E-4**). For both the parametric (quadratic) and non-parametric (k-nearest neighbor) classifiers, generally the mean classification accuracies increased as the size of the feature model increased (**Fig 7.11**). For all examined feature models, except for Hc4, the mean classification accuracies were higher with the parametric classifiers than with the non-parametric classifiers. This was contrary to the corresponding results in the previous classification analyses (**Sections 7.3**, **7.4**, and **7.5**) where the mean classification accuracies were higher with the non-parametric classifiers than with the parametric classifiers. Since the highest mean classification accuracy (85.6%) was achieved using the parametric classifier with the feature model Hc20, this model was chosen for the hold-out grade classification analysis of bulk CWRS wheat samples.

The hold-out grade classification analysis was carried out using the three pairs of training and testing data sets for both the parametric (quadratic) and non-parametric (k-nearest neighbor) statistical classifiers. The results (**Appendix F-4)** are summarized in **Table 7.19(a)** for the parametric classifier and in **Table 7.19(b)** for the non-parametric classifier.

As a comparison to the statistical classifiers, a MNN classifier with a structure of 20-

Table 7.18 The first 32 steps for selecting up to 20 best color features by SAS STEPDISC for grade identification analysis of bulk CWRS wheat samples

Step	Feature In	Feature Out	No.	Partial R^2	F Statistic	Prob* > F	Wilks' λ	Prob > λ	ASCC[‡]	Prob >ASCC
1	hstB1[†]		1	0.4071	60.7760	0.0001	0.5929	0.0001	0.2036	0.0001
2	hstB13		2	0.0941	9.1390	0.0002	0.5371	0.0001	0.2353	0.0001
3	hstR5		3	0.1951	21.2120	0.0001	0.4323	0.0001	0.3052	0.0001
4	hstR19		4	0.1767	18.6700	0.0001	0.3559	0.0001	0.3467	0.0001
5	Var_S		5	0.0879	8.3340	0.0004	0.3246	0.0001	0.3639	0.0001
6	hstB5		6	–0.1370	13.6480	0.0001	0.2802	0.0001	0.4106	0.0001
7	hstG16		7	0.0829	7.7300	0.0006	0.2569	0.0001	0.4246	0.0001
8	hstR17		8	0.0613	5.5510	0.0046	0.2412	0.0001	0.4504	0.0001
9	hstB10		9	0.1150	10.9780	0.0001	0.2135	0.0001	0.4928	0.0001
10	hstR2		10	0.0561	4.9930	0.0078	0.2015	0.0001	0.5032	0.0001
11	hstR6		11	0.0505	4.4440	0.0132	0.1913	0.0001	0.5141	0.0001
12		hstR5	10	0.0140	1.1890	0.3072	0.1940	0.0001	0.5121	0.0001
13	hstR4		11	0.0326	2.8180	0.0626	0.1877	0.0001	0.5167	0.0001
14	hstR7		12	0.0429	3.7220	0.0262	0.1796	0.0001	0.5247	0.0001
15	hstR9		13	0.0442	3.8190	0.0239	0.1717	0.0001	0.5344	0.0001
16	hstB6		14	0.0338	2.8700	0.0596	0.1659	0.0001	0.5390	0.0001
17	hstR15		15	0.0584	5.0580	0.0074	0.1562	0.0001	0.5515	0.0001
18		hstR2	14	0.0203	1.6910	0.1876	0.1594	0.0001	0.5484	0.0001
19	hstG1		15	0.0370	3.1300	0.0464	0.1535	0.0001	0.5531	0.0001
20	H_{mean}		16	0.0435	3.6820	0.0273	0.1469	0.0001	0.5586	0.0001
21		hstR4	15	0.0212	1.7520	0.1766	0.1500	0.0001	0.5561	0.0001
22	hstR1		16	0.0340	2.8540	0.0605	0.1449	0.0001	0.5620	0.0001
23	hstG14		17	0.0323	2.6850	0.0713	0.1402	0.0001	0.5674	0.0001
24	hstR32		18	0.0340	2.8200	0.0626	0.1355	0.0001	0.5775	0.0001
25	hstR18		19	0.0282	2.3080	0.1027	0.1317	0.0001	0.5850	0.0001
26	hstR20		20	0.0365	2.9910	0.0531	0.1268	0.0001	0.5902	0.0001
27		hstR19	19	0.0118	0.9450	0.3909	0.1284	0.0001	0.5887	0.0001
28		hstR15	18	0.0198	1.6070	0.2037	0.1310	0.0001	0.5837	0.0001
29	hstG4		19	0.0255	2.0770	0.1287	0.1276	0.0001	0.5872	0.0001
30	hstG18		20	0.0320	2.6160	0.0763	0.1235	0.0001	0.5971	0.0001
31	hstB14		21	0.0440	3.6120	0.0293	0.1181	0.0001	0.6086	0.0001
32		hstR17	20	0.0170	1.3590	0.2599	0.1201	0.0001	0.6056	0.0001

* Probability. [‡] Average squared canonical correlation. [†] See **Table 5.3** for definitions.

Table 7.19(a) Grade classification of bulk CWRS wheat samples by a parametric statistical classifier (quadratic discriminating function) using 20 selected color features

Class to ⇒ from ⇓	Grade 1 No.	Grade 1 %	Grade 2 No.	Grade 2 %	Grade 3 No.	Grade 3 %	MCA* %
Grade 1							
Set1(21♀)	19	**90.5**	1	4.8	1	4.8	
Set2(21)	16	**76.2**	3	14.3	2	9.5	
Set3(18)	10	**55.6**	3	16.7	5	27.8	
average		**74.1**		11.9		14.0	
Grade 2							
Set1(21)	3	14.3	7	**33.3**	11	52.4	
Set2(21)	3	14.3	14	**66.7**	4	19.1	
Set3(18)	0	0.0	17	**94.4**	1	5.6	
average		9.5		**64.8**		25.7	
Grade 3							
Set1(21)	4	19.1	1	4.8	16	**76.2**	
Set2(21)	5	23.8	9	42.9	7	**33.3**	
Set3(18)	0	0.0	2	11.1	16	**88.9**	
average		14.3		19.6		**66.1**	**68.3**

* Mean classification accuracy ♀ Testing data size

Table 7.19(b) Grade classification of bulk CWRS wheat samples by a non-parametric statistical (k-nearest neighbour) classifier using 20 selected color features

Class to ⇒	Grade 1		Grade 2		Grade 3		Unknown		MCA*
from ⇓	No.	%	No.	%	No.	%	No.	%	%
Grade 1									
Set1(21♀)	18	**85.7**	2	9.5	0	0.0	1	4.8	
Set2(21)	20	**95.2**	1	4.8	0	0.0	0	0.0	
Set3(18)	16	**88.9**	0	0.0	0	0.0	2	11.1	
average		**90.0**		4.8		0.0		5.3	
Grade 2									
Set1(21)	3	14.3	11	**52.4**	4	19.1	3	14.3	
Set2(21)	2	9.5	16	**76.2**	2	9.5	1	4.8	
Set3(18)	0	0.0	6	**33.3**	6	33.3	6	33.3	
average		7.9		**54.0**		20.6			
Grade 3									
Set1(21)	0	0.0	10	47.6	9	**42.9**	2	9.5	
Set2(21)	4	19.1	11	52.4	5	**23.8**	1	4.8	
Set3(18)	0	0.0	2	11.1	16	**88.9**	0	0.0	
average		6.4		37.0		**51.9**		4.8	65.3

* Mean classification accuracy ♀ Testing data size

Table 7.19(c) Grade classification of bulk CWRS wheat samples by a neural network classifier (20-5-5-3) using 20 selected color features

Class to ⇒ from ⇓	Grade 1 No.	Grade 1 %	Grade 2 No.	Grade 2 %	Grade 3 No.	Grade 3 %	MCA* %
Grade 1							
Set1(21♀)	19	**90.5**	2	9.5	0	0.0	
Set2(21)	18	**85.7**	2	9.5	1	4.8	
Set3(18)	12	**66.7**	5	27.8	1	5.6	
average		**81.0**		15.6		3.4	
Grade 2							
Set1(21)	6	28.6	11	**52.4**	4	19.1	
Set2(21)	2	9.5	13	**61.9**	6	28.6	
Set3(18)	0	0.0	16	**88.9**	2	11.1	
average		12.7		**67.7**		19.6	
Grade 3							
Set1(21)	0	0.0	1	4.8	20	**95.2**	
Set2(21)	3	14.3	7	33.3	11	**52.4**	
Set3(18)	0	0.0	0	0.0	18	**100.0**	
average		4.8		12.7		**82.5**	**77.1**

* Mean classification accuracy ♀ Testing data size

5-5-3 (four layers with 20 nodes in the input, 5 nodes in the first hidden, 5 nodes in the second hidden, and 3 nodes in the output layer) was used with the feature model Hc20. The results are summarized in **Table 7.17(c)**. In general, the MNN classifier performed the best with a mean classification accuracy of 77.1%. The parametric classifier with a mean classification accuracy of 68.3% performed better than the non-parametric classifier with a mean classification accuracy of 65.3%. As for the individual grades, the samples of grade 1 were correctly identified with an average rate of 74.1, 90.0, and 81.0% for the parametric, non-parametric, and MNN classifier, respectively. Very large differences (ranging from 9.5 to 65.1%) existed in the classification accuracies using different training and testing data sets, suggesting that either the data set were not large enough to provide adequate class information for training the classifiers or the features used were incapable of representing the class differences. The average classification accuracies of the three training and testing data sets were higher with the non-parametric classifier than with the parametric classifier.

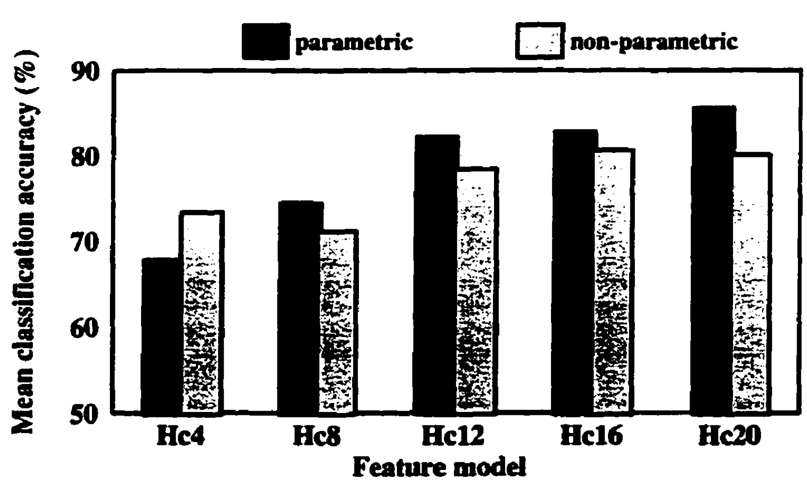


Fig 7.11 Evaluation of color feature models for grade identification analysis of bulk CWRS wheat samples using SAS DISCRIM.

VIII SUMMARY AND CONCLUSIONS

An illumination system was designed and developed to provide consistent, uniform diffused illumination for high quality color imaging of grain samples. Tests showed that the illumination was insensitive to the change in the supply voltage (with a maximum variation of 0.5% in the R, G, and B intensities for a 1 V change from the rated supply voltage), was stable with time (with a maximum variation of 1.20% variations in the R, G, and B intensities over 8 h), and was uniform over the FOV (with the maximum intensity variations of 3.1% across the width and 1.5% down the depth of the FOV).

A software package was developed on a microcomputer (Pentium 166 MHZ) under the DOS environment for grain image processing. The functions of the package includes imaging control, automatic segmentation of individual kernel images, and automatic extractions of 68 morphological and 78 color features for individual kernel images and 114 color feature for bulk sample images.

Using the developed illumination system, individual and bulk grain images of the samples collected in five grain types (CWRS wheat, CWAD wheat, barley, rye, and oats) from 20 different growing regions from the western Canada were acquired. Images of individual CWRS wheat kernels were also acquired for seven damage types (healthy, broken, mildewed, grass-green/green-frosted, black-point/smudged, heated, and bin-/fire-burnt). Morphological and color features were extracted from the acquired images using the developed software package and the classification analysis were conducted to differentiate different grain types and different damage types (for CWRS wheat) using statistical and neural network classification methods with different feature models (morphological, color, and combined). The following conclusions were made from the classification analysis:

1. For the grain type classification of individual kernels, using combined morphological and color features improved the classification accuracies over using morphological or color features alone. For a specific feature model (morphological, color, or combined), the non-parametric (k-nearest neighbor) statistical classifier always gave the best classification result. Using a non-parametric classifier with a selected combined feature model of 15 morphological and 13 color features, the average classification accuracies were 98.2, 96.9, 99.0, 98.2, and 99.0% for CWRS wheat, CWAD wheat, barley, rye, and oats, respectively, when trained and tested with three different training and testing data sets. Similar classification accuracies were achieved using a neural network classifier with the same features.
2. For the classification of damaged CWRS wheat kernels, color features proved to be more efficient than morphological features, however combining morphological features with color features improved the classification accuracies over using the color features alone. Again, the non-parametric (k-nearest neighbor) statistical classifier always gave the best classification result. Using a non-parametric classifier with a selected combined feature model of 24 color and 4 morphological features, the average classification accuracies were 92.5 (healthy), 90.3 (broken), 98.6 (mildewed), 99.0 (grass-green/green-frosted), 99.1 (black-point/smudged), 97.5 (heated), and 100.0 (bin-/fire-burnt) %, when trained and tested with three different training and testing data sets. Similar classification accuracies were achieved using a neural

network classifier with the same features.

3. For the grain type classification of bulk samples, a selected feature model of 8 color features was used with parametric and non-parametric statistical classifiers, and a NN classifier. When tested on three different training and testing data sets, set1, set2, and set3, all the tested bulk sample images were correctly classified by the non-parametric classifier, while 5 out of 21 bulk images of CWAD wheat in set 2 were mis-classified as CWRS wheat by the parametric classifier and 3 out of 21 images of CWAD wheat in set 1 were mis-classified as barley by the neural network classifier.

4. For the grade classification of bulk CWRS wheat samples, a selected feature model of 20 color features was used with parametric and non-parametric statistical classifiers, and a NN classifier. The NN classifier gave the best results with 80.95, 67.72, and 82.52% bulk wheat samples of grade 1, 2, and 3, respectively, correctly classified. However, large variations of 23.81% for grade 1, 36.51% for grade 2, and 47.61% for grade 3 existed in the classification accuracies when using different training and testing data sets, indicating that the grade information is probably not fully represented by the extracted color features.

IX CONTRIBUTION TO KNOWLEDGE

1. Demonstrated that surface color features of individual grain kernels can be used to significantly improve the classification accuracy obtained using the morphological features alone;
2. Demonstrated that surface color features of bulk grain samples can be used for rapid identification of different cereal grains (i.e., CWRS wheat, CWAD wheat, barley, oats, and rye);
3. Demonstrated that color features of individual grain kernels can be used for identification of healthy and some types of damaged wheat kernels (e.g., broken, grass-green/green-frosted, bin-/fire-burnt, black-point/smudged, heated, and mildewed);
4. Demonstrated that neural network classifiers are efficient in classifying different types of cereal grains;
5. Designed and developed a consistent, uniform diffused illumination system for high quality color imaging of grain samples;
6. Developed a color image processing software package on a microcomputer under the DOS environment dedicated to color grain image analysis.

X SUGGESTIONS FOR FUTURE RESEARCH

1. For practical applications, a line-scan color camera, instead of a area-sensing color camera should be used to acquire grain images from continuous grain flow on the belt;
2. To develop a practical system for identifying the constituents of a grain sample using the developed algorithms, more grain types and objects other than grains that are commonly found in uncleaned commercial grains (such as dockages and stone pieces) should be collected and included in the training data set;
3. For the classification of healthy and damaged grain kernels, more damage types and more damaged grain kernels should be collected and included in the training data set and the developed algorithms need to be tested with practical mixed samples (i.e., a small amount of different damaged kernels mixed with a large amount of healthy kernels);
4. An investigation on the effect of growing regions on grain kernel features could be helpful in developing a robust classifier;
5. An statistical analysis of the selected features should be made to determine the probability distributions of the features which could be useful in selecting a proper type of classifiers (parametric or non-parametric).

XI REFERENCES

Ballard, D.H. and C.M. Brown. 1982. *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall Inc. 523 p.

Barker, D.A., T.A. Vouri, M.R. Hegedus, and D.G. Myers. 1992a. The use of ray parameters for the discrimination of Australian wheat varieties. *Plant Varieties and Seeds* 5:35-45.

Barker, D.A., T.A. Vouri, and D.G. Myers. 1992b. The use of slice and aspect ratio parameters for the discrimination of Australian wheat varieties. *Plant Varieties and Seeds* 5:47-52.

Barker, D.A., T.A. Vouri, and D.G. Myers. 1992c. The use of Fourier descriptors for the discrimination of Australian wheat varieties. *Plant Varieties and Seeds* 5:93-102.

Barker, D.A., T.A. Vouri, M.R. Hegedus, and D.G. Myers. 1992d. The use of Chebychev coefficients for the discrimination of Australian wheat varieties. *Plant Varieties and Seeds* 5:103-111.

Brainard, D.H. and B.A. Wandell. 1990. Calibrated processing of image color. *Color Research and Application* 15(5):266-271.

Brogan, W.L. and A.R. Edison. 1974. Automatic classifications of grains via pattern recognition techniques. *Pattern Recognition* 6: 97-103.

Canada Grains Council. 1994. *Statistical Handbook.* Winnipeg, MB: Canada Grains Council.

Casady, W.W. and M.R. Paulsen. 1989. An automated kernel positioning device for computer vision analysis of grain. *Transactions of the ASAE* 32(5): 1821-1826.

Casady, W.W., M.R. Paulsen, J.F. Reid, and J.B. Sinclair. 1992. A trainable algorithm for inspection of soybean seed quality. *Transactions of the ASAE* 35(6): 2027-2034.

Castleman, K.R. 1979. *Digital Image Processing.* Englewood Cliffs, NJ: Prentice-Hall Inc.. 429p.

Chen, C., Y.P. Chiang, and Y. Pomeranz. 1989. Image analysis and characterization of cereal grains with a laser range finder and camera contour extractor. *Cereal Chemistry* 66: 466-470.

Duda, R.O. and P.E. Hart. 1973. *Pattern Classification and Scene Analysis.* New York, NY:

John Wiley and Sons, Inc. 374p

Ganapathy, S. 1984. Decomposition of transformation matrices for robot vision. In *Poceedings of 1984 International Conference on Robotics*, 130-136. Atlanta, GA.

Gershon, R. And A.P. Jepson. 1989. The computation of color constant descriptors in chromatic images. *Color Research and Application* 14(6):325-334.

Gonzalez, R.C. and R.E. Woods. 1992. *Digital Image Processing*. Menlo Park, CA: Addison-Wesley Pub. Co. 716 p.

Gonzalez, R.C. and R. Safabakhsh. 1982. Computer vision techniques for industrial application and robot control. *Computer* 15: 17-32.

Green, R.J. and S.J. Ismail. 1990. Colour error reduction in video systems. *IEEE Transactions on Broadcasting* 36(1):99-107.

Hand, D.J. 1981. *Discrimination and Classification*. New York, NY: John Wiley and Sons, Inc. 218p.

Haney, L., C. Precetti, and H. Gibson. 1994. Color matching of wood with a real-time machine vision system. ASAE Paper No. 94-3579. St. Joseph, MI: ASAE. 15 p.

Haralick, R.M. and L. Shapiro. 1992. *Computer and Robot Vision*. Reading, MA: Addision-Wesley Publishing Co. 672 p.

Hawk, A.L., H.H. Kaufmann, and C.A. Watson. 1970. Reflectance characteristics of various grains. *Cereal Science Today* 15: 381-384.

Hetzroni, A. And G.E. Miles. 1994.Color calibration for RGB video images. ASAE Paper No. 94-3007. St. Joseph, MI: ASAE. 8p.

Humphries, S. and W. Simonton. 1993. Identification of plant parts using color and geometric image data. *Transactions of the ASAE* 36(5): 1493-1500.

Keefe, P.D. 1992. A dedicated wheat grading system. *Plant Varieties and Seeds* 5:27-33.

Keefe, P.D. and S.R. Draper. 1986. The measurement of new characters for cultivar identification in wheat using machine vision. *Seed Science and Technology* 14:715-724.

Keefe, P.D. and S.R. Draper. 1988. An automated machine vision system for the morphometry of new cultivars and plant genebank accessions. *Plant Varieties and*

Seeds 1:1-11.

Kohonen, T. 1988. An introduction to neural computing. *Neural Networks* 1: 3-16.

Kranzler, G.A. 1985. Applying digital image processing in agriculture. *Agricultural Engineering* 66: 11-13.

Lee, Jr. R.L. 1988. Colorimetric calibration of a video digitizing system algorithm and applications. *Color Research and Application* 13(3):180-186.

Lenz, R. and R.Y. Tsai. 1987. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. In *Proceedings of 1987 IEEE International Conference on Robotics and Automation*, 68-75, Raleigh, NC.

Levine, M.D. 1985. *Vision in Man and Machine*. New York, NY: McGraw-Hill Book Co. 574p.

Miller, B.K. and M.J. Delwiche. 1989. A color vision system for peach grading. *Transactions of the ASAE* 32(4): 1484-1490.

Morrow, C.T., P.H. Heinemann, H.J. Sommer, Y. Tao, and Z. Varghese. 1990. Automated inspection of potatoes, apples, and mushrooms. In *Proceeding of International Advanced Robotics Programme*, 179-188. Avignon, France.

Murray, C.E. 1993. An automated seed presentation device for use in machine vision identification of grain. Unpublished B. Sc. thesis, Department of Agricultural Engineering, University of Manitoba, Winnipeg, MB. 38p.

Neuman, M., H.D. Sapirstein, E. Shwedyk, and W. Bushuk. 1987. Discrimination of wheat class and variety by digital image analysis of whole grain samples. *Journal of Cereal Science* 6:125-132.

Neuman, M.R., H.D. Sapirstein, E. Shwedyk, and W. Bushuk. 1989a. Wheat grain color analysis by digital image processing I. Methodology. *Journal of Cereal Science* 10: 175-182.

Neuman, M.R., H.D. Sapirstein, E. Shwedyk, and W. Bushuk. 1989b. Wheat grain color analysis by digital image processing II. Wheat class discrimination. *Journal of Cereal Science* 10: 183-188.

Nevatia, R. 1982. *Machine Perception*. Englewood Cliffs, NJ: Prentice-Hall Inc. 209 p.

Novini, A. 1990. Fundamentals of machine vision component selection. In *Food Processing*

Automation I -- Proceedings of the 1990 conference, 60-71. St. Joseph, MI: ASAE.

Panigrahi, S. and M. Misra. 1989. Color image processing for characterization of corn germplasm. ASAE Paper No. 89-3568. St. Joseph, MI: ASAE. 22 p.

Pao, Y. H. 1989. *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley Pub. Co. 309 p.

Parker, J.R. 1994. *Practical Computer Vision using C*. New York, NY: John Wiley and Sons, Inc. 482 p.

Precètti, C.J. and G.W. Krutz. 1993a. Building a color classification system. ASAE Paper No. 93-3003. St. Joseph, MI: ASAE. 18 p.

Precètti, C.J. and G.W. Krutz. 1993b. Real-time color classification system. ASAE Paper No. 93-3002. St. Joseph, MI: ASAE. 11 p.

Precètti, C.J. and G.W. Krutz. 1993c. A new seed corn husk deduction system using color machine vision. ASAE Paper No. 93-1012. St. Joseph, MI: ASAE. 10 p.

Putnam, D.F. and R.G. Putnam. 1970. *Canada: A Regional Analysis*. Toronto, ON: J.M. Dent and Sons, Inc. 390p

Rosenblatt, F. 1956. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics* 27:632-837.

Rosenblatt, F. 1962. *Principles of Neurodynamics*. Washinton, WA: Spartan Books. 616p.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. Learning internal representation by error propagation, In *Parallel Distributed Processing, vol.1*, ed. D.E. Rumelhart et al., 318-362. Cambridge, MA: MIT Press.

Sapirstein, H.D. 1995. Variety identification by digital image analysis. In *Identification of Food-Grain Varieties*, ed. C.W. Wrigley, 92-130. St. Paul, MN: American Association of Cereal Chemists, Inc.

Sapirstein, H.D. and J.M. Kohler. 1995. Physical uniformity of graded railcar and vessel shipments of Canada Western Red Spring wheat determined by digital image analysis. *Canadian Journal of Plant Science* 75: 363-369.

Sapirstein, H.D., M.R. Neuman, E.H. Wright, E. Shwedyk, and W. Bushuk. 1987. An instrumental system for cereal grain classification using digital image analysis. *Journal of Cereal Science* 6: 3-14.

Sapirstein, H.D. and W. Bushuk. 1989. Quantitative determination of foreign material and vitreosity in wheat by digital image analysis. In *ICC'89 Symposium: Wheat End-Use Properties*, 453-474. Lahti, Finland.

Sarkar, N.R. 1986. Machine vision in the food industry. In *ASAE Food Engineering News, October*, 3-5. St. Joseph, MI: ASAE.

SAS. 1990. *SAS User's Guide: Statistics*. Raleigh, NC: Statistical Analysis System, Inc.

Sejnowski, T.J. and C. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems* 1:145-168.

Shatadal, P., D.S. Jayas, and N.R. Bulley. 1995a. Digital image analysis for software separation and classification of touching grains. I. Disconnect algorithm. *Transactions of the ASAE* 38:635-643.

Shatadal, P., D.S. Jayas, and N.R. Bulley. 1995b. Digital image analysis for software separation and classification of touching grains. II. Classification. *Transactions of the ASAE* 38:645-649.

Shaw, W.E. 1990. Machine vision for detecting defects on fruits and vegetables. In *Food Processing Automation I -- Proceedings of the 1990 conference*, 50-59. St. Joseph, MI: ASAE.

Shearer, S.A. and F.A. Payne. 1990. Color and defect sorting of bell peppers using machine vision. *Transactions of the ASAE* 33(6): 2045-2050.

Shearer, S.A. and R.G. Holmes. 1990. Plant identification using color co-occurrence matrices. *Transactions of the ASAE* 33(6): 2037-2044.

Shyy, Y. and M.K. Misra. 1989. Color image analysis for soybean quality determination. ASAE Paper No. 89-3572. St. Joseph, MI: ASAE. 12 p.

Slaughter, D.C. and R.C. Harrell. 1987. Color vision in robotic fruit harvesting. *Transactions of the ASAE* 30(4): 1144-1148.

Symons, S.J. and R.G. Fulcher. 1988a. Determination of wheat kernel morphological variation by digital image analysis: I. Variation in Eastern Canadian milling quality wheats. *Journal of Cereal Science* 8:211-218.

Symons, S.J. and R.G. Fulcher. 1988b. Determination of wheat kernel morphological variation by digital image analysis: II. Variation in cultivars of soft white winter wheats. *Journal of Cereal Science* 8:219-229.

Tesauro, G. and T.J. Sejnowski. 1989. A parallel network that learns to play backgammon. *Artificial Intelligence* 39:357-390.

Thomas, D.L., F.J.K. daSilva, and W.A. Cromer. 1988. Image processing technique for plant canopy cover evaluation. *Transactions of the ASAE* 31(2): 428-434.

Thomson, W.H. and Y. Pomeranz. 1991. Classification of wheat kernels using three-dimensional image analysis. *Cereal Chemistry* 68: 357-361.

Tillet, R.D. 1990. Image analysis for agricultural processes. *Div. Note DN 1585, AFRC Inst.* Eng. Res., Silsoe, UK. 15 p.

Tillet, R.D. 1991. Image analysis for agricultural process: A review of potential opportunities. *Journal of Agricultural Engineering Research* 50:247-258.

Tominaga, S. 1992. Color classification of natural color images. *Color Research and Application* 17(4):230-239.

Toscani, G. and O.D. Faugeras. 1987. Camera calibration for 3D computer vision. In *International Workshop on Machine Vision and Math. Intelligence.* Tokyo, Japan.

Wigger, W.D., M.R. Paulsen, J.B. Litchfield, and J.B. Sinclair. 1988. Classification of fungal- damaged soybeans using color-image processing. ASAE Paper No. 88-3053. St. Joseph, MI: ASAE. 14 p.

Woebbecke, D.M., G.E. Meyer, K.V. Bargen, and D.A. Mortensen. 1994. Color indices for weed identification under various soil, residue and lighting conditions. ASAE Paper No. 94-3046. St. Joseph, MI: ASAE. 33 p.

Zayas, I., Y. Pomeranz, and F.S. Lai. 1985. Discrimination between Arthur and Arkan wheats by image analysis. *Cereal Chemistry* 62(2): 478-480.

Zayas, I., F.S. Lai, and Y. Pomeranz. 1986. Discrimination between wheat classes and varieties by image analysis. *Cereal* Chemistry 63(1): 52-56.

Zayas, I., Y. Pomeranz, and F.S. Lai. 1989. Discrimination of wheat and non-wheat components in grain samples by image analysis. *Cereal* Chemistry 66:233-237.

Appendix A

C LANGUAGE CODE OF SOFTWARE PACKAGE FOR GRAIN IMAGE PROCESSING

```
/***********************************************************************
 *                              litadj.c                                       *
 *                                                                             *
 *          Program to standardize the lighting configuration.                 *
 *          Monitors a small central square region (50 x 50) and displays      *
 *        graphically the average RGB value differences between                *
 *          desired and actual grey levels as IRIS, W/B BALANCE                *
 *          are adjusted. Completes once the desired and actual                *
 *          green grey levels are within the input error allowed               *
 *          for 10 consecutive frames.                                         *
 *                                                                             *
 *                  Xiangyang Luo, May 01/95                                   *
 *          Modified from UNISTAND.C BY Jeff Hehn MAY 22/93                    *
 *                                                                             *
 ***********************************************************************/
#include <stdio.h>
#include <graph.h>

#include        "c:\aurora\auerrs.h"
#include        "c:\aurora\audefs.h"

#define ON              1
#define OFF             0
#define INTERNAL        0
#define EXTERNAL        1
#define RGB_MODE        1
#define HSI_MODE        0
#define TRUE            1
#define FALSE           0

struct videoconfig screen;
main()
{
   int i,j,x,n, dgrey;
   float r_agrey, g_agrey, b_agrey, all_err;
   int rpixels[512], gpixels[512], bpixels[512];
   long rgn_sum, rgn_size, rpix_sum, gpix_sum, bpix_sum;
   int  rgn_arr[4], over_on[4] = {1,1,1,1};
   char buffer[25],clr_buf[25];
   int  rgrey_set, ggrey_set, bgrey_set;

/* Initialize aurora */
   au_err_msgs (ON);              /* enable display of AURORA error messages */
   au_init ();                    /* initialize AURORA resources */
   au_set_mode (RGB_MODE);
   au_display (ON);               /* enable display */
   au_buf_clear( 3 );             /* clear the overlay buffer */
   au_set_ovl_plns( over_on ); /* overlay is displayed    */
   rgn_arr[0] = 165;
   rgn_arr[1] = 180;
   rgn_arr[2] = 150;
   rgn_arr[3] = 150;
```

```
    rgn_size = (long)rgn_arr[2] * (long)rgn_arr[3];

    printf("\n Illumination Adjusting Program");
    printf("\n desired average grey level: ");
    scanf("%i", &dgrey);
    printf("\n allowable error (+- grey levels): ");
    scanf("%f", &all_err);

    au_set_grfx_pos( rgn_arr[0], rgn_arr[1], 1 );
    au_draw_box( rgn_arr[2], rgn_arr[3], 1 );

/* Set up graphics on screen */
    _setvideomode( _VRES16COLOR );  /* set to vga 640x480 - 16 color */
    _getvideoconfig( &screen );
    _clearscreen( _GCLEARSCREEN );  /* clear the screen */
    _rectangle( _GBORDER, 243, 99, 396, 356 );
    _setcolor( 5 );
    _moveto( 243, (int)(355-dgrey) );
    _lineto( 396, (int)(355-dgrey) );
    sprintf( clr_buf, "                    " );

/* Ok. Lets do it ! */
    au_set_sync (EXTERNAL);        /* select external sync */
    n=0;
    do
    {
        rgrey_set = FALSE, ggrey_set = FALSE, bgrey_set = FALSE;

        while( ! ggrey_set )
        {
          au_acquire(0,1);          /* acquire an image    */
          rpix_sum = 0, gpix_sum = 0, bpix_sum = 0;
          for(i=rgn_arr[0]; i<(rgn_arr[0]+rgn_arr[2]); i++)
          {
            au_get_trixel(0, i, rgn_arr[1], rgn_arr[3], rpixels, gpixels, bpixels);
            for(j=0; j<rgn_arr[3]; j++)
            {
                rpix_sum = (long)rpixels[j] + rpix_sum;
                gpix_sum = (long)gpixels[j] + gpix_sum;
                bpix_sum = (long)bpixels[j] + bpix_sum;
            }
          }
          _setcolor( 0 );
          _rectangle( _GFILLINTERIOR, 244, 100, 395, 355 );

          r_agrey = (float)rpix_sum / (float)rgn_size;
          _setcolor( 4 );
          _rectangle( _GFILLINTERIOR, 244, (355-(int)r_agrey), 293, 355 );
          g_agrey = (float)gpix_sum / (float)rgn_size;
          _setcolor( 2 );
          _rectangle( _GFILLINTERIOR, 295, (355-(int)g_agrey), 344, 355 );
          b_agrey = (float)bpix_sum / (float)rgn_size;
```

```
_setcolor( 1 );
_rectangle( _GFILLINTERIOR, 346, (355-(int)b_agrey), 395, 355 );
_setcolor( 5 );
_moveto( 243, (int)(355-dgrey) );
_lineto( 396, (int)(355-dgrey) );

if( (r_agrey < ( (float)dgrey - all_err) ))
{
	_settextposition( 2,2 );
	sprintf( buffer, "Red too low! avg: %3.2f", r_agrey );
	_outtext( buffer );
}
else if( (r_agrey > ( (float)dgrey + all_err) ))
{
	_settextposition( 2,2 );
	sprintf( buffer, "Red too high! avg: %3.2f", r_agrey );
	_outtext( buffer );
}
else
{
	rgrey_set = TRUE;
	_settextposition( 2,2 );
	sprintf( buffer, "Red is OK! avg: %3.2f", r_agrey );
	_outtext( buffer );
}

if( (g_agrey < ( (float)dgrey - all_err) ))
{
	n=0;
	_settextposition( 4,2 );
	sprintf( buffer, "Green too low! avg: %3.2f", g_agrey );
	_outtext( buffer );
}
else if( (g_agrey > ( (float)dgrey + all_err) ))
{
	n=0;
	_settextposition( 4,2 );
	sprintf( buffer, "Green too high! avg: %3.2f", g_agrey );
	_outtext( buffer );
}
else
{
	ggrey_set = TRUE;
	_settextposition( 4,2 );
	sprintf( buffer, "Green is OK! avg: %3.2f", g_agrey );
	_outtext( buffer );
}

if( (b_agrey < ( (float)dgrey - all_err) ))
{
	_settextposition( 6,2 );
	sprintf( buffer, "Blue too low! avg: %3.2f", b_agrey );
```

```
                _outtext( buffer );
            }
            else if( (b_agrey > ( (float)dgrey + all_err) ))
            {
                _settextposition( 6,2 );
                sprintf( buffer, "Blue too high! avg: %3.2f", b_agrey );
                _outtext( buffer );
            }
            else
            {
                bgrey_set = TRUE;
                _settextposition( 6,2 );
                sprintf( buffer, "Blue is OK! avg: %3.2f", b_agrey );
                _outtext( buffer );
            }

            _settextposition( 8,2 );
            _outtext( clr_buf );

            if( ggrey_set )
            {
                n = n + 1;
                _settextposition( 8,2 );
                sprintf( buffer, "   testing... %i    ", n );
                _outtext( buffer );
            }
        }

    } while( n < 10 );

    au_set_sync (INTERNAL);        /* select internal sync */
    _setvideomode( _DEFAULTMODE );
    au_end();
}

/************************************************************************
 *                              xvsave.c:                               *
 *                                                                      *
 *          Program to save a rectangular_pixel image (in R&W, RGB, or HSI)  *
 *          from DT2871 framegrabber using the Aurora subroutines in the     *
 *          khoros VIFF image format.                                   *
 *                                                                      *
 *          X. Luo, Oct., 1995                                          *
 *          Modified from Jeff's xvsave.c                               *
 *                                                                      *
 ************************************************************************/
#include    <stdio.h>
#include    <string.h>
#include    <dos.h>
#include    "viff.h"            /* VIFF header definitions */
#include    "vdefines.h"        /* more VIFF information   */
#include    "c:\aurora\auerrs.h"  /* Aurora include files    */
```

```
#include     "c:\aurora\audefs.h"
#include          "mousfunt.h"          /* mouse function definitions */

#define OVER_BUF      3     /* the auxiliary buffer */
#define INT_BUF      0     /* the intensity buffer */
#define HUE_BUF      2     /* the hue buffer */
#define SAT_BUF      1     /* the saturation buffer */
#define ABS_VAL      0     /* don't use absolute value in filter operations */
#define HSI_MODE      0
#define RGB_MODE      1

#define ON          1
#define INTERNAL      0
#define EXTERNAL      1

int    over_off[4]={0,0,0,0}, over_on[4]={1,1,1,1};
void   get_roi_mouse( int *, int * );

main()
{
        FILE *fp, *fpsat, *fphue;
        char header1[520];
        unsigned char *image;
        int *temp1, *temp2, *temp3, tem1[512], tem2[512], tem3[512];
        unsigned long header2[5], header4[119], band_form;
        float header3[2];

        int    status,c,freeze;              /* AURORA library return status */
        int    buf_num,final,loop;              /* buffer to save*/
        int    corner[4], file_ok, rgn_arr[4];
        int    pic_num,bands,color,color_mode;  /* picture number */
        char    fname[50],comment[1000],zip[100],ch;
        int    rows=480,cols=512,i,j,k;        /* Default 512x480 pixels */
        int    startx, starty, change_roi;      /* starting ROI coordinates */
        int    x=50,y=50,xn=0,yn=0,box_col,box_row;
        int       height, left_over, lim_32k, num_blocks;
        float     l_w_ratio = 1.275539;

/*      Initialization of Aurora  */
   status = au_err_msgs (ON); /* enable display of AURORA error messages */
   status = au_init();      /* initialize AURORA resources */
   pic_num = 0;
   status = au_pic_clear( pic_num );
   status = au_display(ON);

/* Get an image into buffer using passthru and freezeframe */
   printf("\n Select color mode B&W(2) or RGB(1) or HSI(0): ");
   scanf("%i", &color );
   if( color < 0 || color >= 2 )      /* make sure mode is valid     */
         color_mode = 0;              /* default to HSI            */
   else
         color_mode = color;
```

```
au_set_mode( color_mode );       /* set color mode HSI or RGB   */
au_set_sync (EXTERNAL);          /* select external sync        */
au_passthru();                   /* pass images                 */
printf("\n To freeze frame (1): ");
scanf("%i",&freeze);
au_freeze_frame();               /* freeze the frame            */
au_set_sync (INTERNAL);          /* set back to internal sync   */

/* transfer rectangular pixels to square pixels */
for(i=0; i<480; i++)
{
    j=(int)ceil((float)i*l_w_ratio);
    if( j<480 )
      au_get_trixel(0, j, 0, 512, tem1, tem2, tem3);
    else
      for(k=0; k<=511; k++)
      {
          tem1[k]=0, tem2[k]=0, tem3[k]=0;
      }
    au_put_trixel(0, i, 0, 512, tem1, tem2, tem3);
}

if( color == 2 )                 /* if B&W then clear other bufs */
{
    au_buf_clear( HUE_BUF );
    au_buf_clear( SAT_BUF );
}
final=0;
au_buf_clear( OVER_BUF );        /* clear the overlay for ROI   */

/* Determine ROI to be saved */
while( final < 1 )
{
    printf("\n Use default ROI (512x376)- yes(1) no(0): ");
    scanf("%i",&change_roi);
    if( change_roi == 0)
    {
      printf("\n Mark upper left point:");
      get_roi_mouse( &xn,&yn );
      startx = xn;
      starty = yn;
      printf("point selected (%i,%i)", xn,yn);
      au_set_grfx_pos( starty,startx+10,1 );
      corner[0] = starty; corner[1] = startx;
      corner[2] = starty+10; corner[3]= startx;
      au_draw_lines( 2, corner, 1 );
      printf("\n Mark lower right point:");
      do     /* get 2nd point and make sure it is valid */
      {
        get_roi_mouse( &xn,&yn );
        cols = xn - startx;
        rows = yn - starty;
```

```
            } while ( rows<=1 || cols<=1 );
            printf(" point selected (%i,%i)\n", xn,yn);
          }
          else
          {
            startx=0; starty=0; rows=376; cols=512;
          }
/*  Draw a box on monitor to indicate region to be saved */
          box_col = cols; box_row = rows;
          if( cols>510 )
                  box_col=510;  /* limit 510 for aurora to work */

          au_set_grfx_pos( starty,startx,1 );
          au_draw_box( box_row, box_col, 1);
          printf ("\n Use coordinates (%i,%i) (%i,%i)",
                          startx,starty,startx+cols,starty+rows);
          printf("\n Yes(1) No(0): ");
          scanf("%i", &final );
          au_set_grfx_pos( starty,startx,1 );
          au_draw_box( box_row,box_col,0 );
    }
/* Get filename and comment */
   do
   {
    printf("\n Full file name drive:\\path\\filename\n ");
    flushall();
    gets(fname);
    printf("\n Enter a comment:");
    flushall();
    gets( zip );
    strcpy( comment,"        ");
    strcat( comment, zip );
    strcpy( header1, comment );
    flushall();
    printf(" Save file and comment YES(1) NO(0): " );
    scanf("%i", &file_ok );
   } while( file_ok == 0 ); /* otherwise revise filename and comment */

/* Set viff header for RGB, IHS, or B&W image */
        switch( color )
        {
        case 0:
                bands = 3;
                band_form = VFF_CM_IHS;
                printf("\n Saving color IHS viff image ");
                break;
        case 1:
                bands = 3;
                band_form = VFF_CM_genericRGB;
                printf("\n Saving color RGB viff image ");
                break;
        case 2:
```

```
                bands = 1;
                band_form = VFF_CM_NONE;
                printf("\n Saving B&W viff image ");
                break;
        }

 /* Fill viff header buffers with appropriate values for save */
        for( i=0; i<119; i++)
                header4[i] = 0;    /* fill header4 with zeros */
        header1[0] = XV_FILE_MAGIC_NUM;
        header1[1] = XV_FILE_TYPE_XVIFF;
        header1[2] = XV_IMAGE_REL_NUM;
        header1[3] = XV_IMAGE_VER_NUM;
        header1[4] = VFF_DEP_NSORDER;  /* intel byte ordering */
        header2[0] = cols;             /* row length */
        header2[1] = rows;             /* column length */
        header2[2] = 0;
        header2[3] = 0;
        header2[4] = 0;
        header3[0] = 0.0;
        header3[1] = 0.0;
        header4[0] = VFF_LOC_IMPLICIT;
        header4[1] = 0;
        header4[2] = 1;                /* 1 image */
        header4[3] = bands;            /* three bands I,H,S */
        header4[4] = VFF_TYP_1_BYTE;   /* 1 byte per band */
        header4[5] = VFF_DES_RAW;
        header4[6] = VFF_MS_NONE;
        header4[7] = 0;
        header4[8] = 0;
        header4[9] = 0;
        header4[10] = 0;
        header4[11] = VFF_MAP_OPTIONAL;
        header4[12] = 0;
        header4[13] = band_form;       /* format of bands */
        header4[14] = 0;
        header4[15] = 0;
        header4[16] = 0;
        header4[17] = 0;

/* Save image to disk */
  /* write the viff header (1024 bytes) information to a file first */

  if( (fp = fopen(fname,"wb") ) != NULL)
  {
        printf("\n file %s opened \n writing header", fname );
        fwrite( header1, sizeof( char ), 520, fp );
        fwrite( header2, sizeof( long ), 5, fp );
        fwrite( header3, sizeof( float ), 2, fp );
        fwrite( header4, sizeof( long ),119 , fp );

/* write the image data to the file */
```

```
        height = (int) (16000/cols);  /* 32 kb at 2 bytes/pixel */
        num_blocks = (int) (rows/height + 1);
        lim_32k = height*cols;     /* size of must be under 32 kb */
        temp1 = (int *) calloc( (size_t)lim_32k, (size_t)sizeof(int) );
        temp2 = (int *) calloc( (size_t)lim_32k, (size_t)sizeof(int) );
        temp3 = (int *) calloc( (size_t)lim_32k, (size_t)sizeof(int) );
        image = (unsigned char *) calloc( (size_t)lim_32k, (size_t)sizeof(char) );
        rgn_arr[1]=startx; rgn_arr[2]=height; rgn_arr[3]=cols;

/* check last arrays allocated to see if ok, check both cause one is far  */
        printf("\n buffering image");
        if( image!=NULL || temp3!=NULL)
        {
          loop=0;
          if( (fphue = fopen("f:\hue.buf","wb") ) != NULL )
          {
            if( (fpsat = fopen("f:\sat.buf","wb") ) != NULL )
            for( i=starty; i<starty+rows-height; i=i+height)
            {
                  rgn_arr[0] = i;
                  au_set_act_rgn( rgn_arr );
                  au_get_pic_rgn( 0, temp1,temp2,temp3 );
                  for( j=0; j<lim_32k; j++ )
                    image[j] = (unsigned char) temp1[j];
                  fwrite( image, sizeof(char), lim_32k, fp ); /* write out int directly */
                  fwrite( temp2, sizeof(int), lim_32k, fpsat );
                  fwrite( temp3, sizeof(int), lim_32k, fphue );
                  loop++;
            }
            left_over = starty+rows-i;    /* height is what is left */
            rgn_arr[0] = i;
            rgn_arr[2] = left_over;
            au_set_act_rgn( rgn_arr );
            au_get_pic_rgn( 0, temp1,temp2,temp3 );
            for( j=0; j<left_over*cols; j++ )
                  image[j] = (unsigned char) temp1[j];
            fwrite( image, sizeof(char), left_over*cols, fp ); /* write out int directly */
            fwrite( temp2, sizeof(int), left_over*cols, fpsat );
            fwrite( temp3, sizeof(int), left_over*cols, fphue );
            fclose( fpsat );
            fclose( fphue );
            fphue = fopen("f:\hue.buf","rb");
            fpsat = fopen("f:\sat.buf","rb");
            if( band_form == VFF_CM_genericRGB) /* also output r&b bufs*/
            {
                  for( i=0; i<num_blocks-1; i++)
                  {
                    fread(  temp1, sizeof(int), lim_32k, fpsat );
                    for( j=0; j<lim_32k; j++ )
                      image[j] = (unsigned char) temp1[j];
                    fwrite( image, sizeof(char), lim_32k, fp );
                  }
```

```
        fread( temp1, sizeof(int), left_over*cols, fpsat );
        for( j=0; j<left_over*cols; j++ )
          image[j] = (unsigned char) temp1[j];
        fwrite( image, sizeof(char), left_over*cols, fp );
        for( i=0; i<num_blocks-1; i++)
        {
          fread( temp1, sizeof(int), lim_32k, fphue );
          for( j=0; j<lim_32k; j++ )
            image[j] = (unsigned char) temp1[j];
          fwrite( image, sizeof(char), lim_32k, fp );
        }
        fread( temp1, sizeof(int), left_over*cols, fphue );
        for( j=0; j<left_over*cols; j++ )
         image[j] = (unsigned char) temp1[j];
        fwrite( image, sizeof(char), left_over*cols, fp );
    }
    if( band_form == VFF_CM_IHS) /*also output s&h buffers */
    {
        for( i=0; i<num_blocks-1; i++)
        {
          fread( temp1, sizeof(int), lim_32k, fphue );
          for( j=0; j<lim_32k; j++ )
            image[j] = (unsigned char) temp1[j];
          fwrite( image, sizeof(char), lim_32k, fp );
        }
        fread( temp1, sizeof(int), left_over*cols, fphue );
        for( j=0; j<left_over*cols; j++ )
          image[j] = (unsigned char) temp1[j];
        fwrite( image, sizeof(char), left_over*cols, fp );
        for( i=0; i<num_blocks-1; i++)
        {
          fread( temp1, sizeof(int), lim_32k, fpsat );
          for( j=0; j<lim_32k; j++ )
            image[j] = (unsigned char) temp1[j];
          fwrite( image, sizeof(char), lim_32k, fp );
        }
        fread( temp1, sizeof(int), left_over*cols, fpsat );
        for( j=0; j<left_over*cols; j++ )
          image[j] = (unsigned char) temp1[j];
        fwrite( image, sizeof(char), left_over*cols, fp );
    }
    fclose( fpsat );
    fclose( fphue );
  }
  else
    printf("\n unable to open files on ramdisk f:");
}
else
{
  printf( "\n unable to allocate memory required ");
}
fclose( fp );
```

```
        free( temp1 ); free(temp2); free(temp3); free( image );

  }
  else
    perror("write error");

  au_set_ovl_plns( over_off );
  free( header1 );
  status = au_end();                    /* release AURORA resources */
}



/*****************************************************************
        Function to get a point from the mouse movement.
        Draws a cursor on SONY and moves it with mouse.
*****************************************************************/
void get_roi_mouse( int *xr, int *yr )
{
  int xp,yp,flag;

/* Initialize mouse */
  ms_init();
  xp = *xr;
  yp = *yr;
  au_set_ovl_plns( over_on );
  au_set_curs_pos( yp,xp );
  ms_movecrsr( yp, xp );

  if (rodent.exists)
  {
        flag = 1;
        clearbuttons();
        ms_movement();

        ms_sethrange( 0,510 );
        ms_setvrange( 0,480 );

        do
        {
          ms_getstatus();
          ms_movement();
          switch (rodent.btnstatus)
          {
           case 1:
             flag = 0;
             break;
           case 2:
             flag = 0;
             break;
           case 3:
             flag = 0;
```

```
                break;
              }
              au_set_curs_pos( rodent.row, rodent.column );

          } while (flag);
          yp = rodent.row; xp = rodent.column;
          ms_init();
  }
  else   /* flag that no mouse found */
  {
          printf("\n Sorry no rodent found on this machine");
  }
  *xr = xp;
  *yr = yp;
  au_set_ovl_plns( over_on );
}        /* end of get_roi_mouse  */


#include <stdio.h>
#include <dos.h>

/*   MOUSE FUNCTIONS    */
void clearbuttons(void)
{
  inregs.x.ax = 0x05;
  inregs.x.bx = LEFT;
  MouseCall;

  inregs.x.ax = 0x05;
  inregs.x.bx = RIGHT;
  MouseCall;

  inregs.x.ax = 0x06;
  inregs.x.bx = LEFT;
  MouseCall;

  inregs.x.ax = 0x06;
  inregs.x.bx = RIGHT;
  MouseCall;
}

int ms_btnpress(int button)
{
  inregs.x.ax = 0x05;
  inregs.x.bx = button;

  MouseCall;

  rodent.btnstatus = outregs.x.ax;
  rodent.btnclicks = outregs.x.bx;
  rodent.column = outregs.x.cx;
  rodent.row = outregs.x.dx;
```

```
    return outregs.x.bx;
  }

  int ms_btnrelease(int button)
  {
    inregs.x.ax = 0x06;
    inregs.x.bx = button;

    MouseCall;

    rodent.btnstatus = outregs.x.ax;
    rodent.btnclicks = outregs.x.bx;
    rodent.column = outregs.x.cx;
    rodent.row = outregs.x.dx;

    return outregs.x.bx;
  }

 void ms_exclude(int topleftx, int toplefty, int btmrtx, int btmrty)
 {
   inregs.x.ax = 0x10;
   inregs.x.cx = topleftx;
   inregs.x.dx = toplefty;
   inregs.x.si = btmrtx;
   inregs.x.di = btmrty;

   MouseCall;
 }

 int ms_getstatus(void)
 {
   inregs.x.ax = 0x03;

   MouseCall;

   rodent.btnstatus = outregs.x.bx;
   rodent.column = outregs.x.cx;
   rodent.row = outregs.x.dx;

   return outregs.x.bx;
 }

void ms_hidecrsr(void)
{
  if(rodent.cursor_display)
    {
    inregs.x.ax = 0x02;
    MouseCall;
    }
}

int ms_init(void)
```

```
{
  inregs.x.ax = 0;

  MouseCall;

  rodent.exists = outregs.x.ax;
  return outregs.x.ax;
}

void ms_lightpenoff(void)
{
  inregs.x.ax = 0x0E;
  MouseCall;
}

void ms_lightpenon(void)
{
  inregs.x.ax = 0x0D;
  MouseCall;
}

void ms_movecrsr(int row, int col)
{
  inregs.x.ax = 0x04;
  inregs.x.cx = col;
  inregs.x.dx = row;

  MouseCall;
}

void ms_movement(void)
{
  inregs.x.ax = 0x0B;

  MouseCall;

  rodent.hmovement = outregs.x.cx;
  rodent.vmovement = outregs.x.dx;
}

void ms_sethrange(int leftcol, int rightcol)
{
  inregs.x.ax = 0x07;
  inregs.x.cx = leftcol;
  inregs.x.dx = rightcol;

  MouseCall;
}

void ms_setvrange(int upperrow, int lowerrow)
{
  inregs.x.ax = 0x08;
```

```
  inregs.x.cx = upperrow;
  inregs.x.dx = lowerrow;

  MouseCall;
}

void ms_settextcrsr(int cursortype, int scan1, int scan2)
{
  inregs.x.ax = 0x0A;
  inregs.x.bx = cursortype;
  inregs.x.cx = scan1;
  inregs.x.dx = scan2;

  MouseCall;
}

void ms_showcrsr(void)
{
  int i, counter;

  inregs.x.ax = 0x2A;
  MouseCall;
  counter = inregs.x.ax;

  for (i = 1; i < counter; i++)
    {
    inregs.x.ax = 0x01;
    MouseCall;
    }
  rodent.cursor_display = 1;
}

void waitclick(int button)
{
  char *whichbtn[4] = {"the left button",
                       "the right button",
                       "both buttons",
                       "any button"};

  printf("Click %s to continue.", whichbtn[button]);
  rodent.btnstatus = 0;
  do
  ms_getstatus();
  while (rodent.btnstatus != 0);
  if (button < 3)
    {
    do
          ms_getstatus();
    while (rodent.btnstatus != button + 1);
    }
  else
    {
```

```
      do
            ms_getstatus();
      while (rodent.btnstatus <= 0);
      }
    do
      ms_getstatus();
    while (rodent.btnstatus != 0);
  }

 void clearscreen(void)
 {
   int x;

   for (x =0; x <25; x++)
     printf("\n");
 }

 void locate(char x, char y)
 {
   inregs.h.ah = 0x02;
   inregs.h.dh = y-1;
   inregs.h.dl = x-1;
   inregs.h.bh = 0;

   int86(0x10, &inregs, &outregs);
 }

/**********************************************************************
 *                              gsrgb.c                                      *
 *                                                                           *
 *          Program to calculate the average RGB values over a central       *
 *          area of 50x50 pixels in FOV every ten min. for 8 hrs. Used       *
 *          for light stability testing.                                     *
 *                                                                           *
 *          X. Luo, Jan. 1995                                                *
 *          Modified from UNIFOV.C by Jeff                                   *
 *                                                                           *
 *********************************************************************/
#include <stdio.h>
#include <time.h>

#include        "c:\aurora\auerrs.h"
#include        "c:\aurora\audefs.h"

#define ON              1
#define OFF             0
#define INTERNAL        0
#define EXTERNAL        1
void delay (clock_t wait);
void main(void)
{
  int pic_num,i,j,k,x,y,status;
```

```
    int color_mode,freeze;
    int over_on[4] = {1,1,1,1}, over_off[4] = {0,0,0,0};
    int pixel0[112],pixel1[112],pixel2[112];
    int hbsize, vbsize, hsize, vsize;
    float sum0,sum1,sum2;
    float rgn_size;
    FILE *outfile;
    char fname[256];

    printf("\n*** Program to calculate avg. RGB values over a 50x50 central area ***");

 /*         Initialize aurora  */
    status = au_err_msgs (ON);      /* enable display of AURORA error messages */
    status = au_init();                         /* initialize AURORA resources */
    pic_num = 0;
    status = au_pic_clear(pic_num);
    status = au_display (ON);                   /* enable display */
    color_mode = 1;
    au_set_mode (color_mode);
    au_set_sync (EXTERNAL);
    au_set_ovl_plns(over_on);
    au_buf_clear(3);

    vbsize = 50;
    hbsize = 50;
    vsize = 480;
    hsize = 512;
    rgn_size = (float)vbsize * (float)hbsize;

 /* Draw a box around the 50 x 50 ROI */
    y=215;
    x=231;
    au_set_grfx_pos( y, x, 1 );
    au_draw_box( vbsize, hbsize, 1 );

 /* Get the name of the output file */
    printf("\n File name to save data: ");
    flushall();
    gets( fname );
    flushall();
    if( (outfile = fopen(fname,"at") ) != NULL)
    {
         printf( "\n\nAvg. RGB values over a 50x50 ROI" );
         printf( "\n\n  R\t  G\t  B");
         fprintf( outfile,"\n\nAvg. RGB values over a 50x50 ROI" );
         fprintf( outfile,"\n\n  R\t  G\t  B");

 /* Calculate the average RGB over the 50 x 50 ROI every ten min.*/
         for (k=0; k<49; k++)
         {
           au_passthru();
           au_freeze_frame();
```

```
            sum0 = 0.0; sum1 = 0.0; sum2 = 0.0;
            for(i=y; i<(y+vbsize); i++)
            {
                  au_get_trixel(0, i, x, hbsize, pixel0, pixel1, pixel2);
                  for(j=0; j<hbsize; j++)
                  {
                    sum0 = (float)pixel0[j] + sum0;
                    sum1 = (float)pixel1[j] + sum1;
                    sum2 = (float)pixel2[j] + sum2;
                  }
            }

/* output the results */
           fprintf( outfile, "\n%3.2f\t%3.2f\t%3.2f",
                                        sum0/rgn_size,
                                        sum1/rgn_size,
                                        sum2/rgn_size);
           printf("\n%3.2f\t%3.2f\t%3.2f",
                                        sum0/rgn_size,
                                        sum1/rgn_size,
                                        sum2/rgn_size);

           delay ((clock_t)600*CLOCKS_PER_SEC);
        }
  }
  else
  printf("\n could not open file %s", fname );
  fclose( outfile );

  au_set_sync(INTERNAL);
  au_set_ovl_plns( over_off );
  au_end();
}

void delay(clock_t wait)
{
  clock_t goal;
  goal = wait + clock();
  while ( goal > clock() );
}
```

```
/**********************************************************************
 *                              indiv.c                                    *
 *                                                                         *
 *              Program to extract individual kernel features from an image *
 *              stored in a file                                           *
 *                                                                         *
 *              X. Luo, June. 1996                                         *
 *                                                                         *
 **********************************************************************/
#include "base.c"

void main(int argc, char *argv[])
{
 FILE  *outf;
 struct image *a, *b, *binary_object, *color_object;
 struct feature *objf;
 char  fp1[256], fp2[256], fp3[256];        /* fp1: file name ofinput image
                                                  fp2: file name of calibration image
                                                  fp3: file name of output features  */
 double mm_per_pixel;
 int i, ibegin, err, t, m, n;
 int *ptr;
 int **obj_ptr;

 err = 0;

 if (argc != 4){
         an_error(GET_USAGE);
         exit(0);
 }

 strcpy (fp1, argv[1]);
 strcpy (fp2, argv[2]);
 strcpy (fp3, argv[3]);

/* Read in calibration (coin) image to image A */
// read_img (&a, fp2, &err);
 read_image_in_viff (&a, fp2, &err);
 if (err) {
         an_error(err);
         exit(0);
 }
// disp_image(a,0,&err);

/* Copy red band of coin image A into image B */
 b = 0;
// copy_image (a, &b, 0, &err);
 copy_image (a, &b, 1, &err);
 if (err){
         an_error(err);
         exit(0);
 }
```

```
  free_image (a, &err);
  if (err){
          an_error(err);
          exit(0);
  }
 // disp_image(b,0,&err);

 /* Transfer rectangular pixel image B to square pixel image A */
  rectangular_to_square (b, &a, &err);
  if (err) {
          an_error(err);
          exit(0);
  }

  free_image (b, &err);    __
  if (err){
          an_error(err);
          exit(0);
  }

 // disp_image(a,0,&err);

 /* Get the calibration scale from image A */
  mm_per_pixel = get_scale(a);

  free_image (a, &err);
  if (err){
          an_error(err);
          exit(0);
  }

 /* Read in object image (in viff) to image B */
  read_image_in_viff(&b, fp1, &err);
  if (err) {
          an_error(err);
          exit(0);
  }
 // disp_image(b,0,&err);

 /* Transfer rectangular pixel image B to square pixel image A */
  rectangular_to_square (b, &a, &err);
  if (err) {
          an_error(err);
          exit(0);
  }

  free_image (b, &err);
  if (err){
          an_error(err);
          exit(0);
  }
 // disp_image(a,0,&err);
```

```
/* Copy red band of object image A into image B */
 b = 0;
// copy_image (a, &b, 1, &err);
 copy_image (a, &b, 1, &err);
 if (err){
         an_error(err);
         exit(0);
 }

/* Threshold the red band image B to get a binary image B */
 thresh_is (b, &t,&err);
 if (err){
         an_error(err);
         exit(0);
 }
// disp_image(b,0,&err);

 threshold (b, t, &err);
 if (err){
         an_error(err);
         exit(0);
 }
// disp_image(b,0,&err);

/* Allocate object pointer which contains the coordinates of each region */
 obj_ptr = (int **)malloc((size_t)sizeof(int *)*MAX_OBJECT_NUM);
 if (!obj_ptr){
         an_error(OUT_OF_STORAGE);
         exit(0);
 }

 for (i = 0; i < MAX_OBJECT_NUM; i ++) {
         ptr = (int *) malloc((size_t)sizeof(int)*4);
         if (!ptr){
          an_error(OUT_OF_STORAGE);
          exit(0);
         }
         else obj_ptr[i] = ptr;
 }

/* Mark each seperated regions, ignore very small regions, and fill holes
         in any regions to get a labelled image B.*/
 err = 0; n = 0; ibegin = 0; m = 0;
 /* n: no.of marked regions,m:no. of pixels in a region,
         ibegin: the first row of the last marked region     */
 while (err == 0) {
         region_4 (b, n+1, &ibegin, &m, &err);
         if (err == NO_REGION) break;
         /* Ignore very small regions */
         if (m < 60) {
          del_reg (b, n+1, &err);
          if (err){
```

```
                    an_error(err);
                    exit(0);
              }
              continue;
            }

/* Fill holes in the region marked n+1, and return the coordinates
            of the region in obj_ptr[n] array.                          */
            fill_holes (b, n+1, obj_ptr[n], &err);
            if (err){
              an_error(err);
              exit(0);
            }
            n++;
  }
// disp_image(b,0,&err);

// printf ("\nNo. of objects is %d.\n", n);
// _getch();

/* Allocate feature struct pointer */
  objf = (struct feature *)malloc((size_t)sizeof(struct feature));

/* Open the output feature file */
  outf = fopen(fp3, "ab");
  if (outf == NULL){
            an_error(CANNOT_OPEN_FILE);
            exit(0);
  }
/* write the feature names to the output file */
// write_fname(outf);

  for (i = 0; i < n; i ++){
/* Extract a binary & a color (grey-level) image of the object marked i+1 */
            extract_obj (b, a, &binary_object, &color_object, i+1, obj_ptr[i], &err);
            if (err){
              an_error(err);
              exit(0);
            }
// disp_image(binary_object, 0, &err);

/* Compute size and shape features of the object */
            size_shape_features(binary_object, OBJECT, objf, mm_per_pixel, &err);
            if (err){
              an_error(err);
              exit(0);
            }
//   disp_image(color_object, 0, &err);

/* Compute color features of the object */
            color_features(binary_object, color_object, OBJECT, objf, 16, &err);
            if (err){
```

```
            an_error(err);
            exit(0);
           }

//         fft(objf->radR, 5);
//         fft(objf->areaR, 5);
//         fft(objf->perimR, 5);

/* Write measured features to output file */
           write_feature(outf, objf, fp1, i);
  }

  fclose(outf);

  free (objf);

  free_image (binary_object,&err);
  if (err){
           an_error(err);
           exit(0);
  }

  free_image (color_object,&err);
  if (err){
           an_error(err);
           exit(0);
  }

  free_image (b, &err);
  if (err){
           an_error(err);
           exit(0);
  }

  free_image (a, &err);
  if (err) an_error(err);

  exit(0);
}
```

```
/***********************************************************************
 *                              bulk.c                                  *
 *                                                                      *
 *       Program to extract bulk features from a bulk grain image      *
 *       stored in a file                                               *
 *                                                                      *
 *       X. Luo, June. 1996                                             *
 *                                                                      *
 ***********************************************************************/

#include "base.c"

void main(int argc, char *argv[])
{
 FILE *outf;
 struct image *a, *b;
 struct bfeature *bf;
 char fp1[256], fp2[256], fp3[256];       /*fp1: file name ofinput image
                                            fp2: file name of calibration image
                                            fp3: file name of output features */
 double mm_per_pixel;
 int err;

 err = 0;
 if (argc != 4){
        an_error(GET_USAGE);
        exit(0);
 }
 strcpy (fp1, argv[1]);
 strcpy (fp2, argv[2]);
 strcpy (fp3, argv[3]);

/* Read in calibration (coin) image to image A */
// read_img (&a, fp2, &err);
 read_image_in_viff (&a, fp2, &err);
 if (err) {
        an_error(err);
        exit(0);
 }
// disp_image(a,0,&err);
/* Copy red band of coin image A into image B */
 b = 0;
// copy_image (a, &b, 0, &err);
 copy_image (a, &b, 1, &err);
 if (err){
        an_error(err);
        exit(0);
 }
 free_image (a, &err);
 if (err){
        an_error(err);
        exit(0);
```

```
  }
 // disp_image(b,0,&err);

 /* Transfer rectangular pixel image B to square pixel image A */
  rectangular_to_square (b, &a, &err);
  if (err) {
          an_error(err);
          exit(0);
  }
  free_image (b, &err);
  if (err){
          an_error(err);
          exit(0);
  }
 // disp_image(a,0,&err);
 /* Get the calibration scale from image A */
  mm_per_pixel = get_scale(a);

  free_image (a, &err);
  if (err){
          an_error(err);
          exit(0);
  }

 /* Read in object image (in viff) to image A */
  read_image_in_viff(&a, fp1, &err);
  if (err) {
          an_error(err);
          exit(0);
  }
 // disp_image(a,0,&err);

 /* Transfer rectangular pixel image A to square pixel image B */
  rectangular_to_square (a, &b, &err);
  if (err) {
          an_error(err);
          exit(0);
  }

  free_image (a, &err);
  if (err){
          an_error(err);
          exit(0);
  }
 // disp_image(b,0,&err);

 /* Allocate a bulk image feature struct */
  bf = (struct bfeature *) malloc((size_t)sizeof(struct bfeature));
 /* compute bulk image features */
  bulk_feature(b, bf, 32, &err);
  if (err){
          an_error(err);
```

```
          exit(0);
  }
// disp_image(b,0,&err);
  bf->kn = bf->kn/mm_per_pixel;

/* Open the output feature file */
  outf = fopen(fp3, "ab");
  if (outf == NULL){
          an_error(CANNOT_OPEN_FILE);
          exit(0);
  }

/* write the bulk feature to the output file */
  write_bf(outf,bf,fp1);

  fclose(outf);

  free (bf);

  free_image (b, &err);
  if (err) an_error(err);

  exit(0);
}

/**********************************************************************
 *                          disp_viff.c                               *
 *                                                                    *
 *         Program to display the square, thresholded, and labelled images of   *
 *         a viff-formatted color image stored in a file              *
 *                                                                    *
 *         X. Luo, June. 1996                                         *
 *                                                                    *
 **********************************************************************/

#include "base.c"

void main(int argc, char *argv[])
{
 struct image *a, *b;
 char  fp1[256]; /*file name of input image */

 int i, ibegin, err, t, m, n;
 int *ptr;
 int **obj_ptr;

 err = 0;

 if (argc != 2){
        an_error(GET_USAGE);
        exit(0);
 }
```

```
strcpy (fp1, argv[1]);

// read_img (&a, fp1, &err);
read_image_in_viff (&b, fp1, &err);
if (err) {
        an_error(err);
        exit(0);
}

/* Transfer rectangular pixel image B to square pixel image A */
rectangular_to_square (b, &a, &err);
if (err) {
        an_error(err);
        exit(0);
}

free_image (b, &err);
if (err){
        an_error(err);
        exit(0);
}

disp_image(a,0,&err);

/* Copy red band of object image A into image B */
b = 0;
// copy_image (a, &b, 1, &err);
copy_image (a, &b, 1, &err);
if (err){
        an_error(err);
        exit(0);
}

free_image (a, &err);
if (err){
        an_error(err);
        exit(0);
}

/* Threshold the red band image B to get a binary image B */
thresh_is (b, &t,&err);
if (err){
        an_error(err);
        exit(0);
}
// disp_image(b,0,&err);

threshold (b, t, &err);
if (err){
        an_error(err);
        exit(0);
}
```

```
 // disp_image(b,0,&err);
 /* Allocate object pointer which contains the coordinates of each region */
  obj_ptr = (int **)malloc((size_t)sizeof(int *)*MAX_OBJECT_NUM);
  if (!obj_ptr){
          an_error(OUT_OF_STORAGE);
          exit(0);
  }

  for (i = 0; i < MAX_OBJECT_NUM; i ++) {
          ptr = (int *) malloc((size_t)sizeof(int)*4);
          if (!ptr){
            an_error(OUT_OF_STORAGE);
            exit(0);
          }
          else obj_ptr[i] = ptr;
  }

/* Mark each seperated regions, ignore very small regions, and fill holes
          in any regions to get a labelled image B.*/
 err = 0; n = 0; ibegin = 0; m = 0;
  /* n: no.of marked regions,m:no. of pixels in a region,
          ibegin: the first row of the last marked region    */
 while (err == 0) {
          region_4 (b, n+1, &ibegin, &m, &err);
          if (err == NO_REGION) break;
          /* Ignore very small regions */
          if (m < 30) {
           del_reg (b, n+1, &err);
           if (err){
                  an_error(err);
                  exit(0);
           }
           continue;
          }

/* Fill holes in the region marked n+1, and return the coordinates
          of the region in obj_ptr[n] array.                  */
          fill_holes (b, n+1, obj_ptr[n], &err);
          if (err){
           an_error(err);
           exit(0);
          }
          n++;
 }
// disp_image(b,0,&err);
 printf ("\nNo. of objects is %d.\n", n);
 _getch();
 free_image (b, &err);
 if (err) an_error(err);
 exit(0);
}
```

```
/*************************************************************************
 *                          disp_um.c                                    *
 *                                                                       *
 *          Program to display the square, thresholded, and labelled images of  *
 *          a um-formatted color image stored in a file                  *
 *                                                                       *
 *          X. Luo, June. 1996                                           *
 *                                                                       *
 *************************************************************************/

#include "base.c"

void main(int argc, char *argv[])
{
 struct image *a, *b;
 char  fp1[256]; /*file name of input image */

 int i, ibegin, err, t, m, n;
 int *ptr;
 int **obj_ptr;

 err = 0;

 if (argc != 2){
        an_error(GET_USAGE);
        exit(0);
 }

 strcpy (fp1, argv[1]);

 read_um (&b, fp1, &err);
// read_image_in_viff (&b, fp1, &err);
 if (err) {
        an_error(err);
        exit(0);
 }

/* Transfer rectangular pixel image B to square pixel image A */
 rectangular_to_square (b, &a, &err);
 if (err) {
        an_error(err);
        exit(0);
 }

 free_image (b, &err);
 if (err){
        an_error(err);
        exit(0);
 }

 disp_image(a,0,&err);
```

```
/* Copy red band of object image A into image B */
 b = 0;
// copy_image (a, &b, 1, &err);
 copy_image (a, &b, 1, &err);
 if (err){
        an_error(err);
        exit(0);
 }
 free_image (a, &err);
 if (err){
        an_error(err);
        exit(0);
 }

/* Threshold the red band image B to get a binary image B */
 thresh_is (b, &t,&err);
 if (err){
        an_error(err);
        exit(0);
 }
// disp_image(b,0,&err);

 threshold (b, t, &err);
 if (err){
        an_error(err);
        exit(0);
 }
// disp_image(b,0,&err);

/* Allocate object pointer which contains the coordinates of each region */
 obj_ptr = (int **)malloc((size_t)sizeof(int *)*MAX_OBJECT_NUM);
 if (!obj_ptr){
        an_error(OUT_OF_STORAGE);
        exit(0);
 }

 for (i = 0; i < MAX_OBJECT_NUM; i ++) {
        ptr = (int *) malloc((size_t)sizeof(int)*4);
        if (!ptr){
         an_error(OUT_OF_STORAGE);
         exit(0);
        }
        else obj_ptr[i] = ptr;
 }

/* Mark each seperated regions, ignore very small regions, and fill holes
        in any regions to get a labelled image B.*/
 err = 0; n = 0; ibegin = 0; m = 0;
 /* n: no.of marked regions,m:no. of pixels in a region,
        ibegin: the first row of the last marked region    */
 while (err == 0) {
        region_4 (b, n+1, &ibegin, &m, &err);
```

```
        if (err == NO_REGION) break;
        /* Ignore very small regions */
        if (m < 30) {
         del_reg (b, n+1, &err);
         if (err){
                an_error(err);
                exit(0);
         }
         continue;
        }

/*  Fill holes in the region marked n+1, and return the coordinates
        of the region in obj_ptr[n] array.                    */
        fill_holes (b, n+1, obj_ptr[n], &err);
        if (err){
         an_error(err);
         exit(0);
        }
        n++;
 }
 disp_image(b,0,&err);
 printf ("\nNo. of objects is %d.\n", n);
 _getch();
 free_image (b, &err);
 if (err) an_error(err);
 exit(0);
}


/*************************************************************************
 *                              tstasp.c                                 *
 *                                                                       *
 *          Program to calculate aspect ratio from a Canadian quarter image *
 *          stored in a file                                             *
 *                                                                       *
 *          X. Luo, June. 1996                                           *
 *                                                                       *
 ************************************************************************/
#include "base.c"

void main(int argc, char *argv[])
{
 FILE *outf;
 struct image *a, *b;
 char fp1[256], fp2[256];
 int i, ibegin, err, t, m, n;
 int *ptr;
 int **obj_ptr;

 err = 0;
 if (argc != 3){
  an_error(GET_USAGE);
```

```
    exit(0);
   }
   strcpy (fp1, argv[1]);
   strcpy (fp2, argv[2]);

 /* Read in coin image to A */
  read_image_in_viff (&a, fp1, &err);
  if (err) {
           an_error(err);
           exit(0);
  }
 // disp_image(a,0,&err);

 /* Copy red band of coin image A into image B */
  b = 0;
  copy_image (a, &b, 1, &err);
  if (err){
   an_error(err);
   exit(0);
  }

  free_image (a, &err);
  if (err){
   an_error(err);
   exit(0);
  }
/*  Threshold the red band image B to get a binary image B */
  thresh_is (b, &t,&err);
  if (err){
   an_error(err);
   exit(0);
  }
// disp_image(b,0,&err);
  threshold (b, t, &err);
  if (err){
   an_error(err);
   exit(0);
  }
// disp_image(b,0,&err);
/*  Allocate object pointer which contains the coordinates of each region */
 obj_ptr = (int **)malloc((size_t)sizeof(int *)*MAX_OBJECT_NUM);
 if (!obj_ptr){
  an_error(OUT_OF_STORAGE);
  exit(0);
 }
 for (i = 0; i < MAX_OBJECT_NUM; i ++) {
          ptr = (int *) malloc((size_t)sizeof(int)*4);
          if (!ptr){
           an_error(OUT_OF_STORAGE);
           exit(0);
          }
          else obj_ptr[i] = ptr;
```

```
  }
 /* Mark each seperated regions, ignore very small regions, and fill holes
          in any regions to get a labelled image B.*/
  err = 0; n = 0; ibegin = 0; m = 0;
  /* n: no.of marked regions,m:no. of pixels in a region,
          ibegin: the first row of the last marked region    */
  while (err == 0) {
          region_4 (b, n+1, &ibegin, &m, &err);
          if (err == NO_REGION) break;
          /* Ignore very small regions */
          if (m < 30) {
           del_reg (b, n+1, &err);
    if (err){
     an_error(err);
     exit(0);
    }
           continue;
          }
/* Fill holes in the region marked n+1, and return the coordinates
          of the region in obj_ptr[n] array.                  */
          fill_holes (b, n+1, obj_ptr[n], &err);
   if (err){
    an_error(err);
    exit(0);
   }
          n++;
  }
// disp_image(b,0,&err);
 if(n>1){
  an_error(NO_OR_TOO_MANY_REGIONS);
  exit(0);
 }
/* Open the output feature file */
 outf = fopen(fp2, "ab");
 if (outf == NULL){
  an_error(CANNOT_OPEN_FILE);
  exit(0);
 }
/* write the area and the vertical and horizonal ranges in pixel
  to the output file */
 fprintf(outf, "Coin Area Nr Nc\n");
 fprintf(outf, "%s %d %d %d\n", fp1, area(b,1),
          (obj_ptr[0][2]-obj_ptr[0][0]),(obj_ptr[0][3]-obj_ptr[0][1]));
 fclose(outf);
 free_image (b, &err);
 if (err){
  an_error(err);
  exit(0);
 }
 exit(0);
}
```

```
/**********************************************************************
 *                              uniform.c                             *
 *                                                                    *
 *          Program to check illumination uniformity over FOV         *
 *                                                                    *
 *          X. Luo, June. 1996                                        *
 *                                                                    *
 **********************************************************************/

#include "base.c"

void main(int argc, char *argv[])
{
 FILE *outf;
 struct image *a;
 char fp1[256], fp2[256];
 double r, g, b;
 int i, j, err;

 err = 0;

 if (argc != 3){
        an_error(GET_USAGE);
        exit(0);
 }

 strcpy (fp1, argv[1]);
 strcpy (fp2, argv[2]);

 read_image_in_viff (&a, fp1, &err);
 if (err) {
        an_error(err);
        exit(0);
 }
// disp_image(a,0,&err);

 outf = fopen(fp2, "ab");
 if (outf == NULL){
        an_error(CANNOT_OPEN_FILE);
        exit(0);
 }

 for (i=0; i<a->nr; i++){
  r = 0.0;
  g = 0.0;
  b = 0.0;
  for (j=0; j<(a->nc-4); j++){
    r += a->band1[i][j];
    g += a->band2[i][j];
    b += a->band3[i][j];
  }
  r = r / (double)(a->nc-4);
```

```
    g = g / (double)(a->nc-4);
    b = b / (double)(a->nc-4);
    fprintf(outf,"row %d %f %f %f\n", i, r, g, b);
   }

  fprintf(outf, "\n");
  for (j=0; j<(a->nc-4); j++){
   r = 0.0;
   g = 0.0;
   b = 0.0;
   for (i=0; i<a->nr; i++){
     r += a->band1[i][j];
     g += a->band2[i][j];
     b += a->band3[i][j];
   }
   r = r / (double)a->nr;
   g = g / (double)a->nr;
   b = b / (double)a->nr;
   fprintf(outf, "col %d %f %f %f\n", j, r, g, b);
  }
  fprintf(outf, "\n\n");

  fclose(outf);

  free_image (a, &err);
  if (err) an_error(err);

  exit(0);
}

/*************************************************************************
 *                          head.h                                       *
 *                                                                       *
 *          Header file defining various constants and structures and    *
 *          including DOS header files used in the software              *
 *                                                                       *
 *          X. Luo, June. 1996                                           *
 *                                                                       *
 *************************************************************************/

#include <stdio.h>
#include <math.h>
#include <malloc.h>
//#include <stdlib.h>
#include <graph.h>
#include <conio.h>
#include <process.h>
//#include <fcntl.h>
//#include <io.h>
//#include <dos.h>
//#include <bios.h>
#include <string.h>
```

```
#define R 0x000000001L
#define G 0x000000100L
#define B 0x000010000L

#define BACKGROUND 255
#define OBJECT 0
#define COIN_DIAMETER_IM_MM 23.869
#define WHITE 250.0
#define MAX_OBJECT_NUM 40
#define SQRT2 1.414213562
#define PI 3.1415926535
#define PIX_ASP_RATIO 1.275539

/* The UM (raw) image data structure */
struct image {
        int nc, nr, color;
        unsigned char **band1, **band2, **band3; /* Pixel values */
};

/* The viff image data structure */
struct viff_image {
        char hdr1[510];
        unsigned long hdr2[5], hdr4[119];
        float hdr3[2];
        unsigned char **band1, **band2, **band3; /* Pixel values */
};

/* The bulk image feature structure) */
struct bfeature {
        double meanR;
        double meanG;
        double meanB;
        double meanH;
        double meanS;
        double meanI;
        double varR;
        double varG;
        double varB;
        double varH;
        double varS;
        double varI;
        double rangeR;
        double rangeG;
        double rangeB;
        double histR[32];
        double histG[32];
        double histB[32];
};

/* The feature data structure                    */
struct feature {
```

```
        double area;              /* Object area                          */
        double perimeter;  /* Object perimeter                    */
        double length;     /* Length of the smallest enclosing rectangular box */
        double width;      /* Width of the smallest enclosing rectangular box  */
        double lpa;        /* Length of the principal axis        */
        double wma;        /* Width of the min. axis              */
        double rmin;       /* Min. radius                         */
        double rmax;       /* Max. radius                         */
        double rmean;      /* Mean radius                         */
        double var_r;      /* variance of radius                  */

        double radR[32];   /* Ratio of radius at i*(PI/12) from PA to rmax    */
        double perimR[32]; /* Ratio of perimeter segments within each PI/12
                                                   angle to perimeter            */
        double areaR[32]; _ /* Ratio of subarea within each PI/12 angle to area */

        double asp_R;      /* Aspect ratio = lpa/wma              */
        double rec_R;      /* Rectangular aspect ratio = length/width      */
        double rad_R;      /* Radius ratio = rmax/rmin            */
        double thin_R;     /* Thinness ratio = perimeter*perimeter/area    */
        double area_R;     /* Area ratio = length*width/area      */
        double har_R;      /* Haralick ratio = rmean/var_r        */

        double meanR;      /* mean red component value            */
        double meanG;      /* mean green component value          */
        double meanB;      /* mean blue component value           */
        double meanH;      /* mean hue value                      */
        double meanS;      /* mean sat. value                     */
        double meanI;      /* mean inten. value                   */
        double varR;       /* var. of red component value         */
        double varG;       /* var. of green component value       */
        double varB;       /* var. of blue component value        */
        double varH;       /* var. of hue value                   */
        double varS;       /* var. of sat. value                  */
        double varI;       /* var. of inten. value                */
        double rangeR;
        double rangeG;
        double rangeB;
        double histR[32];  /* histgram of red component           */
        double histG[32];  /* histgram of green component         */
        double histB[32];  /* histgram of blue component          */
};

/*    Error Codes:  */
#define BAD_IMAGE_SIZE              100
#define OUT_OF_STORAGE          101
#define CANNOT_OPEN_FILE        102
#define BAD_DESCRIPTOR1         103
#define BAD_NR_NC               104
#define FILE_TOO_SHORT              105
#define BAD_DESCRIPTOR2         106
#define NO_REGION               107
```

```
#define REGION_INT_BOUND            108
#define INTERNAL_1                  109
#define BAD_IMAGE_COORD             110
#define NO_RESULT                   111
#define IMPOSSIBLE_CLASS            112
#define TOO_MANY_CLASSES            113
#define TOO_MANY_EDGES              114
#define BAD_COLOR_MAP               115
#define IO_ERROR                            116
#define BAD_ARGUMENT1               117
#define BAD_ARGUMENT2               118
#define BAD_ARGUMENT3               119
#define NO_OR_TOO_MANY_REGIONS 120
#define BAD_FEATURE_SIZE            121
#define CANNOT_GET_CALIBR_SCALE             122
#define GET_USAGE                   123

/* Viff header definitions */

#define XV_FILE_MAGIC_NUM  0xab /* Khoros file identifier          */
#define XV_FILE_TYPE_XVIFF 1   /* indicates an image file          */
#define XV_IMAGE_VER_NUM   3   /* Version 3 (3.1)                  */
#define XV_IMAGE_REL_NUM   1   /* Release 1                        */
#define VFF_DEP_NSORDER    0x8 /* NS32000 byte ordering            */
#define VFF_LOC_IMPLICIT   1   /* The location of image pixels     */
#define VFF_TYP_1_BYTE     1   /* pixels are byte (unsigned char)  */
#define VFF_DES_RAW        0   /* Raw - no compression             */
#define VFF_MS_NONE        0 /* No mapping is to be done, & maps are to be stored */. #define
VFF_MAP_OPTIONAL   1   /* The data is valid without being sent
                                thru the color map. If a map is defined,
                                the data may optionally be sent thru it.*/
#define VFF_CM_NONE 0
#define VFF_CM_genericRGB  15  /* an RGB image but not conforming to any
                                                                  standard */
```

```
/*********************************************************************
 *                              base.c                                  *
 *                                                                      *
 *          Included file including functions and routines files used in the software    *
 *                                                                      *
 *          X. Luo, June. 1996                                          *
 *                                                                      *
 *********************************************************************/

#include "head.h"

double angle_2pt(int r1,int c1,int r2,int c2);
double all_dist(struct image *x,double i1,double j1,double i2,double j2,int val);
double dist_2pt(double r1, double c1, double r2, double c2);
double get_scale(struct image *x);
double line_interval(struct image *y,double a, double b, double c);
double max(double a, double b);
double min(double a, double b);
double perimeter(struct image *x,int val,int *sum_pixel,int *error_code);

int  area(struct image *x,int val);
int  is_zero(double x);
int  line2pt(double x1,double y1,double x2,double y2,double *a,double *b,double *c);
int  line_intersect(double a1,double b1,double c1,double a2,double b2,double c2,
                                        double *x, double *y);
int  locate_region(int x, int y, double *a, double *b, double *c, int orient);
int  max2(int i,int j);
int  nay4(struct image *x, int i, int j, int val);
int  nay8(struct image *x, int i, int j, int val);
int  orientation(struct image *x, double *a, double *b, double *c);
int  range(struct image *x,int n,int m);

struct image *new_image(int nr,int nc,int color,int *error_code);

void minmax_dist (struct image *x, int val, double a, double b, double c,
                  int *ii1, int *jj1, int *ii2, int *jj2);
void an_error(int ecode);
void box(struct image *x,int val,int *rxy,int *error_code);
void center_of_mass (struct image *x, int val, double *ii,
                                        double *jj, int *error_code);
void clr_line(struct image *x, double a, double b, double c, int *error_code);
void color_features(struct image *bin_obj,struct image *cl_obj,int val,
                                        struct feature *objf, int n, int *error_code);
void copy_image(struct image *x,struct image **y,int band,int *error_code);
void copy_reg(struct image *x,struct image **y,int val,
                        int *rxy,int *error_code);
void del_reg(struct image *x,int value,int *error_code);
void disp_image(struct image *x,int band,int *error_code);
void draw_line (struct image *im, int x1, int y1, int x2, int y2);
void edge_sobel(struct image *x,int *error_code);
void extract_obj (struct image *marked, struct image *original,
                                struct image **bin_obj,struct image **cl_obj,
```

```
                                    int val,int *rxy,int *error_code);
void fft(double *f, int ln);
void fill_holes(struct image *x,int v,int *rxy,int *error_code);
void frame(struct image *x);
void free_image(struct image  *z,int *error_code);
void histogram(struct image *x,long *hist,int n,int *error_code);
void lines_radius(double *a, double *b, double *c, double x, double y);
void ln_obj_intersec (struct image *y, double a, double b, double c,
                                         int *i1, int *j1, int *i2, int *j2);
void lines_parallel(struct image *z, int value,
                                         double *a, double *b, double *c, int orient);
void mark4(struct image *x,int value,int iseed,int jseed,int *reg_size);
void mark8(struct image *x,int value,int iseed,int jseed,int *reg_size);
void perp (double a, double b, double c, double *a1, double *b1,
                    double *c1, double x, double y);
void principal_axis(struct image *x,int val,double *i1,double *j1,double *i2,
                                         double *j2, double cmi, double cmj, int *error_code);
void radius(struct image *y,double *a,double *b,double *c, int k,
                         double cmi,double cmj,double *r1,double *r2);
void read_um(struct image **x,char *fn,int *error_code);
void read_image_in_viff(struct image  **x,char *fn,int *error_code);
void read_img (struct image  **x, char *fn, int *error_code);
void rectangular_to_square (struct image *x,struct image **y,int *error_code);
void region_4(struct image *x,int value,int *istart,int *r_size,int *error_code);
void region_8(struct image *x,int value,int *istart,int *r_size,int *error_code);
void size_shape_features(struct image *bin_obj,int val,struct feature *objf,
                                               double mm_per_pix,int *error_code);
void thresh_is(struct image *x,int *t,int *error_code);
void threshold(struct image *x,int t,int *error_code);
void write_image(struct image *x, char *fn, int *error_code);
void write_feature(FILE *outfp, struct feature *objf, char *img, int i);
void write_fname(FILE *outfp);
void bulk_feature(struct image *x, struct bfeature *bf, int n, int *error_code);
void write_bf(FILE *outfp, struct bfeature *bf, char *img);


struct image  *new_image (int nr, int nc, int color, int *error_code)
{
 struct image  *x;          /* New image */
 unsigned char *ptr;        /* new pixel array */
 int i;

 *error_code = 0;
 if (nr < 0 || nc < 0) {
        *error_code = BAD_IMAGE_SIZE;
        return 0;
 }

/*    Allocate the image structure    */

 x=(struct image *)malloc((size_t)sizeof(struct image));
```

```
/*  fill appropriate values into headers to create a um file */

  x->nc = nc;
  x->nr = nr;
  x->color = color;

/*     Allocate the pixel array        */

  switch (color){
          case 0:
           x->band1 = (unsigned char **)malloc((size_t)sizeof(unsigned char *)*(int)nr);
           if (!(x->band1)) {
                   *error_code = OUT_OF_STORAGE;
                   return 0;
           }
           for (i=0; i<nr; i++) {
                   ptr = (unsigned char *) malloc((size_t)sizeof(unsigned char)*(int)nc);
/* Allocate one row  */
                   if (!ptr) {
                     *error_code = OUT_OF_STORAGE;
                     return 0;
                   } else x->band1[i] = ptr;
           }
           x->band2 = 0;
           x->band3 = 0;
           break;

          case 1:
           x->band1 = (unsigned char **)malloc((size_t)sizeof(unsigned char *)*(int)nr);
/* Pointers to rows */
           if (!(x->band1)) {
                   *error_code = OUT_OF_STORAGE;
                   return 0;
           }
           for (i=0; i<nr; i++) {
                   ptr = (unsigned char *) malloc((size_t)sizeof(unsigned char)*(int)nc);
/* Allocate one row  */
                   if (!ptr) {
                     *error_code = OUT_OF_STORAGE;
                     return 0;
                   } else x->band1[i] = ptr;
           }
           x->band2 = (unsigned char **)malloc((size_t)sizeof(unsigned char *)*(int)nr);
/* Pointers to rows */
           if (!(x->band2)) {
                   *error_code = OUT_OF_STORAGE;
                   return 0;
           }
           for (i=0; i<nr; i++) {
                   ptr = (unsigned char *) malloc((size_t)sizeof(unsigned char)*(int)nc);
/* Allocate one row  */
                   if (!ptr) {
```

```
                    *error_code = OUT_OF_STORAGE;
                    return 0;
                  } else x->band2[i] = ptr;
           }
           x->band3 = (unsigned char **)malloc((size_t)sizeof(unsigned char *)*(int)nr);
 /* Pointers to rows */
           if (!(x->band3)) {
           *error_code = OUT_OF_STORAGE;
           return 0;
           }
           for (i=0; i<nr; i++) {
                  ptr = (unsigned char *) malloc((size_t)sizeof(unsigned char)*(int)nc);
 /* Allocate one row  */
                  if (!ptr) {
                   *error_code = OUT_OF_STORAGE;
                   return 0;
                  } else x->band3[i] = ptr;
           }
           break;
  }
  return x;
}

/* Free an image Z */
void free_image (struct image  *z, int *error_code)
{
/*    Free the storage associated with the image Z    */
  int i;

  *error_code = 0;
  if (z != 0){
         for (i=0; i< z->nr; i++){
          if (z->color == 0) free (z->band1[i]);
          else{
                 free (z->band1[i]);
                 free (z->band2[i]);
                 free (z->band3[i]);
          }
         }
         free (z->band1);
         free (z->band2);
         free (z->band3);
         free (z);
  }
}

/* Retrieve a viff-format image file from disk into an image structure */
void read_image_in_viff (struct image  **x, char *fn, int *error_code)
{
/* Allocate an um image structure and read an viff image into it */

 FILE * inf;
```

```
  int nr,nc,color,i,j, k;
  unsigned char hdr1[520], hdr2[20];
  unsigned long hdr4[119];
  float hdr3[2];
  unsigned char *buf;

  *x = 0;
  *error_code = 0;

 /* Open the viff file  */
  inf = fopen(fn, "rb");
  if (inf == 0) {
          *error_code = CANNOT_OPEN_FILE;
          return;
  }

 /* Look for XV_FILE_MAGIC_NUM and XV_FILE_TYPE_XVIFF as the first two characters */
  if (fread(hdr1, sizeof (unsigned char), 520, inf) != 520) {
          *error_code = BAD_DESCRIPTOR1;
          fclose (inf);
          return;
  }

 /* Read the image size.  */
  if (fread(hdr2, sizeof (unsigned char), 20, inf) != 20) {
          *error_code = BAD_DESCRIPTOR1;
          fclose(inf);
          return;
  }

 // for(i=0; i<20; i++) printf("HDR2[%i] = %u\n", i, (int)hdr2[i]);
 // _getch();

  nc = 256*((int)hdr2[1] + (int)hdr2[2]) + (int)hdr2[0] + (int)hdr2[3];
  nr = 256*((int)hdr2[5] + (int)hdr2[6]) + (int)hdr2[4] + (int)hdr2[7];
  color = 1;

 // printf("NC: %i, NR: %i\n", nc, nr);
 // _getch();

  if (nr<=0 || nr>9999 || nc<=0 || nc>9999) {
          *error_code = BAD_NR_NC;
          fclose (inf);
          return;
  }

  if (fread(hdr3, sizeof (float), 2, inf) != 2) {
          *error_code = BAD_DESCRIPTOR1;
          fclose(inf);
          return;
  }
```

```
 if (fread(hdr4, sizeof (unsigned long), 119, inf) != 119) {
          *error_code = BAD_DESCRIPTOR1;
          fclose(inf);
          return;
 }

/*    Allocate an urn image and read the data.      */

 *x = new_image (nr, nc, color, error_code);
 if (*error_code) {
          fclose (inf);
          return;
 }

 buf = (unsigned char *)malloc((size_t)sizeof(unsigned char)*nc);

 /* Read in band1 data */
 for (i=0; i<nr; i++) {
          k = fread (buf, 1, nc, inf);
          if (k != nc) {
           *error_code = FILE_TOO_SHORT;
           printf ("Too short at row %d nbytes=%d\n", i,k);
           perror(" message: ");
           scanf ("%d", &j);
           fclose (inf);
           return;
          } else
           for (j=0; j<nc; j++) (*x)->band1[i][j] = buf[j];
 }

 /* Read in band2 data */
 for (i=0; i<nr; i++) {
          k = fread (buf, 1, nc, inf);
          if (k != nc) {
           *error_code = FILE_TOO_SHORT;
           printf ("Too short at row %d nbytes=%d\n", i,k);
           perror(" message: ");
           scanf ("%d", &j);
           fclose (inf);
           return;
          } else
           for (j=0; j<nc; j++) (*x)->band2[i][j] = buf[j];
 }
 /* Read in band3 data */
 for (i=0; i<nr; i++) {
          k = fread (buf, 1, nc, inf);
          if (k != nc) {
           *error_code = FILE_TOO_SHORT;
           printf ("Too short at row %d nbytes=%d\n", i,k);
           perror(" message: ");
           scanf ("%d", &j);
           fclose (inf);
```

```
            return;
          } else
            for (j=0; j<nc; j++) (*x)->band3[i][j] = buf[j];
  }
  free (buf);
  fclose (inf);
}

/* Retrieve a raw-data (UM format) image file from disk into an image structure */
void read_um (struct image  **x, char *fn, int *error_code)
{
/* Allocate an um image structure and read an um image into it */

 FILE * inf;
 int nr,nc,color,i,j, k;
 int num[3];
 unsigned char *buf;

 *x = 0;
 *error_code = 0;

 /* Open the file   */
 inf = fopen(fn, "rb");
 if (inf == 0) {
        *error_code = CANNOT_OPEN_FILE;
        return;
 }

/*  Read the image size and image type (grey or color) indicator  */

 if (fread(num, sizeof (int), 3, inf) != 3) {
        *error_code = BAD_DESCRIPTOR1;
        fclose(inf);
        return;
 }
 nr = num[0];
 nc = num[1];
 color = num[2];

 printf("NR: %d, NC: %d, CL: %d", nr, nc, color);

 if (nr<0 || nr>9999 || nc<0 || nc>9999) {
        *error_code = BAD_NR_NC;
        fclose (inf);
        return;
 }

 /* Allocate image and read the data  */
 *x = new_image (nr, nc, color, error_code);
 if (*error_code) {
        fclose (inf);
        return;
```

```
  }

  buf = (unsigned char *)malloc((size_t)sizeof(unsigned char)*nc);
  /* Read in band1 data */
  for (i=0; i<nr; i++) {
          k = fread (buf, 1, nc, inf);
          if (k != nc) {
           *error_code = FILE_TOO_SHORT;
           printf ("Too short at row %d nbytes=%d\n", i,k);
           perror(" message: ");
           scanf ("%d", &j);
           fclose (inf);
           return;
          } else
           for (j=0; j<nc; j++) (*x)->band1[i][j] = buf[j];
  }
  if (color != 0){
          /* Read in band2 data */
          for (i=0; i<nr; i++) {
           k = fread (buf, 1, nc, inf);
           if (k != nc) {
                   *error_code = FILE_TOO_SHORT;
                   printf ("Too short at row %d nbytes=%d\n", i,k);
                   perror(" message: ");
                   scanf ("%d", &j);
                   fclose (inf);
                   return;
           } else
                   for (j=0; j<nc; j++) (*x)->band2[i][j] = buf[j];
          }
          /* Read in band3 data */
          for (i=0; i<nr; i++) {
           k = fread (buf, 1, nc, inf);
           if (k != nc) {
                   *error_code = FILE_TOO_SHORT;
                   printf ("Too short at row %d nbytes=%d\n", i,k);
                   perror(" message: ");
                   scanf ("%d", &j);
                   fclose (inf);
                   return;
           } else
                   for (j=0; j<nc; j++) (*x)->band3[i][j] = buf[j];
          }
  }
  free (buf);
  fclose (inf);
}

void read_img (struct image  **x, char *fn, int *error_code)
{
/* Allocate an um image structure and read an um image into it */
```

```
 FILE * inf;
 int nr,nc,color,i,j, k;
 unsigned char *buf;

 *x = 0;
 *error_code = 0;

 /* Open the file  */
 inf = fopen(fn, "rb");
 if (inf == 0) {
         *error_code = CANNOT_OPEN_FILE;
         return;
 }

/*  Read the image size and image type (grey or color) indicator  */
 nc = 512;
 nr = 768;
 color = 0;

 /* Allocate image and read the data  */
 *x = new_image (nr, nc, 0, error_code);
 if (*error_code) {
         fclose (inf);
         return;
 }

 buf = (unsigned char *)malloc((size_t)sizeof(unsigned char)*nc);

 /* Read in band1 data */
 for (i=0; i<nr; i++) {
         k = fread (buf, 1, nc, inf);
         if (k != nc) {
          *error_code = FILE_TOO_SHORT;
          printf ("Too short at row %d nbytes=%d\n", i,k);
          perror(" message: ");
          scanf ("%d", &j);
          fclose (inf);
          return;
         } else
          for (j=0; j<nc; j++) (*x)->band1[i][j] = buf[j];
 }

 if (color != 0){
         /* Read in band2 data */
         for (i=0; i<nr; i++) {
          k = fread (buf, 1, nc, inf);
          if (k != nc) {
                  *error_code = FILE_TOO_SHORT;
                  printf ("Too short at row %d nbytes=%d\n", i,k);
                  perror(" message: ");
                  scanf ("%d", &j);
                  fclose (inf);
```

```
                return;
            } else
                for (j=0; j<nc; j++) (*x)->band2[i][j] = buf[j];
          }
          /* Read in band3 data */
          for (i=0; i<nr; i++) {
            k = fread (buf, 1, nc, inf);
            if (k != nc) {
                *error_code = FILE_TOO_SHORT;
                printf ("Too short at row %d nbytes=%d\n", i,k);
                perror(" message: ");
                scanf ("%d", &j);
                fclose (inf);
                return;
            } else
                for (j=0; j<nc; j++) (*x)->band3[i][j] = buf[j];
          }
  }
  free (buf);
  fclose (inf);
}

/*    Write the given um image X to a file named FN      */
void write_image (struct image  *x, char *fn, int *error_code)
{
  FILE *inf;
  int i, k;
  int num[3];
/*    Open the file   */
  *error_code = 0;
  inf = fopen (fn, "wb");
  if (inf == NULL) {
          *error_code = CANNOT_OPEN_FILE;
          return;
  }
/*    Write um image headers  */
  num[0] = x->nr;
  num[1] = x->nc;
  num[2] = x->color;
  if (fwrite (num, sizeof (int), 3, inf) != 3){
          *error_code = FILE_TOO_SHORT;
          return;
  }
/*    Write the image as rows.        */
/* write band1 data */
  for (i=0; i< x->nr; i++) {
          k = fwrite (x->band1[i], 1, x->nc, inf);
          if (k != x->nc) {
            *error_code = FILE_TOO_SHORT;
            return;
          }
  }
```

```
  if (x->color != 0){
/* write band2 data */
          for (i=0; i< x->nr; i++) {
            k = fwrite (x->band2[i], 1, x->nc, inf);
            if (k != x->nc) {
                    *error_code = FILE_TOO_SHORT;
                    return;
            }
          }
/* write band3 data */
          for (i=0; i< x->nr; i++) {
            k = fwrite (x->band3[i], 1, x->nc, inf);
            if (k != x->nc) {
                    *error_code = FILE_TOO_SHORT;
                    return;
            }
          }
  }
  fclose (inf);
}

/* Make a copy of the image X into the image (*Y) if "band = 0) or
   extract one band from image X into the image (*Y). Allocate Y if
   necessary; otherwise copy into the existing storage.          */

void copy_image (struct image *x,struct image **y,int band,int *error_code)
{
 int i,j, new=0;

 *error_code = 0;

 /* check if the specified band is legal */
 if (band != 0 && band != 1 && band != 2 && band != 3){
         *error_code = BAD_ARGUMENT1;
         return;
 }

 if (band == 0){
         /* check if *y exits, if so check the size and image type*/
         if (*y == 0) new =1;
         else if ((*y)->nc != x->nc ||(*y)->nr != x->nr ||(*y)->color != x->color){
           free_image (*y, error_code);
           new = 1;
         } else new =0;

         if (new) *y = new_image (x->nr, x->nc, x->color, error_code);
         if (*error_code) return;

         if (x->color != 0){
           for (i=0; i< x->nr; i++){
                   for (j=0; j< x->nc; j++){
                     (*y)->band1[i][j] = x->band1[i][j];
```

```
                (*y)->band2[i][j] = x->band2[i][j];
                (*y)->band3[i][j] = x->band3[i][j];
               }
         }
        }else {
         for (i=0; i< x->nr; i++)
               for (j=0; j< x->nc; j++)
                (*y)->band1[i][j] = x->band1[i][j];
        }
 }else {
        if (x->color == 0){
         *error_code = BAD_ARGUMENT2;
         return;
        }else {
         /* check if *y exits, if so check the size and image type*/
         if (*y == 0) new =1;
         else if ((*y)->nc != x->nc ||(*y)->nr != x->nr || (*y)->color != 0){
               free_image (*y, error_code);
               new = 1;
         } else new =0;

         if (new) *y = new_image (x->nr, x->nc, 0, error_code);
         if (*error_code) return;

         switch (band){
               case 1:
                for (i=0; i< x->nr; i++)
                      for (j=0; j< x->nc; j++)
                       (*y)->band1[i][j] = x->band1[i][j];
                break;
               case 2:
                for (i=0; i< x->nr; i++)
                      for (j=0; j< x->nc; j++)
                       (*y)->band1[i][j] = x->band2[i][j];
                break;
               case 3:
                for (i=0; i< x->nr; i++)
                      for (j=0; j< x->nc; j++)
                       (*y)->band1[i][j] = x->band3[i][j];
                break;
         }
        }
 }
}

/* Display an image X on screen*/
void disp_image (struct image *x, int band, int *error_code)
{
 struct videoconfig vc;
 int i;
 int col, row;
 long int color[256];
```

```
 *error_code = 0;
 /* check if the specified band is legal */
 if (band != 0 && band != 1 && band != 2 && band != 3){
         *error_code = BAD_ARGUMENT1;
         return;
 }

 _setvideomode(_VRES256COLOR);
 _getvideoconfig(&vc);

/* maxx = vc.numxpixels - 1;
  maxy = vc.numypixels - 1;*/

/* remap colors to 256 level grey scale  */
 for ( i=0; i<256; i++ )
 color[i] = i*(R + G + B);
 _remapallpalette(color);

 switch (band){
         case 0:
/*     if (!(x->color)){
                 *error_code = BAD_ARGUMENT2;
                 return;
         }*/
         for ( row=0; row < x->nr; row++ ){
                 for ( col=0; col< x->nc; col++ ){
                  _setcolor(x->band1[row][col]/4);/*  256 grey levels available  */
                  _setpixel((int)col,(int)row);
                 }
         }
         break;
        case 1:
         if (!(x->color)){
                 *error_code = BAD_ARGUMENT2;
                 return;
         }
         for ( row=0; row < x->nr; row++ ){
                 for ( col=0; col< x->nc; col++ ){
                  _setcolor(x->band1[row][col]/4);/*  256 grey levels available  */
                  _setpixel((int)col,(int)row);
                 }
         }
         break;
        case 2:
         if (!(x->color)){
                 *error_code = BAD_ARGUMENT2;
                 return;
         }
         for ( row=0; row < x->nr; row++ ){
                 for ( col=0; col< x->nc; col++ ){
                  _setcolor(x->band2[row][col]/4);/*  256 grey levels available  */
                  _setpixel((int)col,(int)row);
```

```
                }
        }
        break;
      case 3:
        if (!(x->color)){
                *error_code = BAD_ARGUMENT2;
                return;
        }
        for ( row=0; row < x->nr; row++ ){
                for ( col=0; col< x->nc; col++ ){
                  _setcolor(x->band3[row][col]/4);/*  256 grey levels available  */
                  _setpixel((int)col,(int)row);
                }
        }
        break;
  }

  _settextposition(30,1);
  _outtext("hit any key to exit");

  while(! _kbhit());
  _getch();

  _setvideomode(_DEFAULTMODE);

}

/* Get the n-band histogram from a grey-level image X */
void histogram (struct image *x, long *hist, int n, int *error_code)
{
        long i,j,k,xmin, xmax, t;
        double width, xmean, y;

        *error_code = 0;

        if (x->color){
          *error_code = BAD_ARGUMENT3;
          return;
        }

        xmin = 256L;    xmax = 0L;
        xmean = 0.0;    y = 0.0;
        for (i=0; i<x->nr; i++) {
          for (j=0; j<x->nc; j++) {
                t = (long)(x->band1[i][j]);
                if (t > xmax) xmax = t;
                if (t < xmin) xmin = t;
                y += (double)t;
          }
        }
        printf ("Minimum level is %ld       Maximum level is %ld\n", xmin,xmax);
        _getch();
```

```
        xmean = y/((double)(x->nc)*(double)(x->nr));

        width = 256.0/(double)n;
        for (i=0; i<256; i++) hist[i] = 0;
        for (i=0; i<x->nr; i++)
          for (j=0; j<x->nc; j++) {
                k = (long)(((double)(x->band1[i][j]))/width);
                hist[ k ] += 1;
        }
        xmax = ((long)(x->nr)*(long)(x->nc))/2;

        xmin = 0; i = 0;
        while (xmin < xmax)
          xmin += hist[i++];

        printf ("Mean level is %f   Median level is %d\n", xmean, i);
        _getch();
        printf ("histogram is:\n");
        for (i=0; i<256; i++)
                printf ("%ld %ld\n", i, hist[i]);
}

/* Threshold an image X. Any pixels with a level less than T
   will be set to 0; others will be set to BACKGROUND          */
void threshold (struct image *x, int t, int *error_code)
{
        int i,j;

        *error_code = 0;

        if (x->color){
         *error_code = BAD_ARGUMENT3;
         return;
        }

        for (i=0; i<x->nr; i++)
          for (j=0; j<x->nc; j++)
                if (x->band1[i][j] < t) x->band1[i][j] = (unsigned char)BACKGROUND;
                 else x->band1[i][j] = (unsigned char)OBJECT;
}

/* Automatically choose an optimal thresholding level for a grey-level image X */
void thresh_is (struct image *x, int *t, int *error_code)
{
        static long hist[256], i, j, n, m;
        long tt, tb, to, t1, t2;

/* Create a histogram ... */
        for (i=0; i<256; i++) hist[i] = 0;
        tt = 0;
        for (i=0; i<x->nr; i++)
         for (j=0; j<x->nc; j++) {
```

```
                m = x->band1[i][j];
                tt = tt + m;
                hist[m] += 1;
        }

/* The first threshold is the mean level - then iterate */
        n = (long)(x->nr)*(long)(x->nc);
        tt = tt/n;

        for (m=0; m<40; m++) {          /* MAX of 40 iterations */
         t1 = 0; t2 = 0;
         for (i=0; i<=tt; i++) {
                t1 = t1 + i*hist[i];
                t2 = t2 + hist[i];
         }
         to = t1/(2*t2);

         t1 = 0; t2 = 0;
         for (i=tt+1; i<256; i++) {
                t1 = t1 + i*hist[i];
                t2 = t2 + hist[i];
         }
         tb = t1/(2*t2);

         if (tt == (tb+to)) {
                *t = (int) tt;
                return;
         }
         tt = tb+to;
        }
        printf ("Too many iterations in THRESH_IS!\n");
        *error_code = NO_REGION;
        *t = 127;
}

/* Set the pixels on the frame of a grey-level image X to 0 */
void frame (struct image *x)
{
        int i,j;

        for (i=0; i<x->nr; i++) {
                x->band1[i][0] = 0;
                x->band1[i][x->nc-1] = 0;
        }
        for (j=0; j<x->nc; j++) {
                x->band1[0][j] = 0;
                x->band1[x->nr-1][j] = 0;
        }
}

/*      Mark an 4-connected region, beginning at (iseed, jseed), with VALUE,
                and return the region size in *REG_SIZE                   */
```

```
void mark4 (struct image *x, int value, int iseed, int jseed, int *reg_size)
{
        int i,j,n,m, k, again;

        if (range(x, iseed, jseed)==0) return;

/* Pixels to be marked will all have the value K */
        k = x->band1[iseed][jseed];
        x->band1[iseed][jseed] = value;

        *reg_size = 0;

        do {
          again = 0;
          for (i=iseed; i<x->nr; i++)
                for (j=0; j<x->nc; j++)
                        if (x->band1[i][j] == value)
                                for (n=i-1; n<=i+1; n++)
                                  for (m=j-1; m<=j+1; m++) {
                                        if ( (j-m)*(i-n) != 0) continue;
                                        if (range(x, n, m) == 0) continue;
                                        if (x->band1[n][m] == k) {
                                                x->band1[n][m] = value;
                                                (*reg_size) ++;
                                                again = 1;
                                        }
                                  }
          for (i=x->nr-1; i>=iseed; i--)
                for (j=x->nc-1; j>=0; j--)
                        if (x->band1[i][j] == value)
                                for (n=i-1; n<=i+1; n++)
                                  for (m=j-1; m<=j+1; m++) {
                                        if ( (j-m)*(i-n) != 0) continue;
                                        if (range(x, n, m) == 0) continue;
                                        if (x->band1[n][m] == k) {
                                                x->band1[n][m] = value;
                                                (*reg_size) ++;
                                                again = 1;
                                        }
                                  }
        } while (again);
}

/*    Locate a OBJECT region, mark it with value VALUE,
                and return the value *ISART of the first row. 4-conneceted  */
void region_4 (struct image *x, int value, int *istart, int *r_size, int *error_code)
{
        int i, j, ii, jj;

        *error_code = 0;
        ii= -1; jj = -1;
        for (i = *istart; i<x->nr; i++) {
```

```
          for (j=0; j<x->nc; j++)
                  if (x->band1[i][j] == OBJECT) {
                          ii=i; jj=j;
                          break;
                  }
          if (ii >= 0) break;
        }

        if (ii < 0) {
          *error_code = NO_REGION;
          return;
        }

        *istart = ii;
        mark4 (x, value, ii, jj, r_size);
}

/*    Mark an 8-connected region, beginning at (iseed, jseed), with VALUE,
              and return the region size in REG_SIZE                  */
void mark8 (struct image *x, int value, int iseed, int jseed, int *reg_size)
{
        int i,j,n,m, k, again;

        if (range(x, iseed, jseed)==0) return;

/* Pixels to be marked will all have the value K */
        k = x->band1[iseed][jseed];
        x->band1[iseed][jseed] = value;
        *reg_size = 0;

        do {
          again = 0;
          for (i=iseed; i<x->nr; i++)
                  for (j=0; j<x->nc; j++)
                 if (x->band1[i][j] == value)
                  for (n=i-1; n<=i+1; n++)
                          for (m=j-1; m<=j+1; m++) {
                           if (range(x, n, m) == 0) continue;
                           if (x->band1[n][m] == k) {
                                  x->band1[n][m] = value;
                                  (*reg_size) ++;
                                  again = 1;
                           }
                          }

          for (i=x->nr-1; i>=iseed; i--)
                  for (j=x->nc-1; j>=0; j--)
                 if (x->band1[i][j] == value)
                  for (n=i-1; n<=i+1; n++)
                          for (m=j-1; m<=j+1; m++) {
                           if (range(x, n, m) == 0) continue;
                           if (x->band1[n][m] == k) {
```

```
                                x->band1[n][m] = value;
                                (*reg_size) ++;
                                again = 1;
                            }
                        }
        } while (again);
}

/*  Locate a OBJECT region, mark it with value VALUE,
                and return the value ISART of the first row. 8-conneceted  */

void region_8 (struct image *x, int value, int *istart, int *r_size, int *error_code)
{
        int i,j,ii,jj;

        *error_code = 0;
        ii= -1; jj = -1;
        for (i=*istart; i<x->nr; i++) {
          for (j=0; j<x->nc; j++)
                if (x->band1[i][j] == OBJECT) {
                        ii=i; jj=j;
                        break;
                }
          if (ii >= 0) break;
        }

        if (ii < 0) {
                *error_code = NO_REGION;
                return;
        }
        *istart = ii;
        mark8 (x, value, ii, jj, r_size);
}

/* Fill any holes in the region marked V by marking them too, and return
        the coordinates of the region in RXY array.                  */
void fill_holes (struct image *x, int v, int *rxy, int *error_code)
{
        int i, j, m;
        struct image *z;

        *error_code = 0;
/* copy region marked V into Z, and get the region coordinates in RXY array*/
        copy_reg (x, &z, v, rxy, error_code);
        if (*error_code){
         free_image (z, error_code);
         return;
        }

/* Assume (0,0) is background, and remark it */
        mark4 (z, 254, 0, 0, &m);
```

```
/* Any remaining pixels with value BACKGROUND are holes. Change them to V. */
        for (i=0; i<z->nr; i++)
          for (j=0; j<z->nc; j++)
                if (z->band1[i][j] == BACKGROUND)
                  x->band1[i+rxy[0]-1][j+rxy[1]-1] = v;
/*      mark4 (z, 254, i, j);*/
        free_image (z, error_code);
}

/* Copy the pixels belonging to the region marked VAL into a new
        image (y). All other pixels will be background. The new image
        will be 1 pixel bigger than the region in row & column. Return
        the coordinates of the region in RXY array                  */

void copy_reg (struct image *x, struct image **y, int val,
                int *rxy, int *error_code)
{
        int i,j, rmin, rmax, cmin, cmax;

        *error_code = 0;

        box (x, val, rxy, error_code);
        if (*error_code) return;
        rmin = rxy[0];  cmin = rxy[1];  rmax = rxy[2];  cmax = rxy[3];

        /* Create and initialize the new region image */
        (*y) = new_image (rmax-rmin+3, cmax-cmin+3, 0, error_code);
        if (*error_code) return;
        for (i=0; i<(*y)->nr; i++)
          for (j=0; j<(*y)->nc; j++)
                (*y)->band1[i][j] = BACKGROUND;

        /* Copy VAL pixels into Z */
        for (i=1; i<(*y)->nr-1; i++)
          for (j=1; j<(*y)->nc-1; j++)
                if (range(x,i+rmin-1, j+cmin-1)) {
                 if (x->band1[i+rmin-1][j+cmin-1] == val)
                        (*y)->band1[i][j] = val;
                 else (*y)->band1[i][j] = BACKGROUND;
                } else (*y)->band1[i][j] = BACKGROUND;
}

/* Determine the image-oriented bounding box for the region in the
        image X marked with value VAL. Return coordinates of the region
        in the array RXY                                        */
void box(struct image *x, int val, int *rxy, int *error_code)
{
        int i,j, ip1,jp1,ip2,jp2;

        *error_code = 0;
        ip1 = 10000;  jp1 = 10000;
        ip2 = -1;     jp2 = -1;
```

```
/* Find the min and max coordinates, both row and column */
        for (i=0; i<x->nr; i++)
          for(j=0; j<x->nc; j++)
                if (x->band1[i][j] == val) {
                        if (i < ip1) ip1 = i;
                        if (i > ip2) ip2 = i;
                        if (j < jp1) jp1 = j;
                        if (j > jp2) jp2 = j;
                }
        if (jp2 < 0) {
                *error_code = NO_REGION;
                return;
        }

/* Coordinate array RXY:
        rxy[0],rxy[1] : Upper left (min,min)
        rxy[2],rxy[1] : Lower left (max,min)
        rxy[2],rxy[3] : Lower right (max,max)
        rxy[0],rxy[3] : Upper right (min,max)           */

        rxy[0] = ip1;   rxy[2] = ip2;
        rxy[1] = jp1;   rxy[3] = jp2;
}

/* Delete a region marked VALUE by setting the pixel values to BACKGROUND */
void del_reg (struct image *x, int value, int *error_code)
{
        int i,j;

        *error_code = 0;
        for (i=0; i<x->nr; i++)
          for (j=0; j<x->nc; j++)
                if (x->band1[i][j] == value)
                        x->band1[i][j] = BACKGROUND;
}

/* Extract the object marked VAL (coordinate range indicated by array RXY)
        into a new image (y). The pixels belong to the object will be set to
        OBJECT, and all other pixels will be set to BACKGROUND. The new image
        will be 2 pixel bigger than the object in row & column.         */
void extract_obj (struct image *marked, struct image *original,
                                struct image **bin_obj,struct image **cl_obj,
                                int val,int *rxy,int *error_code)
{
        int i, j, rmin, rmax, cmin, cmax, nr, nc;

        *error_code = 0;

        rmin = rxy[0]; cmin = rxy[1];
        rmax = rxy[2]; cmax = rxy[3];

        /* Create and initialize the new object images */
```

```
        nr = rmax - rmin + 3;
        nc = cmax - cmin + 3;
        (*bin_obj) = new_image (nr, nc, 0, error_code);
        if (*error_code) return;
        (*cl_obj)  = new_image (nr, nc, original->color, error_code);
        if (*error_code) return;

        for (i=0; i<nr; i++)
          for (j=0; j<nc; j++){
                (*bin_obj)->band1[i][j] = BACKGROUND;
                (*cl_obj)->band1[i][j]  = BACKGROUND;
                if (original->color){
                 (*cl_obj)->band2[i][j]  = BACKGROUND;
                 (*cl_obj)->band3[i][j]  = BACKGROUND;
                }
        }

        /* Copy VAL pixels into Z */
        rmin = rxy[0]; cmin = rxy[1];
        for (i=1; i<(nr-1); i++)
          for (j=1; j<(nc-1); j++)
                if (range(marked,i+rmin-1, j+cmin-1)) {
                 if (marked->band1[i+rmin-1][j+cmin-1] == val){
                        (*bin_obj)->band1[i][j] = OBJECT;
                        (*cl_obj)->band1[i][j] = original->band1[i+rmin-1][j+cmin-1];
                        if (original->color){
                         (*cl_obj)->band2[i][j] = original->band2[i+rmin-1][j+cmin-1];
                         (*cl_obj)->band3[i][j] = original->band3[i+rmin-1][j+cmin-1];
                        }
                 }else {
                        (*bin_obj)->band1[i][j] = BACKGROUND;
                        (*cl_obj)->band1[i][j] = BACKGROUND;
                        if (original->color){
                         (*cl_obj)->band2[i][j] = BACKGROUND;
                         (*cl_obj)->band3[i][j] = BACKGROUND;
                        }
                 }
                }else{
                 (*bin_obj)->band1[i][j] = BACKGROUND;
                 (*cl_obj)->band1[i][j] = BACKGROUND;
                 if (original->color){
                        (*cl_obj)->band2[i][j] = BACKGROUND;
                        (*cl_obj)->band3[i][j] = BACKGROUND;
                 }
                }
}

/*    Compute the perimeter of the region(s) marked with VAL  */
double perimeter (struct image *x, int val, int *sum_pixel, int *error_code)
{
        int i,j,k, ii,jj,t;
        double p;
```

```
        struct image *y;

        *error_code = 0;
        *sum_pixel = 0;
        p = 0.0; y = 0;
        copy_image (x, &y, 0, error_code);
        if (*error_code) return 0.0;

/* Remove all pixels except those having value VAL */
        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (x->band1[i][j] != val) {
                 y->band1[i][j] = BACKGROUND;
                 continue;
                }
                (*sum_pixel) ++;
                k = nay4(x, i, j, val); /* How many neighbors are VAL */
                if (k < 4)          /* If not all, this is on perim */
                 y->band1[i][j] = OBJECT;
                else y->band1[i][j] = BACKGROUND;
         }
        }

        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (y->band1[i][j] != OBJECT) continue;

/*    Match one of the templates     */

                k = 1; t = 0;
                for (ii= -1; ii<=1; ii++) {
                  for (jj = -1; jj<=1; jj++) {
                        if (ii==0 && jj==0) continue;
                        if (y->band1[i+ii][j+jj] == OBJECT)
                                t = t + k;
                        k = k << 1;
                  }
                }

/* Templates for 1.207:
       ooo oo# o#o o#o #oo oo# ooo #oo
       ##o ##o o#o o#o o#o o#o o## o##
       oo# ooo #oo oo# o#o o#o #oo ooo
T= 210  014  042  202  101  104  060  021

       Templates for 1.414:
       #oo oo# #oo oo# ooo #o#
       oPo oPo oPo oPo oPo oPo
       oo# #oo #oo oo# #o# ooo
T= 201  044  041  204  240  005

       Templates for 1.0:
```

```
        ooo o#o ooo ooo o#o o#o
        ### o#o ##o o## ##o o##
        ooo o#o o#o o#o ooo ooo
 T= 030   102   72    80    10    18

 */
                if (t==0210 || t == 014 || t == 042 ||
                        t==0202 || t ==0101 || t ==0104 ||
                        t== 060 || t == 021) {
                        p += 1.207;
                        continue;
                }

                if (t == 0201 || t == 044 || t == 041 ||
                        t == 0204 || t ==0240 || t == 005) {
                        p += 1.414;
                        continue;
                }

                if (t == 030 || t == 0102 || t == 80 ||
                        t == 10 || t == 18) {
                        p += 1.0;
                        continue;
                }
                p += 1.207;
          }
        }
        free_image (y, error_code);
        return p;
}

/* Compute the color features of the object marked with VAL. */
void color_features(struct image *bin_obj,struct image *cl_obj,
                                        int val,struct feature *objf, int n, int *error_code)
{
        int j, k;
        double r, g, b;
        double r1, g1, b1;
        double h, s, i;
        double width;
        long np = 0;            /* pixel number of the object */

        if (!cl_obj->color){
          *error_code = BAD_ARGUMENT2;
          return;
        }

        width = 256.0/(double)n;

/* initialize feature struct */
        objf->meanR = 0.0;
        objf->meanG = 0.0;
```

```
        objf->meanB = 0.0;
        objf->meanR3G2B1 = 0.0;
        objf->meanH = 0.0;
        objf->meanS = 0.0;
        objf->meanI = 0.0;

        objf->varR = 0.0;
        objf->varG = 0.0;
        objf->varB = 0.0;
        objf->varR3G2B1 = 0.0;
        objf->varH = 0.0;
        objf->varS = 0.0;
        objf->varI = 0.0;

        for (j=0; j<n; j ++){
         objf->histR[j] = 0.0;
         objf->histG[j] = 0.0;
         objf->histB[j] = 0.0;
        }

        for (j = 0; j < bin_obj->nr; j ++){
         for (k = 0; k < bin_obj->nc; k ++){
                if (bin_obj->band1[j][k] != val) continue;

                np ++;                  /* count object pixel number */

/* Read in RGB grey-level values */
                r1 = (double)(cl_obj->band1[j][k]);
                g1 = (double)(cl_obj->band2[j][k]);
                b1 = (double)(cl_obj->band3[j][k]);

                objf->histR[(int)(r1/width)] += 1.0;
                objf->histG[(int)(g1/width)] += 1.0;
                objf->histB[(int)(b1/width)] += 1.0;

/* Remove Gamma corrections and normalized RGB */
                r = exp((1/2.2)*log(1e-20+r1/WHITE));
                g = exp((1/2.2)*log(1e-20+g1/WHITE));
                b = exp((1/2.2)*log(1e-20+b1/WHITE));

/* Compute HSI values */
                i = (r + g + b) / 3.0;
                if (i == 0){
                 s = 0; h = 0;
                }else{
                 s = 1.0 - (min(min(r, g), b))/i;
                 if (s == 0) h = 0;
                 else h = acos(0.5*((r-g)+(r-b))/sqrt(1e-20+(r-g)*(r-g)+(r-b)*(g-b)));
                }
                if (b/i > g/i) h = 2.0*PI - h;
                h = h / (2.0*PI);
```

```
			objf->meanR = objf->meanR + r;
			objf->meanG = objf->meanG + g;
			objf->meanB = objf->meanB + b;
			objf->meanR3G2B1 = objf->meanR3G2B1 + (3.0*r+2.0*g+b)/6.0;
			objf->meanH = objf->meanH + h;
			objf->meanS = objf->meanS + s;
			objf->meanI = objf->meanI + i;

			objf->varR = objf->varR + r*r;
			objf->varG = objf->varG + g*g;
			objf->varB = objf->varB + b*b;
			objf->varR3G2B1 = objf->varR3G2B1 + (3.0*r+2.0*g+b)*(3.0*r+2.0*g+b)/36.0;
			objf->varH = objf->varH + h*h;
			objf->varS = objf->varS + s*s;
			objf->varI = objf->varI + i*i;

		  }
		}
		objf->meanR = objf->meanR / (double)np;
		objf->meanG = objf->meanG / (double)np;
		objf->meanB = objf->meanB / (double)np;
		objf->meanR3G2B1 = objf->meanR3G2B1 / (double)np;
		objf->meanH = objf->meanH / (double)np;
		objf->meanS = objf->meanS / (double)np;
		objf->meanI = objf->meanI / (double)np;

		objf->varR = (objf->varR - (double)np*(objf->meanR)*(objf->meanR))/((double)np-1.0);
		objf->varG = (objf->varG - (double)np*(objf->meanG)*(objf->meanG))/((double)np-1.0);
		objf->varB = (objf->varB - (double)np*(objf->meanB)*(objf->meanB))/((double)np-1.0);
		objf->varR3G2B1 =
(objf->varR3G2B1-(double)np*(objf->meanR3G2B1)*(objf->meanR3G2B1))/((double)np-1.0);
		objf->varH = (objf->varH -
(double)np*(objf->meanH)*(objf->meanH))/((double)np-1.0);
		objf->varS = (objf->varS - (double)np*(objf->meanS)*(objf->meanS))/((double)np-1.0);
		objf->varI = (objf->varI - (double)np*(objf->meanI)*(objf->meanI))/((double)np-1.0);

		for (j=0; j<n; j ++){
		  objf->histR[j] = objf->histR[j] / (double)np;
		  objf->histG[j] = objf->histG[j] / (double)np;
		  objf->histB[j] = objf->histB[j] / (double)np;
		}
}
/Extract morphological features from a binary image */
void size_shape_features(struct image *bin_obj,int val,struct feature *objf,
							double mm_per_pix, int *error_code)
{
		int i,j,k,m,ii,jj,t, orien;
		long np;			/* np: number of pixels on perimeter */
		double a[35], b[35], c[35];
		double r,ip1,ip2,jp1,jp2;
		double x1[4], y1[4], cmi, cmj;
		struct image *y;
```

```
        *error_code = 0;

 /* Get center of mass of the object */
        center_of_mass (bin_obj, val, &cmi, &cmj, error_code);
        if (*error_code) return;
 //   disp_image(bin_obj,0,error_code);

 /* Find the principal axis; this defines the direction of the 'length'
   dimension, and is a straight line defined by 2 points          */
        principal_axis (bin_obj,val,&ip1,&jp1,&ip2,&jp2,cmi,cmj,error_code);
        if (*error_code) return;
 //   disp_image(bin_obj,0,error_code);

 /* Compute the coefficients of the equation of the PA: a[1]x+b[1]y+c[1]=0. */
        line2pt (ip1, jp1, ip2, jp2, &a[1], &b[1], &c[1]);

 /* Compute the coefficients of the equation of the MA: a[0]x+b[0]y+c[0]=0. */
        a[0] = b[1];
        b[0] = -a[1];
        c[0] = -a[0]*cmi - b[0]*cmj;
        if (c[0] < 0 || (c[0] == 0 && b[0] < 0)){
         a[0] = -a[0];
         b[0] = -b[0];
         c[0] = -c[0];
        }

 /* Get the boundary image Y   */

 /*  make a copy of object image */
        y = 0;
        copy_image (bin_obj, &y, 0, error_code);
        if (*error_code) return;
 // disp_image(y,0,error_code);

 /*  Extract the bounary */
        for (i=0; i<(y->nr); i++) {
         for (j=0; j<(y->nc); j++) {
                if (y->band1[i][j] != val) continue;
                 k = nay4(bin_obj, i, j, val);    /* How many neighbors are VAL      */
                 if (k < 4)                   /* If not all, this is on perimeter */
                        y->band1[i][j] = OBJECT;
                 else y->band1[i][j] = BACKGROUND;
         }
        }
 // disp_image(y,0,error_code);

 /* Determine the orientation of the object */
        orien = orientation(y, a, b, c);

 /* Compute the coefficients of 15 lines a[i]*x+b[i]*y+c[i]=0 (i=2,..16) that
   intersects PA: a[1]x+b[1]y+c[1]=0 at point (x, y) with angle of i*PI/16. */
        lines_radius(a, b, c, cmi, cmj);
```

```
/*   for (i=1; i<17; i ++){
          clr_line(bin_obj, a[i],b[i],c[i],error_code);
          disp_image(bin_obj,0,error_code);
        }*/

/* Get the the area features in term of pixel number.          */
        objf->area = 0.0;
        for (i=0; i<32; i++) objf->areaR[i] = 0.0;

        for (i=0; i<y->nr; i++) {
          for (j=0; j<y->nc; j++) {
                if (bin_obj->band1[i][j] != val) continue;

/*  locate the pixel in which of the 32 subregions divided by a[i]*x+b[i]*y+c[i], i=1..16.*/
                m = locate_region(i, j, a, b, c, orien);

                (objf->area) ++;     /* count the pixel number of the object       */
                (objf->areaR[m]) ++; /* count the pixel number of the subregion m  */
          }
        }
//  disp_image(bin_obj,0,error_code);

/* Compute the coefficients of  the 7 lines: a[i]*x+b[i]*y+c[i]=0 (i=17,..23) parallel
   to PA, equally dividing MA & the 7 lines: a[i]*x+b[i]*y+c[i]=0 (i=24,..30) parallel
   to MA, equally dividing PA */
        lines_parallel(y, OBJECT, a, b, c, orien);

/* MA: a[0]*x+b[0]*y+c[0]=0.
   PA: a[1]*x+b[1]*y+c[1]=0.
   Radius lines: a[i]*x+b[i]*y+c[i]=0, i = 1, ... 16.
   Lines parallel to PA: a[i]*x+b[i]*y+c[i]=0, i=17,...23.
   Lines parallel to MA: a[i]*x+b[i]*y+c[i]=0, i=24,...30.
   MER. Box: L1: a[31]*x+b[31]*y+c[31]=0, L2: a[32]*x+b[32]*y+c[32]=0,
                       W1: a[33]*x+b[33]*y+c[33]=0, L2: a[34]*x+b[34]*y+c[34]=0. */

/* Compute all length/width features */

/*  find the intersection of W1 with L1:  */
        line_intersect (a[31],b[31],c[31], a[33],b[33],c[33], &(x1[0]), &(y1[0]));

/*  find the intersection of W2 with L1:  */
        line_intersect (a[31],b[31],c[31], a[34],b[34],c[34], &(x1[1]), &(y1[1]));

/*  find the intersection of W2 with L2:  */
        line_intersect (a[32],b[32],c[32], a[34],b[34],c[34], &(x1[2]), &(y1[2]));

/*  find the intersection of W1 with L2:  */
        line_intersect (a[32],b[32],c[32], a[33],b[33],c[33], &(x1[3]), &(y1[3]));

        objf->length = dist_2pt(x1[0], y1[0], x1[1], y1[1]);
        objf->width  = dist_2pt(x1[0], y1[0], x1[3], y1[3]);
```

```
        objf->lpa = line_interval(y,a[1],b[1],c[1]);
        objf->wma = line_interval(y,a[0],b[0],c[0]);

        for (i=0; i<7; i++){
         objf->lwR[i]   = line_interval(y, a[i+17], b[i+17], c[i+17])/objf->length;
         objf->lwR[7+i] = line_interval(y, a[i+24], b[i+24], c[i+24])/objf->width;
        }

/* Compute perimeter and all radius related features */
        objf->perimeter = 0.0;
        objf->rmean = 0.0;
        objf->var_r = 0.0;
        objf->rmin = 10000.0;
        objf->rmax = 0.0;
        np = 0;

        for (i=0; i<32; i ++) objf->perimR[i] = 0.0;

        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (y->band1[i][j] != OBJECT) continue;

/* Compute the radius related features              */
                r = dist_2pt((double)i,(double)j,cmi,cmj);

                if (r <= objf->rmin) objf->rmin = r;
                if (r > objf->rmax)  objf->rmax = r;

                objf->rmean = objf->rmean + r;

                objf->var_r = objf->var_r + r * r;

                np ++;

/* Locate the pixel position in the 24 subregions */
                m = locate_region(i, j, a, b, c, orien);

/* Match one of the templates for computing perimeter    */

                k = 1;  t = 0;
                for (ii= -1; ii<=1; ii++) {
                  for (jj = -1; jj<=1; jj++) {
                        if (ii==0 && jj==0) continue;
                        if (y->band1[i+ii][j+jj] == OBJECT)
                                t = t + k;
                        k = k << 1;
                  }
                }

/* Templates for 1.207:
       ooo oo# o#o o#o  #oo  oo#  ooo #oo
       ##o ##o o#o o#o  o#o  o#o  o## o##
```

```
      oo# ooo #oo oo# o#o o#o #oo ooo
T= 210   014   042   202   101   104   060   021
D= 1.2071 1.2071 1.2071 1.2071 1.2071 1.2071 1.2071 1.2071

      Templates for 1.414:
      #oo oo# #oo oo# ooo #o#
      oPo oPo oPo oPo oPo oPo
      oo# #oo #oo oo# #o# ooo
T= 201   044   041   204   240   005
D= 1.4142 1.4142 1.4142 1.4142 1.4142 1.4142

      Templates for 1.0:
      ooo o#o ooo ooo o#o o#o
      ### o#o ##o o## ##o o##
      ooo o#o o#o o#o ooo ooo
T= 030   102   72   80   10   18
D= 1.0   1.0   1.0   1.0   1.0   1.0                 */

            if (t==0210 || t==014  || t==060  || t==021 ||
                t==042  || t==0202 || t==0101 || t==0104) {
                objf->perimeter += 1.2071;
                objf->perimR[m] += 1.2071;
                continue;
            }

            if (t==0201 || t==044  || t==041 ||
                t==0204 || t==0240 || t==005)  {
                objf->perimeter += 1.4142;
                objf->perimR[m] += 1.4142;
                continue;
            }

            if (t==030 || t==0102 || t==72 ||
                t==80  || t==10   || t==18)  {
                objf->perimeter += 1.0;
                objf->perimR[m] += 1.0;
                continue;
            }

            objf->perimeter += 1.2071;
            objf->perimR[m] += 1.2071;

      }
    }

/* compute radius at each PI/16 angle from PA */
    for (k=0; k<16; k++)
      radius(y,a,b,c,k,cmi,cmj,&(objf->radR[k]),&(objf->radR[k+16]));

/* compute radius related ratio features */

    objf->rmean = objf->rmean/np;  /* mean radius */
```

```
        objf->var_r = (objf->var_r - np*(objf->rmean)*(objf->rmean))/(np-1.0); /* Radius variance*/

        for (i=0; i<32; i++){
          objf->radR[i]   = objf->radR[i]/objf->rmax;
          objf->areaR[i]  = objf->areaR[i]/objf->area;
          objf->perimR[i] = objf->perimR[i]/objf->perimeter;
        }
//   disp_image(y,0,error_code);

        /* Space calibration */
        objf->area = objf->area * mm_per_pix * mm_per_pix;
        objf->perimeter = objf->perimeter * mm_per_pix;
        objf->length = objf->length * mm_per_pix;
        objf->width = objf->width * mm_per_pix;
        objf->lpa = objf->lpa * mm_per_pix;
        objf->wma = objf->wma * mm_per_pix;
        objf->rmax = objf->rmax * mm_per_pix;
        objf->rmin = objf->rmin * mm_per_pix;
        objf->rmean = objf->rmean * mm_per_pix;
        objf->var_r = objf->var_r * mm_per_pix * mm_per_pix;
//   disp_image(y,0,error_code);

        /* compute ratio shape features */
        objf->asp_R  = objf->lpa / objf->wma;
        objf->rec_R  = objf->length / objf->width;
        objf->rad_R  = objf->rmax / objf->rmin;
        objf->thin_R = objf->perimeter * objf->perimeter / objf->area;
        objf->area_R = objf->length * objf->width / objf->area;
        objf->har_R  = objf->rmean / objf->var_r;

        *error_code = 0;
        free_image (y, error_code);

/* clear each 24 sub_regions determined by 12 radius lines */
/*  for (k=0; k<32; k ++){
          for (i=0; i<bin_obj->nr; i++) {
                  for (j=0; j<bin_obj->nc; j++) {
                    m = locate_region(i,j,a,b,c, orien);
                    if (bin_obj->band1[i][j] == val && m==k)
                            bin_obj->band1[i][j] = BACKGROUND;
                  }
          }
          disp_image(bin_obj,0,error_code);
        }*/
}

/* Calculate the coordinates of the center of mass of the region(s)
        marked with the value VAL. Return as (II,JJ).              */
void center_of_mass (struct image *x, int val, double *ii,
                    double *jj, int *error_code)
{
        int i,j;
```

```
	long kk;

	*error_code = 0;
	kk = 0;
	*ii = 0.0;	*jj = 0.0;
	for (i=0; i<x->nr; i++) {
	  for (j=0; j<x->nc; j++) {
		if (x->band1[i][j] == val) {
			*ii += (double)i;	*jj += (double)j;
			kk += 1;
		}
	  }
	}

	if (kk==0) {
		*error_code = NO_REGION;
		return;
	}
	*ii = *ii/(double)kk;		*jj = *jj/(double)kk;
}

/* Determine the principal axis of the region marked with VAL in
	the image X. Line will be specified by two points:(i1,j1),(i2,j2) */
void principal_axis(struct image *x,int val,double *i1,double *j1,double *i2,
				double *j2, double cmi, double cmj,int *error_code)
{
	int i,j, di,dj,k;
	struct image *y;
	double dmax,dd,cmi1,cmj1 ;

	*error_code = 0;

/* Make a local copy of the image so it can be changed */
	y = 0;
	copy_image (x, &y, 0, error_code);
	if (*error_code) return;

/* Change (cmi, cmj) into integer coordinate */
	cmi1 = (double)( (int)cmi );	cmj1 = (double)( (int)cmj );

/* Mark candidate pixels: perimeter between 0-row CMI and col CMJ-max */
	for (i=0; i<=(int)(cmi+0.5); i++)
	 for (j=0; j<x->nc; j++)
		if (x->band1[i][j] == val) {
		  if (nay4(x, i,j, val) != 4)
			y->band1[i][j] = 254;
		}

	dmax = 1.0e20; di = -1; dj = -1;

/* The principal axis will pass through the center of mass. Consider
	all candidate pixels, determine the line through it and the COM,
```

```
	and sum the distance between the line an all pixels in the region */
	do {
	  k = 0;
	  for (i=0; i<=(int)(cmi1+0.5); i++)
		for (j=0; j<x->nc; j++)
		 if (y->band1[i][j] == 254) {
			dd = all_dist(x, cmi1,cmj1, (double)i,(double)j, val);
			if (dd < dmax) {
			  dmax = dd;
			  di = i; dj = j;
			  k += 1;
			}
			y->band1[i][j] = val;
		 }
	} while (k);

	*i1 = (double)di;	*j1 = (double)dj;
	*i2 = cmi1;	*j2 = cmj1;
	free_image (y, error_code);
}

/* Compute the distance between two points (r1,c1) & (r2,c2). */
double dist_2pt(double r1, double c1, double r2, double c2)
{
	double r, c, d;

	r = (r1-r2);
	c = (c1-c2);
	d = sqrt(r*r + c*c);

	return d;
}

/* Compute distance between the line given and all pixels in the
  region. Line is specified by two points: (i1,j1) and (i2,j2)	*/
double all_dist (struct image *x, double i1, double j1,
		double i2, double j2, int val)
{
	int i,j;
	double a, b, c, e, f, d;

/* Equation of the line is a*x + b*y + c = 0 */
	a = j2 - j1;
	b = i1 - i2;
	c = - (i1-i2)*j1 + (j1-j2)*i1 ;
	e = a*a + b*b;
	d = 0.0;
/* Sum the residuals, substituting (i,j) for each pixel in place of (x,y) */
	for (i=0; i<x->nr; i++)
	  for (j=0; j<x->nc; j++) {
		if (x->band1[i][j] != val)continue;
	       f = (a*i + b*j + c);
```

```
                f = f*f/e;
                d = d + f;
            }
        return d;
}

/* Calculate the coefficients of the line perpendicular to ax+by+c=0  */
void perp (double a, double b, double c, double *a1, double *b1,
           double *c1, double x, double y)
{
        c = c;
        *a1 = b;
        *b1 = -a;
        *c1 = a*y - b*x;
}

/* Compute the coefficients of 15 line a[i]*x+b[i]*y+c[i]=0 (i=2,..16) that
   intersects PA: a[1]x+b[1]y+c[1]=0 at point (x, y) with angle of i*PI/16. */
void lines_radius(double *a, double *b, double *c, double x, double y)
{
        int i;
        double alpha, di;

        for (i=1; i<16; i ++){
         alpha = i*PI/16;
         if (alpha < PI/2){
                di = tan(alpha);
                a[i+1] = a[1] - b[1] * di;
                b[i+1] = b[1] + a[1] * di;
                c[i+1] = -a[i+1]*x - b[i+1]*y;
         }
         else if (alpha > PI/2){
                        di = tan(alpha);
                        a[i+1] = -a[1] + b[1] * di;
                        b[i+1] = -b[1] - a[1] * di;
                        c[i+1] = -a[i+1]*x - b[i+1]*y;

                }else {
                        a[i+1] = a[0];
                        b[i+1] = b[0];
                        c[i+1] = c[0];
                }
        }
}

/* Compute the coefficients of  the 7 lines: a[i]*x+b[i]*y+c[i]=0 (i=13,..19) parallel
   to PA, equally dividing MA & the 7 lines: a[i]*x+b[i]*y+c[i]=0 (i=20,..26) parallel
   to MA, equally dividing PA */
void lines_parallel(struct image *z, int value,
                                        double *a, double *b, double *c, int orient)
{
  int i, i1, j1, i2, j2, i3, j3, i4, j4;
```

```
/* Find the two pixels farthest (perpendicular) from the PA. One must be positive
   in distance, the other negative. These points will be (i1,j1) =+ve and
   (i2,j2)=-ve, and will lie on opposite sides of the MER.                */
        minmax_dist (z, value, a[1], b[1], c[1], &i1,&j1,&i2,&j2);

/* Find the two pixels farthest (perpendicular) from the MA. One must be positive
   in distance, the other negative. These points will be (i3,j3) =+ve and
   (i4,j4)=-ve, and will lie on opposite sides of the MER.                */
        minmax_dist (z, value, a[0], b[0], c[0], &i3,&j3,&i4,&j4);

/* L1 and L2 are lines forming opposite edges of MER parallel to PA  */
        c[31] = -a[1]*i1-b[1]*j1; a[31] = a[1]; b[31] = b[1]; /* L1 */
        c[32] = -a[1]*i2-b[1]*j2; a[32] = a[1]; b[32] = b[1]; /* L2 */

/* W1 and W2 are lines parallel to MA forming opposite edges of the MER */
        c[33] = -a[0]*i3-b[0]*j3; a[33] = a[0]; b[33] = b[0]; /* W1 */
        c[34] = -a[0]*i4-b[0]*j4; a[34] = a[0]; b[34] = b[0]; /* W2 */

/* Find the seven lines parallel to PA, equally dividing MA
   and the seven lines parallel to MA, equally dividing PA */
        for (i=0; i<7; i++){
         a[i+17] = a[1];
         b[i+17] = b[1];
         a[i+24] = a[0];
         b[i+24] = b[0];
         switch (orient){
                case 1:
                 c[i+17] = (i+1)*(c[32]-c[31])/8 + c[31];
                 c[i+24] = (i+1)*(c[33]-c[34])/8 + c[34];
                break;
                case 2:
                 c[i+17] = (i+1)*(c[32]-c[31])/8 + c[31];
                 c[i+24] = (i+1)*(c[34]-c[33])/8 + c[33];
                break;
                case 3:
                 c[i+17] = (i+1)*(c[31]-c[32])/8 + c[32];
                 c[i+24] = (i+1)*(c[34]-c[33])/8 + c[33];
                break;
                case 4:
                 c[i+17] = (i+1)*(c[31]-c[32])/8 + c[32];
                 c[i+24] = (i+1)*(c[33]-c[34])/8 + c[34];
                break;
         }
        }
}

/* Return the number of 4-connected neighbors of (i,j) with value VAL */
int nay4 (struct image *x, int i, int j, int val)
{
        int n,m,k;

        if (x->band1[i][j] != val) return 0;
```

```
        k = 0;
        for (n= -1; n<=1; n++) {
          for (m= -1; m<=1; m++) {
                if (n*m) continue;
                if (range(x,i+n, j+m))
                  if (x->band1[i+n][j+m] == val) k++;
          }
        }
        return k-1;
}

/* Return the number of 8-connected neighbors of (i,j) having value VAL */
int nay8 (struct image *x, int i, int j, int val)
{
/*    return the number of 8-neighbors of (i,j)       */

        int n,m,k;

        if (x->band1[i][j] != val) return 0;
        k = 0;
        for (n= -1; n<=1; n++) {
          for (m= -1; m<=1; m++) {
                if (range(x,i+n, j+m))
                  if (x->band1[i+n][j+m] == val) k++;
          }
        }
        return k-1;
}

/

/*    Return 1 if (n,m) are legal (row,column) indices for image X    */
int range (struct image *x, int n, int m)
{
        if (n < 0 || n >= x->nr) return 0;
        if (m < 0 || m >= x->nc) return 0;
        return 1;
}

/* Count the total pixel number of a region with a grey level of val in image X */
int area(struct image *x, int val)
{
        int i,j,k;

        k = 0;
        for (i=0; i<x->nr; i++)
          for (j=0; j<x->nc; j++)
                if (x->band1[i][j] == val) k++;
        return k;
}

/* Find the two intersections (i1,j1) & (i2,j2) of the line ax + by + c =0
```

```
  and the given object boundary image X.     */
void ln_obj_intersec (struct image *y, double a, double b, double c,
                                        int *i1, int *j1, int *i2, int *j2)
{
        int i, j;
        int ii2, jj2, ii3, jj3, ii4, jj4;
        double d, dmin, d1, d2;


        dmin = 1000000.0;

        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (y->band1[i][j] != OBJECT) continue;

                d = fabs(a*i + b*j + c);
                if (d < dmin){
                 *i1 = i;
                 *j1 = j;
                 dmin = d;
                }
         }
        }

        dmin = 1000000.0;

        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (y->band1[i][j] != OBJECT) continue;
                if (i == *i1 && j == *j1 ) continue;

                d = fabs(a*i + b*j + c);
                if (d < dmin){
                 ii2 = i;
                 jj2 = j;
                 dmin = d;
                }
         }
        }

        dmin = 1000000.0;
        for (i=0; i<y->nr; i++) {
         for (j=0; j<y->nc; j++) {
                if (y->band1[i][j] != OBJECT) continue;
                if ((i == *i1 && j == *j1) || (i == ii2 && j == jj2)) continue;

                d = fabs(a*i + b*j + c);
                if (d < dmin){
                 ii3 = i;
                 jj3 = j;
                 dmin = d;
                }
```

```
    }
   }

   dmin = 1000000.0;
   for (i=0; i<y->nr; i++) {
    for (j=0; j<y->nc; j++) {
        if (y->band1[i][j] != OBJECT) continue;
        if ((i == *i1 && j == *j1) || (i == ii2 && j == jj2) || (i == ii3 && j == jj3)) continue;

        d = fabs(a*i + b*j + c);
        if (d < dmin){
         ii4 = i;
         jj4 = j;
         dmin = d;
        }
    }
   }

   d1 = dist_2pt((double)(*i1), (double)(*j1), (double)ii2, (double)jj2);
   d2 = dist_2pt((double)(*i1), (double)(*j1), (double)ii3, (double)jj3);

   if (d1 > 2.0){
    *i2 = ii2; *j2 = jj2;
   }
   else {
    if (d2 >2.0){
        *i2 = ii3; *j2 = jj3;
    }
    else {
        *i2 = ii4; *j2 = jj4;
    }
   }
}

/* Compute and return the distance between the two nearest intersections
   of the line ax + by + c =0 and the given object boundary image X.     */
double line_interval (struct image *y,double a, double b, double c)
{
        int i1, i2, j1, j2;
        double d;

        ln_obj_intersec (y, a, b, c, &i1, &j1, &i2, &j2);
        d = dist_2pt(i1, j1, i2, j2);

        return d;
}

/* Compute the radius in direction of line ax+by+c=0 */
void radius(struct image *y, double *a, double *b, double *c, int k,
                        double cmi, double cmj, double *r1, double *r2)
{
        int i, j, i1, i2, j1, j2;
```

```
	double d, dmin;

	dmin = 100000.0;
	i1 = 256; j1 = 256;
	i2 = -1, j2 = -1;
	dmin = 100000.0;

	for (i=0; i<y->nr; i++) {
	  for (j=0; j<y->nc; j++) {
		if (y->band1[i][j] != OBJECT) continue;

		d = fabs(a[k]*i + b[k]*j + c[k]);
		if (d < dmin){
		  i1 = i; j1 = j;
		  dmin = d;
		}
	  }
	}
	dmin = 100000.0;
	for (i=0; i<y->nr; i++) {
	  for (j=0; j<y->nc; j++) {
		if (y->band1[i][j] != OBJECT) continue;
		d = fabs(a[k]*i + b[k]*j + c[k]);
		if ( (d < dmin) && (i != i1) && (j !=j1) ){
		  i2 = i; j2 = j;
		  dmin = d;
		}
	  }
	}
	if (a[0]*i1 + b[0]*j1 + c[0] >= 0){
	  *r1 = dist_2pt(cmi,cmj,(double)i1,(double)j1);
	  *r2 = dist_2pt(cmi,cmj,(double)i2,(double)j2);
	}else{
	  *r2 = dist_2pt(cmi,cmj,(double)i1,(double)j1);
	  *r1 = dist_2pt(cmi,cmj,(double)i2,(double)j2);
	}
}

/* Find the point where two lines intersect	*/
int line_intersect (double a1, double b1, double c1, double a2,
				double b2, double c2, double *x, double *y)
{
	double dt;

	dt = a2*b1 - a1*b2;
	if (is_zero(dt)) return 0;

	*y = (a1*c2 - a2*c1)/dt;
	if (is_zero(a2))
	  *x = (-b1/a1)*(*y) - c1/a1;
	else *x = (-b2/a2)*(*y) - c2/a2;
```

```
        return 1;
}

/* Compute the coefficients a, b, and c of the equation ax+by+c=0
        of the line between (x1,y1) and (x2,y2).                    */
int line2pt (double x1, double y1, double x2, double y2,
                double *a, double *b, double *c)
{
        double dx, dy, dsq, dinv;
        *a = 0.0; *b = 0.0; *c = 0.0;
        dx = x2-x1;    dy = y2-y1;
        dsq = dx*dx + dy*dy;
        if (dsq < 1.0) return 0;
        dinv = -1.0/sqrt(dsq);
        *a = -dy*dinv;
        *b = dx*dinv;
        *c = (x1*y2 - x2*y1)*dinv;
        if (*c < 0 || (*c == 0 && *b < 0)){
          *a = -(*a);
          *b = -(*b);
          *c = -(*c);
        }
        return 1;
}

/* Find the two object pixels farthest (perpendicular) from the line ax+by+c=0.
   One must be positive in distance, the other negative. These points will be
   (ii1,jj1) =+ve and (ii2,jj2)=-ve, and will lie on opposite sides of the MER. */
void minmax_dist (struct image *x, int val, double a, double b, double c,
                int *ii1, int *jj1, int *ii2, int *jj2)
{
        int i,j;
        double f, dmax,dmin;

        dmax = 0.0;    dmin = 100000.0;

/* Locate the pixels with the maximum and minimum residual */
        for (i=0; i<x->nr; i++)
          for (j=0; j<x->nc; j++) {
                if (x->band1[i][j] != val)continue;
                f = (a*i + b*j + c);
                if (f < dmin) {
                  *ii2 = i; *jj2 = j;
                  dmin = f;
                }
                if (f > dmax) {
                  *ii1 = i;      *jj1 = j;
                  dmax = f;
                }
          }
}
```

```
/* Clear (set to BACKGROUND) a line a*x + b*y + c = 0 in the region VAL */
void clr_line (struct image *x,double a, double b, double c,int *error_code)
{
        int i,j,m,n,rn,ibegin,err;
        double f, dmin;

        n = 0; ibegin = 0; m = 0; err = 0;
        while (!err) {
         region_4 (x, n+1, &ibegin, &m, &err);
         if (err == NO_REGION) break;
         n++;
        }
        if (n == 0){
         *error_code = NO_REGION;
         return;
        }

        dmin = 0.0;
        rn = n;
        while (n == rn ){
         dmin += 0.5;
         /* clear the pixels with the minimum residual and set the other
                back to OBJECT                                  */
         for (i = 0; i < x->nr; i++)
                for (j=0; j < x->nc; j++) {
                 if (x->band1[i][j] == BACKGROUND) continue;
                 f = (a*i + b*j + c);
                 if (fabs(f) < dmin) x->band1[i][j] = BACKGROUND;
                 else x->band1[i][j] = OBJECT;
                }

         rn = 0; ibegin = 0; m = 0; err = 0;
         while (!err) {
                region_4 (x, rn+1, &ibegin, &m, &err);
                if (err == NO_REGION) break;
                rn++;
         }
        }
        for (i=0; i < x->nr; i++)
         for (j=0; j < x->nc; j++) {
                if (x->band1[i][j] == BACKGROUND) continue;
                x->band1[i][j] = OBJECT;
         }
}

/*    Is a real value close enough to zero?   */
int is_zero (double x)
{
        if ( (x <= 0.0001) && (x >= -0.0001) ) return 1;
        return 0;
}
```

```
/* Compute the angle between two points. (r1,c1) is the origin
        specified as row, column, and (r2,c2) is the second point.
        Result is between 0-360 degrees, where 0 is horizontal right. */
double angle_2pt (int r1, int c1, int r2, int c2)
{
        double x, dr, dc, conv;

        conv = 180.0/3.1415926535;
        dr = (r2-r1); dc = (c2-c1);

/*   Compute the raw angle based of Drow, Dcolumn          */
        if (dr==0 && dc == 0) x = 0.0;
        else if (dc == 0) x = 90.0;
        else {
                x = fabs(atan (dr/dc));
                x = x * conv;
        }

/*   Adjust the angle according to the quadrant          */
        if (dr <= 0) {                  /* upper 2 quadrants */
          if (dc < 0) x = 180.0 - x;   /* Left quadrant */
        } else if (dr > 0) {            /* Lower 2 quadrants */
          if (dc < 0) x = x + 180.0;   /* Left quadrant */
          else x = 360.0-x;            /* Right quadrant */
        }

        return x;
}

/* Draw a line from (x1,y1) to (x2,y2) with a grey level of OBJECT   */
void draw_line (struct image *im, int x1, int y1, int x2, int y2)
{
        int  x, y, sigx, sigy;
        int absx, absy, d, dx, dy;
        int True = 1;

        dx = x2-x1;
        if (dx < 0) {
          absx = -dx;  sigx = -1;
        } else {
          absx = dx;   sigx = 1;
        }
        absx = absx << 1;

        dy = y2-y1;
        if (dy < 0) {
          absy = -dy;    sigy = -1;
        } else {
          absy = dy;     sigy = 1;
        }
        absy = absy << 1;
```

```
        x = x1; y = y1;
        if (absx > absy) {
         d = absy-(absx>>1);
         while (True) {
                im->band1[x][y] = OBJECT;
                if (x==x2) return;
                if (d>=0) {
                 y += sigy;
                 d -= absx;
                }
                x += sigx;
                d += absy;
         }
        } else {
         d = absx-(absy>>1);
         while (True) {
                im->band1[x][y] = OBJECT;
                if (y==y2) return;
                if (d>=0) {
                 x += sigx;
                 d -= absy;
                }
                y += sigy;
                d += absx;
         }
        }
}

/* Check pixel (x,y) in which of the 32 subregions ( 1 to 32) divided by
   lines a[i]*x + b[i]*y + c[i] = 0, i = 0,... 15, return the No. */
int locate_region(int x, int y, double *a1, double *b1, double *c1, int orient)
{
        int i, m;
        double a[34], b[34], c[34];

        for (i=1; i<17; i++){
         a[i] = a1[i]; b[i] = b1[i]; c[i] = c1[i];
        }
        for (i=17; i<33; i++){
         a[i] = -a1[i-16]; b[i] = -b1[i-16]; c[i] = -c1[i-16];
        }

        a[33] = a1[1]; b[33] = b1[1]; c[33] = c1[1];

        for (i=1; i<33; i ++)
         if (a[i]*x+b[i]*y+c[i] >= 0 && a[i+1]*x+b[i+1]*y+c[i+1] < 0) break;

        switch (orient){
         case 1:
                m = i-1;
         break;
         case 2:
```

```
                if (i<17) m = 16-i;
                else  m = 48-i;
          break;
          case 3:
                if (i<17) m = 15+i;
                else  m = i-17;
          break;
          case 4:
                m = 32-i;
          break;
         }

         return m;
 }

/* Computer the calibration scales from coin image X, and return the row and colum scales */
double get_scale(struct image *x)
{
         int t, n, ibegin, m, rxy[4], error;
         double s;

         error = 0;

/*  Threshold the red band image to get a binary image C */
         thresh_is (x, &t,&error);
         if (error) return 0.0;
         threshold (x, t, &error);
         if (error) return 0.0;
//  disp_image(x,0,&error);

/*  Mark each seperated regions, ignore very small regions, and fill holes
         in any regions to get a labelled image C.*/
         error = 0; n = 0; ibegin = 0; m = 0;
  /* n: no.of marked regions,m:no. of pixels in a region.
         ibegin: the first row of the last marked region     */
         while (error == 0) {
          region_4 (x, n+1, &ibegin, &m, &error);
          if (error == NO_REGION) break;
          /* Ignore very small regions */
          if (m < 30) {
                del_reg (x, n+1, &error);
                if (error) return 0.0;
                continue;
          }

/*  Fill holes in the region marked n+1, and return the coordinates
         of the region in rxy array.                    */
          fill_holes (x, n+1, rxy, &error);
          if (error) return 0.0;
          n++;
         }
```

```
        if (n > 2) return 0.0;

        s = (double)(rxy[2] + rxy[3] - rxy[0] - rxy[1])/2.0;
        s = COIN_DIAMETER_IM_MM /s;

        return s;
}

/* Transfer rectangular pixel image to square pixel image */
void rectangular_to_square (struct image *x, struct image **y, int *error_code)
{
        int i, j, k, nc;
        double t, f;

        nc = (int)floor( (double)(x->nc) * PIX_ASP_RATIO );
        *y = new_image (x->nr, nc, x->color, error_code);
        if (*error_code) return;

        for(j=0; j < (*y)->nc; j++){
         t = (double)j/PIX_ASP_RATIO;
         k = (int)floor(t);
         f = t - (double)k;
         for(i=0; i < (*y)->nr; i++){
                (*y)->band1[i][j] = (unsigned char)((1-f)*(double)x->band1[i][k]
                                                                              +
f*(double)x->band1[i][k+1]);
                if (x->color){
                 (*y)->band2[i][j] = (unsigned char)((1-f)*(double)x->band2[i][k]
                                                                              +
f*(double)x->band2[i][k+1]);
                 (*y)->band3[i][j] = (unsigned char)((1-f)*(double)x->band3[i][k]
                                                                              +
f*(double)x->band3[i][k+1]);
                }
         }
        }
}

/* Determine the orientation of the germ part */
int orientation(struct image *x, double *a, double *b, double *c)
{
        int i, j, orien_ptr;
        int i1, j1, i2, j2, i3, j3, i4, j4;
        double dmax, d1, d2, d3, d4;

        /* Determine the orientation of the object and make the four phases divided by
          PA and MA, phase 1, 2, 3, & 4, in anti-clockwise direction, started with the
          up-right, be (+,-),(+,+),(-,+) & (-,-) */
        /* Remember (0,0) was on the positive side of PA and MA (c[1]>0 & c[0]>0) */
        if (c[1]>0 && a[1]<0 && b[1]<=0 && c[0]>0 && a[0]>=0 && b[0]<0) orien_ptr = 1;
        else if ( c[1]>0 && a[1]<0 && b[1]<0 &&
                        ((c[0]>0 && a[0]<0 && b[0]>0) || c[0] == 0 )){
```

```
                orien_ptr = 2;
                a[0] = - a[0]; b[0] = -b[0]; c[0] = -c[0];
               }
               else if (c[0]>0 && a[0]<0 && b[0]<=0 && c[1]>0 && a[1]>=0 && b[1]<0 ){
                              orien_ptr = 3;
                              a[1] = - a[1]; b[1] = -b[1]; c[1] = -c[1];
                          }
                          else orien_ptr = 4; //(c[0]>0 && a[0]<0 && b[1]<0 && (c[1]>0 && a[1]<0
&& b[1]>0)||c[1]==0))

        /* Find the inersection of PA and MA with object boundary
          (i1,j1),(i2,j2),(i3,j3),(i4,j4)                    */
        ln_obj_intersec(x, a[1], b[1], c[1], &i1, &j1, &i2, &j2);
        ln_obj_intersec(x, a[0], b[0], c[0], &i3, &j3, &i4, &j4);

        /* Make (i1,j1) on left(negative) side & (i2,j2) on right(positive) side of MA */
        /* Make (i3,j3) on up(positive) side & (i4,j4) on down(negative) side of  MA */
        if(a[0]*i1 + b[0]*j1 +c[0] > 0){
         i = i1; j = j1;
         i1 = i2; j1 = j2;
         i2 = i; j2 = j;
        }

        if(a[1]*i3 + b[1]*j3 +c[1] < 0){
         i = i3; j = j3;
         i3 = i4; j3 = j4;
         i4 = i; j4 = j;
        }

        /* Calculate the distances between  (i1,j1) & (i3,j3), d1,
                                            (i2,j2) & (i3,j3), d2,
                                            (i2,j2) & (i4,j4), d3,
                                            (i1,j1) & (i4,j4), d4.    */
        d1 = dist_2pt(i1, j1, i3, j3);
        d2 = dist_2pt(i2, j2, i3, j3);
        d3 = dist_2pt(i2, j2, i4, j4);
        d4 = dist_2pt(i1, j1, i4, j4);

        /* Determine which of d1, d2, d3, and d4 is the longest */
        dmax = max(max(d1, d2), max(d3, d4));
        if (dmax == d1) orien_ptr = 1;
        else if (dmax == d2) orien_ptr = 2;
                else if (dmax == d3) orien_ptr =3;
                        else orien_ptr =4;

        return orien_ptr;
}

/* Compare two numbers, a and b, and return the bigger one */
double max(double a, double b)
{
        double c;
```

```
        c = a;
        if (a < b) c = b;

        return c;
}

/* Compare two numbers, a and b, and return the smaller one */
double min(double a, double b)
{
        double c;

        c = a;
        if (a > b) c = b;

        return c;
}

/* Calculate magnitudes of Fourior transformations of a ln-dimension data vetor F */
void fft(double *f, int ln)
{
        int i, j, k, l;
        int n, nv2, nm1, le, le1, ip;
        double t, tr, ti, ur, ui, wr, wi;
        double fr[256], fi[256];

        n = (int)pow(2.0, (double)ln);
        nv2 = n/2;
        nm1 = n-1;

        j = 0;
        for (i = 0; i < nm1; i ++){
                if (i <= j){
                 t = f[j];
                 f[j] = f[i];
                 f[i] = t;
                }
                k = nv2;
                while (k <= j){
                 j = j - k;
                 k = k / 2;
                }
                j = j + k;
        }

        for (i = 0; i < n; i ++) fr[i] = f[i];

        for (l = 0; l < ln; l ++){
                le = (int)pow(2.0, (double)(l+1));
                le1 = le/2;
                ur = 1.0; ui = 0.0;
                wr = cos(PI/le1); wi = -sin(PI/le1);
```

```
		for (j = 0; j<le1; j ++)
			for (i = j; i < n; i += le){
				ip = i + le1;
				tr = fr[ip]*ur - fi[ip]*ui;
				ti = fr[ip]*ui + fi[ip]*ur;
				fr[ip] = fr[i] - tr;
				fi[ip] = fi[i] - ti;
				fr[i] = fr[i] + tr;
				fi[i] = fi[i] + ti;
			}
			ur = ur*wr - ui*wi;
			ui = ur*wi + ui*wr;
	}

	for (i = 0; i < n; i ++){
		fr[i] = fr[i] / (double)n;
		fi[i] = fi[i] / (double)n;
		f[i] = sqrt(fr[i]*fr[i] + fi[i]*fi[i]);
	}

	return;
}

/* Compute bulk image features which include means, variances and
  histograms of R, G, & B values */
void bulk_feature(struct image *x, struct bfeature *bf, int n, int *error_code)
{
	int j, k, n1, n2;
	long hist[256];
	double width, np, t;
	double r, g, b;
	double r1, g1, b1;
	double h, s, i;
	struct image *y;

	if (!x->color){
	 *error_code = BAD_ARGUMENT2;
	 return;
	}

	width = 256.0/(double)n;

	for (j=0; j<n; j++){
	 bf->histR[j] = 0;
	 bf->histG[j] = 0;
	 bf->histB[j] = 0;
	}

	for (j=0; j<256; j++) hist[j] =0;

	bf->meanR = 0.0;
	bf->meanG = 0.0;
```

```
        bf->meanB = 0.0;
        bf->meanR3G2B1 = 0.0;
        bf->meanH = 0.0;
        bf->meanS = 0.0;
        bf->meanI = 0.0;
        bf->varR = 0.0;
        bf->varG = 0.0;
        bf->varB = 0.0;
        bf->varR3G2B1 = 0.0;
        bf->varH = 0.0;
        bf->varS = 0.0;
        bf->varI = 0.0;
        n1 = 0;
        n2 = 0;

        for (j=0; j<x->nr; j++)
         for (k=0; k<x->nc; k++) {

/* Read in RGB grey-level values */
                r1 = (double)(x->band1[j][k]);
                g1 = (double)(x->band2[j][k]);
                b1 = (double)(x->band3[j][k]);

                bf->histR[(int)(r1/width)] = bf->histR[(int)(r1/width)] + 1.0;
                bf->histG[(int)(g1/width)] = bf->histG[(int)(g1/width)] + 1.0;
                bf->histB[(int)(b1/width)] = bf->histB[(int)(b1/width)] + 1.0;

                hist[(int)r1] += 1;

/* Remove Gamma corrections and get normalized R,G, and B values*/
                r = exp((1.0/2.2)*log(1e-20+r1/WHITE));
                g = exp((1.0/2.2)*log(1e-20+g1/WHITE));
                b = exp((1.0/2.2)*log(1e-20+b1/WHITE));

/* Compute HSI values */
                i = (r + g + b) / 3.0;
                if (i == 0.0){
                 s = 0.0; h = 0.0;
                }
                else{
                 s = 1.0 - (min(min(r, g), b))/i;
                 if (s == 0.0) h = 0.0;
                 else{
                        t = sqrt((r-g)*(r-g)+(r-b)*(g-b)+1e-20);
                        h = acos( 0.5*(2.0*r-g-b)/t );
                 }
                }
                if (b > g) h = 2.0*PI - h;
                h = h / (2.0*PI);

                bf->meanR = bf->meanR + r;
                bf->meanG = bf->meanG + g;
```

```
            bf->meanB = bf->meanB + b;
            bf->meanR3G2B1 = bf->meanR3G2B1 + (3.0*r + 2.0*g + b)/6.0;
            bf->meanH = bf->meanH + h;
            bf->meanS = bf->meanS + s;
            bf->meanI = bf->meanI + i;
            bf->varR = bf->varR + r * r;
            bf->varG = bf->varG + g * g;
            bf->varB = bf->varB + b * b;
            bf->varR3G2B1 = bf->varR3G2B1 + (3.0*r + 2.0*g + b)*(3.0*r + 2.0*g + b)/36.0;
            bf->varH = bf->varH + h * h;
            bf->varS = bf->varS + s * s;
            bf->varI = bf->varI + i * i;
     }

     np = (double)x->nr * (double)x->nc;

     bf->meanR = bf->meanR/np;
     bf->meanG = bf->meanG/np;
     bf->meanB = bf->meanB/np;
     bf->meanR3G2B1 = bf->meanR3G2B1/np;
     bf->meanH = bf->meanH/np;
     bf->meanS = bf->meanS/np;
     bf->meanI = bf->meanI/np;
     bf->varR = (bf->varR - np*(bf->meanR)*(bf->meanR))/(np-1.0);
     bf->varG = (bf->varG - np*(bf->meanG)*(bf->meanG))/(np-1.0);
     bf->varB = (bf->varB - np*(bf->meanB)*(bf->meanB))/(np-1.0);
     bf->varR3G2B1 = (bf->varR3G2B1 - np*(bf->meanR3G2B1)*(bf->meanR3G2B1))/(np-1.0);
     bf->varH = (bf->varH - np*(bf->meanH)*(bf->meanH))/(np-1.0);
     bf->varS = (bf->varS - np*(bf->meanS)*(bf->meanS))/(np-1.0);
     bf->varI = (bf->varI - np*(bf->meanI)*(bf->meanI))/(np-1.0);

     for (j=0; j<n; j ++){
      bf->histR[j] = bf->histR[j] / np;
      bf->histG[j] = bf->histG[j] / np;
      bf->histB[j] = bf->histB[j] / np;
     }

/* get the thresholding level j */
     t = 0;
     for (j=0; j<256; j ++){
      t = t + (double)hist[j];
      if (t >= 3.0*np/10.0) break;
     }
 /* Copy red band of image X to image Y */
     y = 0;
     copy_image (x, &y, 1, error_code);
     if (*error_code) return;

// printf ("threshold is %d", j);
// _getch();

     threshold(y, (int)j, error_code);
```

```
        if (*error_code) return;

//  disp_image(y,0,error_code);

        for (j=0; j<(y->nr); j ++)
          for (k=0; k<(y->nc-1); k ++)
                if ((y->band1[j][k] == OBJECT) && (y->band1[j][k+1] == BACKGROUND)) n1 ++;

        for (k=0; k<(y->nc); k ++)
          for (j=0; j<(y->nr-1); j ++)
                if ((y->band1[j][k] == OBJECT) && (y->band1[j+1][k] == BACKGROUND)) n2 ++;

        bf->kn = 0.5*((double)n1 + (double)n2)/np;

        free_image (y, error_code);
        if (*error_code) return;
}
/* Write calculated bulk features to a output file */
void write_bf(FILE *outfp, struct bfeature *bf, char *img)
{
 fprintf(outfp, "Image ");
 fprintf(outfp, "meanR meanG meanB meanR3G2B1 varR varG varB varR3G2B1 ");
 fprintf(outfp, "meanH meanS meanI varH varS varI Kn ");

 fprintf(outfp, "histR[0] histR[1] histR[2] histR[3] ");
 fprintf(outfp, "histR[4] histR[5] histR[6] histR[7] ");
 fprintf(outfp, "histR[8] histR[9] histR[10] histR[11] ");
 fprintf(outfp, "histR[12] histR[13] histR[14] histR[15] ");
 fprintf(outfp, "histR[16] histR[17] histR[18] histR[19] ");
 fprintf(outfp, "histR[20] histR[21] histR[22] histR[23] ");
 fprintf(outfp, "histR[24] histR[25] histR[26] histR[27] ");
 fprintf(outfp, "histR[28] histR[29] histR[30] histR[31] ");

 fprintf(outfp, "histG[0] histG[1] histG[2] histG[3] ");
 fprintf(outfp, "histG[4] histG[5] histG[6] histG[7] ");
 fprintf(outfp, "histG[8] histG[9] histG[10] histG[11] ");
 fprintf(outfp, "histG[12] histG[13] histG[14] histG[15] ");
 fprintf(outfp, "histG[16] histG[17] histG[18] histG[19] ");
 fprintf(outfp, "histG[20] histG[21] histG[22] histG[23] ");
 fprintf(outfp, "histG[24] histG[25] histG[26] histG[27] ");
 fprintf(outfp, "histG[28] histG[29] histG[30] histG[31] ");

 fprintf(outfp, "histB[0] histB[1] histB[2] histB[3] ");
 fprintf(outfp, "histB[4] histB[5] histB[6] histB[7] ");
 fprintf(outfp, "histB[8] histB[9] histB[10] histB[11] ");
 fprintf(outfp, "histB[12] histB[13] histB[14] histB[15] ");
 fprintf(outfp, "histB[16] histB[17] histB[18] histB[19] ");
 fprintf(outfp, "histB[20] histB[21] histB[22] histB[23] ");
 fprintf(outfp, "histB[24] histB[25] histB[26] histB[27] ");
 fprintf(outfp, "histB[28] histB[29] histB[30] histB[31] \n");

 fprintf(outfp, "%s ", img);
```

```
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->meanR,  bf->meanG,  bf->meanB, bf->meanR3G2B1,
                            bf->varR, bf->varG, bf->varB, bf->varR3G2B1);
 fprintf(outfp, "%f %f %f %f %f %f %f ",
                            bf->meanH,  bf->meanS,  bf->meanI,
                            bf->varH, bf->varS, bf->varI, bf->kn);

 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histR[0], bf->histR[1], bf->histR[2], bf->histR[3],
                            bf->histR[4], bf->histR[5], bf->histR[6], bf->histR[7]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histR[8], bf->histR[9],bf->histR[10], bf->histR[11],
                            bf->histR[12],bf->histR[13],bf->histR[14],bf->histR[15]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histR[16],bf->histR[17],bf->histR[18],bf->histR[19],
                            bf->histR[20],bf->histR[21],bf->histR[22],bf->histR[23]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histR[24],bf->histR[25],bf->histR[26],bf->histR[27],
                            bf->histR[28],bf->histR[29],bf->histR[30],bf->histR[31]);

 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histG[0], bf->histG[1], bf->histG[2], bf->histG[3],
                            bf->histG[4], bf->histG[5], bf->histG[6], bf->histG[7]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histG[8], bf->histG[9], bf->histG[10],bf->histG[11],
                            bf->histG[12],bf->histG[13],bf->histG[14],bf->histG[15]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histG[16],bf->histG[17],bf->histG[18],bf->histG[19],
                            bf->histG[20],bf->histG[21],bf->histG[22],bf->histG[23]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histG[24],bf->histG[25],bf->histG[26],bf->histG[27],
                            bf->histG[28],bf->histG[29],bf->histG[30],bf->histG[31]);

 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histB[0], bf->histB[1], bf->histB[2], bf->histB[3],
                            bf->histB[4], bf->histB[5], bf->histB[6], bf->histB[7]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histB[8], bf->histB[9],bf->histB[10],bf->histB[11],
                            bf->histB[12],bf->histB[13],bf->histB[14],bf->histB[15]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f ",
                            bf->histB[16],bf->histB[17],bf->histB[18],bf->histB[19],
                            bf->histB[20],bf->histB[21],bf->histB[22],bf->histB[23]);
 fprintf(outfp, "%f %f %f %f %f %f %f %f \n",
                            bf->histB[24],bf->histB[25],bf->histB[26],bf->histB[27],
                            bf->histB[28],bf->histB[29],bf->histB[30],bf->histB[31]);
}

/* Write individual features' name to an output file */
void write_fname(FILE *outfp)
{
 fprintf(outfp, "Image Object Area Peri Leng Width Lpa Wma Rmin Rmax Rmean VarR ");
```

```
fprintf(outfp, "AspR RecR RadR ThinR AreaR HarR ");

fprintf(outfp, "lengR[1] lengR[2] lengR[3] lengR[4] lengR[5] lengR[6] lengR[7] ");
fprintf(outfp, "widthR[1] widthR[2] widthR[3] widthR[4] widthR[5] widthR[6] widthR[7] ");

fprintf(outfp, "radR[1] radR[2] radR[3] radR[4] radR[5] radR[6] radR[7] radR[8] ");
fprintf(outfp, "radR[9] radR[10] radR[11] radR[12] radR[13] radR[14] radR[15] radR[16] ");
fprintf(outfp, "radR[17] radR[18] radR[19] radR[20] radR[21] radR[22] radR[23] radR[24] ");
fprintf(outfp, "radR[25] radR[26] radR[27] radR[28] radR[29] radR[30] radR[31] radR[32] ");

fprintf(outfp, "areaR[1] areaR[2] areaR[3] areaR[4] areaR[5] areaR[6] areaR[7] areaR[8] ");
fprintf(outfp, "areaR[9] areaR[10] areaR[11] areaR[12] areaR[13] areaR[14] areaR[15] areaR[16] ");
fprintf(outfp, "areaR[17] areaR[18] areaR[19] areaR[20] areaR[21] areaR[22] areaR[23] areaR[24] ");
fprintf(outfp, "areaR[25] areaR[26] areaR[27] areaR[28] areaR[29] areaR[30] areaR[31] areaR[32] ");

fprintf(outfp, "periR[1] periR[2] periR[3] periR[4] periR[5] periR[6] periR[7] periR[8] ");
fprintf(outfp, "periR[9] periR[10] periR[11] periR[12] periR[13] periR[14] periR[15] periR[16] ");
fprintf(outfp, "periR[17] periR[18] periR[19] periR[20] periR[21] periR[22] periR[23] periR[24] ");
fprintf(outfp, "periR[25] periR[26] periR[27] periR[28] periR[29] periR[30] periR[31] periR[32] ");

fprintf(outfp, "meanR meanG meanB meanR3G2B1 varR varG varB varR3G2B1 ");
fprintf(outfp, "meanH meanS meanI varH varS varI ");

fprintf(outfp, "histR[0] histR[1] histR[2] histR[3] ");
fprintf(outfp, "histR[4] histR[5] histR[6] histR[7] ");
fprintf(outfp, "histR[8] histR[9] histR[10] histR[11] ");
fprintf(outfp, "histR[12] histR[13] histR[14] histR[15] ");

fprintf(outfp, "histG[0] histG[1] histG[2] histG[3] ");
fprintf(outfp, "histG[4] histG[5] histG[6] histG[7] ");
fprintf(outfp, "histG[8] histG[9] histG[10] histG[11] ");
fprintf(outfp, "histG[12] histG[13] histG[14] histG[15] ");

fprintf(outfp, "histB[0] histB[1] histB[2] histB[3] ");
fprintf(outfp, "histB[4] histB[5] histB[6] histB[7] ");
fprintf(outfp, "histB[8] histB[9] histB[10] histB[11] ");
fprintf(outfp, "histB[12] histB[13] histB[14] histB[15] \n");
}

/* Write calculated individual features to a output file */
void write_feature(FILE *outfp, struct feature *objf, char *img, int i)
{
	fprintf(outfp,"%s %d %f %f %f %f %f %f %f %f %f %f ", img, i+1,
			objf->area, objf->perimeter,
			objf->length, objf->width, objf->lpa, objf->wma,
			objf->rmin, objf->rmax, objf->rmean, objf->var_r);

	fprintf(outfp,"%f %f %f %f %f %f ",
			objf->asp_R, objf->rec_R, objf->rad_R,
			objf->thin_R, objf->area_R, objf->har_R);

/*	fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
```

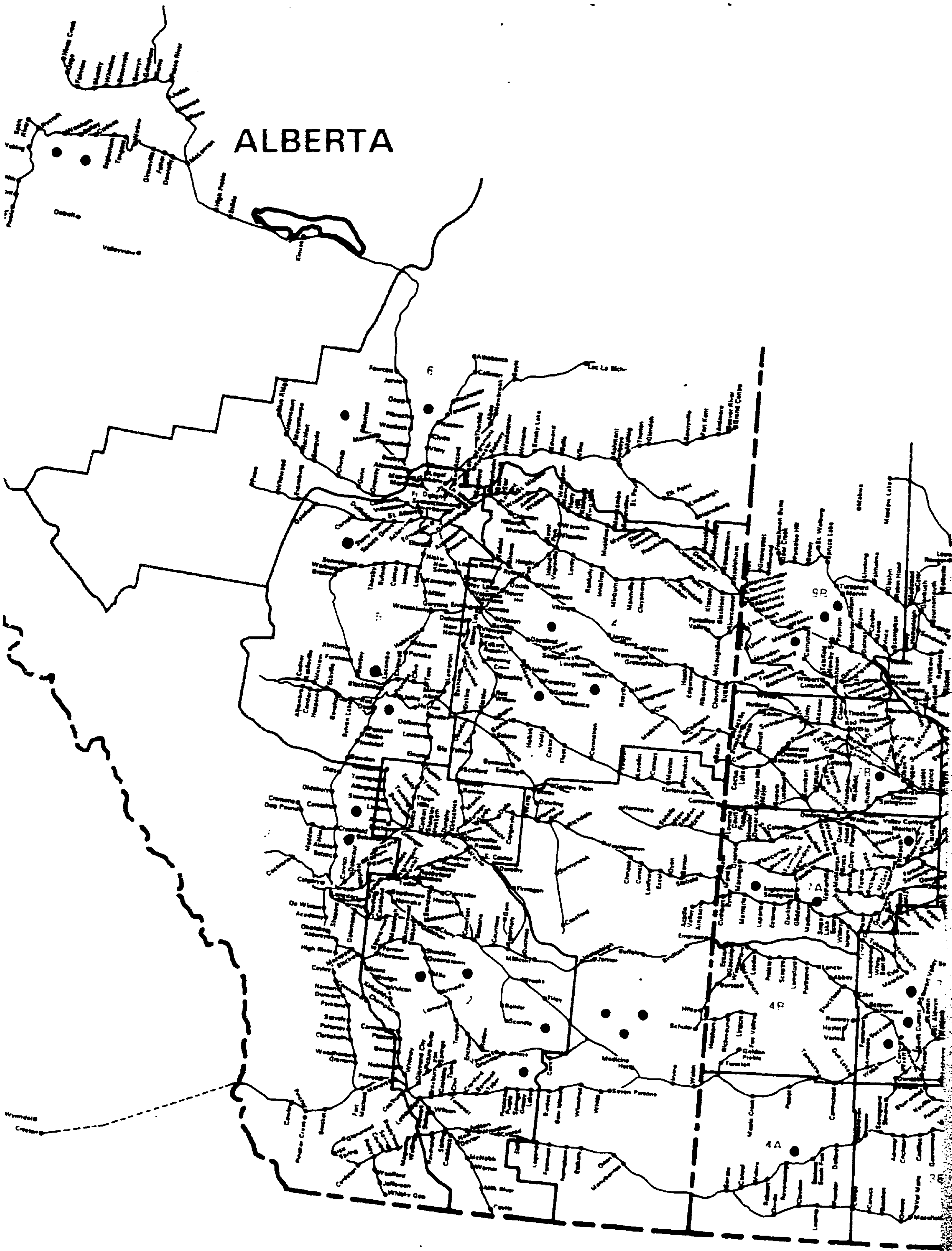
```
                    objf->lwR[0], objf->lwR[1], objf->lwR[2], objf->lwR[3],
                    objf->lwR[4], objf->lwR[5], objf->lwR[6], objf->lwR[7],
                    objf->lwR[8], objf->lwR[9], objf->lwR[10], objf->lwR[11],
                    objf->lwR[12], objf->lwR[13]);

  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->radR[0], objf->radR[1], objf->radR[2], objf->radR[3],
                    objf->radR[4], objf->radR[5], objf->radR[6], objf->radR[7],
                    objf->radR[8], objf->radR[9], objf->radR[10],objf->radR[11],
objf->radR[12],objf->radR[13],objf->radR[14],objf->radR[15]);

  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->radR[16],objf->radR[17],objf->radR[18],objf->radR[19],
                    objf->radR[20],objf->radR[21],objf->radR[22],objf->radR[23],
                    objf->radR[24],objf->radR[25],objf->radR[26],objf->radR[27],
                    objf->radR[28],objf->radR[29],objf->radR[30],objf->radR[31]);
  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->areaR[0], objf->areaR[1], objf->areaR[2], objf->areaR[3],
                    objf->areaR[4], objf->areaR[5], objf->areaR[6], objf->areaR[7],
                    objf->areaR[8], objf->areaR[9], objf->areaR[10],objf->areaR[11],
objf->areaR[12],objf->areaR[13],objf->areaR[14],objf->areaR[15]);

  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->areaR[16],objf->areaR[17],objf->areaR[18],objf->areaR[19],
                    objf->areaR[20],objf->areaR[21],objf->areaR[22],objf->areaR[23],
                    objf->areaR[24],objf->areaR[25],objf->areaR[26],objf->areaR[27],
                    objf->areaR[28],objf->areaR[29],objf->areaR[30],objf->areaR[31]);

  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->perimR[0], objf->perimR[1], objf->perimR[2], objf->perimR[3],
                    objf->perimR[4], objf->perimR[5], objf->perimR[6], objf->perimR[7],
                    objf->perimR[8], objf->perimR[9], objf->perimR[10],objf->perimR[11],
objf->perimR[12],objf->perimR[13],objf->perimR[14],objf->perimR[15]);

  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->perimR[16],objf->perimR[17],objf->perimR[18],objf->perimR[19],
                    objf->perimR[20],objf->perimR[21],objf->perimR[22],objf->perimR[23],
                    objf->perimR[24],objf->perimR[25],objf->perimR[26],objf->perimR[27],
                    objf->perimR[28],objf->perimR[29],objf->perimR[30],objf->perimR[31]); */


  fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f \n",
                    objf->meanR,  objf->meanG,  objf->meanB,
                    objf->varR,   objf->varG,   objf->varB,
                    objf->meanH,  objf->meanS,  objf->meanI,
                    objf->varH,   objf->varS,   objf->varI);

/*   fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                    objf->histR[0], objf->histR[1], objf->histR[2], objf->histR[3],
                    objf->histR[4], objf->histR[5], objf->histR[6], objf->histR[7],
                    objf->histR[8], objf->histR[9], objf->histR[10], objf->histR[11],
                    objf->histR[12], objf->histR[13], objf->histR[14], objf->histR[15]);
```

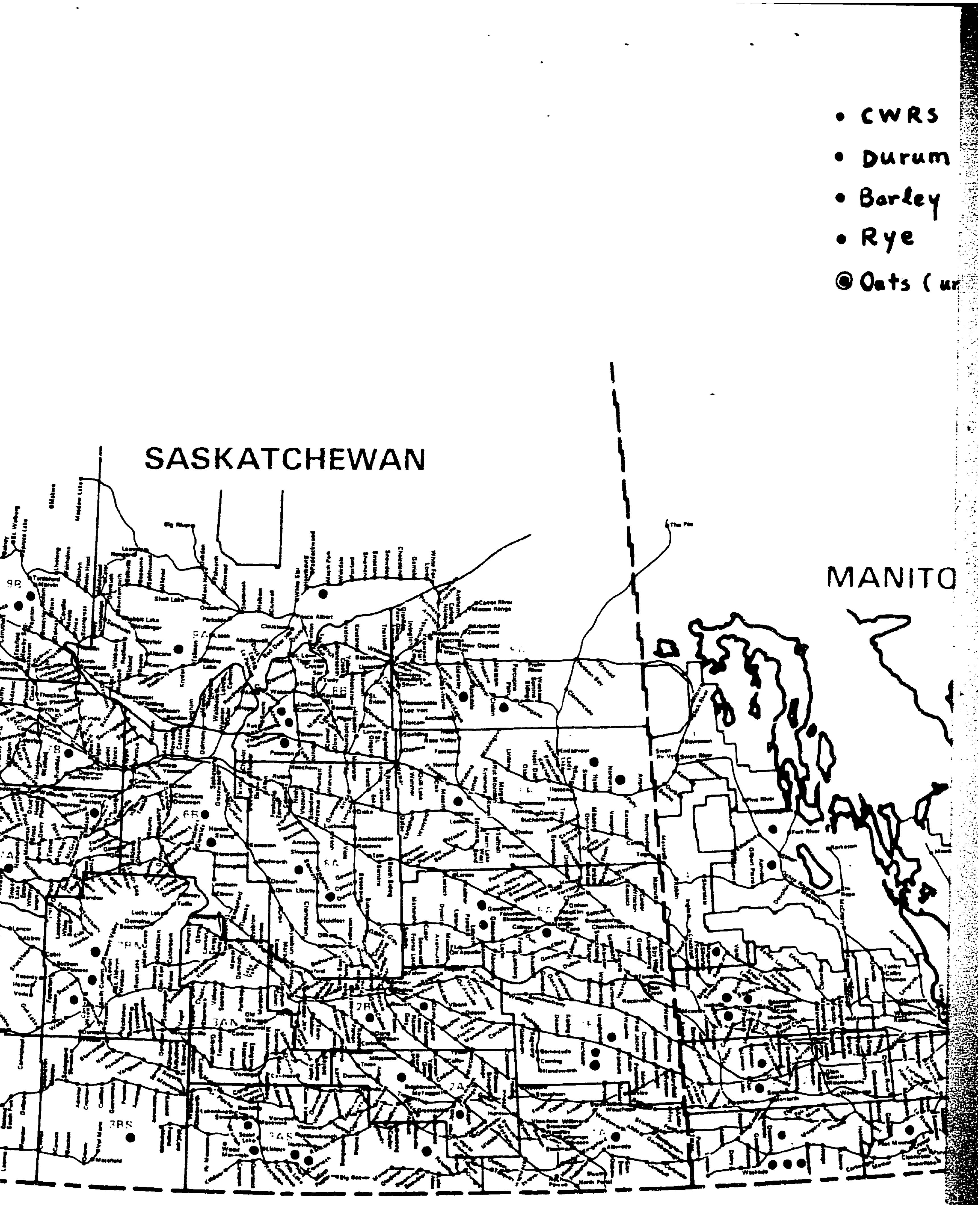
```
        fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f ",
                        objf->histG[0], objf->histG[1], objf->histG[2], objf->histG[3],
                        objf->histG[4], objf->histG[5], objf->histG[6], objf->histG[7],
                        objf->histG[8], objf->histG[9], objf->histG[10], objf->histG[11],
                        objf->histG[12], objf->histG[13], objf->histG[14], objf->histG[15]);

        fprintf(outfp,"%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f \n",
                        objf->histB[0], objf->histB[1], objf->histB[2], objf->histB[3],
                        objf->histB[4], objf->histB[5], objf->histB[6], objf->histB[7],
                        objf->histB[8], objf->histB[9], objf->histB[10], objf->histB[11],
                        objf->histB[12], objf->histB[13], objf->histB[14], objf->histB[15]); */
  }

/* Print corresponding error information on screen according the error code */
void an_error (int ecode)
{
printf("\n- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");
printf(" - - - - - - - - - - - - - - - - - - - - - - - - - - - - -\n");
printf ("\n          UM error # %3d\n", ecode);
switch (ecode) {
case BAD_IMAGE_SIZE:
        printf ("Specified image size is illegal.\n");
        break;
case OUT_OF_STORAGE:
        printf ("Cannot allocate any more storage.\n");
        break;
case CANNOT_OPEN_FILE:
        printf ("Cannot open the specified file.\n");
        break;
case BAD_DESCRIPTOR1:
        printf ("This is not an UM format image file.\n");
        break;
case BAD_NR_NC:
        printf ("Size specified in the file is illegal.\n");
        break;
case FILE_TOO_SHORT:
        printf ("Data is missing from the image file.\n");
        break;
case BAD_DESCRIPTOR2:
        printf ("Synchronization error in image file.\n");
        break;
case NO_REGION:
        printf ("Operator needs a region - none was found with this value.\n");
        break;
case REGION_INT_BOUND:
        printf ("The region intersects the image boundary.\n");
        break;
case INTERNAL_1:
        printf ("INTERNAL ERROR: Should not occur. \n");
        break;
case BAD_IMAGE_COORD:
        printf ("Specified pixel coordinates lie outside of the image.\n");
```

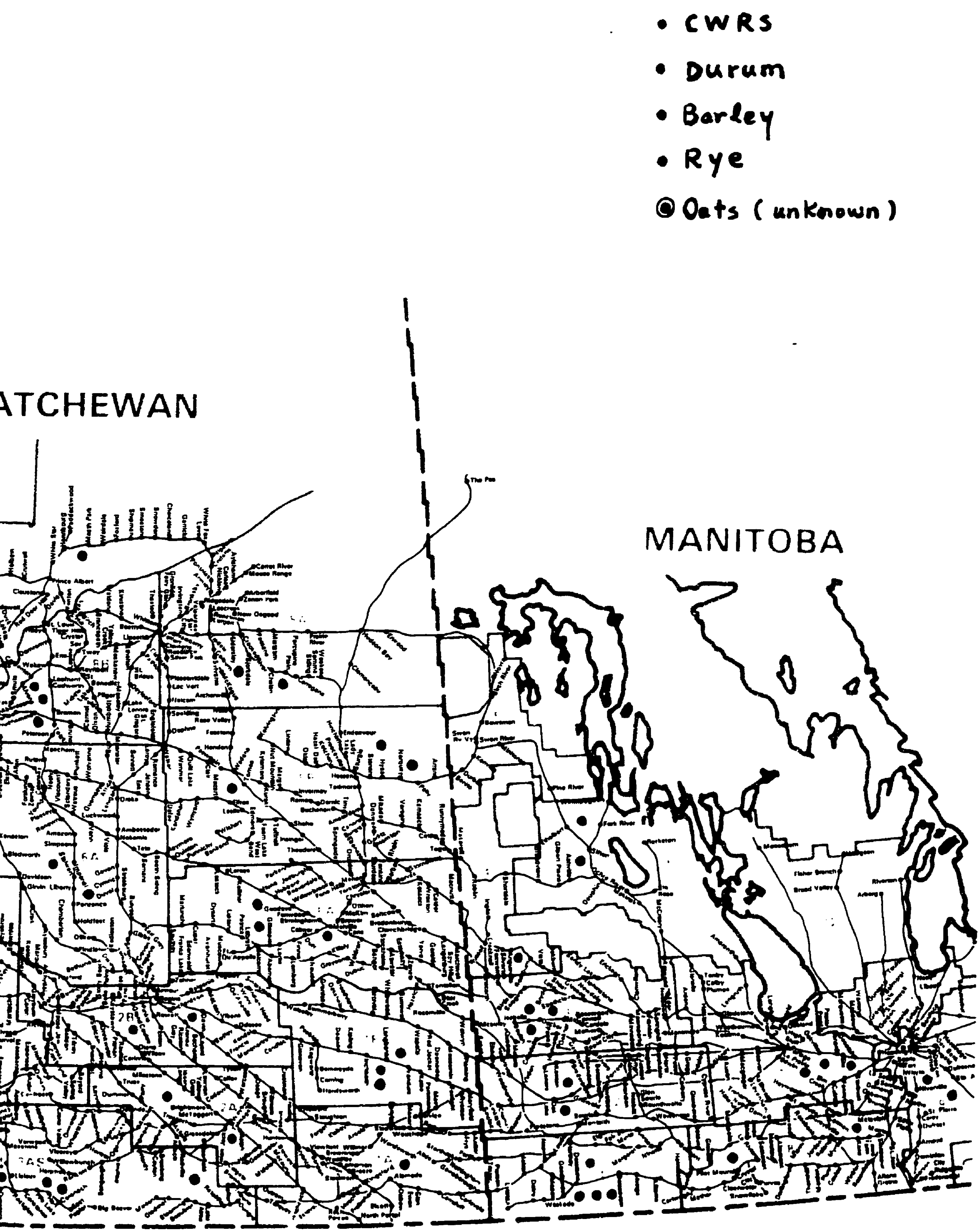
```
        break;
 case NO_RESULT:
        printf ("Can't compute a result for this operation.\n");
        break;
 case IMPOSSIBLE_CLASS:
        printf ("A class number is out of range. Are all classes defined?\n");
        break;
 case TOO_MANY_CLASSES:
        printf ("The standard system allows 200 classes only.\n");
        break;
 case TOO_MANY_EDGES:
        printf ("An internal limit for number of edges has been reached.\n");
        break;
 case BAD_COLOR_MAP:
        printf ("The color map has been omitted or corrupted.\n");
        break;
 case IO_ERROR:
        printf ("An Input/Output error has occurred.\n");
        break;
 case BAD_ARGUMENT1:
        printf ("Band should be 0,1,2, or 3.\n");
        break;
 case BAD_ARGUMENT2:
        printf ("Error: Performing color operations on a grey(single band) image.\n");
        break;
 case BAD_ARGUMENT3:
        printf ("Error: The operation requires a grey(single band) image.\n");
        break;
 case NO_OR_TOO_MANY_REGIONS:
        printf ("No or too many regions.\n");
        break;
 case BAD_FEATURE_SIZE:
        printf ("Feature size should be larger than 1 and less than 100.\n");
        break;
 case CANNOT_GET_CALIBR_SCALE:
        printf ("Cannot get calibration scale for some reason.\n");
        break;
 case GET_USAGE:
        printf ("This program needs 3 arguments.\n");
        break;
default:    printf ("Unknown error code : %d.\n", ecode);
}
printf("\n------------------------------\n");
printf(" ------------------------------\n");
}
```

Appendix B

GRAIN SAMPLE DISTRIBUTION


ALBERTA


CWRS
Durum
Barley
Rye
Oats (ur
SASKATCHEWAN
MANITO



• CWRS
• Durum
• Barley
• Rye
◉ Oats (unknown)
ATCHEWAN
MANITOBA

Appendix C

OPERATION GUIDE FOR GRAIN IMAGING

OPERATION GUIDE FOR GRAIN IMAGING ON "GADGET" --- A COLOR COMPUTER VISION SYSTEM

CAMERA (DXC-3000A) SETTINGS:

Focus ring	max. position
F.f adjustment ring	fixed position
MACRO (close-up) ring	max. position in the direction of the arrow
ZOOM selector	S
Iris selector	A
ABL switch	OFF
VTR selector	1
FILTER selector	1
BARS/WB selector	AUTO
Gain selector	0dB
POWER VF PREHEAT	ON

CAMERA CONTROL UNIT (CCU-M3) SETTINGS:

LOCK switch	OFF
PHASE selector	0°
MODE selector	CAMERA
GAIN selector	0dB
W/B BALANCE selector	AUTO
IRIS selector	MANUAL
MASTER PEDESTAL	zero position

SYSTEM ADJUSTMENT

1. Turn the computer (Gadget), the camera adaptor (CMA-8), the camera control unit(CCU-M3), and the illumination chamber on.
2. Set the power voltage to the illumination chamber at 120.00 V and wait for 30 min. to allow the system to be stable.
3. Login to Gadget and go to directory c:\imaging
4. Run the program "passthru" to put the image on the color video monitor.
5. Focus the camera at a quarter coin by adjusting the motorized zoom switch.
6. Freeze the coin image (Exit from the program"passthru").
7. Point the camera at the white side of the Kodak gray card.
8. Perform black balance (press the BLACK switch under W/B BALANCE on the camera control unit).

9. Make sure that the power voltage to the light box is 120 (+-) 0.1 V.
10. Run the program "litadj" on the computer with the parameters: desired grey level = 250, allowable error = 1.
11. Adjust the IRIS control knob (if necessary) on the camera control unit to get the desired green grey level (250) (when ready, the program will automatically come out).
12. Perform white balance on the white side of the Kodak gray card (press the WHITE switch under W/B BALANCE on the camera control unit).

Now the system is ready to take images.

Don't change any setting on the camera and camera control unit !

TO SAVE THE RESOLUTION INFORMATION

13. Put a quarter coin with a black background under the camera.
14. Make sure that the power voltage to the light box is 120 (+-) 0.1 V.
15. Run the program "xvsave" on the computer to save the coin image in the name of **coinmmdd.xv** for the future use of the spatial calibration.
    Comment: coin quarter image for space calibration, mm/dd/year.

TO TAKE A GRAIN IMAGE

16. Put a grain sample under the camera.
17. Make sure that the power voltage to the light box is 120 (+-) 0.1 V.
18. Run the program "xvsave" on the computer to save the image in the name of ***??.xv** and supply the related information in the comment by indicating the type of grains (such as HRSW), grade, growing location, bulk/sep., M.C, and the corresponding calibration file "**coin_?.xv**".

Repeat procedures 7 to 12 prior to taking each image to adjust the system to the illumination change with time.

Save the resolution information in the image files coin?.xv every working unit.

APPENDIX D-1

STEPDISC ANALYSIS OF KERNEL FEATURES FOR GRAIN TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL KERNELS

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda	Average Squared Canonical Correlation	Prob > ASCC
1	O3		1	0.8886	83785.345	0.0001	0.11135229	0.0001	0.22216193	0.0001
2	A13		2	0.5433	12490.605	0.0001	0.05085157	0.0001	0.33785910	0.0001
3	O10		3	0.5807	14541.236	0.0001	0.02132042	0.0001	0.47371044	0.0001
4	O15		4	0.3231	5010.180	0.0001	0.01443250	0.0001	0.53301804	0.0001
5	O8		5	0.1830	2352.149	0.0001	0.01179066	0.0001	0.55559393	0.0001
6	O7		6	0.1255	1506.013	0.0001	0.01031135	0.0001	0.57080105	0.0001
7	O16		7	0.1564	1946.558	0.0001	0.00869837	0.0001	0.58906181	0.0001
8	O9		8	0.0899	1036.935	0.0001	0.00791636	0.0001	0.59742142	0.0001
9	R1		9	0.0707	798.376	0.0001	0.00735681	0.0001	0.60568792	0.0001
10	P13		10	0.0543	603.068	0.0001	0.00695709	0.0001	0.61112058	0.0001
11	O18		11	0.0484	533.489	0.0001	0.00662059	0.0001	0.61574952	0.0001
12	O4		12	0.0520	575.171	0.0001	0.00627664	0.0001	0.62091703	0.0001
13	A4		13	0.0442	484.913	0.0001	0.00599946	0.0001	0.62447083	0.0001
14	A7		14	0.0364	396.246	0.0001	0.00578119	0.0001	0.62729273	0.0001
15	O2		15	0.0306	330.740	0.0001	0.00560457	0.0001	0.63101390	0.0001
16	O14		16	0.0325	352.506	0.0001	0.00542245	0.0001	0.63403725	0.0001
17	17		17	0.0506	559.055	0.0001	0.00514820	0.0001	0.63950315	0.0001
18	R16		18	0.0289	312.568	0.0001	0.00499930	0.0001	0.64192459	0.0001
19	A15		19	0.0241	258.888	0.0001	0.00487894	0.0001	0.64349414	0.0001
20	R14		20	0.0235	252.909	0.0001	0.00476412	0.0001	0.64724258	0.0001
21	P4		21	0.0197	210.436	0.0001	0.00467046	0.0001	0.64882359	0.0001
22	A6		22	0.0179	190.903	0.0001	0.00458701	0.0001	0.64982710	0.0001
23	O12		23	0.0159	170.054	0.0001	0.00451386	0.0001	0.65077363	0.0001
24	O20		24	0.0143	152.207	0.0001	0.00444932	0.0001	0.65329834	0.0001
25	O19		25	0.0244	261.995	0.0001	0.00434093	0.0001	0.65567324	0.0001
26	R2		26	0.0152	161.414	0.0001	0.00427517	0.0001	0.65689425	0.0001
27	A11		27	0.0128	135.579	0.0001	0.00422063	0.0001	0.65819289	0.0001
28	P10		28	0.0130	138.065	0.0001	0.00416581	0.0001	0.65938331	0.0001
29	R6		29	0.0107	113.079	0.0001	0.00412139	0.0001	0.66101352	0.0001
30	O6		30	0.0080	84.532	0.0001	0.00408845	0.0001	0.66155799	0.0001
31	O1		31	0.0077	81.530	0.0001	0.00405692	0.0001	0.66234779	0.0001
32	P7		32	0.0072	75.916	0.0001	0.00402777	0.0001	0.66279749	0.0001
33	16		33	0.0067	70.697	0.0001	0.00400081	0.0001	0.66324911	0.0001
34	R5		34	0.0063	66.037	0.0001	0.00397579	0.0001	0.66373031	0.0001
35	R13		35	0.0062	65.788	0.0001	0.00395101	0.0001	0.66425265	0.0001
36	P11		36	0.0043	45.336	0.0001	0.00393401	0.0001	0.66448696	0.0001
37	P6		37	0.0040	42.069	0.0001	0.00391829	0.0001	0.66471937	0.0001
38	R8		38	0.0040	42.243	0.0001	0.00390257	0.0001	0.66500575	0.0001
39	A16		39	0.0037	38.503	0.0001	0.00388830	0.0001	0.66524153	0.0001
40	R11		40	0.0036	38.158	0.0001	0.00387421	0.0001	0.66554976	0.0001
41	R10		41	0.0034	35.398	0.0001	0.00386118	0.0001	0.66573422	0.0001
42	P14		42	0.0026	27.505	0.0001	0.00385108	0.0001	0.66614818	0.0001
43	P2		43	0.0028	29.000	0.0001	0.00384046	0.0001	0.66655659	0.0001

44	O13	44	0.0024	25.307	0.0001	0.00383121	0.0001	0.66679524	0.0001
45	A9	45	0.0023	24.435	0.0001	0.00382231	0.0001	0.66701318	0.0001
46	R12	46	0.0022	23.481	0.0001	0.00381377	0.0001	0.66721336	0.0001
47	A10	47	0.0018	19.201	0.0001	0.00380680	0.0001	0.66743251	0.0001
48	P3	48	0.0015	15.576	0.0001	0.00380115	0.0001	0.66751716	0.0001
49	P8	49	0.0015	15.789	0.0001	0.00379544	0.0001	0.66765041	0.0001
50	R4	50	0.0012	13.070	0.0001	0.00379072	0.0001	0.66780383	0.0001
51	O5	51	0.0010	10.845	0.0001	0.00378680	0.0001	0.66786833	0.0001
52	O11	52	0.0025	26.543	0.0001	0.00377724	0.0001	0.66802269	0.0001
53	P9	53	0.0009	9.786	0.0001	0.00377372	0.0001	0.66813269	0.0001
54	R9	54	0.0009	9.922	0.0001	0.00377015	0.0001	0.66818628	0.0001
55	R3	55	0.0013	13.805	0.0001	0.00376519	0.0001	0.66824758	0.0001
56	R7	56	0.0035	36.561	0.0001	0.00375211	0.0001	0.66835978	0.0001
57	R15	57	0.0022	23.419	0.0001	0.00374375	0.0001	0.66853007	0.0001
58	A8	58	0.0009	9.201	0.0001	0.00374046	0.0001	0.66866989	0.0001
59	A14	59	0.0015	16.244	0.0001	0.00373468	0.0001	0.66891682	0.0001
60	A2	60	0.0007	7.024	0.0001	0.00373218	0.0001	0.66894237	0.0001
61	A5	61	0.0006	6.389	0.0001	0.00372990	0.0001	0.66898743	0.0001
62	A12	62	0.0006	6.168	0.0001	0.00372771	0.0001	0.66902765	0.0001
63	P15	63	0.0004	4.272	0.0019	0.00372619	0.0001	0.66907435	0.0001
64	A3	64	0.0004	3.775	0.0045	0.00372485	0.0001	0.66908541	0.0001
65	P12	65	0.0003	3.294	0.0105	0.00372368	0.0001	0.66912543	0.0001

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Average Squared Wilks' Lambda	Prob < Lambda	Canonical Correlation	Prob > ASCC
1	C17		1	0.7064	25258.340	0.0001	0.29361310	0.0001	0.17659673	0.0001
2	C1		2	0.4900	10086.423	0.0001	0.14974538	0.0001	0.26457235	0.0001
3	C21		3	0.3540	5753.326	0.0001	0.09673305	0.0001	0.32931690	0.0001
4	C9		4	0.5376	12203.035	0.0001	0.04473380	0.0001	0.44860012	0.0001
5	HR1		5	0.1908	2475.881	0.0001	0.03619682	0.0001	0.47277369	0.0001
6	HB1		6	0.2426	3362.434	0.0001	0.02741544	0.0001	0.50723083	0.0001
7	C15		7	0.1761	2243.966	0.0001	0.02258707	0.0001	0.53905818	0.0001
8	C16		8	0.1172	1394.078	0.0001	0.01993902	0.0001	0.55204216	0.0001
9	C2		9	0.1068	1255.052	0.0001	0.01780960	0.0001	0.56403609	0.0001
10	C25		10	0.0843	965.986	0.0001	0.01630872	0.0001	0.57278953	0.0001
11	C20		11	0.0735	832.420	0.0001	0.01511037	0.0001	0.57760190	0.0001
12	HG9		12	0.0706	797.130	0.0001	0.01404380	0.0001	0.58327060	0.0001
13	HR9		13	0.0518	573.064	0.0001	0.01331671	0.0001	0.59049233	0.0001
14	C4		14	0.0426	466.886	0.0001	0.01274956	0.0001	0.59574823	0.0001
15	C8		15	0.0376	409.703	0.0001	0.01227055	0.0001	0.59902833	0.0001
16	HB2		16	0.0491	541.829	0.0001	0.01166816	0.0001	0.60328739	0.0001
17	C12		17	0.0413	452.363	0.0001	0.01118600	0.0001	0.60693967	0.0001
18	C11		18	0.0538	596.816	0.0001	0.01058409	0.0001	0.61044193	0.0001
19	C5		19	0.0519	575.044	0.0001	0.01003425	0.0001	0.61621483	0.0001
20	C27		20	0.0379	413.618	0.0001	0.00965375	0.0001	0.62054226	0.0001
21	HR10		21	0.0327	354.229	0.0001	0.00933852	0.0001	0.62380110	0.0001
22	HG8		22	0.0399	436.177	0.0001	0.00896584	0.0001	0.62742832	0.0001
23	HB6		23	0.0425	465.764	0.0001	0.00858479	0.0001	0.63341451	0.0001
24	HG7		24	0.0665	747.096	0.0001	0.00801418	0.0001	0.64174556	0.0001
25	HR5		25	0.0351	381.367	0.0001	0.00773311	0.0001	0.64537159	0.0001
26	HG3		26	0.0232	249.647	0.0001	0.00755340	0.0001	0.64727988	0.0001
27	HR11		27	0.0223	239.068	0.0001	0.00738513	0.0001	0.64920798	0.0001
28	HG10		28	0.0610	681.819	0.0001	0.00693449	0.0001	0.65474485	0.0001
29	C6		29	0.0179	191.552	0.0001	0.00681015	0.0001	0.65610166	0.0001
30	HG1		30	0.0202	215.977	0.0001	0.00667279	0.0001	0.65764898	0.0001
31	HB7		31	0.0172	183.722	0.0001	0.00655795	0.0001	0.65898921	0.0001
32	HB5		32	0.0184	196.153	0.0001	0.00643758	0.0001	0.66161577	0.0001
33	C26		33	0.0146	154.982	0.0001	0.00634386	0.0001	0.66247465	0.0001
34	C24		34	0.0146	155.633	0.0001	0.00625112	0.0001	0.66422592	0.0001
35	C14		35	0.0142	151.593	0.0001	0.00616208	0.0001	0.66548939	0.0001
36	HR12		36	0.0113	119.541	0.0001	0.00609265	0.0001	0.66633521	0.0001
37	HG11		37	0.0155	164.679	0.0001	0.00599848	0.0001	0.66758678	0.0001
38	HR8		38	0.0126	133.744	0.0001	0.00592296	0.0001	0.66892842	0.0001
39	HG6		39	0.0155	164.839	0.0001	0.00583132	0.0001	0.66950771	0.0001
40	HG2		40	0.0119	126.339	0.0001	0.00576192	0.0001	0.67001795	0.0001
41	HR3		41	0.0128	135.841	0.0001	0.00568825	0.0001	0.67038449	0.0001
42	C19		42	0.0106	112.726	0.0001	0.00562776	0.0001	0.67111512	0.0001
43	HB8		43	0.0078	82.534	0.0001	0.00558382	0.0001	0.67174008	0.0001
44	HR7		44	0.0075	78.869	0.0001	0.00554215	0.0001	0.67194875	0.0001

45	HG4		45	0.0092	97.307	0.0001	0.00549120	0.0001	0.67248089	0.0001
46	C22		46	0.0063	66.672	0.0001	0.00545651	0.0001	0.67309028	0.0001
47	C13		47	0.0050	52.310	0.0001	0.00542943	0.0001	0.67356766	0.0001
48	HR6		48	0.0049	52.066	0.0001	0.00540260	0.0001	0.67372592	0.0001
49	C10		49	0.0038	39.970	0.0001	0.00538209	0.0001	0.67423923	0.0001
50	C7		50	0.0235	251.869	0.0001	0.00525585	0.0001	0.67656391	0.0001
51	HB4		51	0.0040	42.360	0.0001	0.00523471	0.0001	0.67684868	0.0001
52	HB3		52	0.0085	90.179	0.0001	0.00519007	0.0001	0.67717312	0.0001
53	HR13		53	0.0027	28.210	0.0001	0.00517615	0.0001	0.67755508	0.0001
54	HB9		54	0.0020	21.528	0.0001	0.00516554	0.0001	0.67770109	0.0001
55	HG13		55	0.0021	21.571	0.0001	0.00515494	0.0001	0.67787907	0.0001
56	HG5		56	0.0020	20.871	0.0001	0.00514470	0.0001	0.67800659	0.0001
57	C23		57	0.0019	19.605	0.0001	0.00513509	0.0001	0.67823244	0.0001
58	HR2		58	0.0015	15.401	0.0001	0.00512756	0.0001	0.67834823	0.0001
59	HR4		59	0.0024	24.849	0.0001	0.00511544	0.0001	0.67850680	0.0001
60		HR5	58	0.0001	1.530	0.1903	0.00511618	0.0001	0.67850024	0.0001
61	C18		59	0.0012	12.237	0.0001	0.00511022	0.0001	0.67857950	0.0001
62	HB11		60	0.0006	6.624	0.0001	0.00510699	0.0001	0.67860195	0.0001
63	HB14		61	0.0006	6.254	0.0001	0.00510395	0.0001	0.67865177	0.0001
64	HG15		62	0.0002	2.544	0.0376	0.00510271	0.0001	0.67866872	0.0001
65	HR16		63	0.0002	2.297	0.0566	0.00510159	0.0001	0.67868365	0.0001
66	HR14		64	0.0003	2.643	0.0318	0.00510031	0.0001	0.67869145	0.0001
67	HG12		65	0.0002	1.712	0.1442	0.00509947	0.0001	0.67870400	0.0001
68		HG13	64	0.0001	0.597	0.6649	0.00509976	0.0001	0.67869921	0.0001
69	HB10		65	0.0002	1.980	0.0946	0.00509880	0.0001	0.67871355	0.0001

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda	Average Squared Canonical Correlation	Prob > ASCC
1	O3		1	0.8886	83785.345	0.0001	0.11135229	0.0001	0.22216193	0.0001
2	A13		2	0.5433	12490.605	0.0001	0.05085157	0.0001	0.33785910	0.0001
3	O10		3	0.5807	14541.236	0.0001	0.02132042	0.0001	0.47371044	0.0001
4	O15		4	0.3231	5010.180	0.0001	0.01443250	0.0001	0.53301804	0.0001
5	HR12		5	0.2128	2838.636	0.0001	0.01136056	0.0001	0.55047571	0.0001
6	O8		6	0.1692	2138.457	0.0001	0.00943794	0.0001	0.56927483	0.0001
7	O7		7	0.1253	1504.240	0.0001	0.00825501	0.0001	0.58443292	0.0001
8	O16		8	0.1436	1760.781	0.0001	0.00706921	0.0001	0.60116317	0.0001
9	C15		9	0.1016	1187.022	0.0001	0.00635101	0.0001	0.61038952	0.0001
10	C1		10	0.0906	1045.201	0.0001	0.00577587	0.0001	0.61902560	0.0001
11	C2		11	0.2591	3670.422	0.0001	0.00427941	0.0001	0.65575952	0.0001
12	C3		12	0.1292	1557.193	0.0001	0.00372654	0.0001	0.66183750	0.0001
13	HB1		13	0.1326	1604.068	0.0001	0.00323251	0.0001	0.67698166	0.0001
14	C8		14	0.2376	3271.157	0.0001	0.00246442	0.0001	0.70325892	0.0001
15	O9		15	0.0853	978.108	0.0001	0.00225433	0.0001	0.70831921	0.0001
16	P13		16	0.0528	584.676	0.0001	0.00213537	0.0001	0.71391265	0.0001
17	HG6		17	0.0423	463.267	0.0001	0.00204509	0.0001	0.71780947	0.0001
18	R1		18	0.0407	445.789	0.0001	0.00196176	0.0001	0.72056677	0.0001
19	O12		19	0.0391	427.083	0.0001	0.00188504	0.0001	0.72406154	0.0001
20	O17		20	0.0393	429.623	0.0001	0.00181091	0.0001	0.72633105	0.0001
21	C13		21	0.0344	374.040	0.0001	0.00174858	0.0001	0.73002581	0.0001
22	A4		22	0.0325	352.827	0.0001	0.00169170	0.0001	0.73207660	0.0001
23	HB6		23	0.0317	343.562	0.0001	0.00163807	0.0001	0.73419269	0.0001
24	A15		24	0.0308	333.310	0.0001	0.00158763	0.0001	0.73624862	0.0001
25	R16		25	0.0283	305.097	0.0001	0.00154278	0.0001	0.73780720	0.0001
26	HG5		26	0.0216	231.192	0.0001	0.00150952	0.0001	0.74041616	0.0001
27	HR14		27	0.0245	263.511	0.0001	0.00147253	0.0001	0.74277933	0.0001
28	HG13		28	0.0261	281.435	0.0001	0.00143407	0.0001	0.74394021	0.0001
29	HB2		29	0.0183	195.784	0.0001	0.00140780	0.0001	0.74546504	0.0001
30	HG11		30	0.0190	202.739	0.0001	0.00138111	0.0001	0.74662479	0.0001
31	HR13		31	0.0219	234.707	0.0001	0.00135088	0.0001	0.74890374	0.0001
32	HG12		32	0.0376	409.809	0.0001	0.00130010	0.0001	0.75166645	0.0001
33	HG10		33	0.0294	318.030	0.0001	0.00126185	0.0001	0.75288575	0.0001
34	HR10		34	0.0283	305.920	0.0001	0.00122609	0.0001	0.75470667	0.0001
35	HB7		35	0.0205	219.015	0.0001	0.00120102	0.0001	0.75565353	0.0001
36	HR5		36	0.0236	253.249	0.0001	0.00117271	0.0001	0.75692705	0.0001
37	O18		37	0.0185	197.407	0.0001	0.00115104	0.0001	0.75788152	0.0001
38	A7		38	0.0317	343.763	0.0001	0.00111452	0.0001	0.76022574	0.0001
39	O2		39	0.0151	160.806	0.0001	0.00109769	0.0001	0.76130117	0.0001
40	O4		40	0.0183	195.323	0.0001	0.00107762	0.0001	0.76254162	0.0001
41	O14		41	0.0344	373.217	0.0001	0.00104060	0.0001	0.76545115	0.0001
42	C4		42	0.0158	168.099	0.0001	0.00102418	0.0001	0.76605486	0.0001
43	P4		43	0.0137	146.010	0.0001	0.00101012	0.0001	0.76715461	0.0001
44	C5		44	0.0124	131.318	0.0001	0.00099763	0.0001	0.76813788	0.0001

45	HR3	45	0.0117	123.778	0.0001	0.00098599	0.0001	0.76850143	0.0001
46	A6	46	0.0116	122.983	0.0001	0.00097456	0.0001	0.76885446	0.0001
47	HG9	47	0.0107	113.392	0.0001	0.00096414	0.0001	0.76930696	0.0001
48	HB5	48	0.0109	115.169	0.0001	0.00095367	0.0001	0.77024382	0.0001
49	C11	49	0.0107	113.537	0.0001	0.00094345	0.0001	0.77092777	0.0001
50	C12	50	0.0125	132.639	0.0001	0.00093167	0.0001	0.77214838	0.0001
51	R14	51	0.0099	104.442	0.0001	0.00092248	0.0001	0.77309389	0.0001
52	O20	52	0.0105	111.523	0.0001	0.00091277	0.0001	0.77426857	0.0001
53	O19	53	0.0173	184.084	0.0001	0.00089702	0.0001	0.77537989	0.0001
54	A11	54	00101	106.772	0.0001	0.00088798	0.0001	0.77617757	0.0001
55	R5	55	0.0097	102.471	0.0001	0.00087939	0.0001	0.77669270	0.0001
56	P16	56	0.0095	100.322	0.0001	0.00087105	0.0001	0.77720340	0.0001
57	R2	57	0.0085	90.077	0.0001	0.00086363	0.0001	0.77758190	0.0001
58	C27	58	0.0077	81.881	0.0001	0.00085694	0.0001	0.77786178	0.0001
59	C14	59	0.0076	80.256	0.0001	0.00085043	0.0001	0.77826654	0.0001
60	C6	60	0.0074	78.186	0.0001	0.00084414	0.0001	0.77843809	0.0001
61	C25	61	0.0083	87.490	0.0001	0.00083715	0.0001	0.77865375	0.0001
62	C24	62	0.0080	84.362	0.0001	0.00083047	0.0001	0.77904082	0.0001
63	HB8	63	0.0065	69.046	0.0001	0.00082503	0.0001	0.77943533	0.0001
64	C20	64	0.0062	65.457	0.0001	0.00081991	0.0001	0.77962768	0.0001
65	C17	65	0.0288	310.394	0.0001	0.00079633	0.0001	0.78099172	0.0001
66	C21	66	0.0390	425.193	0.0001	0.00076529	0.0001	0.78320457	0.0001
67	C16	67	0.0488	538.211	0.0001	0.00072792	0.0001	0.78605255	0.0001
68	HR1	68	0.0111	117.502	0.0001	0.00071985	0.0001	0.78669715	0.0001
69	HG1	69	0.0134	142.308	0.0001	0.00071020	0.0001	0.78747908	0.0001
70	O1	70	00060	62.868	0.0001	0.00070597	0.0001	0.78787342	0.0001
71	P7	71	0.0051	54.156	0.0001	0.00070234	0.0001	0.78818156	0.0001
72	C10	72	0.0049	51.996	0.0001	0.00069887	0.0001	0.78851187	0.0001
73	C7	73	0.0189	201.593	0.0001	0.00068569	0.0001	0.78985876	0.0001
74	R8	74	0.0045	47.660	0.0001	0.00068258	0.0001	0.79002999	0.0001
75	P10	75	0.0044	46.420	0.0001	0.00067957	0.0001	0.79028301	0.0001
76	HG3	76	0.0042	44.176	0.0001	0.00067672	0.0001	0.79050945	0.0001
77	HR11	77	0.0042	43.921	0.0001	0.00067389	0.0001	0.79069636	0.0001
78	HR9	78	0.0057	59.726	0.0001	0.00067008	0.0001	0.79100337	0.0001
79	HG8	79	0.0066	69.370	0.0001	0.00066567	0.0001	0.79128764	0.0001
80	O6	80	0.0041	43.487	0.0001	0.00066292	0.0001	0.79143752	0.0001
81	A16	81	0.0039	41.277	0.0001	0.00066032	0.0001	0.79159339	0.0001
82	C18	82	0.0040	41.919	0.0001	0.00065769	0.0001	0.79192496	0.0001
83	HR8	83	0.0036	38.086	0.0001	0.00065530	0.0001	0.79218076	0.0001
84	C26	84	0.0029	30.339	0.0001	0.00065341	0.0001	0.79242318	0.0001
85	R10	85	0.0026	27.443	0.0001	0.00065170	0.0001	0.79251364	0.0001
86	HB4	86	00026	26.863	0.0001	0.00065004	0.0001	0.79262813	0.0001
87	P6	87	0.0023	24.174	0.0001	0.00064854	0.0001	0.79270170	0.0001
88	P11	88	0.0023	24.034	0.0001	0.00064706	0.0001	0.79283062	0.0001
89	HR2	89	0.0021	21.590	0.0001	0.00064573	0.0001	0.79293724	0.0001
90	HR7	90	0.0020	20.808	0.0001	0.00064445	0.0001	0.79303351	0.0001
91	R13	91	0.0019	20.446	0.0001	0.00064319	0.0001	0.79311188	0.0001
92	R11	92	0.0023	24.246	0.0001	0.00064171	0.0001	0.79324883	0.0001
93	R6	93	0.0020	21.494	0.0001	0.00064039	0.0001	0.79343644	0.0001
94	HR4	94	0.0018	18.710	0.0001	0.00063925	0.0001	0.79354823	0.0001
95	HR6	95	0.0017	17.970	0.0001	0.00063816	0.0001	0.79364348	0.0001
96	A10	96	0.0016	17.076	0.0001	0.00063712	0.0001	0.79374082	0.0001

97	O13	97	0.0016	16.594	0.0001	0.00063611	0.0001	0.79381979	0.0001
98	HB11	98	0.0015	15.910	0.0001	0.00063514	0.0001	0.79389379	0.0001
99	A9	99	0.0015	15.240	0.0001	0.00063422	0.0001	0.79400289	0.0001
100	HB3	100	0.0013	13.920	0.0001	0.00063338	0.0001	0.79406149	0.0001
101	HB9	101	0.0014	14.414	0.0001	0.00063251	0.0001	0.79414296	0.0001
102	HG7	102	0.0013	13.242	0.0001	0.00063171	0.0001	0.79418623	0.0001
103	HG14	103	0.0013	13.406	0.0001	0.00063090	0.0001	0.79423184	0.0001
104	O5	104	0.0013	13.428	0.0001	0.00063010	0.0001	0.79429797	0.0001
105	O11	105	0.0019	19.503	0.0001	0.00062892	0.0001	0.79438203	0.0001
106	C23	106	0.0012	12.801	0.0001	0.00062816	0.0001	0.79447302	0.0001
107	P8	107	0.0012	12.365	0.0001	0.00062742	0.0001	0.79452444	0.0001
108	P2	108	0.0013	14.070	0.0001	0.00062657	0.0001	0.79457525	0.0001
109	P14	109	0.0015	15.644	0.0001	0.00062564	0.0001	0.79473324	0.0001
110	HG2	110	0.0011	11.378	00001	0.00062496	0.0001	0.79477200	0.0001
111	R3	111	0.0010	10.593	0.0001	0.00062433	0.0001	0.79479988	0.0001
112	R7	112	0.0032	33.957	0.0001	0.00062231	0.0001	0.79488172	0.0001
113	R15	113	0.0015	16.108	0.0001	0.00062135	0.0001	0.79494412	0.0001
114	R9	114	0.0009	9.761	0.0001	0.00062078	0.0001	0.79497868	0.0001
115	R12	115	0.0009	9.387	0.0001	0.00062022	0.0001	0.79501853	0.0001
116	P9	116	0.0008	8.097	0.0001	0.00061974	0.0001	0.79507035	0.0001
117	R4	117	0.0007	7.332	0.0001	0.00061931	0.0001	0.79510030	0.0001
118	HB14	118	0.0007	6.957	0.0001	0.00061890	0.0001	0.79513471	0.0001
119	A3	119	0.0006	5.940	0.0001	0.00061854	0.0001	0.79515288	0.0001
120	A5	120	0.0007	7.153	0.0001	0.00061812	0.0001	0.79517481	0.0001
121	C19	121	0.0005	5.557	0.0002	0.00061779	0.0001	0.79520909	0.0001
122	A8	122	0.0005	4.743	0.0008	0.00061751	0.0001	0.79525269	0.0001
123	A12	123	0.0007	7.068	0.0001	0.00061710	0.0001	0.79530063	0.0001
124	A2	124	0.0005	5.753	0.0001	0.00061676	0.0001	0.79531913	0.0001
125	P3	125	0.0004	3.936	0.0034	0.00061653	0.0001	0.79533032	0.0001
126	P15	126	0.0003	3.194	00124	0.00061634	0.0001	0.79534568	0.0001
127	A14	127	0.0002	2.062	0.0829	0.00061622	0.0001	0.79536175	0.0001
128	P12	128	0.0002	1.899	0.1077	0.00061611	0.0001	0.79537735	0.0001
129	HR16	129	0.0002	1.692	0.1487	0.00061601	0.0001	0.79538369	0.0001
130	HR14	128	0.0002	1.670	0.1539	0.00061610	0.0001	0.79538041	0.0001
131	HB12	129	0.0002	1.797	0.1263	0.00061600	0.0001	0.79539196	0.0001

APPENDIX D-2

STEPDISC ANALYSIS OF KERNEL FEATURES FOR DAMAGE TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL CWRS WHEAT KERNELS

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda	Average Squared Canonical Correlation	Prob > ASCC
1	F7		1	0.4326	888.618	0.0001	0.56739680	0.0001	0.07210053	0.0001
2	F21		2	0.2955	488.680	0.0001	0.39975879	0.0001	0.12109561	0.0001
3	F43		3	0.2736	438.899	0.0001	0.29037816	0.0001	0.16241080	0.0001
4	F19		4	0.1416	192.143	0.0001	0.24926678	0.0001	0.18206295	0.0001
5	F5		5	0.0851	108.400	0.0001	0.22804489	0.0001	0.19214480	0.0001
6	F36		6	0.0758	95.476	0.0001	0.21076679	0.0001	0.20267099	0.0001
7	F59		7	0.0620	76.949	0.0001	0.19770278	0.0001	0.20965432	0.0001
8	F34		8	0.0565	69.699	0.0001	0.18653647	0.0001	0.21598314	0.0001
9	F25		9	0.0429	52.215	0.0001	0.17852913	0.0001	0.22190697	0.0001
10	F49		10	0.0521	63.923	0.0001	0.16923536	0.0001	0.23001125	0.0001
11	F16		11	0.0426	51.797	0.0001	0.16202433	0.0001	0.23478427	0.0001
12	F63		12	0.0419	50.828	0.0001	0.15524347	0.0001	0.23896136	0.0001
13	F15		13	0.0356	43.011	0.0001	0.14970919	0.0001	0.24308992	0.0001
14	F1		14	0.0349	42.064	0.0001	0.14448483	0.0001	0.24658745	0.0001
15	F9		15	0.0740	92.986	0.0001	0.13378946	0.0001	0.25360290	0.0001
16	F3		16	0.0452	55.115	0.0001	0.12773603	0.0001	0.25943738	0.0001
17	F2		17	0.0372	44.903	0.0001	0.12298685	0.0001	0.26328037	0.0001
18	F12		18	0.0344	41.424	0.0001	0.11875578	0.0001	0.26759597	0.0001
19	F8		19	0.0450	54.724	0.0001	0.11341678	0.0001	0.27245296	0.0001
20	F26		20	0.0287	34.381	0.0001	0.11015833	0.0001	0.27496773	0.0001
21	F14		21	0.0270	32.236	0.0001	0.10718528	0.0001	0.27814962	0.0001
22	F10		22	0.0286	34.265	0.0001	0.10411512	0.0001	0.28063750	0.0001
23	F17		23	0.0425	51.550	0.0001	0.09969186	0.0001	0.28532501	0.0001
24	F18		24	0.0303	36.251	0.0001	0.09667500	0.0001	0.28876320	0.0001
25	F20		25	0.0293	35.055	0.0001	0.09384274	0.0001	0.29209652	0.0001
26	F4		26	0.0255	30.434	0.0001	0.09144632	0.0001	0.29482542	0.0001
27	F13		27	0.0209	24.753	0.0001	0.08953759	0.0001	0.29696645	0.0001
28	F41		28	0.0184	21.755	0.0001	0.08789070	0.0001	0.29858884	0.0001
29	F65		29	0.0118	13.863	0.0001	0.08685344	0.0001	0.30001407	0.0001
30	F58		30	0.0110	12.883	0.0001	0.08590002	0.0001	0.30106692	0.0001
31	F42		31	0.0110	12.881	0.0001	0.08495705	0.0001	0.30210552	0.0001
32	F29		32	0.0078	9.072	0.0001	0.08429799	0.0001	0.30299314	0.0001
33	F68		33	0.0076	8.924	0.0001	0.08365451	0.0001	0.30372514	0.0001
34	F56		34	0.0094	10.954	0.0001	0.08287195	0.0001	0.30470930	0.0001
35	F52		35	0.0074	8.674	0.0001	0.08225681	0.0001	0.30550588	0.0001
36	F39		36	0.0073	8.578	0.0001	0.08165285	0.0001	0.30612574	0.0001
37	F32		37	0.0073	8.542	0.0001	0.08105571	0.0001	0.30668422	0.0001
38	F46		38	0.0057	6.620	0.0001	0.08059551	0.0001	0.30723643	0.0001
39	F24		39	0.0054	6.275	0.0001	0.08016158	0.0001	0.30779707	0.0001
40	F28		40	0.0054	6.271	0.0001	0.07973017	0.0001	0.30833386	0.0001
41	F51		41	0.0051	5.887	0.0001	0.07932717	0.0001	0.30875465	0.0001
42	F35		42	0.0050	5.852	0.0001	0.07892852	0.0001	0.30920300	0.0001
43	F27		43	0.0050	5.799	0.0001	0.07853540	0.0001	0.30971330	0.0001
44	F62		44	0.0043	5.056	0.0001	0.07819410	0.0001	0.31010055	0.0001
45	F22		45	0.0038	4.461	0.0002	0.07789406	0.0001	0.31052193	0.0001

46	F55		46	0.0029	3.399	0.0024	0.07766609	0.0001	0.31080634	0.0001
47	F64		47	0.0027	3.082	0.0052	0.07745992	0.0001	0.31118340	0.0001
48	F47		48	0.0024	2.740	0.0117	0.07727704	0.0001	0.31140909	0.0001
49	F23		49	0.0027	3.078	0.0052	0.07707206	0.0001	0.31169703	0.0001
50	F40		50	0.0024	2.757	0.0112	0.07688889	0.0001	0.31188742	0.0001
51	F30		51	0.0024	2.755	0.0112	0.07670626	0.0001	0.31212385	0.0001
52	F33		52	0.0019	2.196	0.0404	0.07656092	0.0001	0.31238675	0.0001
53	F31		53	0.0021	2.447	0.0230	0.07639931	0.0001	0.31265358	0.0001
54	F50		54	0.0020	2.284	0.0332	0.07624872	0.0001	0.31282525	0.0001
55	F67		55	0.0016	1.809	0.0931	0.07612963	0.0001	0.31295333	0.0001
56		F5	54	0.0014	1.572	0.1510	0.07623309	0.0001	0.31282977	0.0001
57	F11		55	0.0017	1.930	0.0722	0.07610607	0.0001	0.31297014	0.0001
58	F57		56	0.0014	1.628	0.1349	0.07599907	0.0001	0.31311860	0.0001
59	F6		57	0.0014	1.615	0.1385	0.07589305	0.0001	0.31326364	0.0001

Stepwise Selection: Summary

Step	Variable Entered	Removed	Number In	Partial R**2	F Statistic	Prob > F	Average Squared Wilks' Lambda	Prob < Lambda	Canonical Correlation	Prob > ASCC
1	F75		1	0.9722	40792.328	0.0001	0.02777789	0.0001	0.16203702	0.0001
2	F121		2	0.6223	1920.386	0.0001	0.01049042	0.0001	0.26474982	0.0001
3	F81		3	0.4137	822.063	0.0001	0.00615082	0.0001	0.32134245	0.0001
4	F101		4	0.3531	635.791	0.0001	0.00397920	0.0001	0.37934454	0.0001
5	F137		5	0.3061	513.792	0.0001	0.00276125	0.0001	0.41112572	0.0001
6	F94		6	0.2781	448.634	0.0001	0.00199339	0.0001	0.45244427	0.0001
7	F138		7	0.2276	343.045	0.0001	0.00153979	0.0001	0.47820239	0.0001
8	F80		8	0.2514	390.976	0.0001	0.00115271	0.0001	0.49091756	0.0001
9	F69		9	0.1993	289.789	0.0001	0.00092297	0.0001	0.49554228	0.0001
10	F72		10	0.3700	683.594	0.0001	0.00058148	0.0001	0.50658032	0.0001
11	F140		11	0.2172	322.898	0.0001	0.00045519	0.0001	0.50958325	0.0001
12	F70		12	0.2015	293.706	0.0001	0.00036345	0.0001	0.52925377	0.0001
13	F92		13	0.1590	219.984	0.0001	0.00030566	0.0001	0.54518034	0.0001
14	F95		14	0.1683	235.401	0.0001	0.00025422	0.0001	0.56433282	0.0001
15	F115		15	0.1382	186.591	0.0001	0.00021908	0.0001	0.56549830	0.0001
16	F114		16	0.2058	301.312	0.0001	0.00017400	0.0001	0.56741799	0.0001
17	F128		17	0.1771	250.259	0.0001	0.00014318	0.0001	0.56856743	0.0001
18	F100		18	0.1599	221.326	0.0001	0.00012028	0.0001	0.57209014	0.0001
19	F113		19	0.1275	169.936	0.0001	0.00010494	0.0001	0.57264654	0.0001
20	F107		20	0.1033	133.863	0.0001	0.00009411	0.0001	0.58261376	0.0001
21	F135		21	0.1005	129.842	0.0001	0.00008465	0.0001	0.58967873	0.0001
22	F111		22	0.0964	123.907	0.0001	0.00007649	0.0001	0.59060236	0.0001
23	F76		23	0.0973	125.165	0.0001	0.00006905	0.0001	0.59758407	0.0001
24	F110		24	0.1069	139.012	0.0001	0.00006167	0.0001	0.60503993	0.0001
25	F108		25	0.0853	108.359	0.0001	0.00005641	0.0001	0.60948680	0.0001
26	F84		26	0.0854	108.465	0.0001	0.00005159	0.0001	0.61563092	0.0001
27	F71		27	0.2579	403.472	0.0001	0.00003829	0.0001	0.63116456	0.0001
28	F88		28	0.0792	99.898	0.0001	0.00003525	0.0001	0.63573864	0.0001
29	F73		29	0.0914	116.802	0.0001	0.00003203	0.0001	0.63883507	0.0001
30	F136		30	0.0783	98.598	0.0001	0.00002952	0.0001	0.64746546	0.0001
31	F93		31	0.0682	84.989	0.0001	0.00002751	0.0001	0.65231938	0.0001
32	F85		32	0.1119	146.132	0.0001	0.00002443	0.0001	0.65766659	0.0001
33	F99		33	0.0636	78.797	0.0001	0.00002288	0.0001	0.66002322	0.0001
34	F98		34	0.0717	89.556	0.0001	0.00002124	0.0001	0.66063007	0.0001
35	F120		35	0.0484	58.977	0.0001	0.00002021	0.0001	0.66468099	0.0001
36	F134		36	0.0580	71.368	0.0001	0.00001904	0.0001	0.66741076	0.0001
37	F74		37	0.0507	61.866	0.0001	0.00001807	0.0001	0.67016120	0.0001
38	F141		38	0.0600	73.948	0.0001	0.00001699	0.0001	0.67050661	0.0001
39	F97		39	0.0522	63.848	0.0001	0.00001610	0.0001	0.67124715	0.0001
40	F123		40	0.0379	45.717	0.0001	0.00001549	0.0001	0.67279896	0.0001
41	F78		41	0.0365	43.941	0.0001	0.00001493	0.0001	0.67338349	0.0001
42	F106		42	0.0330	39.516	0.0001	0.00001443	0.0001	0.67555995	0.0001
43	F82		43	0.0326	39.102	0.0001	0.00001396	0.0001	0.67718002	0.0001
44	F79		44	0.0339	40.640	0.0001	0.00001349	0.0001	0.67801318	0.0001

45	F91		45	0.0305	36.399	0.0001	0.00001308	0.0001	0.67938363	0.0001
46	F109		46	0.0287	34.157	0.0001	0.00001270	0.0001	0.68069902	0.0001
47	F96		47	0.0290	34.547	0.0001	0.00001234	0.0001	0.68144720	0.0001
48	F112		48	0.0267	31.746	0.0001	0.00001201	0.0001	0.68303856	0.0001
49	F83		49	0.0203	23.947	0.0001	0.0000117	0.0001	0.68370766	0.0001
50	F86		50	0.0182	21.416	0.0001	0.00001155	0.0001	0.68534022	0.0001
51	F133		51	0.0147	17.234	0.0001	0.00001138	0.0001	0.68610355	0.0001
52	F102		52	0.0139	16.350	0.0001	0.00001122	0.0001	0.68660861	0.0001
53	F142		53	0.0131	15.316	0.0001	0.00001107	0.0001	0.68705749	0.0001
54	F87		54	0.0114	13.347	0.0001	0.00001095	0.0001	0.68743169	0.0001
55	F105		55	0.0093	10.806	0.0001	0.00001085	0.0001	0.68828342	0.0001
56	F117		56	0.0077	9.005	0.0001	0.00001076	0.0001	0.68862699	0.0001
57	F116		57	0.0077	8.961	0.0001	0.00001068	0.0001	0.68903666	0.0001
58	F118		58	0.0086	9.984	0.0001	0.00001059	0.0001	0.68943522	0.0001
59	F124		59	0.0054	6.226	0.0001	0.00001053	0.0001	0.68969119	0.0001
60	F130		60	0.0063	7.368	0.0001	0.00001047	0.0001	0.68995494	0.0001
61	F122		61	0.0058	6.772	0.0001	0.00001040	0.0001	0.69017553	0.0001
62	F126		62	0.0060	6.981	0.0001	0.00001034	0.0001	0.69053838	0.0001
63	F139		63	0.0058	6.732	0.0001	0.00001028	0.0001	0.69077663	0.0001
64		F113	62	0.0011	1.302	0.2524	0.00001029	0.0001	0.69072493	0.0001
65	F125		63	0.0050	5.793	0.0001	0.0000102	0.0001	0.69076424	0.0001
66	F131		64	0.0043	4.975	0.0001	0.00001020	0.0001	0.69102609	0.0001
67	F104		65	0.0039	4.575	0.0001	0.00001016	0.0001	0.69128834	0.0001
68	F90		66	0.0023	2.695	0.0129	0.00001013	0.0001	0.69134818	0.0001
69	F89		67	0.0020	2.307	0.0316	0.00001011	0.0001	0.69145050	0.0001
70	F132		68	0.0016	1.900	0.0769	0.00001010	0.0001	0.69153155	0.0001
71	F129		69	0.0014	1.621	0.1368	0.00001008	0.0001	0.69163367	0.0001

Stepwise Selection: Summary

Step	Variable Entered	Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda	Average Squared Canonical Correlation	Prob > ASCC
1	F75		1	0.9722	40792.328	0.0001	0.02777789	0.0001	0.16203702	0.0001
2	F121		2	0.6223	1920.386	0.0001	0.01049042	0.0001	0.26474982	0.0001
3	F1		3	0.4230	854.344	0.0001	0.00605250	0.0001	0.33362819	0.0001
4	F81		4	0.3477	620.856	0.0001	0.00394834	0.0001	0.37008790	0.0001
5	F101		5	0.3519	632.438	0.0001	0.00255897	0.0001	0.42512700	0.0001
6	F137		6	0.2663	422.700	0.0001	0.00187754	0.0001	0.44655154	0.0001
7	F138		7	0.2697	430.132	0.0001	0.00137110	0.0001	0.48310919	0.0001
8	F80		8	0.2409	369.504	0.0001	0.00104080	0.0001	0.49600018	0.0001
9	F34		9	0.2073	304.463	0.0001	0.00082503	0.0001	0.51805420	0.0001
10	F72		10	0.2049	300.022	0.0001	0.00065596	0.0001	0.52751004	0.0001
11	F69		11	0.3645	667.634	0.0001	0.00041684	0.0001	0.53102427	0.0001
12	F140		12	0.2090	307.480	0.0001	0.00032971	0.0001	0.53516480	0.0001
13	F70		13	0.1719	241.590	0.0001	0.00027302	0.0001	0.55016370	0.0001
14	F95		14	0.1450	197.330	0.0001	0.00023343	0.0001	0.56744942	0.0001
15	F92		15	0.1645	229.047	0.0001	0.00019502	0.0001	0.58310936	0.0001
16	F115		16	0.1390	187.808	0.0001	0.00016791	0.0001	0.58436535	0.0001
17	F114		17	0.2103	309.682	0.0001	0.00013260	0.0001	0.58598388	0.0001
18	F128		18	0.1686	235.815	0.0001	0.00011024	0.0001	0.58692796	0.0001
19	F100		19	0.1632	226.736	0.0001	0.00009225	0.0001	0.59037250	0.0001
20	F15		20	0.1311	175.331	0.0001	0.00008016	0.0001	0.60256063	0.0001
21	F17		21	0.1273	169.590	0.0001	0.00006995	0.0001	0.61229517	0.0001
22	F84		22	0.4624	999.582	0.0001	0.00003760	0.0001	0.65393943	0.0001
23	F71		23	0.2075	304.125	0.0001	0.00002980	0.0001	0.66116706	0.0001
24	F76		24	0.1678	234.298	0.0001	0.00002480	0.0001	0.66932742	0.0001
25	F113		25	0.1650	229.452	0.0001	0.00002071	0.0001	0.67388340	0.0001
26	F126		26	0.1388	187.123	0.0001	0.00001783	0.0001	0.68056571	0.0001
27	F73		27	0.1324	177.189	0.0001	0.00001547	0.0001	0.68401614	0.0001
28	F107		28	0.1145	150.104	0.0001	0.00001370	0.0001	0.69179565	0.0001
29	F108		29	0.0778	97.920	0.0001	0.00001264	0.0001	0.69517550	0.0001
30	F99		30	0.0712	88.986	0.0001	0.00001174	0.0001	0.69665759	0.0001
31	F98		31	0.0842	106.657	0.0001	0.00001075	0.0001	0.69773032	0.0001
32	F97		32	0.0697	86.999	0.0001	0.00001000	0.0001	0.69907270	0.0001
33	F111		33	0.0712	88.921	0.0001	0.00000929	0.0001	0.69923592	0.0001
34	F133		34	0.0626	77.494	0.0001	0.00000871	0.0001	0.70458839	0.0001
35	F12		35	0.0526	64.337	0.0001	0.00000825	0.0001	0.70863829	0.0001
36	F136		36	0.0511	62.427	0.0001	0.00000783	0.0001	0.71227015	0.0001
37	F74		37	0.0479	58.329	0.0001	0.00000745	0.0001	0.71406946	0.0001
38	F141		38	0.0575	70.782	0.0001	0.00000702	0.0001	0.71451606	0.0001
39	F94		39	0.0423	51.225	0.0001	0.00000673	0.0001	0.71635046	0.0001
40	F3		40	0.0358	43.086	0.0001	0.00000648	0.0001	0.71746184	0.0001
41	F2		41	0.0412	49.820	0.0001	0.00000622	0.0001	0.71989073	0.0001
42	F123		42	0.0368	44.223	0.0001	0.00000599	0.0001	0.72130914	0.0001
43	F14		43	0.0345	41.376	0.0001	0.00000578	0.0001	0.72420503	0.0001
44	F79		44	0.0335	40.113	0.0001	0.00000559	0.0001	0.72504201	0.0001

45	F87	45	0.0301	35.947	0.0001	0.00000542	0.0001	0.72604948	0.0001
46	F82	46	0.0290	34.637	0.0001	0.00000526	0.0001	0.72784873	0.0001
47	F132	47	0.0297	35.494	0.0001	0.00000511	0.0001	0.72964395	0.0001
48	F83	48	0.0297	35.417	0.0001	0.00000495	0.0001	0.73037038	0.0001
49	F96	49	0.0308	36.735	0.0001	0.00000480	0.0001	0.73119305	0.0001
50	F78	50	0.0325	38.863	0.0001	0.00000465	0.0001	0.73139774	0.0001
51	F125	51	0.0288	34.335	0.0001	0.00000451	0.0001	0.73242961	0.0001
52	F36	52	0.0285	33.944	0.0001	0.00000438	0.0001	0.73429122	0.0001
53	F135	53	0.0284	33.754	0.0001	0.00000426	0.0001	0.73594064	0.0001
54	F10	54	0.0220	26.061	0.0001	0.00000417	0.0001	0.73668835	0.0001
55	F43	55	0.0211	24.968	0.0001	0.00000408	0.0001	0.73776858	0.0001
56	F93	56	0.0201	23.767	0.0001	0.00000400	0.0001	0.73850188	0.0001
57	F18	57	0.1470	199.169	0.0001	0.00000341	0.0001	0.74435274	0.0001
58	F85	58	0.0231	27.343	0.0001	0.00000333	0.0001	0.74522523	0.0001
59	F102	59	0.0199	23.478	0.0001	0.00000326	0.0001	0.74577000	0.0001
60	F88	60	0.0175	20.541	0.0001	0.00000321	0.0001	0.74654410	0.0001
61	F120	61	0.0154	18.079	0.0001	0.00000316	0.0001	0.74739877	0.0001
62	F106	62	0.0169	19.898	0.0001	0.00000310	0.0001	0.74840653	0.0001
63	F49	63	0.0143	16.731	0.0001	0.00000306	0.0001	0.74920846	0.0001
64	F41	64	0.0149	17.434	0.0001	0.00000301	0.0001	0.75002681	0.0001
65	F58	65	0.0139	16.260	0.0001	0.00000297	0.0001	0.75075042	0.0001
66	F13	66	0.0138	16.210	0.0001	0.00000293	0.0001	0.75124975	0.0001
67	F134	67	0.0128	15.016	0.0001	0.00000289	0.0001	0.75165320	0.0001
68	F112	68	0.0144	16.856	0.0001	0.00000285	0.0001	0.75213933	0.0001
69	F86	69	0.0127	14.890	0.0001	0.00000282	0.0001	0.75320495	0.0001
70	F9	70	0.0108	12.571	0.0001	0.00000278	0.0001	0.75391390	0.0001
71	F142	71	0.0097	11.286	0.0001	0.00000276	0.0001	0.75415261	0.0001
72	F26	72	0.0096	11.205	0.0001	0.00000273	0.0001	0.75469611	0.0001
73	F16	73	0.0097	11.256	0.0001	0.00000271	0.0001	0.75505263	0.0001
74	F21	74	0.0088	10.181	0.0001	0.00000268	0.0001	0.75545992	0.0001
75	F110	75	0.0087	10.107	0.0001	0.00000266	0.0001	0.75551451	0.0001
76	F105	76	0.0082	9.569	0.0001	0.00000264	0.0001	0.75611819	0.0001
77	F91	77	0.0079	9.155	0.0001	0.00000262	0.0001	0.75657548	0.0001
78	F8	78	0.0071	8.207	0.0001	0.00000260	0.0001	0.75683467	0.0001
79	F56	79	0.0061	7.059	0.0001	0.00000258	0.0001	0.75721153	0.0001
80	F62	80	0.0063	7.336	0.0001	0.00000256	0.0001	0.75752541	0.0001
81	F129	81	0.0061	7.018	0.0001	0.00000255	0.0001	0.75775881	0.0001
82	F117	82	0.0057	6.612	0.0001	0.00000253	0.0001	0.75793436	0.0001
83	F116	83	0.0077	8.884	0.0001	0.00000252	0.0001	0.75813813	0.0001
84	F65	84	0.0058	6.661	0.0001	0.00000250	0.0001	0.75835478	0.0001
85	F63	85	0.0055	6.381	0.0001	0.00000249	0.0001	0.75854709	0.0001
86	F4	86	0.0056	6.434	0.0001	0.00000247	0.0001	0.75884290	0.0001
87	F59	87	0.0062	7.164	0.0001	0.00000246	0.0001	0.75901792	0.0001
88	F20	88	0.0054	6.207	0.0001	0.00000244	0.0001	0.75922663	0.0001
89	F19	89	0.0108	12.539	0.0001	0.00000242	0.0001	0.75971387	0.0001
90	F35	90	0.0052	6.030	0.0001	0.00000241	0.0001	0.75993912	0.0001
91	F118	91	0.0050	5.810	0.0001	0.00000239	0.0001	0.76010324	0.0001
92	F27	92	0.0049	5.634	0.0001	0.00000238	0.0001	0.76025368	0.0001
93	F130	93	0.0048	5.594	0.0001	0.00000237	0.0001	0.76035348	0.0001
94	F119	94	0.0047	5.378	0.0001	0.00000236	0.0001	0.76049364	0.0001
95	F113	93	0.0007	0.799	0.5709	0.00000236	0.0001	0.76046438	0.0001
96	F52	94	0.0041	4.771	0.0001	0.00000235	0.0001	0.76068296	0.0001

97	F89		95	0.0041	4.779	0.0001	0.00000234	0.0001	0.76082860	0.0001
98	F25		96	0.0039	4.525	0.0001	0.00000233	0.0001	0.76092485	0.0001
99	F51		97	0.0037	4.313	0.0002	0.00000232	0.0001	0.76109263	0.0001
100	F39		98	0.0055	6.314	0.0001	0.00000231	0.0001	0.76137020	0.0001
101	F68		99	0.0038	4.420	0.0002	0.00000230	0.0001	0.76159608	0.0001
102	F42		100	0.0034	3.930	0.0006	0.00000229	0.0001	0.76172179	0.0001
103	F7		101	0.0034	3.937	0.0006	0.00000229	0.0001	0.76191856	0.0001
104	F32		102	0.0030	3.455	0.0021	0.00000228	0.0001	0.76204914	0.0001
105	F30		103	0.0028	3.267	0.0033	0.00000227	0.0001	0.76211376	0.0001
106	F28		104	0.0029	3.369	0.0026	0.00000227	0.0001	0.76215899	0.0001
107	F124		105	0.0027	3.105	0.0049	0.00000226	0.0001	0.76227746	0.0001
108	F109		106	0.0030	3.421	0.0023	0.00000225	0.0001	0.76233120	0.0001
109	F104		107	0.0029	3.373	0.0025	0.00000225	0.0001	0.76245933	0.0001
110	F24		108	0.0025	2.929	0.0074	0.00000224	0.0001	0.76259105	0.0001
111	F46		109	0.0024	2.774	0.0108	0.00000224	0.0001	0.76267729	0.0001
112	F66		110	0.0019	2.152	0.0446	0.00000223	0.0001	0.76271700	0.0001
113	F139		111	0.0016	1.871	0.0818	0.00000223	0.0001	0.76279357	0.0001
114	F40		112	0.0016	1.826	0.0899	0.00000222	0.0001	0.76285170	0.0001
115	F64		113	0.0015	1.775	0.1001	0.00000222	0.0001	0.76295211	0.0001
116	F131		114	0.0014	1.661	0.1262	0.00000222	0.0001	0.76304802	0.0001
117		F138	113	0.0012	1.417	0.2038	0.00000222	0.0001	0.76296838	0.0001

APPENDIX D-3

STEPDISC ANALYSIS OF BULK GRAIN IMAGE FEATURES FOR GRAIN TYPE IDENTIFICATION ANALYSIS OF BULK GRAIN SAMPLES

1

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda	Average Squared Canonical Correlation	Prob > ASCC
1	F5		1	0.9453	1791.900	0.0001	0.05473056	0.0001	0.23631736	0.0001
2	F6		2	0.8393	540.358	0.0001	0.00879792	0.0001	0.44547635	0.0001
3	F19		3	0.8019	417.842	0.0001	0.00174323	0.0001	0.62946143	0.0001
4	F10		4	0.6635	203.128	0.0001	0.00058653	0.0001	0.67702512	0.0001
5	F32		5	0.6439	185.824	0.0001	0.00020884	0.0001	0.82815636	0.0001
6	F36		6	0.4462	82.592	0.0001	0.00011565	0.0001	0.84433295	0.0001
7	F93		7	0.4763	92.98	0.0001	0.00006057	0.0001	0.87640787	0.0001
8	F55		8	0.3607	57.548	0.0001	0.00003872	0.0001	0.89468469	0.0001
9	F17		9	0.2816	39.879	0.0001	0.00002782	0.0001	0.90004377	0.0001
10	F9		10	0.2286	30.086	0.0001	0.00002146	0.0001	0.90525328	0.0001
11	F16		11	0.2035	25.861	0.0001	0.00001709	0.0001	0.91357422	0.0001
12		F19	10	0.0091	0.930	0.4462	0.00001725	0.0001	0.91331416	0.0001
13	F7		11	0.1659	20.144	0.0001	0.00001439	0.0001	0.91845130	0.0001
14	F104		12	0.1405	16.510	0.0001	0.00001237	0.0001	0.92073288	0.0001
15	F15		13	0.1444	17.009	0.0001	0.00001058	0.0001	0.92438182	0.0001
16	F66		14	0.1117	12.640	0.0001	0.00000940	0.0001	0.92662499	0.0001
17	F94		15	0.1755	21.342	0.0001	0.00000775	0.0001	0.92791229	0.0001
18	F29		16	0.1090	12.239	0.0001	0.00000690	0.0001	0.93024823	0.0001
19	F14		17	0.2066	25.975	0.0001	0.00000548	0.0001	0.93406406	0.0001
20	F79		18	0.3866	62.715	0.0001	0.00000336	0.0001	0.93777685	0.0001
21	F67		19	0.1102	12.291	0.0001	0.00000299	0.0001	0.94008670	0.0001
22	F38		20	0.1961	24.153	0.0001	0.00000240	0.0001	0.94418166	0.0001
23	F18		21	0.0817	8.70	0.0001	0.00000221	0.0001	0.94490052	0.0001
24	F48		22	0.1053	11.597	0.0001	0.00000197	0.0001	0.94599636	0.0001
25	F96		23	0.0719	7.617	0.0001	0.00000183	0.0001	0.94707638	0.0001
26	F28		24	0.0647	6.781	0.0001	0.00000171	0.0001	0.94798468	0.0001
27	F12		25	0.0617	6.427	0.0001	0.00000161	0.0001	0.94848672	0.0001
28	F65		26	0.0556	5.740	0.0002	0.00000152	0.0001	0.94901655	0.0001
29		F66	25	0.0108	1.061	0.3754	0.00000154	0.0001	0.94891145	0.0001
30	F35		26	0.0639	6.654	0.0001	0.00000144	0.0001	0.94966391	0.0001
31	F34		27	0.0708	7.414	0.0001	0.00000134	0.0001	0.95044598	0.0001
32	F71		28	0.0557	5.724	0.0002	0.00000126	0.0001	0.95074467	0.0001
33	F88		29	0.0430	4.348	0.0019	0.00000121	0.0001	0.95175609	0.0001
34		F17	28	0.0164	1.615	0.1697	0.00000123	0.0001	0.95128320	0.0001
35	F60		29	0.0646	6.680	0.0001	0.00000115	0.0001	0.95216873	0.0001
36		F32	28	0.0128	1.250	0.2893	0.00000116	0.0001	0.95203770	0.0001
37	F100		29	0.0545	5.578	0.0002	0.00000110	0.0001	0.95242244	0.0001
38	F26		30	0.0534	5.442	0.0003	0.00000104	0.0001	0.95314679	0.0001
39	F85		31	0.0427	4.25	0.0021	0.00000100	0.0001	0.95347067	0.0001
40	F80		32	0.0654	6.723	0.0001	0.00000093	0.0001	0.95432791	0.0001
41	F68		33	0.0617	6.291	0.0001	0.00000087	0.0001	0.95528709	0.0001
42		F65	32	0.0038	0.366	0.8326	0.00000088	0.0001	0.95526041	0.0001
43	F41		33	0.0451	4.526	0.0014	0.00000084	0.0001	0.95572539	0.0001
44	F89		34	0.0415	4.134	0.0027	0.00000080	0.0001	0.95604489	0.0001

45	F57		35	0.0356	3.515	0.0078	0.00000077	0.0001	0.95614492	0.0001
46	F92		36	0.0353	3.481	0.0083	0.00000075	0.0001	0.95698348	0.0001
47	F106		37	0.0254	2.473	0.0441	0.00000073	0.0001	0.95730423	0.0001
48	F77		38	0.0241	2.334	0.0552	0.00000071	0.0001	0.95788073	0.0001
49	F84		39	0.0232	2.240	0.0642	0.00000069	0.0001	0.95802564	0.0001
50	F2		40	0.0292	2.829	0.0246	0.00000067	0.0001	0.95830301	0.0001
51	F21		41	0.0337	3.271	0.0118	0.00000065	0.0001	0.95899392	0.0001
52	F49		42	0.0242	2.320	0.0565	0.00000063	0.0001	0.95908566	0.0001
53	F53		43	0.0373	3.609	0.0067	0.00000061	0.0001	0.95940219	0.0001
54	F82		44	0.0269	2.569	0.0378	0.00000059	0.0001	0.95984642	0.0001
55	F19		45	0.0207	1.958	0.1004	0.00000058	0.0001	0.95997707	0.0001
56	F50		46	0.0225	2.132	0.0764	0.00000057	0.0001	0.96002679	0.0001
57		F92	45	0.0176	1.655	0.1599	0.00000058	0.0001	0.95966751	0.0001
58	F91		46	0.0240	2.274	0.0609	0.00000057	0.0001	0.96013877	0.0001
59	F8		47	0.0218	2.060	0.0855	0.00000055	0.0001	0.96038044	0.0001
60	F52		48	0.0226	2.124	0.0772	0.00000054	0.0001	0.96064023	0.0001
61	F61		49	0.0198	1.852	0.1184	0.00000053	0.0001	0.96079923	0.0001
62	F13		50	0.0186	1.731	0.1425	0.00000052	0.0001	0.96099494	0.0001
63	F1		51	0.0239	2.232	0.0651	0.00000051	0.0001	0.96141950	0.0001
64	F3		52	0.0391	3.698	0.0058	0.00000049	0.0001	0.96220401	0.0001
65		F96	51	0.0171	1.586	0.1773	0.00000050	0.0001	0.96205754	0.0001
66	F99		52	0.0255	2.377	0.0516	0.00000048	0.0001	0.96248823	0.0001
67		F21	51	0.0182	1.683	0.1533	0.00000049	0.0001	0.96213340	0.0001
68	F42		52	0.0195	1.812	0.1259	0.00000048	0.0001	0.96230815	0.0001
69		F71	51	0.0104	0.953	0.4336	0.00000049	0.0001	0.96216876	0.0001
70	F30		52	0.0216	2.009	0.0927	0.00000048	0.0001	0.96229249	0.0001
71	F58		53	0.0197	1.820	0.1244	0.00000047	0.0001	0.96256496	0.0001
72	F105		54	0.0199	1.836	0.1213	0.00000046	0.0001	0.96288331	0.0001
73	F21		55	0.0191	1.757	0.1370	0.00000045	0.0001	0.96321991	0.0001

APPENDIX D-4

STEPDISC ANALYSIS OF BULK GRAIN IMAGE FEATURES FOR GRADE IDENTIFICATION ANALYSIS OF BULK CWRS WHEAT SAMPLES

1

Stepwise Selection: Summary

Step	Variable Entered	Removed	Number In	Partial R**2	F Statistic	Prob > F	Average Squared Wilks' Lambda	Prob < Lambda	Canonical Correlation	Prob > ASCC
1	F80		1	0.4071	60.776	0.0001	0.59286193	0.0001	0.20356904	0.0001
2	F92		2	0.0941	9.139	0.0002	0.53708717	0.0001	0.23532677	0.0001
3	F20		3	0.1951	21.212	0.0001	0.43229176	0.0001	0.30516216	0.0001
4	F34		4	0.1767	18.670	0.0001	0.35591186	0.0001	0.34670758	0.0001
5	F13		5	0.0879	8.334	0.0004	0.32463328	0.0001	0.36390367	0.0001
6	F84		6	0.1370	13.648	0.0001	0.28017201	0.0001	0.41062602	0.0001
7	F63		7	0.0829	7.730	0.0006	0.25694097	0.0001	0.42459680	0.0001
8	F32		8	0.0613	5.551	0.0046	0.24118982	0.0001	0.45040804	0.0001
9	F89		9	0.1150	10.978	0.0001	0.21345825	0.0001	0.49276086	0.0001
10	F17		10	0.0561	4.993	0.0078	0.20148282	0.0001	0.50318549	0.0001
11	F21		11	0.0505	4.444	0.0132	0.19130054	0.0001	0.51414584	0.0001
12		F20	10	0.0140	1.189	0.3072	0.19402396	0.0001	0.51213779	0.0001
13	F19		11	0.0326	2.818	0.0626	0.18768956	0.0001	0.51672308	0.0001
14	F22		12	0.0429	3.722	0.0262	0.17963322	0.0001	0.52474300	0.0001
15	F24		13	0.0442	3.819	0.0239	0.17168479	0.0001	0.53435518	0.0001
16	F85		14	0.0338	2.870	0.0596	0.16587988	0.0001	0.53900169	0.0001
17	F30		15	0.0584	5.058	0.0074	0.15618673	0.0001	0.55153403	0.0001
18		F17	14	0.0203	1.691	0.1876	0.15942697	0.0001	0.54840781	0.0001
19	F48		15	0.0370	3.130	0.0464	0.15353081	0.0001	0.55312316	0.0001
20	F9		16	0.0435	3.682	0.0273	0.14685592	0.0001	0.55863986	0.0001
21		F19	15	0.0212	1.752	0.1766	0.15003319	0.0001	0.55610050	0.0001
22	F16		16	0.0340	2.854	0.0605	0.14492606	0.0001	0.56198361	0.0001
23	F61		17	0.0323	2.685	0.0713	0.14024901	0.0001	0.56742714	0.0001
24	F47		18	0.0340	2.820	0.0626	0.13547396	0.0001	0.57751000	0.0001
25	F33		19	0.0282	2.308	0.1027	0.13165112	0.0001	0.58497276	0.0001
26	F35		20	0.0365	2.991	0.0531	0.12684832	0.0001	0.59015260	0.0001
27		F34	19	0.0118	0.945	0.3909	0.12836548	0.0001	0.58865599	0.0001
28		F30	18	0.0198	1.607	0.2037	0.13096010	0.0001	0.58365893	0.0001
29	F51		19	0.0255	2.077	0.1287	0.12762591	0.0001	0.58717840	0.0001
30	F65		20	0.0320	2.616	0.0763	0.12353554	0.0001	0.59708449	0.0001
31	F93		21	0.0440	3.612	0.0293	0.11810176	0.0001	0.60855434	0.0001
32		F32	20	0.0170	1.359	0.2599	0.12014661	0.0001	0.60559369	0.0001

APPENDIX E-1

EVALUATIONS OF FEATURE MODELS FOR GRAIN TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL KERNELS

Parametric Method, Using 4 mof Features   14:21 Friday, February 7, 1997 83

From SPECIES	1	2	3	4	5	Total
1	17170	579	37	206	8	18000
	95.39	3.22	0.21	1.14	0.04	100.00
2	349	4961	48	639	3	6000
	5.82	82.68	0.80	10.65	0.05	100.00
3	6	174	5548	177	95	6000
	0.10	2.90	92.47	2.95	1.58	100.00
4	142	771	98	4985	4	6000
	2.37	12.85	1.63	83.08	0.07	100.00
5	0	3	348	58	5591	6000
	0.00	0.05	5.80	0.97	93.18	100.00
Total	17667	6488	6079	6065	5701	42000
Percent	42.06	15.45	14.47	14.44	13.57	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0461	0.1732	0.0753	0.1692	0.0682	0.1064
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 4 mof Features   14:21 Friday, February 7, 1997 88

From SPECIES	1	2	3	4	5	OTHER	Total
1	17096	667	28	188	6	15	18000
	94.98	3.71	0.16	1.04	0.03	0.08	100.00
2	385	4790	63	757	0	5	6000
	6.42	79.83	1.05	12.62	0.00	0.08	100.00
3	6	117	5611	117	135	14	6000
	0.10	1.95	93.52	1.95	2.25	0.23	100.00
4	124	707	83	5062	14	10	6000
	2.07	11.78	1.38	84.37	0.23	0.17	100.00
5	0	4	178	48	5768	2	6000
	0.00	0.07	2.97	0.80	96.13	0.03	100.00

Total	17611	6285	5963	6172	5923	46	42000
Percent	41.93	14.96	14.20	14.70	14.10	0.11	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0502	0.2017	0.0648	0.1563	0.0387	0.1023
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 8 mof Features 14:21 Friday, February 7, 1997 97

From SPECIES	1	2	3	4	5	Total
1	16949	968	9	69	5	18000
	94.16	5.38	0.05	0.38	0.03	100.00
2	259	5204	16	521	0	6000
	4.32	86.73	0.27	8.68	0.00	100.00
3	5	48	5723	121	103	6000
	0.08	0.80	95.38	2.02	1.72	100.00
4	39	418	61	5458	24	6000
	0.65	6.97	1.02	90.97	0.40	100.00
5	0	2	193	40	5765	6000
	0.00	0.03	3.22	0.67	96.08	100.00
Total	17252	6640	6002	6209	5897	42000
Percent	41.08	15.81	14.29	14.78	14.04	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0584	0.1327	0.0462	0.0903	0.0392	0.0733
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17376	504	14	89	5	12	18000
	96.53	2.80	0.08	0.49	0.03	0.07	100.00
2	315	5192	26	464	0	3	6000
	5.25	86.53	0.43	7.73	0.00	0.05	100.00
3	4	50	5763	91	87	5	6000
	0.07	0.83	96.05	1.52	1.45	0.08	100.00
4	50	429	44	5464	10	3	6000
	0.83	7.15	0.73	91.07	0.17	0.05	100.00
5	0	8	84	46	5859	3	6000
	0.00	0.13	1.40	0.77	97.65	0.05	100.00
Total	17745	6183	5931	6154	5961	26	42000
Percent	42.25	14.72	14.12	14.65	14.19	0.06	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0347	0.1347	0.0395	0.0893	0.0235	0.0643
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 12 mof Features 14:21 Friday, February 7, 1997 111

From SPECIES	1	2	3	4	5	Total
1	16902	962	7	123	6	18000
	93.90	5.34	0.04	0.68	0.03	100.00
2	260	4891	18	831	0	6000
	4.33	81.52	0.30	13.85	0.00	100.00
3	6	38	5746	85	125	6000
	0.10	0.63	95.77	1.42	2.08	100.00
4	23	245	48	5658	26	6000
	0.38	4.08	0.80	94.30	0.43	100.00
5	0	2	168	41	5789	6000
	0.00	0.03	2.80	0.68	96.48	100.00

Total	17191	6138	5987	6738	5946	42000
Percent	40.93	14.61	14.25	16.04	14.16	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0610	0.1848	0.0423	0.0570	0.0352	0.0761
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHR	Total
1	17434	460	11	86	4	5	18000
	96.86	2.56	0.06	0.48	0.02	0.03	100.00
2	310	5247	10	433	0	0	6000
	5.17	87.45	0.17	7.22	0.00	0.00	100.00
3	2	34	5799	70	89	6	6000
	0.03	0.57	96.65	1.17	1.48	0.10	100.00
4	47	452	35	5457	4	5	6000
	0.78	7.53	0.58	90.95	0.07	0.08	100.00
5	0	9	75	41	5875	0	6000
	0.00	0.15	1.25	0.68	97.92	0.00	100.00
Total	17793	6202	5930	6087	5972	16	42000
Percent	42.36	14.77	14.12	14.49	14.22	0.04	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0314	0.1255	0.0335	0.0905	0.0208	0.0604
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	16988	868	13	125	6	18000
	94.38	4.82	0.07	0.69	0.03	100.00
2	244	5036	12	708	0	6000
	4.07	83.93	0.20	11.80	0.00	100.00
3	7	27	5765	101	100	6000
	0.12	0.45	96.08	1.68	1.67	100.00
4	29	277	49	5621	24	6000
	0.48	4.62	0.82	93.68	0.40	100.00
5	0	2	113	44	5841	6000
	0.00	0.03	1.88	0.73	97.35	100.00
Total	17268	6210	5952	6599	5971	42000
Percent	41.11	14.79	14.17	15.71	14.22	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0562	0.1607	0.0392	0.0632	0.0265	0.0691
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17463	438	11	78	3	7	18000
	97.02	2.43	0.06	0.43	0.02	0.04	100.00
2	296	5337	6	361	0	0	6000
	4.93	88.95	0.10	6.02	0.00	0.00	100.00
3	4	23	5875	55	41	2	6000
	0.07	0.38	97.92	0.92	0.68	0.03	100.00
4	41	473	40	5432	11	3	6000
	0.68	7.88	0.67	90.53	0.18	0.05	100.00
5	0	6	42	45	5906	1	6000
	0.00	0.10	0.70	0.75	98.43	0.02	100.00

Total	17804	6277	5974	5971	5961	13	42000
Percent	42.39	14.95	14.22	14.22	14.19	0.03	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0298	0.1105	0.0208	0.0947	0.0157	0.0543
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 20 mof Features 14:21 Friday, February 7, 1997 142

From SPECIES	1	2	3	4	5	Total
1	17006	814	11	164	5	18000
	94.48	4.52	0.06	0.91	0.03	100.00
2	209	4933	10	848	0	6000
	3.48	82.22	0.17	14.13	0.00	100.00
3	5	28	5791	93	83	6000
	0.08	0.47	96.52	1.55	1.38	100.00
4	19	226	51	5690	14	6000
	0.32	3.77	0.85	94.83	0.23	100.00
5	0	2	117	50	5831	6000
	0.00	0.03	1.95	0.83	97.18	100.00
Total	17239	6003	5980	6845	5933	42000
Percent	41.05	14.29	14.24	16.30	14.13	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0552	0.1778	0.0348	0.0517	0.0282	0.0695
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 20 mof Features 14:21 Friday, February 7, 1997
148

From SPECIES	1	2	3	4	5	OTHER	Total
1	17473	437	11	69	4	6	18000
	97.07	2.43	0.06	0.38	0.02	0.03	100.00
2	281	5386	6	327	0	0	6000
	4.68	89.77	0.10	5.45	0.00	0.00	100.00
3	3	27	5859	63	44	4	6000
	0.05	0.45	97.65	1.05	0.73	0.07	100.00
4	38	534	27	5389	8	4	6000
	0.63	8.90	0.45	89.82	0.13	0.07	100.00
5	0	10	41	37	5905	7	6000
	0.00	0.17	0.68	0.62	98.42	0.12	100.00
Total	17795	6394	5944	5885	5961	21	42000
Percent	42.37	15.22	14.15	14.01	14.19	0.05	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0293	0.1023	0.0235	0.1018	0.0158	0.0546
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 24 mof Features 14:21 Friday, February 7, 1997 161

From SPECIES	1	2	3	4	5	Total
1	16942	864	12	177	5	18000
	94.12	4.80	0.7	0.98	0.03	100.00
2	199	5047	11	743	0	6000
	3.32	84.12	0.18	12.38	0.00	100.00
3	6	25	5806	77	86	6000
	0.10	0.42	96.77	1.28	1.43	100.00
4	21	222	45	5697	15	6000
	0.35	3.70	0.75	94.95	0.25	100.00
5	0	5	97	44	5854	6000
	0.00	0.08	1.62	0.73	97.57	100.00

Total	17168	6163	5971	6738	5960	42000
Percent	40.88	14.67	14.22	16.04	14.19	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0588	0.1588	0.0323	0.0505	0.0243	0.0650
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17495	404	10	76	4	11	18000
	97.19	2.24	0.06	0.42	0.02	0.06	100.00
2	282	5382	4	332	0	0	6000
	4.70	89.70	0.07	5.53	0.00	0.00	100.00
3	2	20	5873	60	41	4	6000
	0.03	0.33	97.88	1.00	0.68	0.07	100.00
4	31	503	25	5434	5	2	6000
	0.52	8.38	0.42	90.57	0.08	0.03	100.00
5	0	16	43	29	5909	3	6000
	0.00	0.27	0.72	0.48	98.48	0.05	100.00
Total	17810	6325	5955	5931	5959	20	42000
Percent	42.40	15.06	14.18	14.12	14.19	0.05	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0281	0.1030	0.0212	0.0943	0.0152	0.0523
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 28 mof Features 14:21 Friday, February 7, 1997 183

From SPECIES	1	2	3	4	5	Total
1	16946	870	10	169	5	18000
	94.14	4.83	0.06	0.94	0.03	100.00
2	195	5084	11	710	0	6000
	3.25	84.73	0.18	11.83	0.00	100.00
3	7	24	5810	74	85	6000
	0.12	0.40	96.83	1.23	1.42	100.00
4	21	215	43	5706	15	6000
	0.35	3.58	0.72	95.10	0.25	100.00
5	0	5	94	45	5856	6000
	0.00	0.08	1.57	0.75	97.60	100.00
Total	17169	6198	5968	6704	5961	42000
Percent	40.88	14.76	14.21	15.96	14.19	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0586	0.1527	0.0317	0.0490	0.0240	0.0632
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 28 mof Features 14:21 Friday, February 7, 1997
192

From SPECIES	1	2	3	4	5	OTHER	Total
1	17477	422	10	81	4	6	18000
	97.09	2.34	0.06	0.45	0.02	0.03	100.00
2	296	5376	5	320	0	3	6000
	4.93	89.60	0.08	5.33	0.00	0.05	100.00
3	3	28	5874	53	37	5	6000
	0.05	0.47	97.90	0.88	0.62	0.08	100.00
4	46	532	30	5382	8	2	6000
	0.77	8.87	0.50	89.70	0.13	0.03	100.00
5	0	16	34	28	5910	12	6000

	0.00	0.27	0.57	0.47	98.50	0.20	100.00
Total	17822	6374	5953	5864	5959	28	42000
Percent	42.43	15.18	14.17	13.96	14.19	0.07	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0291	0.1040	0.0210	0.1030	0.0150	0.0544
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 4 color Features 14:21 Friday, February 7, 1997 201

From SPECIES	1	2	3	4	5	Total
1	17441	505	24	30	0	18000
	96.89	2.81	0.13	0.17	0.00	100.00
2	154	5279	544	23	0	6000
	2.57	87.98	9.07	0.38	0.00	100.00
3	4	418	5287	89	202	6000
	0.07	6.97	88.12	1.48	3.37	100.00
4	51	134	75	5739	1	6000
	0.85	2.23	1.25	95.65	0.02	100.00
5	0	0	97	15	5888	6000
	0.00	0.00	1.62	0.25	98.13	100.00
Total	17650	6336	6027	5896	6091	42000
Percent	42.02	15.09	14.35	14.04	14.50	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0311	0.1202	0.1188	0.0435	0.0187	0.0664
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 4 color Features 14:21 Friday, February 7, 1997
206

From SPECIES	1	2	3	4	5	OTHER	Total
1	17514	400	24	57	1	4	18000
	97.30	2.22	0.13	0.32	0.01	0.02	100.00
2	103	5417	445	29	1	5	6000
	1.72	90.28	7.42	0.48	0.02	0.08	100.00
3	2	440	5273	91	189	5	6000
	0.03	7.33	87.88	1.52	3.15	0.08	100.00
4	33	69	96	5794	2	6	6000
	0.55	1.15	1.60	96.57	0.03	0.10	100.00
5	0	0	112	7	5880	1	6000
	0.00	0.00	1.87	0.12	98.00	0.02	100.00
Total	17652	6326	5950	5978	6073	21	42000
Percent	42.03	15.06	14.17	14.23	14.46	0.05	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0270	0.0972	0.1212	0.0343	0.0200	0.0599
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 8 color Features 14:21 Friday, February 7, 1997 215

From SPECIES	1	2	3	4	5	Total
1	15605	767	223	1405	0	18000
	86.69	4.26	1.24	7.81	0.00	100.00
2	73	3128	1010	1785	4	6000
	1.22	52.13	16.83	29.75	0.07	100.00
3	34	25	5370	316	255	6000
	0.57	0.42	89.50	5.27	4.25	100.00
4	4	13	47	5926	10	6000
	0.07	0.22	0.78	98.77	0.17	100.00
5	5	0	60	9	5926	6000
	0.08	0.00	1.00	0.15	98.77	100.00

Total	15721	3933	6710	9441	6195	42000
Percent	37.43	9.36	15.98	22.48	14.75	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.1331	0.4787	0.1050	0.0123	0.0123	0.1483
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17676	250	20	52	0	2	18000
	98.20	1.39	0.11	0.2	0.00	0.01	100.00
2	96	5630	234	33	3	4	6000
	1.60	93.83	3.90	0.55	0.05	0.07	100.00
3	3	200	5599	41	153	4	6000
	0.05	3.33	93.32	0.68	2.55	0.07	100.00
4	33	91	42	5831	0	3	6000
	0.55	1.52	0.70	97.18	0.00	0.05	100.00
5	0	0	107	5	5886	2	6000
	0.00	0.00	1.78	0.08	98.10	0.03	100.00
Total	17808	6171	6002	5962	6042	15	42000
Percent	42.40	14.69	14.29	14.20	14.39	0.04	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0180	0.0617	0.0668	0.0282	0.0190	0.0387
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 12 color Features 14:21 Friday, February 7, 1997 229

From SPECIES	1	2	3	4	5	Total
1	14806	2676	278	240	0	18000
	82.26	14.87	1.54	1.33	0.00	100.00
2	49	4840	663	448	0	6000
	0.82	80.67	11.05	7.47	0.00	100.00
3	30	47	5571	155	197	6000
	0.50	0.78	92.85	2.58	3.28	100.00
4	10	14	32	5943	1	6000
	0.17	0.23	0.53	99.05	0.02	100.00
5	3	0	73	5	5919	6000
	0.05	0.00	1.22	0.08	98.65	100.00
Total	14898	7577	6617	6791	6117	42000
Percent	35.47	18.04	15.75	16.17	14.56	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.1774	0.1933	0.0715	0.0095	0.0135	0.0931
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 12 color Features 14:21 Friday, February 7, 1997
234

From SPECIES	1	2	3	4	5	OTHER	Total
1	17738	216	6	35	0	5	18000
	98.54	1.20	0.03	0.19	0.00	0.03	100.00
2	57	5775	143	22	1	2	6000
	0.95	96.25	2.38	0.37	0.02	0.03	100.00
3	1	90	5763	21	124	1	6000
	0.02	1.50	96.05	0.35	2.07	0.02	100.00
4	16	72	24	5887	0	1	6000
	0.27	1.20	0.40	98.12	0.00	0.02	100.00
5	0	0	85	4	5911	0	6000
	0.00	0.00	1.42	0.07	98.52	0.00	100.00

Total	17812	6153	6021	5969	6036	9	42000
Percent	42.41	14.65	14.34	14.21	14.37	0.02	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0146	0.0375	0.0395	0.0188	0.0148	0.0250
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 16 color Features 14:21 Friday, February 7, 1997 244

From SPECIES	1	2	3	4	5	Total
1	13875	3512	334	279	0	18000
	77.08	19.51	1.86	1.55	0.00	100.00
2	55	4664	914	367	0	6000
	0.92	77.73	15.23	6.12	0.00	100.00
3	33	51	5660	96	160	6000
	0.55	0.85	94.33	1.60	2.67	100.00
4	29	7	30	5930	4	6000
	0.48	0.12	0.50	98.83	0.07	100.00
5	4	0	79	12	5905	6000
	0.07	0.00	1.32	0.20	98.42	100.00
Total	13996	8234	7017	6684	6069	42000
Percent	33.32	19.60	16.71	15.91	14.45	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.2292	0.2227	0.0567	0.0117	0.0158	0.1072
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 16 color Features 14:21 Friday, February 7, 1997

250

From SPECIES	1	2	3	4	5	OTHER	Total
1	17770 98.72	179 0.99	8 0.04	40 0.22	1 0.01	2 0.01	18000 100.00
2	56 0.93	5783 96.38	133 2.22	28 0.47	0 0.00	0 0.00	6000 100.00
3	1 0.02	90 1.50	5763 96.05	19 0.32	126 2.10	1 0.02	6000 100.00
4	16 0.27	67 1.12	24 0.40	5890 98.17	2 0.03	1 0.02	6000 100.00
5	0 0.00	1 0.02	65 1.08	3 0.05	5931 98.85	0 0.00	6000 100.00
Total Percent	17843 42.48	6120 14.57	5993 14.27	5980 14.24	6060 14.43	4 0.01	42000 100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total	
Rate	0.0128	0.0362	0.0395	0.0183	0.0115	0.0237	
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Parametric Method, Using 20 color Features 14:21 Friday, February 7, 1997 260

From SPECIES	1	2	3	4	5	Total
1	13022 72.34	4202 23.34	447 2.48	329 1.83	0 0.00	18000 100.00
2	56 0.93	4508 75.13	973 16.22	463 7.72	0 0.00	6000 100.00
3	40 0.67	19 0.32	5675 94.58	115 1.92	151 2.52	6000 100.00
4	29 0.48	11 0.18	23 0.38	5933 98.88	4 0.07	6000 100.00
5	5 0.08	0 0.00	80 1.33	14 0.23	5901 98.35	6000 100.00

Total	13152	8740	7198	6854	6056	42000
Percent	31.31	20.81	17.14	16.32	14.42	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.2766	0.2487	0.0542	0.0112	0.0165	0.1214
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 20 color Features 14:21 Friday, February 7, 1997
266

From SPECIES	1	2	3	4	5	OTHER	Total
1	17751	206	12	27	1	3	18000
	98.62	1.14	0.07	0.15	0.01	0.02	100.00
2	63	5813	105	17	1	1	6000
	1.05	96.88	1.75	0.28	0.02	0.02	100.00
3	1	94	5803	12	87	3	6000
	0.02	1.57	96.72	0.20	1.45	0.05	100.00
4	15	66	15	5899	1	4	6000
	0.25	1.10	0.25	98.32	0.02	0.07	100.00
5	0	0	65	3	5932	0	6000
	0.00	0.00	1.08	0.05	98.87	0.00	100.00
Total	17830	6179	6000	5958	6022	11	42000
Percent	42.45	14.71	14.29	14.19	14.34	.03	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0138	0.0312	0.0328	0.0168	0.0113	0.0212
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 24 color Features 14:21 Friday, February 7, 1997 279

From SPECIES	1	2	3	4	5	Total
1	13968	3235	48	316	0	18000
	77.60	17.97	2.67	1.76	0.00	100.00
2	62	4595	1019	324	0	6000
	1.03	76.58	16.98	5.40	0.00	100.00
3	39	40	5659	97	165	6000
	0.65	0.67	94.32	1.62	2.75	100.00
4	27	24	46	5897	6	6000
	0.45	0.40	0.77	98.28	0.10	100.00
5	6	0	102	12	5880	6000
	0.10	0.00	1.70	0.20	98.00	100.00
Total	14102	7894	7307	6646	6051	42000
Percent	33.58	18.80	17.40	15.82	14.41	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.2240	0.2342	0.0568	0.0172	0.0200	0.1104
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 24 color Features 14:21 Frida, February 7, 1997 288

From SPECIES	1	2	3	4	5	OTHER	Total
1	17706	239	12	39	1	3	18000
	98.37	1.33	0.07	0.22	0.01	0.02	100.00
2	54	5807	114	24	0	1	6000
	0.90	96.78	1.90	0.40	0.00	0.02	100.00
3	1	91	5810	11	87	0	6000
	0.02	1.52	96.83	0.18	1.45	0.00	100.00
4	16	80	18	5880	1	5	6000
	0.27	1.33	0.30	98.00	0.02	0.08	100.00
5	0	1	0	3	5924	2	6000
	0.00	0.02	1.17	0.05	98.73	0.03	100.00

Total	17777	6218	6024	5957	6013	11	42000
Percent	42.33	14.80	14.34	14.18	14.32	0.03	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0163	0.0322	0.0317	0.0200	0.0127	0.0226
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 28 color Features 14:21 Friday, ebruary 7, 1997 301

From SPECIES	1	2	3	4	5	Total
1	14318	2824	517	341	0	18000
	79.54	15.69	2.87	1.89	0.00	100.00
2	70	4611	959	360	0	6000
	1.17	76.85	15.98	6.00	0.00	100.00
3	41	45	5656	106	152	6000
	0.68	0.75	94.27	1.77	2.53	100.00
4	27	15	55	5902	1	6000
	0.45	0.25	0.92	98.37	0.02	100.00
5	4	0	112	12	5872	6000
	0.07	0.00	1.87	0.20	97.87	100.00
Total	14460	7495	7299	6721	6025	42000
Percent	34.43	17.85	17.38	16.00	14.35	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.2046	0.2315	0.0573	0.0163	0.0213	0.1062
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 28 color Features 14:21 Friday, February 7, 1997

310

From SPECIES	1	2	3	4	5	OTHER	Total
1	17704	249	10	34	1	2	18000
	98.36	1.38	0.06	0.19	0.01	0.01	100.00
2	54	5807	118	18	0	3	6000
	0.90	96.78	1.97	0.30	0.00	0.05	100.00
3	1	90	5803	12	94	0	6000
	0.02	1.50	96.72	0.20	1.57	0.00	100.00
4	16	79	19	5881	2	3	6000
	0.27	1.32	0.32	98.02	0.03	0.05	100.00
5	0	0	57	2	5941	0	6000
	0.00	0.00	0.95	0.03	99.02	0.00	100.00
Total	17775	6225	6007	5947	6038	8	42000
Percent	42.32	14.82	14.30	14.16	14.38	0.02	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0164	0.0322	0.0328	0.0198	0.0098	0.0222
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 4 combined Features 14:21 Friday, February 7, 1997

319

From SPECIES	1	2	3	4	5	Total
1	17170	579	37	206	8	18000
	95.39	3.22	0.21	1.14	0.04	100.00
2	349	4961	48	639	3	6000
	5.82	82.68	0.80	10.65	0.05	100.00
3	6	174	5548	177	95	6000
	0.10	2.90	92.47	2.95	1.58	100.00
4	142	771	98	4985	4	6000
	2.37	12.85	1.63	83.08	0.07	100.00
5	0	3	348	58	5591	6000
	0.00	0.05	5.80	0.97	93.18	100.00

Total	17667	6488	6079	6065	5701	42000
Percent	42.06	15.45	14.47	14.44	13.57	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0461	0.1732	0.0753	0.1692	0.0682	0.1064
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17096	667	28	188	6	15	18000
	94.98	3.71	0.16	1.04	0.03	0.08	100.00
2	385	4790	63	757	0	5	6000
	6.42	79.83	1.05	12.62	0.00	0.08	100.00
3	6	117	5611	117	135	14	6000
	0.10	1.95	93.52	1.95	2.25	0.23	100.00
4	124	707	83	5062	14	10	6000
	2.07	11.78	1.38	84.37	0.23	0.17	100.00
5	0	4	178	48	5768	2	6000
	0.00	0.07	2.97	0.80	96.13	0.03	100.00
Total	17611	6285	5963	6172	5923	46	42000
Percent	41.93	14.96	14.20	14.70	14.10	0.11	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0502	0.2017	0.0648	0.1563	0.0387	0.1023
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 8 combined Features 14:21 Friday, February 7, 1997

333

From SPECIES	1	2	3	4	5	Total
1	16622	1200	18	156	4	18000
	92.34	6.67	0.10	0.87	0.02	100.00
2	303	4925	41	731	0	6000
	5.05	82.08	0.68	12.18	0.00	100.00
3	4	61	5756	85	94	6000
	0.07	1.02	95.93	1.42	1.57	100.00
4	41	216	69	5666	8	6000
	0.68	3.60	1.15	94.43	0.13	100.00
5	0	2	156	0	5842	6000
	0.00	0.03	2.60	0.00	97.37	100.00
Total	16970	6404	6040	6638	5948	42000
Percent	40.40	15.25	14.38	15.80	14.16	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0766	0.1792	0.0407	0.0557	0.0263	0.0757
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 8 combined Features 14:21 Friday, February 7, 1997

338

From SPECIES	1	2	3	4	5	OTHER	Total
1	17245	604	16	120	5	10	18000
	95.81	3.36	0.09	0.67	0.03	0.06	100.00
2	374	5159	18	446	1	2	6000
	6.23	85.98	0.30	7.43	0.02	0.03	100.00
3	3	63	5769	81	79	5	6000
	0.05	1.05	96.15	1.35	1.32	0.08	100.00
4	48	357	31	5559	2	3	6000
	0.80	5.95	0.52	92.65	0.03	0.05	100.00
5	0	5	57	5	5928	5	6000
	0.00	0.08	0.95	0.08	98.80	0.08	100.00

Total	17670	6188	5891	6211	6015	25	42000
Percent	42.07	14.73	14.03	14.79	14.32	0.06	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0419	0.1402	0.0385	0.0735	0.0120	0.0612
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 12 combined Features 14:21 Friday, February 7, 1997
347

From SPECIES	1	2	3	4	5	Total
1	17637	243	2	115	3	18000
	97.98	1.35	0.01	0.64	0.02	100.00
2	202	5704	14	80	0	6000
	3.37	95.07	0.23	1.33	0.00	100.00
3	4	29	5906	2	59	6000
	0.07	0.48	98.43	0.03	0.98	100.00
4	29	84	32	5854	1	6000
	0.48	1.40	0.53	97.57	0.02	100.00
5	0	2	99	0	5899	6000
	0.00	0.03	1.65	0.00	98.32	100.00
Total	17872	6062	6053	6051	5962	42000
Percent	42.55	14.43	14.41	14.41	14.20	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0202	0.0493	0.0157	0.0243	0.0168	0.0253
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 12 combined Features 14:21 Friday, February 7, 1997

352

From SPECIES	1	2	3	4	5	OTHER	Total
1	17816	138	1	40	3	2	18000
	98.98	0.77	0.01	0.22	0.02	0.01	100.00
2	84	5878	4	33	0	1	6000
	1.40	97.97	0.07	0.55	0.00	0.02	100.00
3	1	21	5930	3	45	0	6000
	0.02	0.35	98.83	0.05	0.75	0.00	100.00
4	21	75	1	5903	0	0	6000
	0.35	1.25	0.02	98.38	0.00	0.00	100.00
5	0	7	28	0	5964	1	6000
	0.00	0.12	0.47	0.00	99.40	0.02	100.00
Total	17922	6119	5964	5979	6012	4	42000
Percent	42.67	14.57	14.20	14.24	14.31	0.01	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0102	0.0203	0.0117	0.0162	0.0060	0.0129
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 16 combined Features 14:21 Friday, February 7, 1997

362

From SPECIES	1	2	3	4	5	Total
1	17476	216	0	308	0	18000
	97.09	1.20	0.00	1.71	0.00	100.00
2	92	5545	21	342	0	6000
	1.53	92.42	0.35	5.70	0.00	100.00
3	15	17	5890	3	75	6000
	0.25	0.28	98.17	0.05	1.25	100.00
4	19	49	24	5906	2	6000
	0.32	0.82	0.40	98.43	0.03	100.00
5	0	3	60	0	5937	6000
	0.00	0.05	1.00	0.00	98.95	100.00

Total	17602	5830	5995	6559	6014	42000
Percent	41.91	13.88	14.27	15.62	14.32	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SECIES:

	1	2	3	4	5	Total
Rate	0.0291	0.0758	0.0183	0.0157	0.0105	0.0299
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	17867	110	1	20	2	18000
	99.26	0.61	0.01	0.11	0.01	100.00
2	63	5909	3	24	1	6000
	1.05	98.48	0.05	0.40	0.02	100.00
3	1	21	5939	2	37	6000
	0.02	0.35	98.98	0.03	0.62	100.00
4	16	56	1	5927	0	6000
	0.27	0.93	0.02	98.78	0.00	100.00
5	0	6	26	1	5967	6000
	0.00	0.10	0.43	0.02	99.45	100.00
Total	17947	6102	5970	5974	6007	42000
Percent	42.73	14.53	14.21	14.22	14.30	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0074	0.0152	0.0102	0.0122	0.0055	0.0101
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 20 combined Features 14:21 Friday, February 7, 1997

378

From SPECIES	1	2	3	4	5	Total
1	17500	230	0	270	0	18000
	97.22	1.28	0.00	1.50	0.00	100.00
2	97	5580	18	305	0	6000
	1.62	93.00	0.30	5.08	0.00	100.00
3	15	21	5869	3	92	6000
	0.25	0.35	97.82	0.05	1.53	100.00
4	17	55	21	5906	1	6000
	0.28	0.92	0.35	98.43	0.02	100.00
5	0	3	54	0	5943	6000
	0.00	0.05	0.90	0.00	99.05	100.00
Total	17629	5889	5962	6484	6036	42000
Percent	41.97	14.02	14.20	15.44	14.37	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0278	0.0700	0.0218	0.0157	0.0095	0.0290
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 20 combined Features 14:21 Friday, February 7, 1997

384

From SPECIES	1	2	3	4	5	OTHER	Total
1	17851	114	1	32	2	0	18000
	99.17	0.63	0.01	0.18	0.01	0.00	100.00
2	68	5902	2	27	0	1	6000
	1.13	98.37	0.03	0.45	0.00	0.02	100.00
3	1	14	5950	2	33	0	6000
	0.02	0.23	99.17	0.03	0.55	0.00	100.00
4	11	60	1	5928	0	0	6000
	0.18	1.00	0.02	98.80	0.00	0.00	100.00
5	0	6	28	2	5964	0	6000
	0.00	0.10	0.47	0.03	99.40	0.00	100.00

Total	17931	6096	5982	5991	5999	1	42000
Percent	42.69	14.51	14.24	14.26	14.28	0.00	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0083	0.0163	0.0083	0.0120	0.0060	0.0102
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 24 combined Features 14:21 Friday, February 7, 1997

397

From SPECIES	1	2	3	4	5	Total
1	17540	198	0	262	0	18000
	97.44	1.10	0.00	1.46	0.00	100.00
2	96	5574	18	312	0	6000
	1.60	92.90	0.30	5.20	0.00	100.00
3	15	18	5909	5	53	6000
	0.25	0.30	98.48	0.08	0.88	100.00
4	17	52	23	5907	1	6000
	0.28	0.87	0.38	98.45	0.02	100.00
5	0	5	50	0	5945	6000
	0.00	0.08	0.83	0.00	99.08	100.00
Total	17668	5847	6000	6486	5999	42000
Percent	42.07	13.92	14.29	15.44	14.28	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0256	0.0710	0.0152	0.0155	0.0092	0.0273
Priors	0.200	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 24 combined Features 14:21 Friday, February 7, 1997

406

From SPECIES	1	2	3	4	5	OTHER	Total
1	17864	106	1	27	2	0	18000
	99.24	0.59	0.01	0.15	0.01	0.00	100.00
2	67	5905	1	26	0	1	6000
	1.12	98.42	0.02	0.43	0.00	0.02	100.00
3	1	16	5971	0	11	1	6000
	0.02	0.27	99.52	0.00	0.18	0.02	100.00
4	18	70	1	5911	0	0	6000
	0.30	1.17	0.02	9.52	0.00	0.00	100.00
5	0	5	27	3	5965	0	6000
	0.00	0.08	0.45	0.05	99.42	0.00	100.00
Total	17950	6102	6001	5967	5978	2	42000
Percent	42.74	14.53	14.29	14.21	14.23	0.00	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0076	0.0158	0.0048	0.0148	0.0058	0.0098
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 28 combined Features 14:21 Friday, February 7, 1997

419

From SPECIES	1	2	3	4	5	Total
1	17555	195	0	250	0	18000
	97.53	1.08	0.00	1.39	0.00	100.00
2	258	4795	36	910	1	6000
	4.30	79.92	0.60	15.17	0.02	100.00
3	18	16	5882	10	74	6000
	0.30	0.27	98.03	0.17	1.23	100.00
4	22	35	42	5900	1	6000
	0.37	0.58	0.70	98.33	0.02	100.00
5	0	3	57	0	5940	6000
	0.00	0.05	0.95	0.00	99.00	100.00

Total	17853	5044	6017	7070	6016	42000
Percent	42.51	12.01	14.33	16.83	14.32	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0247	0.2008	0.0197	0.0167	0.0100	0.0544
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	OTHER	Total
1	17870	103	1	24	2	0	18000
	99.28	0.57	0.01	0.13	0.01	0.00	100.00
2	69	5909	2	17	2	1	6000
	1.15	98.48	0.03	0.28	0.03	0.02	100.00
3	1	14	5969	0	16	0	6000
	0.02	0.23	99.48	0.00	0.27	0.00	100.00
4	10	75	1	5913	0	1	6000
	0.17	1.25	0.02	98.55	0.00	0.02	100.00
5	0	6	24	3	5967	0	6000
	0.00	0.10	0.40	0.05	99.45	0.00	100.00
Total	17950	6107	5997	5957	5987	2	42000
Percent	42.74	14.54	14.28	14.18	14.25	0.00	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000		

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0072	0.0152	0.0052	0.0145	0.0055	0.0095
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

APPENDIX E-2

EVALUATIONS OF FEATURE MODELS FOR DAMAGE TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL CWRS WHEAT KERNELS

Parametric Method, Using 4 mof Features 16:07 Sunday, February 16. 1997 9

From SPECIES	1	2	3	4	5	6	7	Total
1	546	20	272	16	72	45	29	1000
	54.60	2.00	27.20	1.60	7.20	4.50	2.90	100.00
2	71	598	134	113	6	14	64	1000
	7.10	59.80	13.40	11.30	0.60	1.40	6.40	100.00
3	244	14	554	55	78	29	26	1000
	24.40	1.40	55.40	5.50	7.80	2.90	2.60	100.00
4	5	67	24	759	4	55	86	1000
	0.50	6.70	2.40	75.90	0.40	5.50	8.60	100.00
5	32	5	47	5	786	48	77	1000
	3.20	0.50	4.70	0.50	78.60	4.80	7.70	100.00
6	91	28	105	84	292	207	193	1000
	9.10	2.80	10.50	8.40	29.20	20.70	19.30	100.00
7	26	29	65	130	288	76	386	1000
	2.60	2.90	6.50	13.00	28.80	7.60	38.60	100.00

NonParametric Method, Using 4 mof Features 16:07 Sunday, February 16, 1997
15

From SPECIES	1	2	3	4	5
1	556	25	204	14	42
	55.60	2.50	20.40	1.40	4.20
2	60	687	75	85	5
	6.00	68.70	7.50	8.50	0.50
3	210	32	526	38	46
	21.00	3.20	52.60	3.80	4.60
4	4	47	10	779	2
	0.40	4.70	1.00	77.90	0.20
5	27	5	34	12	664
	2.70	0.50	3.40	1.20	66.40
6	69	18	68	81	145
	6.90	1.80	6.80	8.10	14.50
7	23	21	29	115	134
	2.30	2.10	2.90	11.50	13.40

From SPECIES	6	7	OTHER	Total
1	62 6.20	26 2.60	71 7.10	1000 100.00
2	11 1.10	26 2.60	51 5.10	1000 100.00
3	63 6.30	21 2.10	64 6.40	1000 100.00
4	53 5.30	68 6.80	37 3.70	1000 100.00
5	103 10.30	100 10.00	55 5.50	1000 100.00
6	378 37.80	161 16.10	80 8.00	1000 100.00
7	127 12.70	463 46.30	88 8.80	1000 100.00

Parametric Method, Using 8 mof Features 16:07 Sunday, February 16, 1997 26

From SPECIES	1	2	3	4	5	6	7	Total
1	560 56.00	27 2.70	266 26.60	7 0.70	64 6.40	22 2.20	54 5.40	1000 100.00
2	62 6.20	668 66.80	100 10.00	71 7.10	15 1.50	40 4.00	44 4.40	1000 100.00
3	263 26.30	16 1.60	549 54.90	24 2.40	72 7.20	21 2.10	55 5.50	1000 100.00
4	5 0.50	52 5.20	46 4.60	708 70.80	8 0.80	35 3.50	146 14.60	1000 100.00
5	25 2.50	11 1.10	53 5.30	10 1.00	771 77.10	17 1.70	113 11.30	1000 100.00
6	119 11.90	35 3.50	111 11.10	43 4.30	312 31.20	104 10.40	276 27.60	1000 100.00
7	25 2.50	29 2.90	73 7.30	40 4.00	206 20.60	28 2.80	599 59.90	1000 100.00

NonParametric Method, Using 8 mof Features 16:07 Sunday, February 16, 1997

33

From SPECIES	1	2	3	4	5
1	518	26	236	11	40
	51.80	2.60	23.60	1.10	4.00
2	54	717	55	58	14
	5.40	71.70	5.50	5.80	1.40
3	201	25	533	42	48
	20.10	2.50	53.30	4.20	4.80
4	5	23	30	793	5
	0.50	2.30	3.00	79.30	0.50
5	21	4	28	9	684
	2.10	0.40	2.80	0.90	68.40
6	66	22	61	58	168
	6.60	2.20	6.10	5.80	16.80
7	31	16	35	61	132
	3.10	1.60	3.50	6.10	13.20

NonParametric Method, Using 8 mof Features 16:07 Sunday, February 16, 1997

34

From SPECIES	6	7	OTHER	Total
1	71	36	62	1000
	7.10	3.60	6.20	100.00
2	28	20	54	1000
	2.80	2.00	5.40	100.00
3	32	45	74	1000
	3.20	4.50	7.40	100.00
4	46	75	23	1000
	4.60	7.50	2.30	100.00
5	92	107	55	1000
	9.20	10.70	5.50	100.00
6	385	169	71	1000
	38.50	16.90	7.10	100.00
7	103	563	59	1000
	10.30	56.30	5.90	100.00

Parametric Method, Using 12 mof Features 16:07 Sunday. February 16, 1997 44

From SPECIES	1	2	3	4	5	6	7	Total
1	505	9	383	8	50	15	30	1000
	50.50	0.90	38.30	0.80	5.00	1.50	3.00	100.00
2	70	617	120	92	19	51	31	1000
	7.00	61.70	12.00	9.20	1.90	5.10	3.10	100.00
3	204	10	668	20	53	15	30	1000
	20.40	1.00	66.80	2.00	5.30	1.50	3.00	100.00
4	7	20	93	729	14	25	112	1000
	0.70	2.00	9.30	72.90	1.40	2.50	11.20	100.00
5	30	12	80	10	790	14	64	1000
	3.00	1.20	8.00	1.00	79.00	1.40	6.40	100.00
6	118	44	194	53	284	88	219	1000
	11.80	4.40	19.40	5.30	28.40	8.80	21.90	100.00
7	29	28	148	45	166	21	563	1000
	2.90	2.80	14.80	4.50	16.60	2.10	56.30	100.00

NonParametric Method, Using 12 mof Features 16:07 Sunday, February 16, 1997

51

From SPECIES	1	2	3	4	5
1	531	14	245	7	37
	53.10	1.40	24.50	0.70	3.70
2	51	693	70	56	14
	5.10	69.30	7.00	5.60	1.40
3	200	19	550	24	45
	20.00	1.90	55.00	2.40	4.50
4	5	32	34	763	6
	0.50	3.20	3.40	76.30	0.60
5	17	6	27	9	747
	1.70	0.60	2.70	0.90	74.70
6	59	24	59	40	175
	5.90	2.40	5.90	4.00	17.50
7	34	13	40	50	82
	3.40	1.30	4.00	5.00	8.20

From SPECIES	6	7	OTHER	Total
1	53	38	75	1000
	5.30	3.80	7.50	100.00
2	36	28	52	1000
	3.60	2.80	5.20	100.00
3	54	45	63	1000
	5.40	4.50	6.30	100.00
4	55	75	30	1000
	5.50	7.50	3.00	100.00
5	80	75	39	1000
	8.00	7.50	3.90	100.00
6	384	172	87	1000
	38.40	17.20	8.70	100.00
7	114	594	73	1000
	11.40	59.40	7.30	100.00

Parametric Method, Using 16 mof Features 16:07 Sunday, February 16, 1997 63

From SPECIES	1	2	3	4	5	6	7	Total
1	602	9	314	8	39	9	19	1000
	60.20	0.90	31.40	0.80	3.90	0.90	1.90	100.00
2	105	633	134	88	14	11	15	1000
	10.50	63.30	13.40	8.80	1.40	1.10	1.50	100.00
3	253	8	643	28	50	3	15	1000
	25.30	0.80	64.30	2.80	5.00	0.30	1.50	100.00
4	10	20	82	797	16	9	66	1000
	1.00	2.00	8.20	79.70	1.60	0.90	6.60	100.00
5	23	13	80	4	822	11	47	1000
	2.30	1.30	8.00	0.40	82.20	1.10	4.70	100.00
6	126	24	230	63	329	68	160	1000
	12.60	2.40	23.00	6.30	32.90	6.80	16.00	100.00
7	35	15	160	63	231	36	460	1000
	3.50	1.50	16.00	6.30	23.10	3.60	46.00	100.00

From SPECIES	1	2	3	4	5
1	547 54.70	9 0.90	234 23.40	2 0.20	49 4.90
2	68 6.80	658 65.80	86 8.60	71 7.10	11 1.10
3	193 19.30	12 1.20	574 57.40	19 1.90	57 570
4	10 1.00	14 1.40	30 3.00	769 76.90	9 0.90
5	9 0.90	2 0.20	18 1.80	7 0.70	774 77.40
6	55 5.50	12 1.20	62 6.20	33 3.30	170 17.00
7	23 2.30	5 0.50	35 3.50	47 4.70	84 8.40

From SPECIES	6	7	OTHER	Total
1	61 6.10	41 4.10	57 570	1000 100.00
2	36 3.60	25 2.50	45 4.50	1000 100.00
3	51 5.10	46 4.60	48 4.80	1000 100.00
4	50 5.00	90 9.00	28 2.80	1000 100.00
5	75 7.50	75 7.50	40 4.00	1000 100.00
6	388 38.80	208 20.80	72 7.20	1000 100.00
7	95 9.50	658 65.80	53 5.30	1000 100.00

Parametric Method, Using 20 mof Features 16:07 Sunday, February 16, 1997 83

From SPECIES	1	2	3	4	5	6	7	Total
1	579	11	323	5	48	9	25	1000
	57.90	1.10	32.30	0.50	4.80	0.90	2.50	100.00
2	100	642	125	89	9	18	17	1000
	10.00	64.20	12.50	8.90	0.90	1.80	1.70	100.00
3	235	9	647	30	55	3	21	1000
	23.50	0.90	64.70	3.00	5.50	0.30	2.10	100.00
4	10	24	78	799	16	7	66	1000
	1.00	2.40	7.80	79.90	1.60	0.7	6.60	100.00
5	23	12	74	3	836	7	45	1000
	2.30	1.20	7.40	0.30	83.60	0.70	4.50	100.00
6	117	24	207	58	328	68	198	1000
	11.70	2.40	20.70	5.80	32.80	6.80	19.80	100.00
7	45	10	143	45	220	32	505	1000
	4.50	1.00	14.30	4.50	22.00	3.20	50.50	100.00

NonParametric Method, Using 20 mof Features 16:07 Sunday, February 16, 1997

91

From SPECIES	1	2	3	4	5
1	567	12	227	7	43
	56.70	1.20	22.70	0.70	4.30
2	56	671	87	71	14
	5.60	67.10	8.70	7.10	1.40
3	190	7	550	24	43
	19.00	0.70	55.00	2.40	4.30
4	12	15	36	752	8
	1.20	1.50	3.60	75.20	0.80
5	7	0	19	10	785
	0.70	0.00	1.90	1.00	78.50
6	62	6	60	26	169
	6.20	0.60	6.00	2.60	16.90
7	26	4	36	39	82
	2.60	0.40	3.60	3.90	8.20

From SPECIES	6	7	OTHER	Total		
1	54	36	54	1000	5.40	3.60
5.40 100.00						
2	35	23	43	1000		
	3.50	2.30	4.30	100.00		
3	61	63	62	1000		
	6.10	6.30	6.20	100.00		
4	45	90	42	1000		
	4.50	9.00	4.20	100.00		
5	67	83	29	1000		
	6.70	8.30	2.90	100.00		
6	394	205	78	1000		
	39.40	20.50	7.80	100.00		
7	105	660	48	1000		
	10.50	66.00	4.80	100.00		

Parametric Method, Using 24 mof Features 16:07 Sunday, February 16, 1997 107

From SPECIES	1	2	3	4	5	6	7	Total
1	577	7	337	8	61	2	8	1000
	57.70	0.70	33.70	0.80	6.10	0.20	0.80	100.00
2	112	594	144	119	12	7	12	1000
	11.20	59.40	14.40	11.90	1.20	0.70	1.20	100.00
3	230	8	657	36	67	0	2	1000
	23.00	0.80	65.70	3.60	6.70	0.00	0.20	100.00
4	9	20	74	842	17	2	36	1000
	0.90	2.00	7.40	84.20	1.70	0.20	3.60	100.00
5	21	12	66	4	869	6	22	1000
	2.10	1.20	6.60	0.40	86.90	0.60	2.20	100.00
6	120	21	236	79	375	41	128	1000
	12.00	2.10	23.60	7.90	37.50	4.10	12.80	100.00
7	44	8	146	75	337	22	368	1000
	4.40	0.80	14.60	7.50	33.70	2.20	36.80	100.00

NonParametric Method, Using 24 mof Features 16:07 Sunday, February 16, 1997

119

From SPECIES	1	2	3	4	5
1	566	6	230	6	43
	56.60	0.60	23.00	0.60	4.30
2	61	659	85	59	13
	6.10	65.90	8.50	5.90	1.30
3	202	11	545	19	46
	20.20	1.10	54.50	1.90	4.60
4	9	7	29	766	12
	0.90	0.70	2.0	76.60	1.20
5	12	2	20	6	789
	1.20	0.20	2.00	0.60	78.90
6	45	5	58	28	174
	4.50	0.50	5.80	2.80	17.40
7	20	2	45	40	73
	2.00	0.20	4.50	4.00	7.30

NonParametric Method, Using 24 mof Features 16:07 Sunday, February 16, 1997

120

From SPECIES	6	7	OTHER	Total
1	51	35	63	1000
	5.10	3.50	6.30	100.00
2	37	34	52	1000
	3.70	3.40	5.0	100.00
3	60	60	57	1000
	6.00	6.00	5.70	100.00
4	44	100	33	1000
	4.40	10.00	3.30	100.00
5	64	78	29	1000
	6.40	7.80	2.90	100.00
6	405	198	87	1000
	40.50	19.80	8.70	100.00
7	105	671	44	1000
	10.50	67.10	4.40	100.00

Parametric Method, Using 28 mof Features 16:07 Sunday, February 16, 1997

135

From SPECIES	1	2	3	4	5	6	7	Total
1	607 60.70	5 0.50	312 31.20	10 1.00	59 5.90	0 0.00	7 0.70	1000 100.00
2	112 11.20	586 58.60	146 14.60	133 13.30	13 1.30	3 0.30	7 0.70	1000 100.00
3	253 25.30	7 0.70	636 63.60	37 3.70	65 6.50	0 0.00	2 0.20	1000 100.00
4	10 1.00	19 1.90	77 7.70	848 84.80	14 1.40	1 0.10	31 3.10	1000 100.00
5	22 2.20	12 1.20	55 5.50	5 0.50	883 88.30	4 0.40	19 1.90	1000 100.00
6	147 14.70	18 1.80	216 21.60	92 9.20	403 40.30	30 3.00	94 9.40	1000 100.00
7	63 6.30	9 0.90	154 15.40	78 7.80	382 38.20	16 1.60	298 29.80	1000 100.00

NonParametric Method, Using 28 mof Features 16:07 Sunday, February 16, 1997

147

From SPECIES	1	2	3	4	5
1	572 57.20	13 1.30	219 21.90	5 0.50	52 5.20
2	55 5.50	675 67.50	88 8.8	56 5.60	8 0.80
3	211 21.10	13 1.30	549 54.90	22 2.20	41 4.10
4	7 0.70	5 0.50	32 3.20	755 75.50	10 1.00
5	9 0.90	0 0.00	28 2.80	6 0.60	795 79.50
6	47 4.70	10 1.00	55 5.50	36 3.60	166 16.60
7	23 2.30	3 0.30	33 3.30	30 3.00	65 6.50

From SPECIES	6	7	OTHER	Total
1	52 5.20	35 3.50	52 5.20	1000 100.00
2	40 4.00	24 2.40	54 5.40	1000 100.00
3	56 5.60	52 5.20	56 5.60	1000 100.00
4	52 5.20	98 9.80	41 4.10	1000 100.00
5	61 6.10	76 7.60	25 2.50	1000 100.00
6	379 37.90	217 21.70	90 9.00	1000 100.00
7	92 9.20	716 71.60	38 3.80	1000 100.00

From SPECIES	1	2	3	4	5	6	7	Total
1	533 53.30	162 16.20	232 23.20	1 0.10	48 4.80	24 2.40	0 0.00	1000 100.00
2	230 23.00	548 54.80	89 8.90	24 2.40	80 8.00	29 2.90	0 0.00	1000 100.00
3	60 6.00	29 2.90	893 89.30	1 0.10	0 0.00	17 1.70	0 0.00	1000 100.00
4	44 4.40	45 4.50	1 0.10	859 85.90	17 1.70	34 3.40	0 0.00	1000 100.00
5	46 4.60	75 7.50	0 0.00	8 0.80	848 84.80	23 2.30	0 0.00	1000 100.00
6	53 5.30	120 12.00	20 2.00	221 22.10	13 1.30	573 57.30	0 0.00	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	2 0.20	998 99.80	1000 100.00

From SPECIES	1	2	3	4	5
1	614 61.40	152 15.20	142 14.20	5 0.50	44 4.40
2	181 18.10	632 63.20	49 4.90	7 0.70	71 7.10
3	119 11.90	18 1.80	837 83.70	0 0.00	0 0.00
4	27 2.70	32 3.20	2 0.20	830 83.00	11 1.10
5	27 2.70	73 7.30	0 0.00	4 0.40	868 86.80
6	29 2.90	75 7.50	11 1.10	150 15.00	8 0.80
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	22 2.20	0 0.00	21 2.10	1000 100.00
2	44 4.40	0 0.00	16 1.60	1000 100.00
3	8 0.80	0 0.00	18 1.80	1000 100.00
4	84 8.40	0 0.00	14 1.40	1000 100.00
5	17 1.70	0 0.00	11 1.10	1000 100.00
6	710 71.00	0 0.00	17 1.70	1000 100.00
7	2 0.20	998 99.80	0 0.00	1000 100.00

Parametric Method, Using 8 color Features 16:07 Sunday, February 16, 1997 174

From SPECIES	1	2	3	4	5	6	7	Total
1	765	36	137	6	29	27	0	1000
	76.50	3.60	13.70	0.60	2.90	2.70	0.00	100.00
2	149	713	94	20	14	10	0	1000
	14.90	71.30	9.40	2.00	1.40	1.00	0.00	100.00
3	73	2	912	0	0	13	0	1000
	7.30	0.20	91.20	0.00	0.00	1.30	0.00	100.00
4	18	20	0	927	17	18	0	1000
	1.80	2.00	0.00	92.70	1.70	1.80	0.00	100.00
5	9	17	0	15	949	10	0	1000
	0.90	1.70	0.00	1.50	94.90	1.00	0.00	100.00
6	123	4	20	119	13	721	0	1000
	12.30	0.40	2.00	11.90	1.30	72.10	0.00	100.00
7	0	0	0	0	0	2	998	1000
	0.00	0.00	0.00	0.00	0.00	0.20	99.80	100.00

NonParametric Method, Using 8 color Features 16:07 Sunday, February 16, 1997

181

From SPECIES	1	2	3	4	5
1	796	22	121	8	25
	79.60	2.20	12.10	0.80	2.50
2	64	863	37	8	9
	6.40	86.30	3.70	0.80	0.90
3	61	14	918	0	0
	6.10	1.40	91.80	0.00	0.00
4	14	25	0	908	20
	1.40	2.50	0.00	90.80	2.00
5	13	5	0	10	961
	1.30	0.50	0.00	1.00	96.10
6	55	9	15	50	13
	5.50	0.90	1.50	5.00	1.30
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

NonParametric Method, Using 8 color Features 16:07 Sunday, February 16, 1997

182

From SPECIES	6	7	OTHER	Total
1	22	0	6	1000
	2.20	0.00	0.60	100.00
2	11	0	8	1000
	1.10	0.00	0.80	100.00
3	6	0	1	1000
	0.60	0.00	0.10	100.00
4	19	0	14	1000
	1.90	0.00	1.40	100.00
5	9	0	2	1000
	0.90	0.00	0.20	100.00
6	849	0	9	1000
	84.90	0.00	0.90	100.00
7	1	999	0	1000
	0.10	99.90	0.00	100.00

Parametric Method, Using 12 color Features 16:07 Sunday, February 16, 1997

192

From SPECIES	1	2	3	4	5	6	7	Total
1	835	11	110	0	35	9	0	1000
	83.50	1.10	11.00	0.00	3.50	0.90	0.00	100.00
2	297	544	84	30	41	4	0	1000
	29.70	54.40	8.40	3.00	4.10	0.40	0.00	100.00
3	44	3	946	0	0	7	0	1000
	4.40	0.30	94.60	0.00	0.00	0.70	0.00	100.00
4	3	7	0	971	10	9	0	1000
	0.30	0.70	0.00	97.10	1.00	0.90	0.00	100.00
5	4	2	0	12	981	1	0	1000
	0.40	0.20	0.00	1.20	98.10	0.10	0.00	100.00
6	202	1	36	156	20	585	0	1000
	20.20	0.10	3.60	15.60	2.00	58.50	0.00	100.00
7	0	0	0	0	0	1	999	1000
	0.00	0.00	0.00	0.00	0.00	0.10	99.90	100.00

NonParametric Method, Using 12 color Features 16:07 Sunday, February 16, 1997

199

From SPECIES	1	2	3	4	5
1	861	27	71	0	22
	86.10	2.70	7.10	0.00	2.20
2	51	875	43	9	6
	5.10	87.50	4.30	0.90	0.60
3	33	10	942	0	0
	3.30	1.00	94.20	0.00	0.00
4	1	24	0	940	11
	0.10	2.40	0.00	94.00	1.10
5	13	2	0	2	975
	1.30	0.20	0.00	0.20	97.50
6	22	7	23	33	8
	2.20	0.70	2.30	3.30	0.80
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

NonParametric Method, Using 12 color Features 16:07 Sunday, February 16, 1997

200

From SPECIES	6	7	OTHER	Total
1	11	0	8	1000
	1.10	0.00	0.80	100.00
2	8	0	8	1000
	0.80	0.00	0.80	100.00
3	13	0	2	1000
	1.30	0.00	0.20	100.00
4	22	0	2	1000
	2.20	0.00	0.20	100.00
5	6	0	2	1000
	0.60	0.00	0.20	100.00
6	904	0	3	1000
	90.40	0.00	0.30	100.00
7	1	999	0	1000
	0.10	99.90	0.00	100.00

Parametric Method, Using 16 color Features 16:07 Sunday, February 16, 1997
211

From SPECIES	1	2	3	4	5	6	7	Total
1	862	14	95	1	17	11	0	1000
	86.20	1.40	9.50	0.10	1.70	1.10	0.00	100.00
2	188	710	59	22	15	6	0	1000
	18.80	71.00	5.90	2.20	1.50	0.60	0.00	100.00
3	52	2	940	0	0	6	0	1000
	5.20	0.20	94.00	0.00	0.00	0.60	0.00	100.00
4	6	8	0	973	0	13	0	1000
	0.60	0.80	0.00	97.30	0.00	1.30	0.00	100.00
5	4	3	0	8	983	2	0	1000
	0.40	0.30	0.00	0.80	98.30	0.20	0.00	100.00
6	153	7	31	180	7	622	0	1000
	15.30	0.70	3.10	18.00	0.70	2.20	0.00	100.00
7	0	0	0	0	0	1	999	1000
	0.00	0.00	0.00	0.00	0.00	0.10	99.90	100.00

NonParametric Method, Using 16 color Features 16:07 Sunday, February 16, 1997
219

From SPECIES	1	2	3	4	5
1	874	13	84	0	12
	87.40	1.30	8.40	0.00	1.20
2	50	887	42	3	1
	5.00	88.70	4.20	0.30	0.10
3	40	2	950	0	0
	4.00	0.20	95.00	0.00	0.00
4	5	13	0	954	5
	0.50	1.30	0.00	95.40	0.50
5	8	2	0	1	988
	0.80	0.20	0.00	0.10	98.80
6	18	6	25	24	7
	1.80	0.60	2.50	2.40	0.70
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

NonParametric Method, Using 16 color Features 16:07 Sunday, February 16, 1997

220

From SPECIES	6	7	OTHER	Total
1	13	0	4	1000
	1.30	0.00	0.40	100.00
2	11	0	6	1000
	1.10	0.00	0.60	100.00
3	7	0	1	1000
	0.70	0.00	0.10	100.00
4	22	0	1	1000
	2.20	0.00	0.10	100.00
5	0	0	1	1000
	0.00	0.00	0.10	100.00
6	914	0	6	1000
	91.40	0.00	0.60	100.00
7	1	999	0	1000
	0.10	99.90	0.00	100.00

Parametric Method, Using 20 color Features 16:07 Sunday, February 16, 1997

231

From SPECIES	1	2	3	4	5	6	7	Total
1	785	18	171	0	16	10	0	1000
	78.50	1.80	17.10	0.00	1.60	1.00	0.00	100.00
2	161	719	87	12	16	5	0	1000
	16.10	71.90	8.70	1.20	1.60	0.50	0.00	100.00
3	44	2	952	0	0	2	0	1000
	4.40	0.20	95.20	0.00	0.00	0.20	0.00	100.00
4	7	13	0	959	3	18	0	1000
	0.70	1.30	0.00	95.90	0.30	1.80	0.00	100.00
5	4	1	0	10	980	5	0	1000
	0.40	0.10	0.00	1.00	98.00	0.50	0.00	100.00
6	67	3	39	208	7	676	0	1000
	6.70	0.30	3.90	20.80	0.70	67.60	0.00	100.00
7	1	0	0	0	0	0	999	1000
	0.10	0.00	0.00	0.00	0.00	0.00	99.90	100.00

From SPECIES	1	2	3	4	5
1	876 87.60	17 1.70	79 7.90	0 0.00	11 1.10
2	55 5.50	878 87.80	44 4.40	4 0.40	3 0.30
3	38 3.80	2 0.20	954 95.40	0 0.00	0 0.00
4	5 0.50	11 1.10	0 0.00	961 96.10	3 0.30
5	11 1.10	0 0.00	0 0.00	1 0.10	985 98.50
6	16 1.60	2 0.20	22 2.20	25 2.50	6 0.60
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	14 1.40	0 0.00	3 0.30	1000 100.00
2	10 1.00	0 0.00	6 0.60	1000 100.00
3	6 0.60	0 0.00	0 0.00	1000 100.00
4	20 2.00	0 0.00	0 0.00	1000 100.00
5	1 0.10	0 0.00	2 0.20	1000 100.00
6	925 92.50	0 0.00	4 0.40	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 24 color Features 16:07 Sunday, February 16, 1997

255

From SPECIES	1	2	3	4	5	6	7	Total
1	744 74.40	10 1.00	228 22.80	2 0.20	14 1.40	2 0.20	0 0.00	1000 100.00
2	212 21.20	601 60.10	115 11.50	39 3.90	31 3.10	2 0.20	0 0.00	1000 100.00
3	31 3.10	2 0.20	967 96.70	0 0.00	0 0.00	0 0.00	0 0.00	1000 100.00
4	7 0.70	7 0.70	0 0.00	978 97.80	2 0.20	6 0.60	0 0.00	1000 100.00
5	1 0.10	1 0.10	0 0.00	14 1.40	983 98.30	1 0.10	0 0.00	1000 100.00
6	176 17.60	9 0.90	87 8.70	263 26.30	16 1.60	448 44.80	1 0.10	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

NonParametric Method, Using 24 color Features 16:07 Sunday, February 16, 1997

267

From SPECIES	1	2	3	4	5
1	873 87.30	17 1.70	94 9.40	0 0.00	7 0.70
2	56 5.60	875 87.50	54 5.40	1 0.10	3 0.30
3	37 3.70	4 0.40	954 95.40	0 0.00	0 0.00
4	3 0.30	6 0.60	0 0.00	976 97.60	6 0.60
5	6 0.60	2 0.20	0 0.00	2 0.20	987 98.70
6	13 1.30	5 0.50	12 1.20	15 1.50	9 0.90
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

NonParametric Method, Using 24 color Features 16:07 Sunday, February 16, 1997

268

From SPECIES	6	7	OTHER	Total
1	5 0.50	0 0.00	4 0.40	1000 100.00
2	3 0.30	0 0.00	8 0.80	1000 100.00
3	5 0.50	0 0.00	0 0.00	1000 100.00
4	8 0.80	0 0.00	1 0.10	1000 100.00
5	3 0.30	0 0.00	0 0.00	1000 100.00
6	942 94.20	0 0.00	4 0.40	1000 100.00
7	0 0.00	1000 100.00	0 0.00	1000 100.00

Parametric Method, Using 28 color Features 16:07 Sunday, February 16, 1997

283

From SPECIES	1	2	3	4	5	6	7	Total
1	780 78.00	10 1.00	201 20.10	0 0.00	8 0.80	1 0.10	0 0.00	1000 100.00
2	216 21.60	629 62.90	113 11.30	20 2.00	21 2.10	1 0.10	0 0.00	1000 100.00
3	29 2.90	2 0.20	969 96.90	0 0.00	0 0.00	0 0.00	0 0.00	1000 100.00
4	3 0.30	7 0.70	0 0.00	979 97.90	4 0.40	7 0.70	0 0.00	1000 100.00
5	0 0.00	2 0.20	0 0.00	9 0.90	988 98.80	1 0.10	0 0.00	1000 100.00
6	203 20.30	1 0.10	8 0.80	268 26.80	7 0.70	512 51.20	1 0.10	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

From SPECIES	1	2	3	4	5
1	898 89.80	13 1.30	79 7.90	0 0.00	5 0.50
2	56 5.60	888 88.80	44 4.40	3 0.30	4 0.40
3	26 2.60	2 0.20	972 97.20	0 0.00	0 0.00
4	2 0.20	5 0.50	0 0.00	978 97.80	3 0.30
5	0 0.00	1 0.10	0 0.00	1 0.10	997 99.70
6	4 0.40	1 0.10	0 0.00	14 1.40	8 0.80
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	2 0.20	0 0.00	3 0.30	1000 100.00
2	2 0.20	0 0.00	3 0.30	1000 100.00
3	0 0.00	0 0.00	0 0.00	1000 100.00
4	11 1.10	0 0.00	1 0.10	1000 100.00
5	0 0.00	0 0.00	1 0.10	1000 100.00
6	972 97.20	0 0.00	1 0.10	1000 100.00
7	0 0.00	1000 100.00	0 0.00	1000 100.00

Parametric Method, Using 4 combined Features 16:07 Sunday. February 16, 1997
305

From SPECIES	1	2	3	4	5	6	7	Total
1	615	67	229	1	70	18	0	1000
	61.50	6.70	22.90	0.10	7.00	1.80	0.00	100.00
2	204	613	54	70	33	26	0	1000
	20.40	61.30	5.40	7.00	3.30	2.60	0.00	100.00
3	76	9	896	9	0	10	0	1000
	7.60	0.90	89.60	0.90	0.00	1.00	0.00	100.00
4	3	115	0	856	3	23	0	1000
	0.30	11.50	0.00	85.60	0.30	2.30	0.00	100.00
5	87	44	0	0	858	11	0	1000
	8.70	4.40	0.00	0.00	85.80	1.10	0.00	100.00
6	287	182	104	123	34	270	0	1000
	28.70	18.20	10.40	12.30	3.40	27.00	0.00	100.00
7	0	0	0	0	0	2	998	1000
	0.00	0.00	0.00	0.00	0.00	0.20	99.80	100.00

NonParametric Method, Using 4 combined Features 16:07 Sunday, February 16,
1997 311

From SPECIES	1	2	3	4	5
1	579	79	154	3	79
	57.90	7.90	15.40	0.30	7.90
2	72	701	27	45	37
	7.20	70.10	2.70	4.50	3.70
3	107	12	840	1	0
	10.70	1.20	84.00	0.10	0.00
4	3	77	0	871	0
	0.30	7.70	0.00	87.10	0.00
5	41	48	0	1	869
	4.10	4.80	0.00	0.10	86.90
6	120	105	38	92	17
	12.00	10.50	3.80	9.20	1.70
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

NonParametric Method, Using 4 combined Features 16:07 Sunday, February 16.
1997 312

From SPECIES	6	7	OTHER	Total
1	70	0	36	1000
	7.00	0.00	3.60	100.00
2	84	0	34	1000
	8.40	0.00	3.40	100.00
3	30	0	10	1000
	3.00	0.00	1.00	100.00
4	38	0	11	1000
	3.80	0.00	1.10	100.00
5	23	0	18	1000
	2.30	0.00	1.80	100.00
6	588	0	4	1000
	58.80	0.00	4.00	100.00
7	4	996	0	1000
	0.40	99.60	0.00	100.00

Parametric Method, Using 8 combined Features 16:07 Sunday, February 16, 1997
322

From SPECIES	1	2	3	4	5	6	7	Total
1	743	41	154	1	32	29	0	1000
	74.30	4.10	15.40	0.10	3.20	2.90	0.00	100.00
2	161	704	82	18	20	15	0	1000
	16.10	70.40	8.20	1.80	2.00	1.50	0.00	100.00
3	69	2	915	0	0	14	0	1000
	6.90	0.20	91.50	0.00	0.00	1.40	0.00	100.00
4	8	17	0	956	3	16	0	1000
	0.80	1.70	0.00	95.60	0.30	1.60	0.00	100.00
5	11	17	0	2	961	9	0	1000
	1.10	1.70	0.00	0.20	96.10	0.90	0.00	100.00
6	123	24	27	78	20	728	0	1000
	12.30	2.40	2.70	7.80	2.00	72.80	0.00	100.00
7	0	0	0	0	0	2	998	1000
	0.00	0.00	0.00	0.00	0.00	0.20	99.80	100.00

From SPECIES	1	2	3	4	5
1	789 78.90	38 3.80	119 11.90	1 0.10	21 2.10
2	68 6.80	838 83.80	34 3.40	17 1.70	6 0.60
3	57 5.70	12 1.20	922 92.20	0 0.00	0 0.00
4	4 0.40	16 1.60	0 0.00	964 96.40	5 0.50
5	21 2.10	9 0.90	0 0.00	1 0.10	961 96.10
6	43 4.30	34 3.40	22 2.20	31 3.10	13 1.30
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	24 2.40	0 0.00	8 0.80	1000 100.00
2	27 2.70	0 0.00	10 1.00	1000 100.00
3	7 0.70	0 0.00	2 0.20	1000 100.00
4	11 1.10	0 0.00	0 0.00	1000 100.00
5	7 0.70	0 0.00	1 0.10	1000 100.00
6	850 85.00	0 0.00	7 0.70	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 12 combined Features 16:07 Sunday, February 16, 1997
340

From SPECIES	1	2	3	4	5	6	7	Total
1	812 81.20	15 1.50	124 12.40	1 0.10	37 3.70	11 1.10	0 0.00	1000 100.00
2	201 20.10	708 70.80	42 4.20	20 2.00	26 2.60	3 0.30	0 0.00	1000 100.00
3	45 4.50	4 0.40	945 94.50	0 0.00	0 0.00	6 0.60	0 0.00	1000 100.00
4	6 0.60	5 0.50	0 0.00	976 97.60	5 0.50	8 0.80	0 0.00	1000 100.00
5	2 0.20	4 0.40	0 0.00	1 0.10	992 99.20	1 0.10	0 0.00	1000 100.00
6	269 26.90	1 0.10	32 3.20	107 10.70	51 5.10	540 54.00	0 0.00	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

NonParametric Method, Using 12 combined Features 16:07 Sunday, February 16,
1997 347

From SPECIES	1	2	3	4	5
1	831 83.10	22 2.20	105 10.50	1 0.10	17 1.70
2	55 5.50	884 88.40	33 3.30	12 1.20	5 0.50
3	34 3.40	7 0.70	950 95.00	0 0.00	0 0.00
4	5 0.50	11 1.10	0 0.00	968 96.80	1 0.10
5	14 1.40	4 0.40	0 0.00	1 0.10	974 97.40
6	34 3.40	16 1.60	32 3.20	29 2.90	11 1.10
7	0 0.00	0 0.00	0 0.00	0 0.00	0 .00

NonParametric Method, Using 12 combined Features 16:07 Sunday, February 16, 1997 348

From SPECIES	6	7	OTHER	Total
1	17 1.70	0 0.00	7 0.70	1000 100.00
2	5 0.50	0 0.00	6 0.60	1000 100.00
3	8 0.80	0 0.00	1 0.10	1000 100.00
4	15 1.50	0 0.00	0 0.00	1000 100.00
5	5 0.50	0 0.00	2 0.20	1000 100.00
6	870 87.00	0 0.00	8 0.80	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 16 combined Features 16:07 Sunday, February 16, 1997 359

From SPECIES	1	2	3	4	5	6	7	Total
1	845 84.50	15 1.50	110 11.00	0 0.00	18 1.80	12 1.20	0 0.00	1000 100.00
2	138 13.80	799 79.90	34 3.40	13 1.30	10 1.00	6 0.60	0 0.00	1000 100.00
3	43 4.30	3 0.30	950 95.00	0 0.00	0 0.00	4 0.40	0 0.00	1000 100.00
4	1 0.10	6 0.60	0 0.00	983 98.30	0 0.00	10 1.00	0 0.00	1000 100.00
5	5 0.50	3 0.30	0 0.00	4 0.40	985 98.50	3 0.30	0 0.00	1000 100.00
6	40 14.00	8 0.80	29 2.90	161 16.10	10 1.00	652 65.20	0 0.00	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

From SPECIES	1	2	3	4	5
1	853 85.30	13 1.30	103 10.30	0 0.00	15 1.50
2	32 3.20	884 88.40	45 4.50	9 0.90	4 0.40
3	44 4.40	4 0.40	945 94.50	0 0.00	0 0.00
4	2 0.20	6 0.60	0 0.00	980 98.00	0 0.00
5	8 0.80	2 0.20	0 0.00	1 0.10	988 98.80
6	17 1.70	6 0.60	18 1.80	26 2.60	7 0.70
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	11 1.10	0 0.00	5 0.50	1000 100.00
2	17 1.70	0 0.00	9 0.90	1000 100.00
3	5 0.50	0 0.00	2 0.20	1000 100.00
4	11 1.10	0 0.00	1 0.10	1000 100.00
5	1 0.10	0 0.00	0 0.00	1000 100.00
6	922 92.20	0 0.00	4 0.40	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 20 combined Features 16:07 Sunday, February 16, 1997
379

From SPECIES	1	2	3	4	5	6	7	Total
1	808	16	155	0	11	10	0	1000
	80.80	1.60	15.50	0.00	1.10	1.00	0.00	100.00
2	86	846	47	6	7	8	0	1000
	8.60	84.60	4.70	0.60	0.70	0.80	0.00	100.00
3	20	3	973	0	0	4	0	1000
	2.00	0.30	97.30	0.00	0.00	0.40	0.00	100.00
4	2	14	0	969	2	13	0	1000
	0.20	1.40	0.00	96.90	0.20	1.30	0.00	100.00
5	4	4	0	5	985	2	0	1000
	0.40	0.40	0.00	0.50	98.50	0.20	0.00	100.00
6	50	8	40	194	11	697	0	1000
	5.00	0.80	4.00	19.40	1.10	69.70	0.00	100.00
7	1	0	0	0	0	0	999	1000
	0.10	0.00	0.00	0.00	0.00	0.00	99.90	100.00

NonParametric Method, Using 20 combined Features 16:07 Sunday, February 16,
1997 387

From SPECIES	1	2	3	4	5
1	875	12	93	0	7
	87.50	1.20	9.30	0.00	0.70
2	34	888	37	7	7
	3.40	88.80	3.70	0.70	0.70
3	35	3	959	0	0
	3.50	0.30	95.90	0.00	0.00
4	1	6	0	980	0
	0.10	0.60	0.00	98.00	0.00
5	5	1	0	1	990
	0.50	0.10	0.00	0.10	99.00
6	14	3	16	27	6
	1.40	0.30	1.60	2.70	0.60
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

NonParametric Method, Using 20 combined Features 16:07 Sunday, February 16, 1997 388

From SPECIES	6	7	OTHER	Total
1	12 1.20	0 0.00	1 0.10	1000 100.00
2	17 1.70	0 0.00	10 1.00	1000 100.00
3	2 0.20	0 0.00	1 0.10	1000 100.00
4	11 1.10	0 0.00	2 0.20	1000 100.00
5	3 0.30	0 0.00	0 0.00	1000 100.00
6	930 93.00	0 0.00	4 0.40	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 24 combined Features 16:07 Sunday, February 16, 1997 403

From SPECIES	1	2	3	4	5	6	7	Total
1	893 89.30	15 1.50	86 8.60	0 0.00	3 0.30	3 0.30	0 0.00	1000 100.00
2	86 8.60	886 88.60	23 2.30	0 0.00	5 0.50	0 0.00	0 0.00	1000 100.00
3	19 1.90	3 0.30	978 97.80	0 0.00	0 0.00	0 0.00	0 0.00	1000 100.00
4	1 0.10	11 1.10	0 0.00	981 98.10	2 0.20	5 0.50	0 0.00	1000 100.00
5	0 0.00	4 0.40	0 0.00	5 0.50	990 99.00	1 0.10	0 0.00	1000 100.00
6	2 0.20	0 0.00	0 0.00	189 18.90	9 0.90	800 80.00	0 0.00	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

From SPECIES	1	2	3	4	5
1	934 93.40	11 1.10	50 5.00	0 0.00	2 0.20
2	58 5.80	924 92.40	12 1.20	2 0.20	3 0.30
3	12 1.20	1 0.10	987 98.70	0 0.00	0 0.00
4	1 0.10	2 0.20	0 0.00	989 98.90	0 0.00
5	1 0.10	1 0.10	0 0.00	2 0.20	993 99.30
6	1 0.10	0 0.00	0 0.00	15 1.50	7 0.70
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	0 0.00	0 0.00	3 0.30	1000 100.00
2	0 0.00	0 0.00	1 0.10	1000 100.00
3	0 0.00	0 0.00	0 0.00	1000 100.00
4	8 0.80	0 0.00	0 0.00	1000 100.00
5	3 0.30	0 0.00	0 0.00	1000 100.00
6	977 97.70	0 0.00	0 0.00	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

Parametric Method, Using 28 combined Features 16:07 Sunday, February 16, 1997
431

From SPECIES	1	2	3	4	5	6	7	Total
1	886 88.60	9 0.90	99 9.90	0 0.00	5 0.50	1 0.10	0 0.00	1000 100.00
2	135 13.50	812 81.20	43 4.30	0 0.00	8 0.80	2 0.20	0 0.00	1000 100.00
3	28 2.80	2 0.20	970 97.00	0 0.00	0 0.00	0 0.00	0 0.00	1000 100.00
4	2 0.20	6 0.60	0 0.00	982 98.20	2 0.20	8 0.80	0 0.00	1000 100.00
5	0 0.00	2 0.20	0 0.00	5 0.50	992 99.20	1 0.10	0 0.00	1000 100.00
6	20 2.00	0 0.00	0 0.00	248 24.80	12 1.20	720 72.00	0 0.00	1000 100.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00	1 0.10	999 99.90	1000 100.00

NonParametric Method, Using 28 combined Features 16:07 Sunday, February 16,
1997 443

From SPECIES	1	2	3	4	5
1	95 93.50	10 1.00	49 4.90	0 0.00	3 0.30
2	60 6.00	931 93.10	4 0.40	1 0.10	2 0.20
3	16 1.60	1 0.10	983 98.30	0 0.00	0 0.00
4	0 0.00	1 0.10	0 0.00	993 99.30	0 0.00
5	1 0.10	1 0.10	0 0.00	1 0.10	994 99.40
6	1 0.10	0 0.00	0 0.00	12 1.20	7 0.70
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

From SPECIES	6	7	OTHER	Total
1	0 0.00	0 0.00	3 0.30	1000 100.00
2	0 0.00	0 0.00	2 0.20	1000 100.00
3	0 0.00	0 0.00	0 0.00	1000 100.00
4	6 0.60	0 0.00	0 0.00	1000 100.00
5	3 0.30	0 0.00	0 0.00	1000 100.00
6	980 98.00	0 0.00	0 0.00	1000 100.00
7	1 0.10	999 99.90	0 0.00	1000 100.00

APPENDIX E-3

EVALUATIONS OF FEATURE MODELS FOR GRAIN TYPE IDENTIFICATION ANALYSIS OF BULK GRAIN SAMPLES

From SPECIES	1	2	3	4	5	Total
1	179 99.44	1 0.56	0 0.00	0 0.00	0 0.00	180 100.00
2	0 0.00	60 100.00	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	0 0.00	0 0.00	0 0.00	60 100.00	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00
Total Percent	179 42.62	61 14.52	60 14.29	60 14.29	60 14.29	420 100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0000	0.0000	0.0000	0.0000	0.0011
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	179 99.44	0 0.00	1 0.56	0 0.00	0 0.00	180 100.00
2	0 0.00	60 100.00	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	0 0.00	0 0.00	0 0.00	60 100.00	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00

Total	179	60	61	60	60	420
Percent	42.62	14.29	14.52	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0000	0.0000	0.0000	0.0000	0.0011
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	180 100.00	0 0.00	0 0.00	0 0.00	0 0.00	180 100.00
2	0 0.00	60 100.00	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	0 0.00	0 0.00	0 0.00	60 100.00	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00
Total Percent	180 42.86	60 14.29	60 14.29	60 14.29	60 14.29	420 100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00
Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 12 color Features 08:33 Thursday, April 10, 1997 37

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00

Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00
Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 16 color Features 08:33 Thursday, April 10, 1997 52

From SPECIES	1	2	3	4	5	Total
1	179	0	1	0	0	180
	99.44	0.00	0.56	0.00	0.0	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00
Total	179	60	61	60	60	420
Percent	42.62	14.29	14.52	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0000	0.0000	0.0000	0.0000	0.0011
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 16 color Features 08:33 Thursday, April 10, 1997
58

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00

Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	179 99.44	0 0.00	0 0.00	1 0.56	0 0.00	180 100.00
2	1 1.67	59 98.33	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	1 1.67	0 0.00	0 0.00	59 98.33	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00
Total Percent	181 43.10	59 14.05	60 14.29	60 14.29	60 14.29	420 100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0167	0.0000	0.0167	0.0000	0.0078
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	180 100.00	0 0.00	0 0.00	0 0.00	0 0.00	180 100.00
2	0 0.00	60 100.00	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	0 0.00	0 0.00	0 0.00	60 100.00	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00
Total Percent	180 42.86	60 14.29	60 14.29	60 14.29	60 14.29	420 100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 24 color Features 08:33 Thursday, April 10, 1997 87

From SPECIES	1	2	3	4	5	Total
1	179 99.44	0 0.00	0 0.00	1 0.56	0 0.00	180 100.00
2	0 0.00	60 100.00	0 0.00	0 0.00	0 0.00	60 100.00
3	0 0.00	0 0.00	60 100.00	0 0.00	0 0.00	60 100.00
4	1 1.67	0 0.00	0 0.00	59 98.33	0 0.00	60 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	60 100.00	60 100.00

Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0000	0.0000	0.0167	0.0000	0.0044
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00
Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Parametric Method, Using 28 color Features 08:33 Thursday, April 10, 1997 109

From SPECIES	1	2	3	4	5	Total
1	179	0	0	1	0	180
	99.44	0.00	0.00	0.56	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	1	0	0	59	0	60
	1.67	0.00	0.00	98.33	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00
Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0056	0.0000	0.0000	0.0167	0.0000	0.0044
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

NonParametric Method, Using 28 color Features 08:33 Thursday, April 10, 1997
118

From SPECIES	1	2	3	4	5	Total
1	180	0	0	0	0	180
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	60	0	0	0	60
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	60	0	0	60
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	60	0	60
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	60	60
	0.00	0.00	0.00	0.00	100.00	100.00

Total	180	60	60	60	60	420
Percent	42.86	14.29	14.29	14.29	14.29	100.00
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

Error Count Estimates for SPECIES:

	1	2	3	4	5	Total
Rate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Priors	0.2000	0.2000	0.2000	0.2000	0.2000	

APPENDIX E-4

EVALUATIONS OF FEATURE MODELS FOR GRADE IDENTIFICATION ANALYSIS OF BULK CWRS WHEAT SAMPLES

Parametric Method, Using 4 color Features 08:35 Thursday, April 10, 1997 8

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)j}^{-1} \, (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}|$$

Posterior Probability of Membership in each SPECIES:

$$Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	55 91.67	5 8.33	0 0.00	60 100.00
2	12 20.00	43 71.67	5 8.33	60 100.00
3	10 16.67	26 43.33	24 40.00	60 100.00
Total Percent	77 42.78	74 41.11	29 16.11	180 100.00
Priors	0.3333	0.3333	0.3333	

12 NonParametric Method, Using 4 color Features 08:35 Thursday, April 10, 1997

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using 5 Nearest Neighbors

Squared Distance Function:

$$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$$

Posterior Probability of Membership in each SPECIES:

$m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X)\, PRIOR_j \,/\, SUM_k \,(\, m_k(X)\, PRIOR_k \,)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	54	4	2	0	60

	90.00	6.67	3.33	0.00	100.00
2	8 13.33	44 73.33	6 10.00	2 3.33	60 100.00
3	11 18.33	13 21.67	34 56.67	2 3.33	60 100.00
Total Percent	73 40.56	61 33.89	42 23.33	4 2.22	180 100.00
Priors	0.3333	0.3333	0.3333		

Parametric Method, Using 8 color Features 08:35 Thursday, April 10, 1997 21

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)j}^{-1} \, (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}|$$

Posterior Probability of Membership in each SPECIES:

$$Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	57 95.00	3 5.00	0 0.00	60 100.00
2	9 15.00	45 75.00	6 10.00	60 100.00
3	6 10.00	22 36.67	32 53.33	60 100.00
Total Percent	72 40.00	70 38.89	38 21.11	180 100.00
Priors	0.3333	0.3333	0.3333	

NonParametric Method, Using 8 color Features 08:35 Thursday, April 10, 1997

26

Discriminant Analysis Classification Summry for Calibration Data: WORK.CALIB

Cross-validation Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y) \qquad m_k(X) = \text{Proportion of obs in group k in 5 nearest neighbors of X}$$

$$Pr(j|X) = m_j(X) \, PRIOR_j \, / \, SUM_k (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	57 95.00	3 5.00	0 0.00	0 0.00	60 100.00
2	11 18.33	40 66.67	9 15.00	0 0.00	60 100.00
3	6 10.00	21 35.00	31 51.67	2 3.33	60 100.00
Total Percent	74 41.11	64 35.56	40 22.22	2 1.11	180 100.00
Priors	0.3333	0.3333	0.3333		

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' COV_{(X)j}^{-1} (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}| \qquad Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	59 98.33	1 1.67	0 0.00	60 100.00
2	5 8.33	51 85.00	4 6.67	60 100.00
3	3 5.00	19 31.67	38 63.33	60 100.00
Total	67	71	42	180

Percent	37.22	39.44	23.33	100.00
Priors	0.3333	0.3333	0.3333	

40 NonParametric Method, Using 12 color Features 08:35 Thursday, April 10, 1997

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)'\ COV^{-1}\ (X-Y) \qquad m_k(X) = \text{Proportion of obs in group k in 5 nearest neighbors of X}$$

$$Pr(j|X) = m_j(X)\ PRIOR_j\ /\ SUM_k\ (\ m_k(X)\ PRIOR_k\ )$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	57	3	0	0	60
	95.00	5.00	0.00	0.00	100.00
2	4	47	8	1	60
	6.67	78.33	13.33	1.67	100.00
3	4	17	37	2	60
	6.67	28.33	61.67	3.33	100.00
Total	65	67	45	3	180
Percent	36.11	37.22	25.00	1.67	100.00
Priors	0.3333	0.3333	0.3333		

Parametric Method, Using 16 color Features 08:35 Thursday, April 10, 1997 49

Discriminant Analysis Classification Summary or Calibration Data: WORK.CALIB

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_{(X)j})'\ COV_{(X)j}^{-1}\ (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}| \qquad Pr(j|X) = \exp(-.5\ D_j^2(X))\ /\ SUM_k\ \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	57 95.00	3 5.00	0 0.00	60 100.00
2	8 13.33	47 78.33	5 8.33	60 100.00
3	5 8.33	10 16.67	45 75.00	60 100.00
Total Percent	70 38.89	60 33.33	50 27.78	180 100.00
Priors	0.3333	0.3333	0.3333	

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)'\ COV^{-1}\ (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X)\ PRIOR_j \ /\ SUM_k\ (\ m_k(X)\ PRIOR_k\ )$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	58 96.67	2 3.33	0 0.00	0 0.00	60 100.00
2	7 11.67	47 78.33	5 8.33	1 1.67	60 100.00
3	4 6.67	16 26.67	40 66.67	0 0.00	60 100.00
Total Percent	69 38.33	65 36.11	45 25.00	1 0.56	180 100.00
Priors	0.3333	0.3333	0.3333		

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)j}^{-1} \, (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}|$$

$$Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations andPercent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	58 96.67	2 3.33	0 0.00	60 100.00
2	5 8.33	47 78.33	8 13.33	60 100.00
3	4 6.67	7 11.67	49 81.67	60 100.00
Total Percent	67 37.22	56 31.11	57 31.67	180 100.00
Priors	0.3333	0.3333	0.3333	

Discriminant Analysis Classification Summary for Calibration Data: WORK.CALIB

Cross-validation Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$$

$m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X) \, PRIOR_j \,/\, SUM_k (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total

1	59 98.33	1 1.67	0 0.00	0 0.00	60 100.00
2	3 5.00	49 81.67	6 10.00	2 3.33	60 100.00
3	3 5.00	20 33.33	36 60.00	1 1.67	60 100.00
Total Percent	65 36.11	70 38.89	42 23.33	3 1.67	180 100.00
Priors	0.3333	0.3333	0.3333		

APPENDIX F-1

RESULTS OF GRAIN TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL GRAIN KERNELS USING STATISTICAL CLASSIFIERS

Parametric Method, Using 24 mof features, Group1: tn1 ts1 14
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS11

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	5926 94.06	301 4.78	6 0.10	66 1.05	1 0.02	6300 100.00
2	29 1.38	1760 83.81	3 0.14	308 14.67	0 0.00	2100 100.00
3	1 0.05	3 0.14	1999 95.19	38 1.81	59 2.81	2100 100.00
4	5 0.24	89 4.24	13 0.62	1989 94.71	4 0.19	2100 100.00
5	0 0.00	3 0.14	51 2.43	18 0.86	2028 96.57	2100 100.00
Total	5961	2156	2072	2419	2092	14700

Parametric Method, Using 24 mof features, Group2: tn2 ts2 28
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS12

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	5887 93.44	370 5.87	3 0.05	39 0.62	1 0.02	6300 100.00

2	152 7.24	1820 86.67	8 0.38	120 5.71	0 0.00	2100 100.00
3	0 0.00	16 0.76	2049 97.57	16 0.76	19 0.90	2100 100.00
4	20 0.95	163 7.76	18 0.86	1891 90.05	8 0.38	2100 100.00
5	0 0.00	0 0.00	22 1.05	17 0.81	2061 98.14	2100 100.00
Total	6059	2369	2100	2083	2089	14700

Parametric Method, Using 24 mof features, Group3: tn3 ts3 42

10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS13

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	5030 93.15	286 5.30	6 0.11	75 1.39	3 0.06	5400 100.00
2	43 2.39	1483 82.39	9 0.50	265 14.72	0 0.00	1800 100.00
3	6 0.33	32 1.78	1716 95.33	24 1.33	22 1.22	1800 100.00
4	1 0.06	59 3.28	14 0.78	1721 95.61	5 0.28	1800 100.00
5	0 0.00	1 0.06	36 2.00	13 0.72	1750 97.22	1800 100.00
Total	5080	1861	1781	2098	1780	12600

Parametric Method, Using 20 color features, Group1: tn1 ts1 53

10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS21

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	4975	1080	101	144	0	6300
	78.97	17.14	1.60	.29	0.00	100.00
2	17	1652	329	102	0	2100
	0.81	78.67	15.67	4.86	0.00	100.00
3	40	12	1909	58	81	2100
	1.90	0.57	90.90	2.76	3.86	100.00
4	10	5	11	2074	0	2100
	0.48	0.24	0.52	98.76	0.00	100.00
5	4	0	25	5	2066	2100
	0.19	0.00	1.19	0.24	98.38	100.00
Total	5046	2749	2375	2383	2147	14700

Discriminant Analysis Classification Summary for Test Data: WORK.TS22

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	2782	2569	23	26	0	5400
	51.52	47.57	0.43	0.48	0.00	100.00
2	3	1784	4	9	0	1800
	0.17	99.11	0.22	0.50	0.00	100.00
3	7	174	1611	5	3	1800
	0.39	9.67	89.50	0.28	0.17	100.00
4	3	9	6	1782	0	1800

From SPECIES	1	2	3	4	5	Total
	0.17	0.50	0.33	99.00	0.00	100.00
5	0	0	8	0	1792	1800
	0.00	0.00	0.44	0.00	99.56	100.00
Total	2795	4536	1652	1822	1795	12600

Parametric Method, Using 20 color features, Group3: tn3 ts3 75

10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS23

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)'\ COV_j^{-1}\ (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\ D_j^2(X))\ /\ SUM_k \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	4897	385	33	85	0	5400
	90.69	7.13	0.61	1.57	0.00	100.00
2	15	1370	221	194	0	1800
	0.83	76.11	12.28	10.78	0.00	100.00
3	6	29	1757	3	5	1800
	0.33	1.61	97.61	0.17	0.28	100.00
4	0	2	11	1783	4	1800
	0.00	0.11	0.61	99.06	0.22	100.00
5	0	0	6	0	1794	1800
	0.00	0.00	0.33	0.00	99.67	100.00
Total	4918	1786	2028	2065	1803	12600

Parametric Method, Using 28 selected features, Group1: tn1 ts1 89

10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS31

Classification Summary using Quadratic Discriminant Function

Generalizd Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)'\ COV_j^{-1}\ (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\ D_j^2(X))\ /\ SUM_k \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	6083	99	6	112	0	6300
	96.56	1.57	0.10	1.78	0.00	100.00
2	21	1815	17	119	128	2100
	1.00	86.43	0.81	5.67	6.10	100.00
3	20	2	2006	5	67	2100
	0.95	0.10	95.52	0.24	3.19	100.00
4	5	22	18	2054	1	2100
	0.24	1.05	0.86	97.81	0.05	100.00
5	0	4	21	0	2075	2100
	0.00	0.19	1.00	0.00	98.81	100.00
Total	6129	1942	2068	2290	2271	14700

Parametric Method, Using 28 selected features, Group2: tn2 ts2 103
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS32

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5 \, D_j^2(X)) \, / \, SUM_k \exp(-.5 \, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	6171	51	5	73	0	6300
	97.95	0.81	0.08	1.16	0.00	100.00
2	140	1801	12	147	0	2100
	6.67	85.76	0.57	7.00	0.00	100.00
3	1	3	207	6	18	2100
	0.05	0.14	98.67	0.29	0.86	100.00
4	17	17	15	2051	0	2100
	0.81	0.81	0.71	97.67	0.00	100.00
5	0	1	47	0	2052	2100
	0.00	0.05	2.24	0.00	97.71	100.00
Total	6329	1873	2151	2277	2070	14700

Parametric Method, Using 28 selected features, Group3: tn3 ts3 117
10:33 Friday, February 14, 1997

Discriminant Analysis   Classification Summary for Test Data: WORK.TS33

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5 \, D_j^2(X)) \, / \, SUM_k \, \exp(-.5 \, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	5241	84	0	75	0	5400
	97.06	1.56	0.00	1.39	0.00	100.00
2	31	1327	24	418	0	1800
	1.72	73.72	1.33	23.22	0.00	100.00
3	1	22	1769	3	5	1800
	0.06	1.22	98.28	0.17	0.28	100.00
4	6	7	11	1776	0	1800
	0.33	0.39	0.61	98.67	0.00	100.00
5	0	1	4	0	1795	1800
	0.00	0.06	0.22	0.00	99.72	100.00
Total	5279	1441	1808	2272	1800	12600

Nonparametric Method, Using 24 mof features, Group1: tn1 ts1   127
10:33 Friday, February 14, 1997

Discriminant Analysis   Classification Summary for Test Data: WORK.TS11

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y) \qquad m_k(X) = \text{Proportion of obs in group } k \text{ in 5 nearest neighbors of } X$$

$$Pr(j|X) = m_j(X) \, PRIOR_j \, / \, SUM_k \, (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6041	206	9	29	1	14	6300
	95.89	3.27	0.14	0.46	0.02	0.22	100.00
2	64	1883	0	137	0	16	2100

	3.05	8967	0.00	6.52	0.00	0.76	100.00
3	0	7	2040	24	24	5	2100
	0.00	0.33	97.14	1.14	1.14	0.24	100.00
4	6	195	12	1849	4	34	2100
	0.29	9.29	0.57	88.05	0.19	1.62	100.00
5	0	9	21	14	2050	6	2100
	0.00	0.43	1.00	0.67	97.62	0.29	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS12

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6173	94	2	12	1	18	6300
	97.98	1.49	0.03	0.19	0.02	0.29	100.00
2	221	1740	4	109	0	26	2100
	10.52	82.86	0.19	5.19	0.00	1.24	100.00
3	1	14	2046	18	13	8	2100
	0.05	0.67	97.43	0.86	0.62	0.38	100.00
4	27	167	10	1868	1	27	2100
	1.29	7.95	0.48	88.95	0.05	1.29	100.00
5	0	5	11	14	2068	2	2100
	0.00	0.24	0.52	0.67	98.48	0.10	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS13

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$ $\qquad$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X) \, PRIOR_j \; / \; SUM_k \, (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	5131	188	6	30	2	43	5400
	95.02	3.48	0.11	0.56	0.04	0.80	100.00
2	46	1637	0	106	0	11	1800
	2.56	90.94	0.00	5.89	0.00	0.61	100.00
3	1	8	1761	14	10	6	1800
	0.06	0.44	97.83	0.78	0.56	0.33	100.00
4	5	165	6	1586	2	36	1800
	0.28	9.17	0.33	88.11	0.11	2.00	100.00
5	0	3	16	11	1764	6	1800
	0.00	0.17	0.89	0.61	98.00	0.33	100.00

Nonparametric Method, Using 20 color features, Group1: tn1 ts1 154
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS21

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$ $\qquad$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X) \, PRIOR_j \; / \; SUM_k \, (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6190	35	1	70	0	4	6300
	98.25	0.56	0.02	1.11	0.00	0.06	100.00
2	23	2033	33	8	1	2	2100
	1.10	96.81	1.57	0.38	0.05	0.10	100.00
3	6	111	1858	16	98	11	2100
	0.29	5.29	88.48	0.76	4.67	0.52	100.00

4	17	74	5	1990	0	14	2100
	0.81	3.52	0.24	94.76	0.00	0.67	100.00
5	0	2	11	0	2085	2	2100
	0.00	0.10	0.52	0.00	99.29	0.10	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS22

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neihbors of X

$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6241	41	2	11	0	5	6300
	99.06	0.65	0.03	0.17	0.00	0.08	100.00
2	74	1922	81	9	1	13	2100
	3.52	91.52	3.86	0.43	0.05	0.62	100.00
3	1	20	2057	5	13	4	2100
	0.05	0.95	97.95	0.24	0.62	0.19	100.00
4	5	13	2	2068	0	12	2100
	0.24	0.62	0.10	98.48	0.00	0.57	100.00
5	0	0	108	3	1984	5	2100
	0.00	0.00	5.14	0.14	94.48	0.24	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS23

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m (X) PRIOR / SUM (m (X) PRIOR)$

j j k k k

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	5008	351	23	10	0	8	5400
	92.74	6.50	0.43	0.19	0.00	0.15	100.00
2	8	1761	20	5	0	6	1800
	0.44	97.83	1.11	0.28	0.00	0.33	100.00
3	0	29	1763	0	6	2	1800
	0.00	1.61	97.94	0.00	0.33	0.11	100.00
4	0	10	8	1778	1	3	1800
	0.00	0.56	0.44	98.78	0.06	0.17	100.00
5	0	0	3	0	1797	0	1800
	0.00	0.00	0.17	0.00	99.83	0.00	100.00

Nonparametric Method, Using 28 selected features, Group1: tn1 ts1 178
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS31

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$

$m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m_j(X) PRIOR_j \ / \ SUM_k (m_k(X) PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6212	48	0	31	1	8	6300
	98.60	0.76	0.00	0.49	0.02	0.13	100.00
2	9	2023	0	6	60	2	2100
	0.43	96.33	0.00	0.29	2.86	0.10	100.00
3	0	11	2056	1	27	5	2100
	0.00	0.52	97.90	0.05	1.29	0.24	100.00
4	6	55	2	2030	1	6	2100
	0.29	2.62	0.10	96.67	0.05	0.29	100.00
5	0	7	4	0	2083	6	2100

0.00	0.33	0.19	0.00	99.19	0.29	100.00

Nonparametric Method, Using 28 selected features, Group2: tn2 ts2 188
10:33 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS32

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m_j(X) \, PRIOR_j \, / \, SUM_k (m_k(X) \, PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	6280 99.68	9 0.14	0 0.00	9 0.14	0 0.00	2 0.03	6300 100.00
2	86 4.10	2002 95.33	1 0.05	7 0.33	0 0.00	4 0.19	2100 100.00
3	1 0.05	4 0.19	2091 99.57	0 0.00	4 0.19	0 0.00	2100 100.00
4	9 0.43	15 0.71	0 0.00	2072 98.67	0 0.00	4 0.19	2100 100.00
5	0 0.00	3 0.14	33 1.57	2 0.10	2058 98.00	4 0.19	2100 100.00

Nonparametric Method, Using 28 selected features, Group3: tn3 ts3 198
10:12 Friday, February 14, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS33

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' \, COV^{-1} \, (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m_j(X) \, PRIOR_j \, / \, SUM_k (m_k(X) \, PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	OTHER	Total
1	5196	191	1	5	0	7	5400
	96.22	3.54	0.02	0.09	0.00	0.13	100.00
2	2	1784	2	7	0	5	1800
	0.11	99.11	0.11	0.39	0.00	0.28	100.00
3	0	8	1791	0	1	0	1800
	0.00	0.44	99.50	0.00	0.06	0.00	100.00
4	0	13	0	1784	0	3	1800
	0.00	0.72	0.00	99.11	0.00	0.17	100.00
5	0	0	2	1	1797	0	1800
	0.00	0.00	0.11	0.06	99.83	0.00	100.00

APPENDIX F-2

RESULTS OF DAMAGE TYPE IDENTIFICATION ANALYSIS OF INDIVIDUAL CWRS WHEAT KERNELS USING STATISTICAL CLASSIFIERS

Parametric Method, Using 28 mof features, Group1: tn11 ts11 16
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS11

From SPECIES	1	2	3	4	5	6	7	Total
1	175 58.33	1 0.33	81 27.00	3 1.00	12 4.00	15 5.00	13 4.33	300 100.00
2	30 10.00	160 53.33	46 15.33	39 13.00	5 1.67	18 6.00	2 0.67	300 100.00
3	64 21.33	3 1.00	175 58.33	7 2.33	16 5.33	28 9.33	7 2.33	300 100.00
4	0 0.00	5 1.67	14 4.67	224 74.67	3 1.00	26 8.67	28 9.33	300 100.00
5	4 1.33	5 1.67	14 4.67	0 0.00	227 75.67	16 5.33	34 11.33	300 100.00
6	16 5.33	6 2.00	21 7.00	11 3.67	66 22.00	103 34.33	77 25.67	300 100.00
7	6 2.00	2 0.67	15 5.00	7 2.33	22 7.33	38 12.67	210 70.00	300 100.00

Parametric Method, Using 28 mof features, Group2: tn12 ts12 32
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS12

From SPECIES	1	2	3	4	5	6	7	Total
1	181 60.33	2 0.67	62 20.67	4 1.33	19 6.33	14 4.67	18 6.00	300 100.00
2	25 8.33	183 61.00	39 13.00	34 11.33	2 0.67	14 4.67	3 1.00	300 100.00
3	94 31.33	1 0.33	164 54.67	5 1.67	17 5.67	7 2.33	12 4.00	300 100.00
4	1 0.33	11 3.67	24 8.00	217 72.33	1 0.33	24 8.00	22 7.33	300 100.00
5	5 1.67	4 1.33	8 2.67	1 0.33	257 85.67	7 2.33	18 6.00	300 100.00
6	31 10.33	1 0.33	40 13.33	9 3.00	64 21.33	48 16.00	107 35.67	300 100.0

7	7	0	22	9	39	13	210	300
	2.33	0.00	7.33	3.00	13.00	4.33	70.00	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS13

From SPECIES	1	2	3	4	5	6	7	Total
1	212	3	146	2	10	15	12	400
	53.00	0.75	36.50	0.50	2.50	3.75	3.00	100.00
2	29	255	52	50	1	8	5	400
	7.25	63.75	13.00	12.50	0.25	2.00	1.25	100.00
3	77	2	251	15	11	17	27	400
	19.25	0.50	62.75	3.75	2.75	4.25	6.75	100.00
4	1	7	18	330	1	21	22	400
	0.25	1.75	4.50	82.50	0.25	5.25	5.50	100.00
5	11	3	22	3	274	21	66	400
	2.75	0.75	5.50	0.75	68.50	5.25	16.50	100.00
6	26	1	45	24	64	103	137	400
	6.50	0.25	11.25	6.00	16.00	25.75	34.25	100.00
7	4	2	18	26	24	25	301	400
	1.00	0.50	4.50	6.50	6.00	6.25	75.25	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS11

From SPECIES	1	2	3	4	5
1	132	1	68	0	15
	44.00	0.33	22.67	0.00	5.00
2	15	164	27	13	6
	5.00	54.67	9.00	4.33	2.00
3	48	0	126	4	19
	16.00	0.00	42.00	1.33	6.33
4	0	1	3	178	7
	0.00	0.33	1.00	59.33	2.33
5	1	0	12	0	210
	0.33	0.00	4.00	0.00	70.00

6	16 5.33	1 0.33	10 3.33	12 4.00	65 21.67
7	4 1.33	1 0.33	16 5.33	4 1.33	29 9.67

Nonparametric Method, Using 28 mof features, Group1: tn11 ts11 63
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS11

From SPECIES	6	7	OTHER	Total
1	17 5.67	14 4.67	53 17.67	300 100.00
2	25 8.33	9 3.00	41 13.67	300 100.00
3	30 10.00	16 5.33	57 19.00	300 100.00
4	19 6.33	36 12.00	56 18.67	300 100.00
5	16 5.33	27 9.00	34 11.33	300 100.00
6	73 24.33	62 20.67	61 20.33	300 100.00
7	31 10.33	153 51.00	62 20.67	300 100.00

Nonparametric Method, Using 28 mof features, Group2: tn12 ts12 77
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS12

From SPECIES	1	2	3	4	5
1	123 41.00	5 1.67	62 20.67	1 0.33	21 7.00
2	15 5.00	183 61.00	25 8.33	20 6.67	2 0.67
3	53 17.67	5 1.67	138 46.00	5 1.67	11 3.67
4	4 1.33	1 0.33	17 5.67	195 65.00	2 0.67

5	2 0.67	0 0.00	6 2.00	6 2.00	202 67.33
6	19 6.33	0 0.00	18 6.00	9 3.00	39 13.00
7	2 0.67	0 0.00	3 1.00	9 3.00	23 7.67

Nonparametric Method, Using 28 mof features, Group2: tn12 ts12 78
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS12

1	18 6.00	15 5.00	55 18.33	300 100.00
2	7 2.33	5 1.67	43 14.33	300 100.00
3	15 5.00	17 5.67	56 18.67	300 100.00
4	17 5.67	24 8.00	40 13.33	300 100.00
5	19 6.33	23 7.67	42 14.00	300 100.00
6	68 22.67	76 25.33	71 23.67	300 100.00
7	25 8.33	186 62.00	52 17.33	300 100.00

Nonparametric Method, Using 28 mof features, Group3: tn13 ts13 92
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS13

From SPECIES	1	2	3	4	5
1	182 45.50	4 1.00	106 26.50	1 0.25	17 4.25
2	33 8.25	226 56.50	37 9.25	33 8.25	1 0.25
3	71 17.75	3 0.75	184 46.00	18 4.50	10 2.50
4	2 0.50	4 1.00	15 3.75	273 68.25	0 0.00

5	3 0.75	1 0.25	14 3.50	0 0.00	264 66.00
6	15 3.75	0 0.00	19 4.75	17 4.25	70 17.50
7	4 1.00	0 0.00	10 2.50	18 4.50	29 7.25

Nonparametric Method, Using 28 mof features, Group3: tn13 ts13 93
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS13

From SPECIES	6	7	OTHER	Total
1	16 4.00	7 1.75	67 16.75	400 100.00
2	18 4.50	11 2.75	41 10.25	400 100.00
3	21 5.25	18 4.50	75 18.75	400 100.00
4	16 4.00	36 9.00	54 13.50	400 100.00
5	30 7.50	36 9.00	52 13.00	400 100.00
6	107 26.75	78 19.50	94 23.50	400 100.00
7	36 9.00	213 53.25	90 22.50	400 100.00

Parametric Method, Uing 28 col features, Group1: tn21 ts21 109
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS21

From SPECIES	1	2	3	4	5	6	7	Total
1	200 66.67	3 1.00	72 24.00	0 0.00	10 3.33	15 5.00	0 0.00	300 100.00
2	51 17.00	151 50.33	38 12.67	12 4.00	18 6.00	30 10.00	0 0.00	300 100.00
3	7 2.33	0 0.00	293 97.67	0 0.00	0 0.00	0 0.00	0 0.00	300 100.00

4	0	1	0	287	9	3	0	300
	0.00	0.33	000	95.67	3.00	1.00	0.00	100.00
5	0	1	0	0	298	1	0	300
	0.00	0.33	0.00	0.00	99.33	0.33	0.00	100.00
6	1	0	0	18	4	277	0	300
	0.33	0.00	0.00	6.00	1.33	92.33	0.00	100.00
7	0	0	0	0	0	0	300	300
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00

Parametric Method, Using 28 col features, Group2: tn22 ts22 125
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS22

From SPECIES	1	2	3	4	5	6	7	Total
1	206	4	80	0	3	7	0	300
	68.67	1.33	26.67	0.00	1.00	2.33	0.00	100.00
2	57	175	33	14	8	13	0	300
	19.00	58.33	11.00	4.67	2.67	4.33	0.00	100.00
3	6	1	293	0	0	0	0	300
	2.00	0.33	97.67	0.00	0.00	0.00	0.00	100.00
4	0	1	0	286	1	12	0	300
	0.00	0.33	0.00	95.33	0.33	4.00	0.00	100.00
5	0	0	0	3	282	15	0	300
	0.00	0.00	0.00	1.00	94.00	5.00	0.00	100.00
6	0	0	0	23	0	277	0	300
	0.00	0.00	0.00	7.67	0.00	92.33	0.00	100.00
7	0	0	0	0	0	0	300	300
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00

Parametric Method, Using 28 col features, Group3: tn23 ts23 141
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS23

From SPECES	1	2	3	4	5	6	7	Total
1	304	4	84	0	4	4	0	400
	76.00	1.00	21.00	0.00	1.00	1.00	0.00	100.00
2	108	187	55	28	6	16	0	400
	27.00	46.75	13.75	7.00	1.50	4.00	0.00	100.00

3	14	1	385	0	0	0	0	400
	3.50	0.25	96.25	0.00	0.00	0.00	0.00	100.00
4	0	1	0	391	0	8	0	400
	0.00	0.25	0.00	97.75	0.00	2.00	0.00	100.00
5	0	0	0	10	374	16	0	400
	0.00	0.00	0.00	2.50	93.50	4.00	0.00	100.00
6	0	0	0	35	2	363	0	400
	0.00	0.00	0.00	8.75	0.50	90.75	0.00	100.00
7	0	0	0	0	0	0	400	400
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS21

From SPECIES	1	2	3	4	5
1	269	3	14	0	1
	89.67	1.00	4.67	0.00	0.33
2	18	250	13	3	1
	6.00	83.33	4.33	1.00	0.33
3	5	0	294	0	0
	1.67	0.00	98.00	0.00	0.00
4	0	0	0	287	5
	0.00	0.00	0.00	95.67	1.67
5	0	2	0	1	297
	0.00	0.67	0.00	0.33	99.00
6	1	0	0	9	3
	0.33	0.00	0.00	3.00	1.00
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

Nonparametric Method, Using 28 col features, Group1: tn21 ts21 156
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS21

From SPECIES	6	7	OTHER	Total
1	2 0.67	0 0.00	11 3.67	300 100.00
2	0 0.00	0 0.00	15 5.00	300 100.00
3	0 0.00	0 0.00	1 0.33	300 100.00
4	7 2.33	0 0.00	1 0.33	300 100.00
5	0 0.00	0 0.00	0 0.00	300 100.00
6	285 95.00	0 0.00	2 0.67	300 100.00
7	0 0.00	300 100.00	0 0.00	300 100.00

Nonparametric Method, Using 28 col features, Group2: tn22 ts22 170
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS22

From SPECIES	1	2	3	4	5
1	264 88.00	4 1.33	21 7.00	0 0.00	1 0.33
2	21 7.00	260 86.67	10 3.33	0 0.00	0 0.00
3	4 1.33	1 0.33	291 97.00	0 0.00	0 0.00
4	0 0.00	2 0.67	0 0.00	293 97.67	1 0.33
5	0 0.00	0 0.00	0 0.00	0 0.00	297 99.00
6	0 0.00	0 0.00	0 0.00	2 0.67	1 0.33

7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

Nonparametric Method, Using 28 col features, Group2: tn22 ts22 171
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS22

From SPECIES	6	7	OTHER	Total
1	0 0.00	0 0.00	10 3.33	300 100.00
2	1 0.33	0 0.00	8 2.67	300 100.00
3	0 0.00	0 0.00	4 1.33	300 100.00
4	2 0.67	0 0.00	2 0.67	300 100.00
5	3 1.00	0 0.00	0 0.00	300 100.00
6	297 99.00	0 0.00	0 0.00	300 100.00
7	0 0.00	300 100.00	0 0.00	300 100.00

Nonparametric Method, Using 28 col features, Group3: tn23 ts23 185
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS23

From SPECIES	1	2	3	4	5
1	337 84.25	13 3.25	35 8.75	0 0.00	1 0.25
2	17 4.25	339 84.75	21 5.25	1 0.25	2 0.50
3	0 2.50	0 0.00	389 97.25	0 0.00	0 0.00
4	0 0.00	1 0.25	0 0.00	391 97.75	0 0.00
5	0 0.00	0 0.00	0 0.00	0 0.00	396 99.00

6	1	1	0	1	2
	0.25	0.25	0.00	0.25	0.50
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS23

From SPECIES	6	7	OTHER	Total
1	0	0	14	400
	0.00	0.00	3.50	100.00
2	1	0	19	400
	0.25	0.00	4.75	100.00
3	0	0	1	400
	0.00	0.00	0.25	100.00
4	6	0	2	400
	1.50	0.00	0.50	100.00
5	4	0	0	400
	1.00	0.00	0.00	100.00
6	389	0	6	400
	97.25	0.00	1.50	100.00
7	0	400	0	400
	0.00	100.00	0.00	100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS31

From SPECIES	1	2	3	4	5	6	7	Total
1	272	0	26	0	2	0	0	300
	90.67	0.00	8.67	0.00	0.67	0.00	0.00	100.00
2	62	211	15	0	11	1	0	300
	20.67	70.33	5.00	0.00	3.67	0.33	0.00	100.00
3	8	0	292	0	0	0	0	300
	2.67	0.00	97.33	0.00	0.00	0.00	0.00	100.00
4	0	1	0	294	2	3	0	300
	0.00	0.33	0.00	98.00	0.67	1.00	0.00	100.00

5	0	2	0	1	296	1	0	300
	0.00	0.67	0.00	0.33	98.67	0.33	0.00	100.00
6	0	0	0	8	4	288	0	300
	0.00	0.00	0.00	2.67	1.33	96.00	0.00	100.00
7	0	0	0	0	0	0	300	300
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00

Parametric Method, Using 28 cmb features, Group2: tn32 ts32 218
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS32

From SPECIES	1	2	3	4	5	6	7	Total
1	256	3	39	0	2	0	0	300
	85.33	1.00	13.00	0.00	0.67	0.00	0.00	100.00
2	62	228	9	0	1	0	0	300
	20.67	76.00	3.00	0.00	0.33	0.00	0.00	100.00
3	10	1	289	0	0	0	0	300
	3.33	0.33	96.33	0.00	0.00	0.00	0.00	100.00
4	0	0	0	288	1	11	0	300
	0.00	0.00	0.00	96.00	0.33	3.67	0.00	100.00
5	0	0	0	2	291	7	0	300
	0.00	0.00	0.00	0.67	97.00	2.33	0.00	100.00
6	0	0	0	17	1	282	0	300
	0.00	0.00	0.00	5.67	0.33	94.00	0.00	100.00
7	0	0	0	0	0	0	300	300
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00

Parametric Method, Using 28 cmb features, Group3: tn33 ts33 234
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS33

From SPECIES	1	2	3	4	5	6	7	Total
1	328	3	66	0	2	1	0	400
	82.00	0.75	16.50	0.00	0.50	0.25	0.00	100.00
2	68	291	33	4	3	1	0	400
	17.00	72.75	8.25	1.00	0.75	0.25	0.00	100.00
3	11	1	388	0	0	0	0	400
	2.75	0.25	97.00	0.00	0.00	0.00	0.00	100.0

4	0	1	0	394	0	5	0	400	
	0.00	0.25	0.00	98.50	0.00	1.25	0.00	100.00	
5	0	0	0	3	392	5	0	400	
	0.00	0.00	0.00	0.75	98.00	1.25	0.00	100.00	
6	0	0	0	31	2	367	0	400	
	0.00	0.00	0.00	7.75	0.50	91.75	0.00	100.00	
7	0	0	0	0	0	0	400	400	
	0.00	0.00	0.00	0.00	0.00	0.00	100.00	100.00	

Discriminant Analysis Classification Summary for Test Data: WORK.TS31

From SPECIES	1	2	3	4	5
1	278	2	16	0	0
	92.67	0.67	5.33	0.00	0.00
2	16	271	3	0	1
	5.33	90.33	1.00	0.00	0.33
3	2	0	297	0	0
	0.67	0.00	99.00	0.00	0.00
4	0	0	0	297	1
	0.00	0.00	0.00	99.00	0.33
5	2	0	0	0	296
	0.67	0.00	0.00	0.00	98.67
6	0	0	0	6	3
	0.00	0.00	0.00	2.00	1.00
7	0	0	0	0	0
	0.00	0.00	0.00	0.00	0.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS31

From SPECIES	6	7	OTHER	Total
1	0	0	4	300
	0.00	0.00	1.33	100.00
2	0	0	9	300
	0.00	0.00	3.00	100.00

3	0 0.00	0 0.00	1 0.33	300 100.00
4	2 0.67	0 0.00	0 0.00	300 100.00
5	1 0.33	0 0.00	1 0.33	300 100.00
6	289 96.33	0 0.00	2 0.67	300 100.00
7	0 0.00	300 100.00	0 0.00	300 100.00

Nonparametric Method, Using 28 cmb features, Group2: tn32 ts32 263
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS32

From SPECIES	1	2	3	4	5
1	283 94.33	5 1.67	10 3.33	0 0.00	1 0.33
2	19 6.33	272 90.67	5 1.67	1 0.33	0 0.00
3	4 1.33	1 0.33	295 98.33	0.00	0 0.00
4	0 0.00	0 0.00	0 0.00	295 98.33	0 0.00
5	1 0.33	0 0.00	0 0.00	0 0.00	298 99.33
6	0 0.00	0 0.00	0 0.00	5 1.67	3 1.00
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

Nonparametric Method, Using 28 cmb features, Group2: tn32 ts32 264
10:19 Wednesday, March 19, 1997

Discriminant Analysis Classification Summary for Test Data: WORK.TS32

From PECIES	6	7	OTHER	Total

1	0 0.00	0 0.00	1 0.33	300 100.00
2	0 0.00	0 0.00	3 1.00	300 100.00
3	0 0.00	0 0.00	0 0.00	300 100.00
4	5 1.67	0 0.00	0 0.00	300 100.00
5	1 0.33	0 0.00	0 0.00	300 100.00
6	292 97.33	0 0.00	0 0.00	300 100.00
7	0 0.00	300 100.00	0 0.00	300 100.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS33

From SPECIES	1	2	3	4	5
1	362 90.50	4 1.00	25 6.25	0 0.00	2 0.50
2	27 6.75	359 89.75	6 1.50	0 0.00	1 0.25
3	6 1.50	0 0.00	394 98.50	0 0.00	0 0.00
4	0 0.00	0 0.00	0 0.00	399 99.75	0 0.00
5	0 0.00	0 0.00	0 0.00	0 0.00	397 99.25
6	0 0.00	0 0.00	0 0.00	1 0.25	2 0.50
7	0 0.00	0 0.00	0 0.00	0 0.00	0 0.00

Discriminant Analysis Classification Summary for Test Data: WORK.TS33

From SPECIES	6	7	OTHER	Total
1	0	0	7	400
	0.00	0.00	1.75	100.00
2	0	0	7	400
	0.00	0.00	1.75	100.00
3	0	0	0	400
	0.00	0.00	0.00	100.00
4	1	0	0	400
	0.25	0.00	0.00	100.00
5	2	0	1	400
	0.50	0.00	0.25	100.00
6	395	0	2	400
	98.75	0.00	0.50	100.00
7	0	400	0	400
	0.00	100.00	0.00	100.00

APPENDIX F-3

RESULTS OF GRAIN TYPE IDENTIFICATION ANALYSIS OF BULK GRAIN SAMPLES USING STATISTICAL CLASSIFIERS

Parametric Method, Using 8 slc features, Group1: tn1 ts1   15:57 Sunday, April 13, 1997

10

Discriminant Analysis   Classification Summary for Test Data: WORK.TS1

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)'\ COV_j^{-1}\ (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\ D_j^2(X))\ /\ SUM_k\ \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	63	0	0	0	0	63
	100.00	0.00	0.00	0.00	0.00	100.00
2	5	16	0	0	0	21
	23.81	76.19	0.00	0.00	0.00	100.00
3	0	0	21	0	0	21
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	21	0	21
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	21	21
	0.00	0.00	0.00	0.00	100.00	100.00
Total	68	16	21	21	21	147
Percent	46.26	10.88	14.29	14.29	14.29	100.00

Parametric Method, Using 8 slc features, Group2: tn2 ts2   15:57 Sunday, April 13, 1997

20

Discriminant Analysis   Classification Summary for Test Data: WORK.TS2

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)'\ COV_j^{-1}\ (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\ D_j^2(X))\ /\ SUM_k\ \exp(-.5\ D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	63	0	0	0	0	63

	100.00	0.00	0.00	0.00	0.00	100.00
2	0 0.00	21 100.00	0 0.00	0 0.00	0 0.00	21 100.00
3	0 0.00	0 0.00	21 100.00	0 0.00	0 0.00	21 100.00
4	0 0.00	0 0.00	0 0.00	21 100.00	0 0.00	21 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	21 100.00	21 100.00
Total Percent	63 42.86	21 14.29	21 14.29	21 14.29	21 14.29	147 100.00

Parametric Method, Using 8 slc features, Group3: tn3 ts3 15:57 Sunday, April 13, 1997

30

Discriminant Analysis Classification Summary for Test Data: WORK.TS3

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j|$ $\quad Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	54 100.00	0 0.00	0 0.00	0 0.00	0 0.00	54 100.00
2	0 0.00	18 100.00	0 0.00	0 0.00	0 0.00	18 100.00
3	0 0.00	0 0.00	18 100.00	0 0.00	0 0.00	18 100.00
4	0 0.00	0 0.00	0 0.00	18 100.00	0 0.00	18 100.00
5	0 0.00	0 0.00	0 0.00	0 0.00	18 100.00	18 100.00
Total Percent	54 42.86	18 14.29	18 14.29	18 14.29	18 14.29	126 100.00

Nonparametric Method, Using 8 slc features, Group1: tn1 ts1 15:57 Sunday, April 13, 1997 36

Discriminant Analysis Classification Summary for Test Data: WORK.TS1

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$

$m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	63	0	0	0	0	63
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	21	0	0	0	21
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	21	0	0	21
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	21	0	21
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	21	21
	0.00	0.00	0.00	0.00	100.00	100.00

Nonparametric Method, Using 8 slc features, Group2: tn2 ts2 15:57 Sunday, April 13, 1997 42

Discriminant Analysis Classification Summary for Test Data: WORK.TS2

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y)$

$m_k(X)$ = Proportion of obs in group k in 5 nearest neighbos of X

$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	63	0	0	0	0	63

	100.00	0.00	0.00	0.00	0.00	100.00
2	0	21	0	0	0	21
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	21	0	0	21
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	21	0	21
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	21	21
	0.00	0.00	0.00	0.00	100.00	100.00

Nonparametric Method, Using 8 slc features, Group3: tn3 ts3   15:57 Sunday, April 13, 1997 48

Discriminant Analysis   Classification Summary for Test Data: WORK.TS3

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$D^2(X,Y) = (X-Y)'\ COV^{-1}\ (X-Y)$ $m_k(X)$ = Proportion of obs in group k in 5 nearest neighbors of X

$$Pr(j|X) = m_j(X)\ PRIOR_j\ /\ SUM_k\ (\ m_k(X)\ PRIOR_k\ )$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	4	5	Total
1	54	0	0	0	0	54
	100.00	0.00	0.00	0.00	0.00	100.00
2	0	18	0	0	0	18
	0.00	100.00	0.00	0.00	0.00	100.00
3	0	0	18	0	0	18
	0.00	0.00	100.00	0.00	0.00	100.00
4	0	0	0	18	0	18
	0.00	0.00	0.00	100.00	0.00	100.00
5	0	0	0	0	18	18
	0.00	0.00	0.00	0.00	100.00	100.00

APPENDIX F-4

RESULTS OF GRADE IDENTIFICATION ANALYSIS OF BULK CWRS WHEAT SAMPLES USING STATISTICAL CLASSIFIERS

58 Parametric Method, Using 20 slc features, Group1: tn1 ts1   15:57 Sunday, April 13, 1997

Discriminant Analysis   Classification Summary for Test Data: WORK.TS1

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\, D_j^2(X)) / SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	19 90.48	1 4.76	1 4.76	21 100.00
2	3 14.29	7 33.33	11 52.38	21 100.00
3	4 19.05	1 4.76	16 76.19	21 100.0
Total Percent	26 41.27	9 14.29	28 44.44	63 100.00
Priors	0.3333	0.3333	0.3333	

Error Count Estimates for SPECIES:

68 Parametric Method, Using 20 slc features, Group2: tn2 ts2   15:57 Sunday, April 13, 1997

Discriminant Analysis   Classification Summary for Test Data: WORK.TS2

Classification Summary using Quadratic Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\, D_j^2(X)) / SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	16 76.19	3 14.29	2 9.52	21 100.00

2	3 14.29	14 66.67	4 19.05	21 100.00
3	5 23.81	9 42.86	7 33.33	21 100.00
Total Percent	24 38.10	26 41.27	13 20.63	63 100.00
Priors	0.3333	0.3333	0.3333	

Error Count stimates for SPECIES:

Parametric Method, Using 20 slc features, Group3: tn3 ts3 15:57 Sunday, April 13, 1997

78

Discriminant Analysis Classification Summary for Test Data: WORK.TS3

Classification Summary using Quadratic Discriminant Function

Generalized Suared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| \qquad Pr(j|X) = \exp(-.5\,D_j^2(X)) \,/\, SUM_k \exp(-.5\,D_k^2(X))$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	Total
1	10 55.56	3 16.67	5 27.78	18 100.00
2	0 0.00	17 94.44	1 5.56	18 100.00
3	0 0.00	2 11.11	16 88.89	18 100.00
Total Percent	10 18.52	22 40.74	22 40.74	54 100.00
Priors	0.3333	0.3333	0.3333	

Error Count Estimates for SPECIES:

Nonparametric Method, Using 20 slc features, Group1: tn1 ts1 15:57 Sunday, April 13,

1997 84

Discriminant Analysis Classification Summary for Test Data: WORK.TS1

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y) \qquad m_k(X) = \text{Proportion of obs in group k in 5 nearest neighbors of X}$$

$$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	18 85.71	2 9.52	0 0.00	1 4.76	21 100.00
2	3 14.29	11 52.38	4 19.05	3 14.29	21 100.00
3	0 0.00	10 47.62	9 42.86	2 9.52	21 100.00
Total Percent	21 33.33	23 36.51	13 20.63	6 9.52	63 100.00
Priors	0.3333	0.3333	0.3333		

Nonparametric Method, Using 20 slc features, Group2: tn2 ts2 15:57 Sunday, April 13, 1997 90

Discriminant Analysis Classification Summary for Test Data: WORK.TS2

Classification Summary using 5 Nearest eighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' COV^{-1} (X-Y) \qquad m_k(X) = \text{Proportion of obs in group k in 5 nearest neighbors of X}$$

$$Pr(j|X) = m_j(X) PRIOR_j / SUM_k (m_k(X) PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	20 95.24	1 4.76	0 0.00	0 0.00	21 100.00
2	2 9.52	16 76.19	2 9.52	1 4.76	21 100.00

3	4	11	5	1	21
	19.05	52.38	23.81	4.76	100.00
Total	26	28	7	2	63
Percent	41.27	44.44	11.11	3.17	100.00
Priors	0.3333	0.3333	0.3333		

Discriminant Analysis Classification Summary for Test Data: WORK.TS3

Classification Summary using 5 Nearest Neighbors

Squared Distance Function: Posterior Probability of Membership in each SPECIES:

$$D^2(X,Y) = (X-Y)' \, COV^{-1} (X-Y) \qquad m_k(X) = \text{Proportion of obs in group k in 5 nearest neighbors of X}$$

$$Pr(j|X) = m_j(X) \, PRIOR_j \, / \, SUM_k \, (m_k(X) \, PRIOR_k)$$

Number of Observations and Percent Classified into SPECIES:

From SPECIES	1	2	3	OTHER	Total
1	16	0	0	2	18
	88.89	0.00	0.00	11.11	100.00
2	0	6	6	6	18
	0.00	33.33	33.33	33.33	100.00
3	0	2	16	0	18
	0.00	11.11	88.89	0.00	100.00
Total	16	8	22	8	54
Percent	29.63	14.81	40.74	14.81	100.00
Priors	0.3333	0.3333	0.3333		