CLASSIFICATION OF WHEAT KERNELS BY MACHINE-VISION MEASUREMENT

20s

by

TERRY EUGENE SCHMALZRIED

B. S., Kansas State University, 1983

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1985

Approved by:

Stephen A. Dyer

Major Professor

To Professor Jacob J. Smaltz.

ACKNOWLEDGEMENTS

The Electrical and Computer Engineering and the Grain Science and Industry departments financially supported this research. Julie Liska-Mason patiently positioned and took pictures of hundreds of kernels of wheat. Gerald Schmalzried proofread and made grammatical corrections on the rough draft of this report. Dr. Arthur Davis, Dr. Jon Faubion, Dr. Carl Hoseney, and Dr. Stephen Dyer took care of the politics and offered technical advice. To each of them, I would like to express my sincerest thanks.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

"The [U.S.] Grain Standards Act provides in part that all grain shipped in interstate or foreign commerce to or from a point at which an inspector licensed under the act is located must be officially inspected and graded if the grain be merchandised by grade."[6] The inspector uses mechanical measurements as much as possible to grade the wheat, but "there is no mechanical test for determining the class to which any particular lot of grain belongs. Identification of the different classes of grain depends almost entirely upon the grader's knowledge of and familiarity with the different kinds of grain that come under his observation."[6] A device that could recognize and itemize the representative varieties or classes present in a sample of wheat would be of great use to the grain inspector.

This paper describes the computer programs that were written by the author for a machine-vision system that classified kernels of wheat by variety. The computer programs gathered and analyzed data on four varieties of wheat and then used that information to choose a minimal set of features to be used to differentiate between the varieties. The results and some suggestions for further research are included at the end of this report.

Historical Kernel Characteristics

The Grain Grading Primer[6], published by the Department of Agriculture, gives an indication of the kernel characteristics used by experienced grain inspectors to grade wheat. "Color, kernel texture, and variety characters are helpful indexes in determining the class or subclass."[6] The "relative size and shape of the germs, the appearance of the crease -- whether open or tightly closed -- and the outline of the kernel from both the side and top views"[6] are also helpful. The indexes of color, texture, kernel shape, and variety characters are discussed in the following paragraphs.

Color. The first of these indexes, color of the wheat kernel, is due to the amount of pigment in the bran layer (see Figure 1) and can vary with variety, texture, and environment. Wheat can be found throughout the world in varying shades and mixtures of white, red, yellow, and purple.[5]

Texture. The second index, kernel texture, refers to the hardness of the endosperm (see Figure 1) and is usually tested by cutting or biting the kernel and examining the inside. "A soft wheat is one which, when normally developed, has an endosperm entirely soft, mealy, or starchy. A hard kernel, when normally developed, has a corneous, horny, or vitreous endosperm throughout."[5]

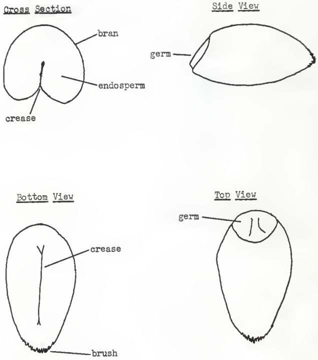


Figure 1. Parts of a Wheat Kernel.

__Shape__. Some fairly constant kernel shapes for all of the wheat classes, except the mixed class, are described in the following excerpt from the Grain Grading Primer:

> ... fully developed Durums are pointed sharply at the germ end and are widest back of the center of the kernel; most other wheats are widest near the germ. The crease in Durums is tightly closed and the seed coat over the germ is not as wrinkled as in other classes. The White class ... is always yellowish white or tannish in color, the germs are large and the creases are wide open. The Soft Red Winter class ... is generally characterized by soft texture, large germs, open creases with rounded cheeks, and a red or yellowish red color.
>
> The Hard Red Winter class is represented by slender elliptical kernels, with a small germ, and tightly closed crease. When viewed from the side the bottom line of the kernel is relatively straight or slightly rocker in outline. The Hard Red Spring class is represented by short kernels, usually hard and vitreous, and dark red in color. The germ is midsized, the crease deep and open, and the kernels have a prominent hump near the germ at one side of the center line of the back.[6]

The primer then goes on to say that class descriptions are not always sufficient, especially in "some of the more recent wheat varieties, resulting from crossing varieties of different classes." Inspectors must recognize varietal traits as well. Some varieties are so similiar to others that they are virtually indistinguishable; classifying them "requires experience and study."[6]

__Variety characters__. John H. Martin[5], in 1922, investigated the use of kernel characteristics in classifying wheat by variety. He designates the major distinctions between kernels to be color, length, and texture. Of minor importance, but also useful, are kernel shape and "differences of the germ, crease, cheeks, and brush."[5] He recommends that length measurements be taken only from the normal kernels, and not from kernels taken "from the top spikelets on a spike and from the upper florets in the spikelet" because they are "below normal in length."[5]

Selection of Discriminators

Noting that, unlike thumbtacks, the kernels of a single variety of wheat are not mass produced from the same mold, characteristics chosen to identify wheat varieties must be tolerant of the differences that occur among the kernels of the same variety. One such difference is in overall kernel size, and can be attributed to the growing region, weather, and position in the wheat head. A normalization technique, in which ratios of measurements from a kernel are compared rather than absolute measurements, might be useful in getting around the size differences. But which measurements should the vision system use as discriminators?

The first ideas for discriminators were simple measurements such as height, width, and length. To these were added radial measurements from the centroid in both top and side views. Then, the possibility of trying to measure some of the more abstract shape features such as pointedness, degree of curvature, surface texture, relative germ size, and germ angle could be considered if time permitted. Three-dimensional mapping for crease, cheek, and back-peaking measurements had to be shelved due to their complexity. Kernel color was dismissed because of its variability with environment. Kernel texture was discarded because the testing procedure involved destroying the kernel by cutting it before examination.

The method chosen for classification of the four varieties of wheat was to take radial measurements from the centroid at half degree increments in two views (top and side); to normalize the measurements with respect to the kernel's length and width (height for side view); and then to find a small number of measurements that, in combination, would give good classification results for the four variety test case. The total number of measurements taken for each kernel was 1440, which was more than enough data to analyze for one project.

An Explanation of the Procedure

Program steps. The following three steps are used to find which measurements will be the best discriminators and give the best classification results: 1) collection of edge contours for the wheat sample set, 2) measurement of the kernel contours, and 3) exhaustive search for optimal discriminators. The inner workings of the programs that accomplish each of these three steps are detailed in the following chapter entitled "Vision System Internals."

Operator intervention. In the final classification system, no operator intervention should be required. But, for this research system, the operator is required to do several tasks. First, the operator must enter the commands that run the programs. Second, he must orient the kernels crease side down before taking pictures of them and must keep them in the same order for both views. Third, computer chosen perimeters are shown to him on a video monitor and he selects or rejects them so that boundaries of touching or broken kernels are not processed. And fourth, he indicates the germ end of the kernel using a joystick so that the measurements will be taken starting from the same end for of all of the kernels.

Chapter 2

VISION SYSTEM INTERNALS

Obtaining the Data

Hardware. The vision system hardware consists of a digitizing camera, a general-purpose computer, and a video display terminal. The camera passes the image to the computer as a 512 by 512 array of integers as shown in Figure 2. Each integer represents the average gray level (light intensity) at the corresponding point in the picture and can range in value from 0 (black) to 255 (white). An element of the image array is called a pixel, which comes from the words picture and element. Although looking at a picture and picking out the outline of an individual grain of wheat from it is a simple task for a human, it is not nearly so simple for a computer. Computers do not posess the abstract concept of objects that humans do. The computer must find the outline mathematically, using that array of integers.

Object silhouette. One method of showing an object to a computer that is easy for it to understand is to show the object in silhouette. The object is placed on a light table with the camera above, and looking down on, the object. The object will appear dark on a bright background to the camera. The image array passed to the computer will hold large integers for the

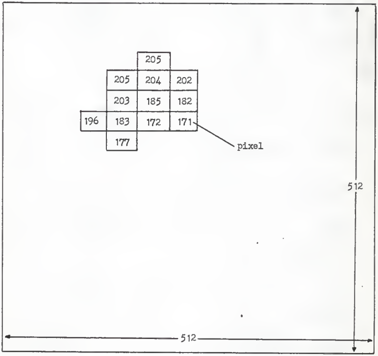


Figure 2. Image Array.

background pixels and small integers for the object pixels. A small value next to a large value indicates an edge. But what is a small value, and what is a large value? In order to determine what these relative values are for a specific image, a histogram is used.

Image histogram. An image histogram is a function consisting of the number of times that a pixel intensity occurs in an image versus the pixel intensity value (see Figure 3). It is created by scanning through the image array and tallying up the number of occurrences of each intensity present. The tallies are kept in a histogram array. When the number of occurrences is plotted versus the intensity values, the resulting function will have two humps for a back-lit image. Such a function is called bimodal.[2] The two humps occur in the function because most of the points in the picture are either object (dark) or background (light) points and only a few of the points occur in the transition between object and background. Therefore, in order to separate the object regions of the image from the background regions, the intensity value in the center of the valley can be chosen as a threshold between small and large pixel intensities.[2]

Automatic thresholding. Locating the valley intensity in the histogram is begun by smoothing the histogram by averaging each value with its four neighbors (two on either side). Then, the smoothed function is scanned from intensity 255 to intensity 0, comparing the occurrence tallies against a minimum peak

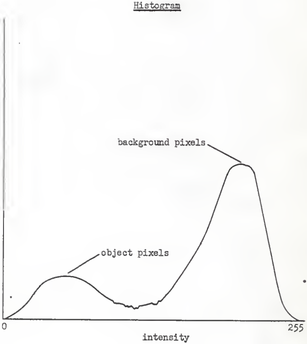


Figure 3. Image Histogram.

constant until a tally is found to be large enough to be considered on a peak (see Figure 4). From there, scanned tallies are compared with a maximum valley constant until a tally is found to be small enough to be considered in a valley. Scanning continues from this valley edge point as long as the next tally is smaller than its predecessor. When a low spot is encountered, the intensity at which it occurs is saved. The computer resumes scanning and searches for another peak tally. When a peak is found, the scanning direction is reversed and a second valley edge and low spot are found in succession. The midpoint between the two low spots is declared to be the threshold.

Object detection. Once the threshold has been selected, edge determination can begin. Recall that an edge pixel is a dark object pixel which neighbors a light background pixel. The edge contour is found by grouping neighboring edge pixels together. This grouping is done by the edge-tracking algorithm described in the following paragraph. But, in order to begin tracking an edge, the edge-tracking algorithm must be given a root edge pixel to grow from. The root could be found by scanning through the array pixel by pixel, searching for an edge pixel, but that would be very slow because of the large number of comparisons involved. A pattern-bombing, scanning procedure works much faster. Image array pixels whose indexes are multiples of 25 are tested sequentially until a dark object pixel is found. Then, the scanning direction is reversed and done pixel by pixel until a background pixel is found. The previous

Histogram

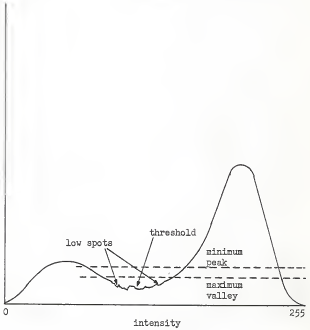
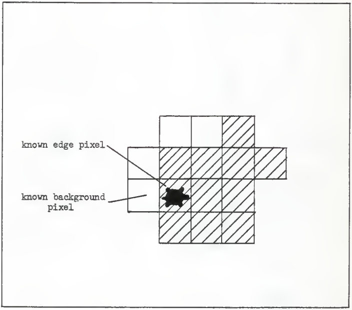


Figure 4. Automatic Threshold Detection with a Bimodal Histogram.

object pixel is passed to the edge-tracking algorithm to be used as the root edge pixel.

Edge tracking. The edge-tracking algorithm connects edge pixels together as if a turtle were traversing the image array. First, the far outside edge pixels of the image array are set to background values so the turtle cannot wander off of the image. The turtle begins its journey on the known object edge pixel with its tail to the neighboring background pixel as in Figure 5. It looks back over its left shoulder and scans from left to right looking for an adjoining object pixel. When it finds one, it turns toward it and steps forward to the new edge pixel. The turtle then repeats the scan from its new orientation and location until its movement returns it to the original, root pixel. The edge contour is stored in a one-dimensional array as a list of the directions that the turtle moves when it advances from one pixel to the next. This list is called a boundary chain code and is a very compact method of storing edge contours.[2]

Determining and marking interior pixels. Now that an edge has been found, it is important for later processing to determine which of the image array pixels are interior to this particular edge. To do this, a logical array of the same dimensions as the image array is used. This array indicates interior membership. A rectangular portion of this array, extending from the minimum to the maximum x and y indexes of the edge contour traced by the turtle, is initialized to values of false, meaning not interior. Next, the edge contour is traversed and, with some methodological



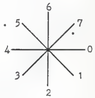


Figure 5. Initial Turtle Position in Edge Tracking.

toggling, the elements of the logical array corresponding to interior pixels of the edge contour are set to true while those corresponding to exterior pixels are left in a resultant state of false. The toggling rules used during traversal are as follows:

1. If the direction of motion to the next pixel is a downward motion, then all of the elements to the right of and in the row of the pixel being moved from are toggled.

2. If the direction of motion to the next pixel is an upward motion, then all of the elements to the right of and in the row of the pixel being moved to are toggled.

3. Otherwise, no toggling is performed for this motion.

Once the contour has been entirely traversed and all toggling is complete, the edge is traversed once more and all of the edge pixels are set true to indicate that they are also a part of the object. Note that only the pixels interior to this specific edge contour have values of true in the interior-membership array; pixels interior to other objects have values of false along with the background pixels.

Ensuring externality of the contour. It is possible that the edge-tracking procedure could have traced around a light area (due to a hole in the kernel or noise in the image) inside the wheat kernel, which would result in the light area being enclosed by the edge contour rather than the wheat kernel (see Figure 6). This is easily tested by checking to see if the background pixel adjacent to the root edge pixel is interior or exterior to the edge by using the interior-membership array. If the background pixel is interior to the edge, then the contour encloses a hole,

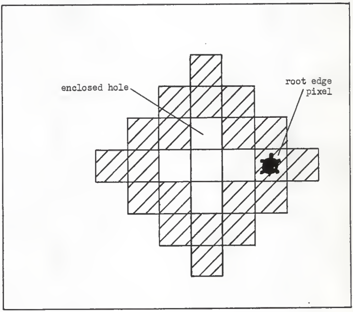


Figure 6. Edge Tracking around a Hole.

not a kernel. To correct this, searching is continued from the leftmost pixel of the edge contour back to the left for another root edge pixel to send to the edge-tracking algorithm for another try.

Contour rejection. If an edge contour touches the edge of the image, is longer than the boundary chain code array, or is shorter than 175 pixels, it is discarded. Otherwise, the centroid is calculated. (If the centroid turns out to be exterior to the edge, then the contour is also discarded. This sometimes happens when two or more kernels touch and overlap each other on the light table and all of them are traced around as one edge.) The outline of the kernel that has been found by the computer is traced in green on the display screen and crosshairs are centered over the kernel's centroid. Then, the operator decides whether or not the boundary will be stored. If so, the boundary chain code is written to disk. Next, the points interior to this edge that would be hit by the pattern-bombing procedure are set to background values in the image array so that this kernel will not be considered again. Searching continues for other kernels in the image from the point where the pattern-bombing procedure left off when it found this latest kernel. When all of the kernels in an image have been found and shown to the operator, the operator may take another image.

Kernel Measurement

Now that the edge contours have been found, the kernels can be measured. Recall that it was decided that radial measurements would be taken of both top and side views, in half-degree angular increments, from the centroid to the kernel's edge. This section describes how these measurements are taken.

Centroid calculation. The centroid, or center of gravity, of the wheat kernel is found using moments. The density function, $f(x,y)$, of the silhouette is considered to have a value of one inside the edge and zero outside. The centroid is located at the point

$$\bar{x} = \frac{M_{10}}{M_{00}}, \qquad \bar{y} = \frac{M_{01}}{M_{00}} \tag{1}$$

where

$$M_{jk} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^j\, y^k\, f(x,y)\, dx\, dy \tag{2}$$

is the moment of order $j + k$. The zero-order moment is the area of the silhouette.[2]

Determination of the principal axes. The principal axes of the wheat kernel are found using central moments. Central moments are found in the same manner as regular moments, except the centroid is used as the origin. The expression for a central moment is

$$u_{jk} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} (x-\bar{x})^j\, (y-\bar{y})^k\, f(x,y)\, dx\, dy \tag{3}$$

The angle, w, that the principal axes differ in rotation from the present axes may be solved for from equation 4.[2]

$$\tan 2w = \frac{2\,u_{11}}{u_{20} - u_{02}} \tag{4}$$

Indication of germ end of kernel. Before the radial measurements are begun, the operator is asked to indicate the germ end of the kernel with a joystick. Crosshairs are positioned over the centroid of an image of the kernel on the display screen. The operator then moves the crosshairs toward the germ end of the kernel. (Although the major axis of the kernel has already been calculated, the computer does not know which end of the kernel is the brush end and which is the germ end.)

Taking radial measurements. The radial measurements are begun at the germ end of the kernel. A vector is followed outward from the centroid in the internal-membership array until a background pixel is found. Then, the length of the vector is calculated and saved. This is repeated for all 720 measurements in the view.

Normalization methods. Absolute as well as normalized measurements are used in the statistic and classifier programs. The normalization techniques used are length- and width-normalization. For length-normalization, the radial measurements are divided by the sum of the 0 degree and the 180 degree angular measurements; for width-normalization, the radial measurements are divided by the sum of the 90 degree and the 270 degree angular radii. The absolute radial measurements are in units of pixels while the normalized measurements are, of course, simple ratios.

Classifiers

Minimum-distance and nearest-neighbor classifiers. Two types of classifiers are used on the wheat data: a minimum-distance classifier[1] and a nearest-neighbor classifier. The minimum-distance classifier compares a measurement from the unknown kernel against the corresponding mean measurement for each of the varieties known to it. The unknown kernel is assumed to be of the same variety as the variety with the closest mean measurement. The nearest-neighbor classifier keeps, on disk, a set of measurements of sample kernels of known varieties. It then compares a measurement of the unknown kernel against the corresponding measurement of each of the sample kernels in its known set. It assigns the unknown kernel to the variety of the sample kernel whose corresponding measurement is closest.

Elective classifier. A third, elective, classifier is also used, but it can be considered a modification of the minimum-distance classifier. It is run six times for each unknown kernel with the results being put to a vote. The six times are for each of the possible combinations of two varieties out of the four. If a single variety wins each of its comparisons against each of the other three varieties, then the voting will come out with a perfect three for that variety. If there is not a unanimous decision, then there will at least be one variety that gets fewer votes than the others and a choice can be made among the leaders in the voting. The unknown kernel is considered to belong to the variety that wins the voting.

Training and testing sets. All classifiers are trained using 100 kernels of each of the four varieties: measurements are written to disk for the nearest-neighbor classifier and mean measurements are calculated for the minimum-distance classifier. The classifiers are also tested using these same kernels. During the testing of the nearest-neighbor classifier, the unknown kernel is taken out of its known set in an odd man out fashion so that there will not be an exact match.

Use of multiple dimensions. More than one measurement is often required to distinguish one variety of wheat from another. Each of these discriminating measurements is used as a coordinate in an n-dimensional space. To determine the distance between two points in the n-space, the normal Euclidean definition of distance is used: the square root of the sum of the squares of each coordinate difference.

Exhaustive discriminator selection. Selection of the measurements to be used as discriminators in the final classifiers is done exhaustively. Each possible measurement (all angles and all normalization methods) is added, in turn, to the current set of classifier dimensions and all of the training kernels classified using the resulting set of dimensions. The additional measurement that gives the best classification rate is then added to the list of discriminators that make up the classifier dimension set. In this way, the classifier dimension is incremented from zero (searching for the best first dimension) until the addition of a new dimension does not increase the rate of classification.

Chapter 3

PROBLEMS AND SOLUTIONS

Several problems came about during the course of this research. They, and their solutions, are described below.

The first group of problems involved the image arrays produced by the digitizing camera. The camera filled the rightmost elements of the image array with erroneous values and scattered salt-and-pepper noise (white and black pixels scattered over the image) throughout the image array. Simply having the programs ignore the rightmost columns of pixels turned out to be easier than getting the camera fixed and was quicker too. The salt-and-pepper noise occurred in a regular pattern throughout the image array and was attributed to poor shielding. Mean and approximate-mean filters were programmed and applied to the image array to rid it of the noise before other processing, but this preprocessing slowed down the whole procedure unbearably. It ended up that the noise was easily removed by coiling the extra cable between the camera and the digitizing circuitry so that the preprocessing filter was not necessary.

Another problem had to do with obtaining the side view of each kernel. The camera is positioned over the light table in a rigid frame and is not easily moved to a side position. Changing the camera position for each kernel was unthinkable. The solution was to tape down the loose glass on the light table and then to tip the entire camera and light table assembly on its side. A small piece of plexiglass was taped to the glass of the light table and its unpolished edge served as a nonreflective shelf for the wheat kernel. The plexiglass also allowed light to pass through it so that another edge-tracking program was not required that would ignore the support. Small wooden rockers were screwed to the side of two of the legs of the assembly to facilitate easy movement between orientations (shown in Figure 7). Also, the data-gathering programs were written so that top views of several kernels could be taken before their side views were taken. This allowed the operator to line the kernels up on a table rather than madly see-sawing the apparatus for each kernel.

The third set of problems had to do with spurious program crashing and loss of data. The first of these was with the input routines of FORTRAN -- they abort if the operator's input does not match the data type of the input variable. This was compounded by bad keyboards that unpredictably inserted extraneous control characters and occasionally misinterpreted some keys, causing total garbage to be input in some cases. This problem was tolerated by copying special input routines from another program that are tolerant of data mismatches and by error-checking all input. (In this case, the dummy-proofing is for dumb equipment and not for untrained operators.) The second of these problems was that an occasional zap of static charge would kill executing programs. So that the amount of data lost in such an instance would be minimal, the operator stopped and resumed the input program at regular intervals so that all of the

I/O memory buffers would be written out to disk. The operator was also careful not to touch the display screen because of the static charge on it.

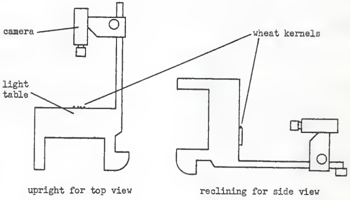


Figure 7. Camera Orientations for Top and Side Views.

Chapter 4

RESULTS

Kernel Measurement Statistics

Statistics for the radial measurements were calculated and plotted. These plots appear in Appendix A. These include mean, variance, minimum and maximum, mean plus-and-minus one standard deviation, and sorted variance plots for each variety. All of these are plotted versus measurement angle, except for sorted variance. The mean curve shows what the measurements are for an average kernel of each variety. The variance plot shows which measurements are the most reliable, or constant, for each variety sample. The minimum and maximum as well as the mean plus-and-minus one standard deviation plots give a general impression of what the distribution of the measurements are for the sample. True distributions of specific measurements were also plotted as necessary.

These statistical plots give an idea of which measurements and normalization techniques are good discriminators for classifying the varieties: ones which show high separation between mean measurements of different varieties while having small variance among the same variety. The best multiple-dimension discriminators, however, are chosen by the exhaustive classifier programs.

Classifiers

Selected single-dimension classification plots are included in Appendix B for the minimum-distance classifier. These plots show the classification rates for each of the normalization methods and each of the angles of measurement for varietal as well as textural classification. Also included in Appendix B are tables showing the best dimensions and their rates of classification for the minimum-distance and the nearest-neighbor classifiers of increasing dimensions.

Minimum-distance. The minimum-distance classifier classified with rates varying from 68.50% for one dimension to 80.75% for nine dimensions when classifying kernels by variety. Classification by texture varied from 84.50% for one dimension to 94.75% for eight dimensions.

Nearest-neighbor. The nearest-neighbor classification program was halted after several days of calculation when its classification rates began to level off. Its classification rates varied from 64.75% for a single dimension to 82.00% for three dimensions when classifying by variety. Classification by texture was not attempted due to the length of program execution time.

Single-dimension elective. Several modifications of the minimum-distance classifier were tried that used only a single dimension. Each of these compared a measurement against two of the mean measurements out of the four. The first method cast a whole vote for the nearest mean in each of the six cases and selected the variety with three votes as the winner. If none of the varieties got three votes, then the first of the varieties with two votes was selected. This method resulted in an 86.00% classification rate, compared with the 68.50% rate given above. 4.25% was due to guesses among the two-vote varieties. The second method gave percentage votes equal to the fraction of the distance away from each of the means. Therefore, if a measurement was one-quarter of the distance from mean 1 to mean 2, then mean 2 would get a vote of 0.25 and mean 1 would get a vote of 0.75. If the measurement was greater than or less than both of the means, then the closest mean would get a whole vote. This method produced a classification rate of 84.00%. A third method, similiar to the second, gave a classification rate of 83.00%. This method gave a percentage vote only to the nearest mean and was based on the distance from the midpoint between the means. A measurement that occurred at the midpoint gave a vote of zero, while one that occurred 90% of the way from the midpoint to the mean gave a vote of 0.90. These methods improved on the single dimension classification rate, so similiar modifications were tried for multiple dimensions.

Multiple-dimension elective. In this third, multiple-dimension classifier, the best set of minimum-distance discriminators between two varieties were used, with the maximum number of dimensions. For example, when testing a kernel between the Arkan and the Arthur means, a four-dimensional space was used that gave a 97.5% classification rate when tested with the 100

Arkan and 100 Arthur sample kernels. (These dimensions are given in the tables in Appendix B.) The three schemes varied in their method of voting.

The first method connects the two mean points in the n-dimensional space with a line. A hyperplane, perpendicular to this line and passing through the unknown kernel's point in the n-space is set up. The percentage of the distance along this line that the hyperplane cuts it determines the vote that each variety gets in the same manner as in the second, single-dimension voting classifier above. A two-dimensional case is shown in Figure 8. (Figuring this proportion is not nearly as difficult as it sounds since it only involves calculating an n-dimensional dot product between a vector from mean 1 to mean 2 and a vector from mean 1 to the unknown kernel's point.) The classification rate for this method was 92.00%.

The second method uses equipotential curves to give partial votes to both varieties. This involves calculating the distances between mean 1 and the unknown point and mean 2 and the unknown point. The fraction described by the distance from mean 1 to the unknown point divided by the sum of the two distances is given to variety 2 while the distance from mean 2 to the unknown point divided by the same sum is given as a vote to variety 1. This method resulted in a classification rate of 86.00%.

The third method involves giving the whole vote to the nearest of the two means. If classification occurs only when there are three votes, the classification rate is 92.75%. If, however, an educated guess is made among the varieties based on

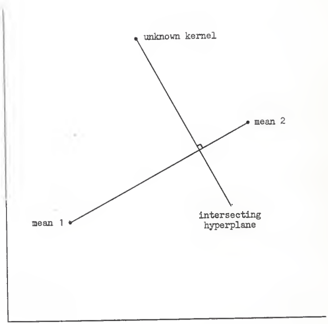


Figure 8. Hyperplane Intersection with Line, 2-D Case.

the number of times that that sequence of votes has come up for each variety, the classification rate increases to 94.00%. If classification is done by texture rather than variety, then the rate is 96.25% with educated guessing. This method gives the best classification rates of all the methods for this four-variety test case.

As noted before, the test case for all of these classifiers was the same set of measurements used to train them. Therefore, if a different sample of wheat were to be classified with one of these classifiers, the classification rate is expected to decrease.

Chapter 5

CONCLUSIONS

This paper has described a machine-vision system designed to classify wheat on a kernel-by-kernel basis. The system measures a set of known kernels and tries to find a set of those measurements that can be used to discriminate between the varieties.

Several classification schemes were used: a minimum-distance, a nearest-neighbor, and an elective classifier. The elective classifier did not attempt to find a single set of measurements to discriminate between all varieties as the others did, but chose sets of features to discriminate between each of the possible pairs of varieties. A tally was kept of the winners of each pair's comparison and the kernel was classified as the variety that won the most comparisons. This classifier gave the best results of any of them with a varietal classification rate of 94% correct. But, this percentage was for a single test set which was the same set that the classifier was trained with, so the percentage is not representative of what could be expected in all cases.

Regardless of the accuracy of the classification rate, it would seem that the machine-vision system is promising as a classification tool for use in wheat grading. Therefore, a few suggestions are given below for possible directions in future research.

Additional Features

There are many additional features that can be measured from a wheat kernel. First, additional measurements that could be made from the silhouette images include a series of vertical and horizontal measurements; area; fit of the contour to a rectangle, circle, ellipse or mean contour[3,4]; and comparison of abstract features that humans use, like pointedness or angle of germ, roundedness of the kernel's back, and so on. Second, different methods of analyzing edge data could be used, such as taking the Cosine or Fourier Transforms of the boundary chain code as described by Castleman.[2] Third, three-dimensional comparisons of crease width, crease depth, cheek curvature, slope of the kernel's back, and wrinkling of the outsize layer could be done by building up a representation of a kernel from multiple silhouette views, stereoscopic views, or by analyzing the reflection from the kernel's surface when in the presence of a point light source.[2,3]

More Efficient Methods of Discriminator Selection

More efficient means of discriminator selection should be used. Exhaustive methods require huge amounts of calculation. Castleman talks about feature variance and correlation and how these can be used to determine class separation and for dimension reduction. Features with high variance within a variety should not be considered as possible discriminators. Features that are correlated with other features are redundant and only one of them should be kept. Castleman gives equations for measuring these parameters.[2]

More Representative Test Sets

Rather than testing a classifier on the sample that was used to train it, a classifier should be tested with a different sample. Classifier training should also be done with samples of the same variety of wheat from different portions of the state or country and, if possible, grown in different climates or years.

REFERENCES

1. Ahmed, Nasir, and Kamisetty Ramamohan Rao. Orthogonal Transforms for Digital Signal Processing. New York, New York: Springer-Verlag, 1975, pp. 225-232.

2. Castleman, Kenneth R. Digital Image Processing. Signal Processing Series, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979, pp. 305, 316-317, 327, 329, 332-333, 347-377.

3. Cohen, Paul R., and Edward A. Feigenbaum, ed. The Handbook of Artificial Intelligence. vol 3. Los Altos, California: William Kaufmann, Inc., 1982, pp. 220-224, 230-237, 260-278.

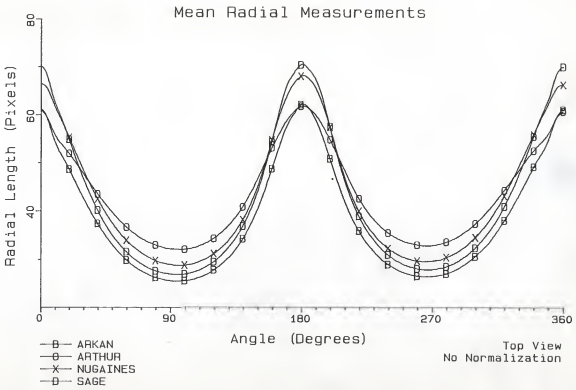
4. Hall, Ernest L. Computer Image Processing and Recognition. Computer Science and Applied Mathematics, New York, New York: Academic Press, 1979, p. 414.

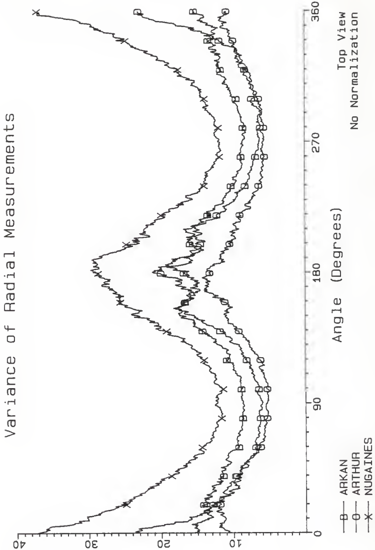
5. Martin, John H. "Kernel Classification of United States Wheat Varieties," Thesis, 1922, pp. 2-4.

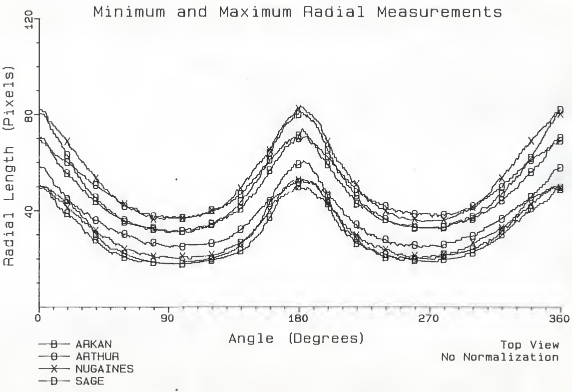
6. U.S. Department of Agriculture. Grain Grading Primer. Washington, D.C.: Government Printing Office, 1957, pp. 3, 10, 14-15.

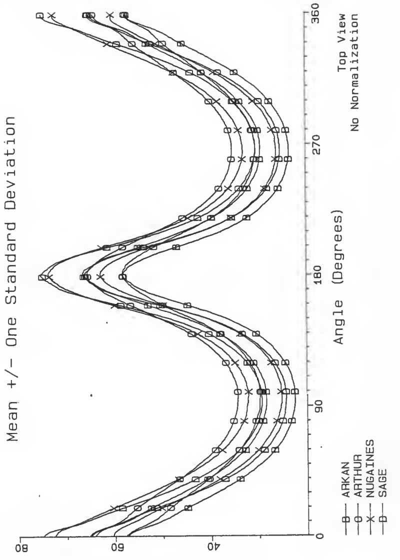
APPENDIX A

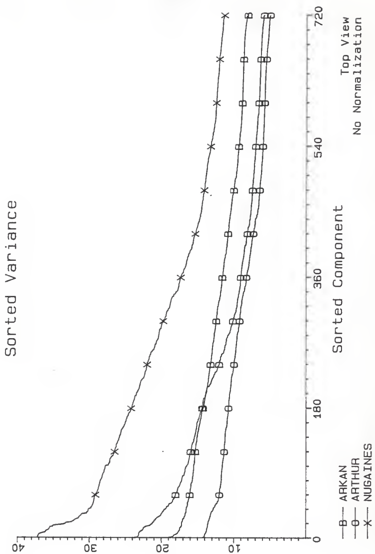
STATISTICAL PLOTS

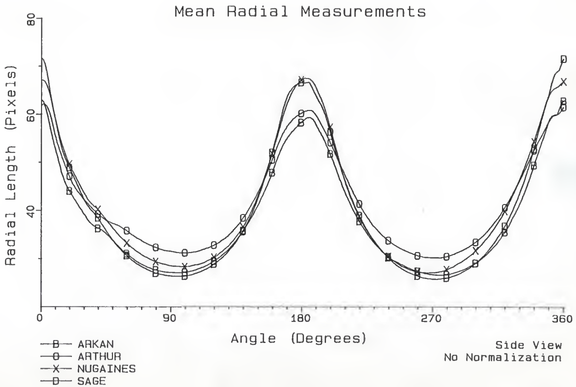

Mean Radial Measurements
Radial Length (Pixels)
80
60
40
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
No Normalization

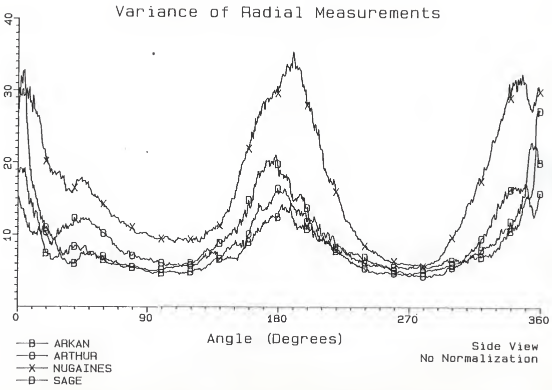

Variance of Radial Measurements
40
30
20
10
0
90
180
270
360
Angle (Degrees)
Top View
No Normalization
ARKAN
ARTHUR
NUGAINES
SAGE


Minimum and Maximum Radial Measurements
120
80
40
Radial Length (Pixels)
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
No Normalization

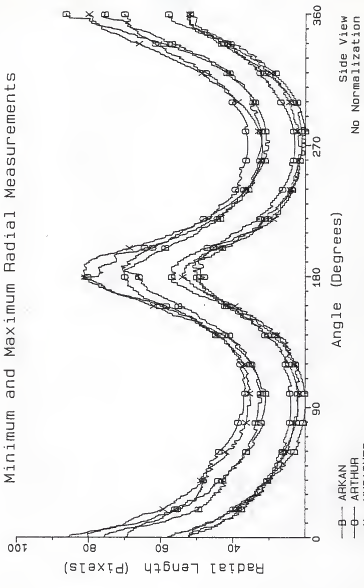

Mean +/- One Standard Deviation
Radial Length (Pixels)
80
60
40
0
90
180
270
360
Angle (Degrees)
Top View
No Normalization
ARKAN
ARTHUR
NUGAINES
SAGE

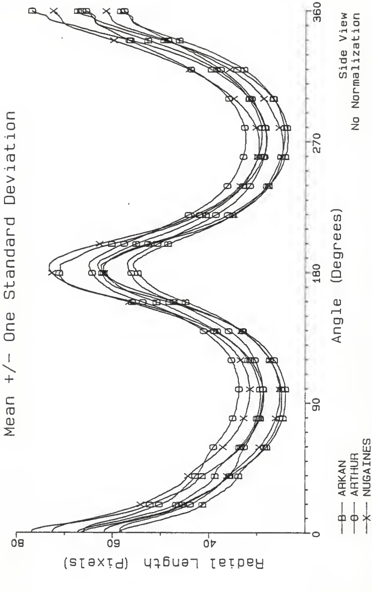

Sorted Variance
Sorted Component
Top View
No Normalization
ARKAN
ARTHUR
NUGAINES
SAGE
0
180
360
540
720
10
20
30
40


Mean Radial Measurements
Radial Length (Pixels)
80
60
40
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
No Normalization

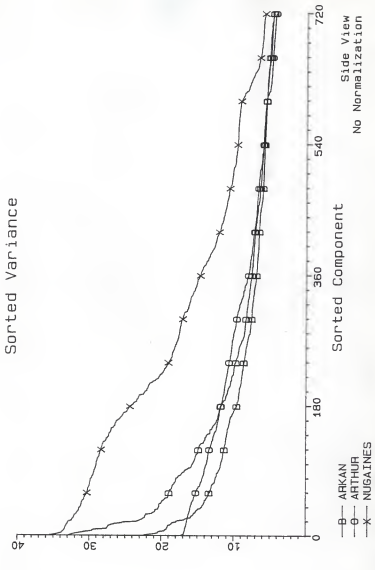


Variance of Radial Measurements
40
30
20
10
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
No Normalization

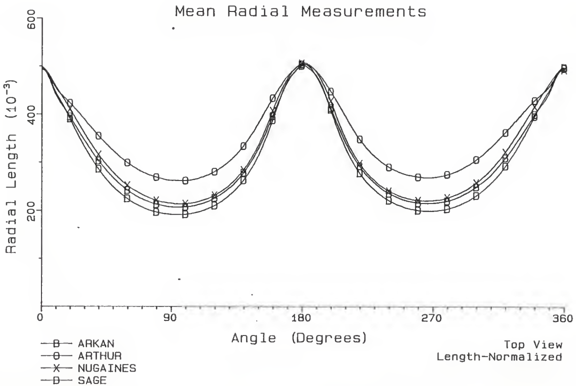

Minimum and Maximum Radial Measurements
Radial Length (Pixels)
100
80
60
40
0
90
180
270
360
Angle (Degrees)
Side View
No Normalization
ARKAN
ARTHUR
NUGAINES
SAGE

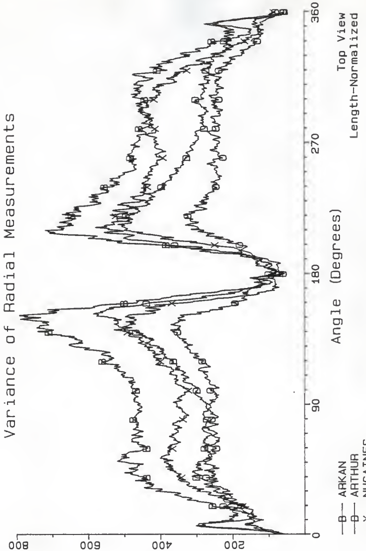


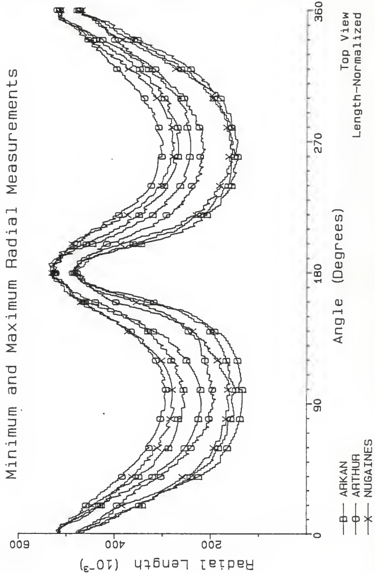
Mean +/- One Standard Deviation
Radial Length (Pixels)
80
60
40
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
No Normalization

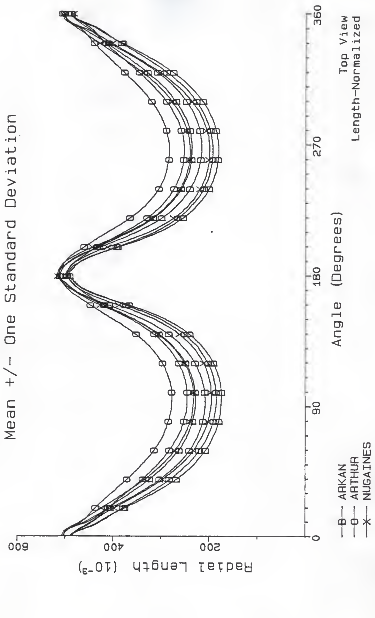


Sorted Variance
40
30
20
10
0
180
360
540
720
Sorted Component
Side View
No Normalization
ARKAN
ARTHUR
NUGAINES
SAGE

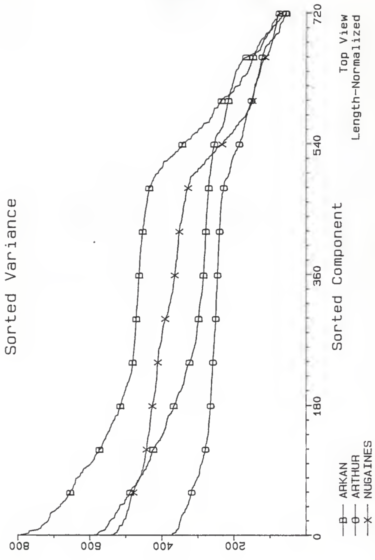

Mean Radial Measurements
Radial Length (10^{-3})
600
400
200
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
Length-Normalized

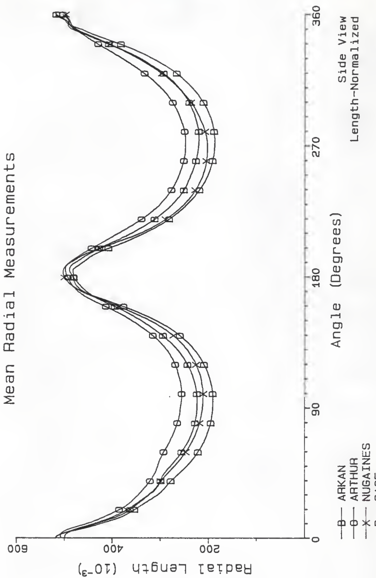

Variance of Radial Measurements
(10^-6)
800
600
400
200
0
90
180
270
360
Angle (Degrees)
Top View
Length-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

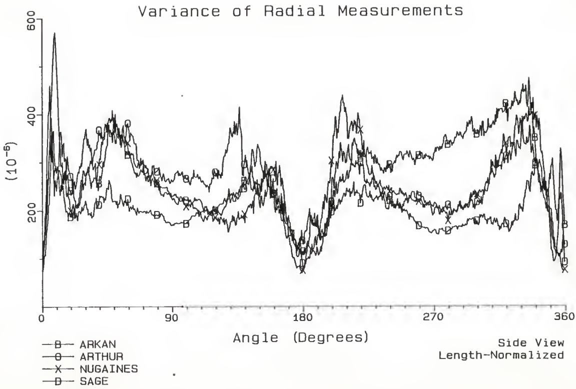

Minimum and Maximum Radial Measurements
Radial Length (10^{-3})
600
400
200
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
Length-Normalized



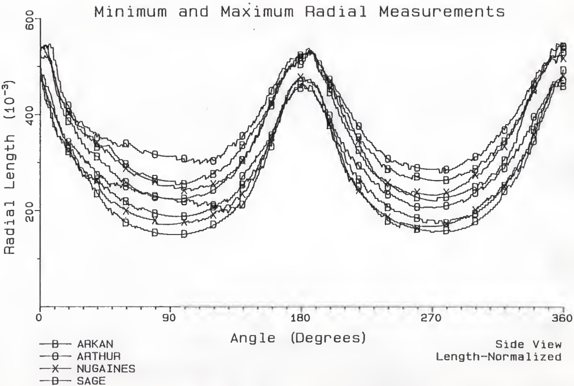
Mean +/- One Standard Deviation
Radial Length (10^-3)
0
200
400
600
0
90
180
270
360
Angle (Degrees)
Top View
Length-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

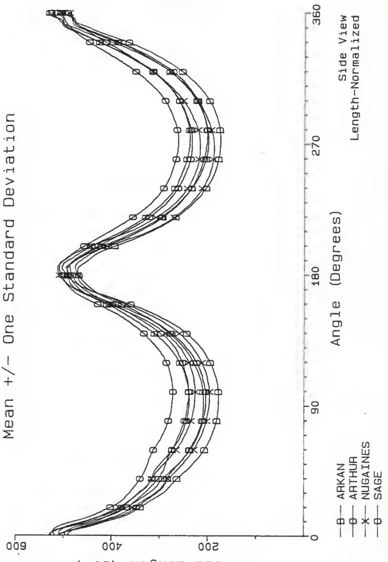

Sorted Variance
800
600
400
200
0
(10^-6)
0
180
360
540
720
Sorted Component
Top View
Length-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

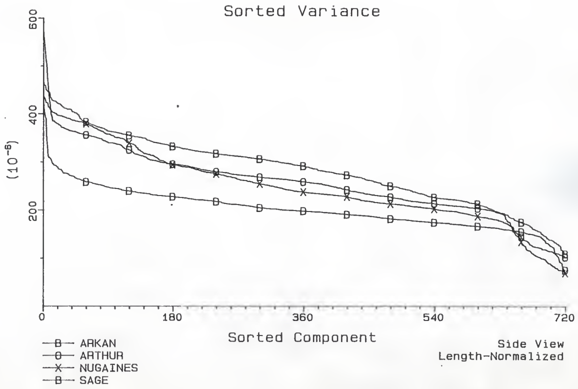

Mean Radial Measurements
Side View
Length-Normalized
Angle (Degrees)
Radial Length (10^{-3})
0
90
180
270
360
200
400
600
ARKAN
ARTHUR
NUGAINES
SAGE

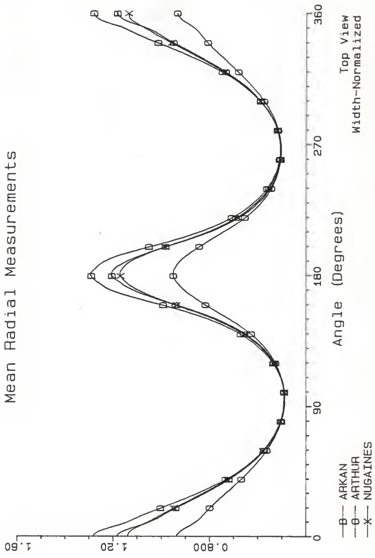


Variance of Radial Measurements
600
400
200
(10^-6)
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Length-Normalized

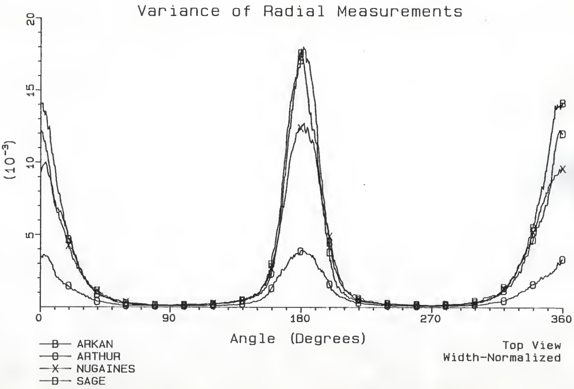

Minimum and Maximum Radial Measurements
Radial Length (10^{-3})
600
400
200
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Length-Normalized

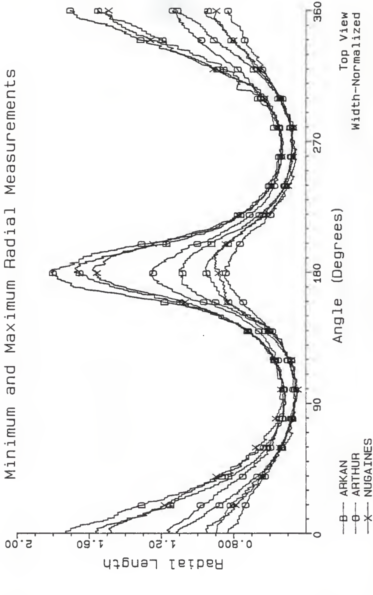

Mean +/- One Standard Deviation
Radial Length (10^-3)
600
400
200
0
90
180
270
360
Angle (Degrees)
Side View
Length-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

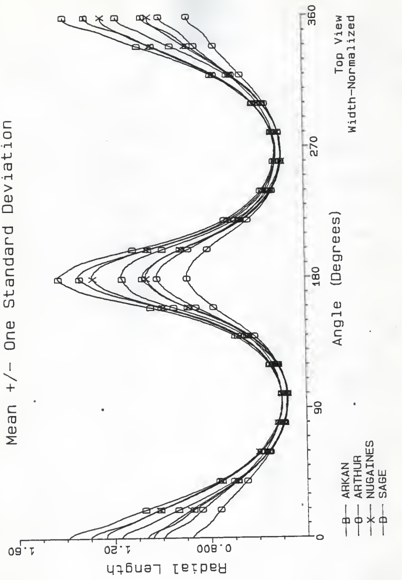

Sorted Variance
600
400
200
(10^-6)
0
180
360
540
720
Sorted Component
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Length-Normalized



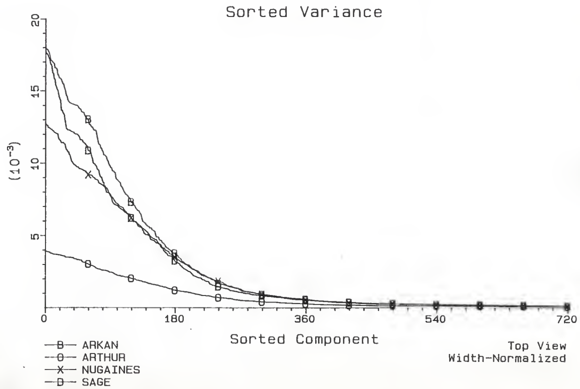
Mean Radial Measurements
Radial Length
1.60
1.20
0.800
0
90
180
270
360
Angle (Degrees)
Top View
Width-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

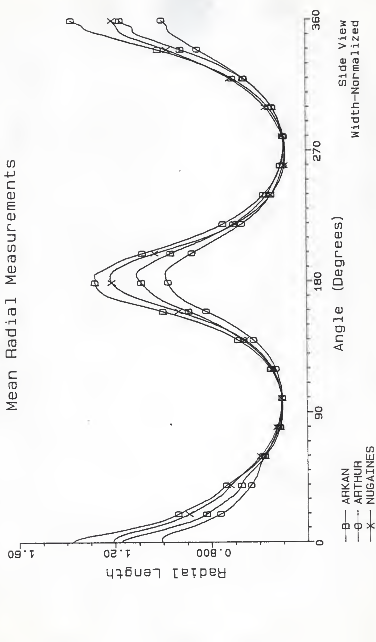

Variance of Radial Measurements
(10^-3)
0
5
10
15
20
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
Width-Normalized

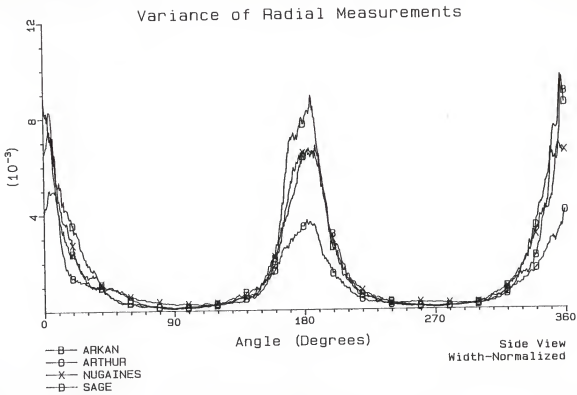

Minimum and Maximum Radial Measurements
Radial Length
2.00
1.60
1.20
0.800
0
90
180
270
360
Angle (Degrees)
Top View
Width-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

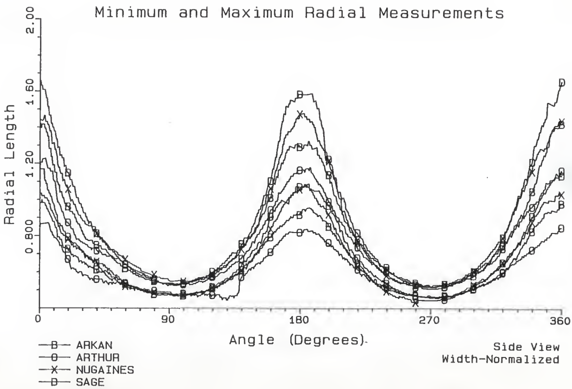


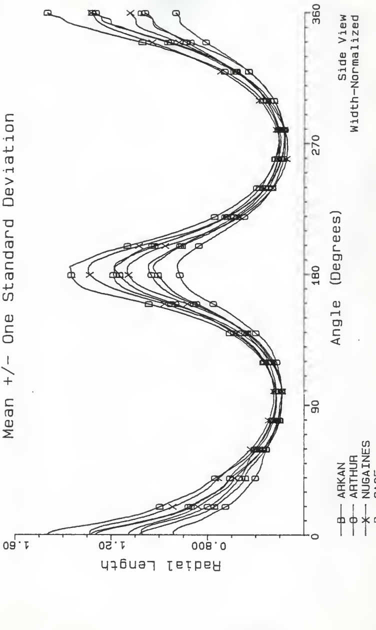
Mean +/- One Standard Deviation
Radial Length
0.800
1.20
1.60
0
90
180
270
360
Angle (Degrees)
Top View
Width-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE

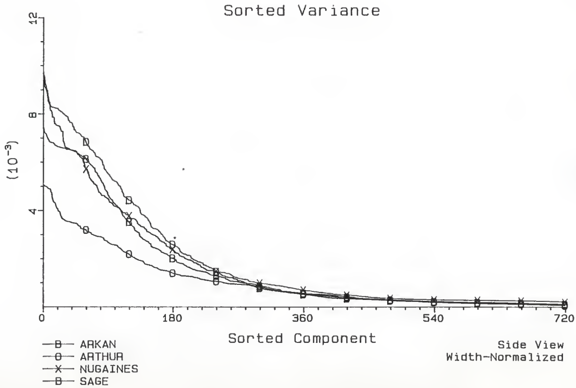

Sorted Variance
(10^{-3})
20
15
10
5
0
180
360
540
720
Sorted Component
ARKAN
ARTHUR
NUGAINES
SAGE
Top View
Width-Normalized


Mean Radial Measurements
Radial Length
1.60
1.20
0.800
0
90
180
270
360
Angle (Degrees)
Side View
Width-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE


Variance of Radial Measurements
12
8
4
(10^-3)
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Width-Normalized


Minimum and Maximum Radial Measurements
Radial Length
2.00
1.60
1.20
0.800
0
90
180
270
360
Angle (Degrees)
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Width-Normalized


Mean +/- One Standard Deviation
Radial Length
1.60
1.20
0.800
0
90
180
270
360
Angle (Degrees)
Side View
Width-Normalized
ARKAN
ARTHUR
NUGAINES
SAGE


Sorted Variance
12
8
4
(10^-3)
0
180
360
540
720
Sorted Component
ARKAN
ARTHUR
NUGAINES
SAGE
Side View
Width-Normalized

APPENDIX B

CLASSIFIER RESULTS (TABLES AND PLOTS)

FOUR-VARIETY MINIMUM-DISTANCE CLASSIFIER

Arkan vs. Arthur vs. Nugaines vs. Sage

Dimension	View	Normalization	Index	Percentage
1	Side	Width	337	68.50
2	Top	Width	6	76.25
3	Side	Width	350	77.25
4	Side	Width	429	78.00
5	Side	Width	426	78.75
6	Side	Width	432	79.25
7	Side	Width	437	79.75
8	Side	Width	497	80.50
9	Side	Width	436	80.75

Hard vs. Soft

Dimension	View	Normalization	Index	Percentage
1	Top	Length	661	84.50
2	Side	Width	500	89.00
3	Side	Width	577	92.25
4	Side	Length	568	93.25
5	Top	Width	460	93.75
6	Side	Length	3	94.00
7	Top	Length	378	94.50
8	Top	Width	234	94.75

TWO-VARIETY NEAREST-NEIGHBOR CLASSIFIER

Arkan vs. Arthur vs. Nugaines vs. Sage

Dimension	View	Normalization	Index	Percentage
1	Side	Width	339	64.75
2	Side	None	536	79.50
3	Top	Length	352	82.00

TWO-VARIETY MINIMUM-DISTANCE CLASSIFIER

Arkan vs. Arthur

Dimension	View	Normalization	Index	Percentage
1	Top	Length	144	95.5
2	Side	Width	65	96.5
3	Top	Width	111	97.0
4	Top	Length	380	97.5

Arkan vs. Nugaines

Dimension	View	Normalization	Index	Percentage
1	Side	Length	720	87.0
2	Side	Length	276	91.0
3	Top	Length	695	97.0
4	Top	Width	434	98.0
5	Side	Width	199	98.5

Arkan vs. Sage

Dimension	View	Normalization	Index	Percentage
1	Side	Width	344	89.5
2	Side	None	474	93.0
3	Side	Length	154	93.5
4	Side	Length	650	94.0

Arthur vs. Nugaines

Dimension	View	Normalization	Index	Percentage
1	Side	Width	697	97.0
2	Top	Width	401	99.0

Arthur vs. Sage

Dimension	View	Normalization	Index	Percentage
1	Side	Width	713	100.0

Nugaines vs. Sage

Dimension	View	Normalization	Index	Percentage
1	Side	Length	379	90.0
2	Side	Width	235	93.0
3	Side	Width	193	94.5
4	Top	Width	95	95.0

FOUR-VARIETY ELECTIVE CLASSIFIER

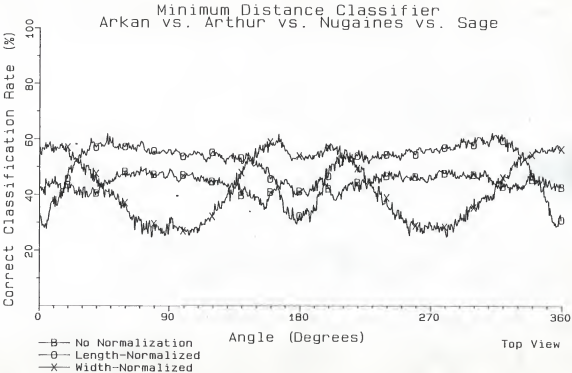
Single Dimension Comparisons

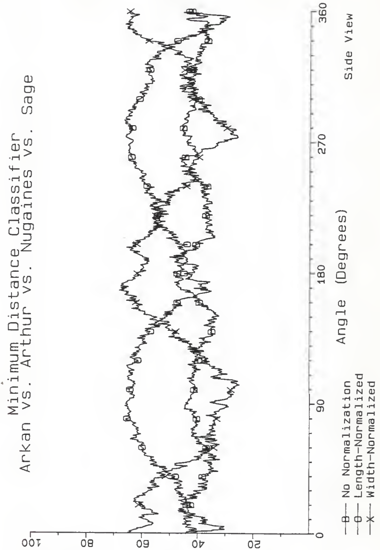
Method	No. Measurements	Percentage
1	6	86.00
2	6	84.00
3	6	83.00

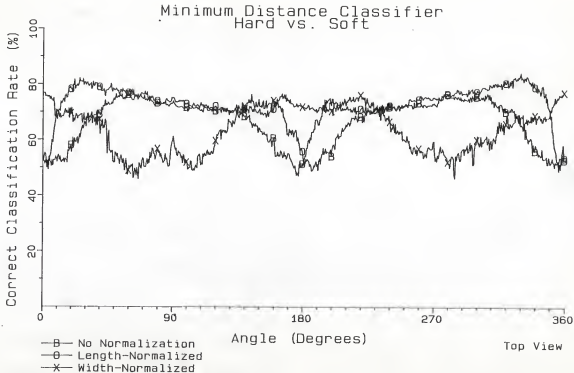
Multiple Dimension Comparisons

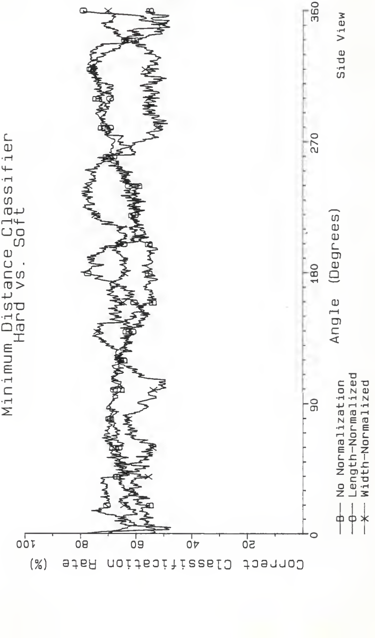
Method	No. Measurements	Percentage
1	20	92.00
2	20	86.00
3	20	92.75
4	20	94.00

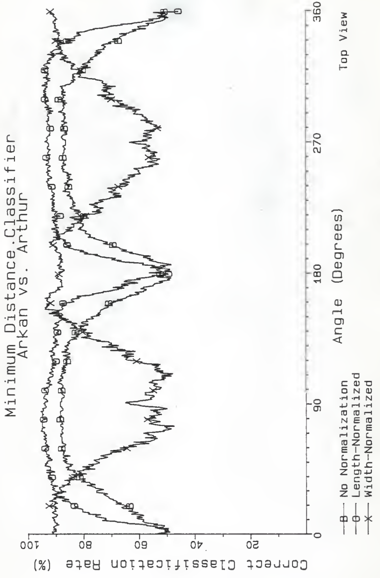
Multiple Dimension Method 4 used for Hard vs. Soft: 96.25%

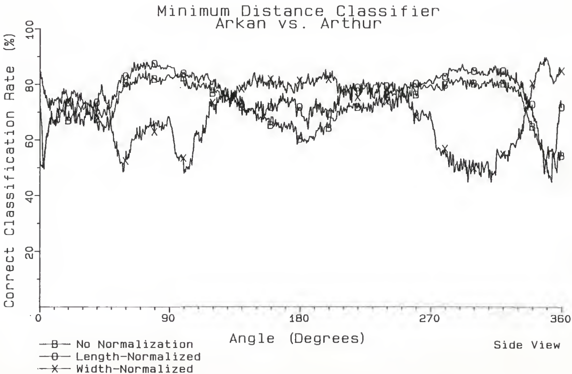

Minimum Distance Classifier
Arkan vs. Arthur vs. Nugaines vs. Sage
Correct Classification Rate (%)
100
80
60
40
20
0
90
180
270
360
Angle (Degrees)
B No Normalization
0 Length-Normalized
X Width-Normalized
Top View


Minimum Distance Classifier
Arkan vs. Arthur vs. Nugaines vs. Sage
Correct Classification Rate (%)
100
80
60
40
20
0
90
180
270
360
Angle (Degrees)
Side View
No Normalization
Length-Normalized
Width-Normalized


Minimum Distance Classifier
Hard vs. Soft
Correct Classification Rate (%)
100
80
60
40
20
0
90
180
270
360
Angle (Degrees)
No Normalization
Length-Normalized
Width-Normalized
Top View


Minimum Distance Classifier
Hard vs. Soft
Correct Classification Rate (%)
0
20
40
60
80
100
0
90
180
270
360
Angle (Degrees)
Side View
No Normalization
Length-Normalized
Width-Normalized


Minimum Distance.Classifier
Arkan vs. Arthur
Correct Classification Rate (%)
Angle (Degrees)
Top View
No Normalization
Length-Normalized
Width-Normalized
0
20
40
60
80
100
90
180
270
360


Minimum Distance Classifier
Arkan vs. Arthur
Correct Classification Rate (%)
100
80
60
40
20
0
90
180
270
360
Angle (Degrees)
B No Normalization
0 Length-Normalized
X Width-Normalized
Side View

APPENDIX C

COMPUTER PROGRAMS

```
C*********************************************************************
C
C       WHEAT
C
C       VAX-11 FORTRAN SOURCE FILENAME:          WHEAT.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                     PROGRAMMER(S)
C       --------        ----                     -------------
C       1.0             OCTOBER 24, 1984         TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985            TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN WHEAT
C
C       PURPOSE
C
C               This program captures and processes an image array,
C               locating and storing edge contours of wheat kernels.
C               Multiple contours may occur in the image.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               GRDDG  - Grinnell Systems Routine
C               GRDOP  - Grinnell Systems Routine
C               GRDSH  - Grinnell Systems Routine
C               GRDTH  - Grinnell Systems Routine
C               GRFAR  - Grinnell Systems Routine
C               GRSBFD - Grinnell Systems Routine
C               GRSEND - Grinnell Systems Routine
C               GRZCB  - Grinnell Systems Routine
C               GRZCL  - Grinnell Systems Routine
C               GRZCO  - Grinnell Systems Routine
C               GRZCR  - Grinnell Systems Routine
C               GRZCW  - Grinnell Systems Routine
C               IMDISP - KSU Image Display Routine
C               IMINIT - KSU Display Device Initialization Routine
C               PLAWT  - Histogram Plotting Routine
C               TTYINC - Ralph's Character Input Routine
C               TTYINI - Ralph's Integer Input Routine
C               TTYINY - Ralph's Yes/No Input Routine
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
```

```
C*********************************************************************
C
      IMPLICIT NONE

      INTEGER*2   BLACK, WHITE, THRESH
      PARAMETER   (BLACK=0, WHITE=255)
      INTEGER     AREA, CHAINX, CHAINY, CTHRSH, DELTA/25/, DIRECT,
     &            EDGCNT, EDGMAX, EDGMIN, ENTER, FUN1, FUN2,
     &            HIST(BLACK:WHITE), HRANGE, I, ICKX, ICKY, ICOLOR,
     &            ICUR, IERR, ISTAT, ITEMP, ITMP1, ITMP2, ITMP3,
     &            ITMP4, ITMP5, J, JCUR, K, KB, KL, L, LRANGE, LSUM,
     &            MAXLX, MAXLY, MINLX, MINLY, NANGLS, NBIN, NELEM,
     &            NKERN1, NKERN3, NLINE, OFSTX, OFSTY, RM, ROOTX,
     &            ROOTY, SCANX, SCANY, SUM, TESTX, THR1, THR2, UN1,
     &            UN3, XCENT, XLOC, XSPAN, XSUM, YCENT, YLOC, YSPAN,
     &            YSUM,
     &            XDELT(0:7)/1, 1, 0,-1,-1,-1,0,1/,
     &            YDELT(0:7)/0,-1,-1,-1, 0, 1,1,1/
      PARAMETER   (EDGMAX=2000, EDGMIN=175, NELEM=512, NLINE=512,
     &            RM=5, NANGLS=72)
      REAL        ANG, IX, IXY, IY, MAJANG, MINANG, PI,
     &            RADIAL(NANGLS), TIX, TIXY, TIY, X, Y, Y2
      PARAMETER   (PI=3.141592654)
      INTEGER*2   EDGCHN(EDGMAX), IMAGE(NELEM,NLINE),
     &            TIMAGE(NELEM,NLINE)
      LOGICAL*1   LIMAGE(NELEM,NLINE), SAVEDG, SAVSID, TPVIEW
      CHARACTER*1 DUMMY
      CHARACTER*24 AC1, FN1, FN3, STAT1

C     Set up the peak-detection thresholds
      NBIN = 5
      THR1 = NBIN*650
      THR2 = NBIN*550

C *** Open the data files (One for edge data, the other for
C     measurement data)
      UN1 = 54
      UN3 = 56
      SAVEDG = .FALSE.
      SAVSID = .FALSE.

      TYPE 1,' Welcome'

      CALL TTYINC('Enter file name for TOP edge data:  ',
     &            FN1,IERR,0,' ')
      IF (IERR.NE.0) FN1 = ' '
      IF (FN1.NE.' ') SAVEDG = .TRUE.

      CALL TTYINC('Enter file name for SIDE edge data:  ',
     &            FN3,IERR,0,' ')
      IF (IERR.NE.0) FN3 = ' '
      IF (FN3.NE.' ') SAVSID = .TRUE.

      IF (SAVEDG .OR. SAVSID) THEN
        CALL TTYINY('Append data? (Y/<N>):  ',
     &            ITMP1,IERR,0,'NO')
```

```
          DUMMY = 'N'
          IF (IERR.EQ.0 .AND. ITMP1.EQ.1) DUMMY = 'Y'

          AC1   = 'SEQUENTIAL'
          IF (DUMMY.EQ.'Y' .OR. DUMMY.EQ.'y') THEN
            STAT1 = 'UNKNOWN'
          ELSE
            STAT1 = 'NEW'
          ENDIF
        ENDIF

        IF (SAVEDG) OPEN(UN1,FILE=FN1,ERR=9999,STATUS=STAT1,ACCESS=AC1,
     &                   FORM='UNFORMATTED')

        IF (SAVSID) OPEN(UN3,FILE=FN3,ERR=9999,STATUS=STAT1,ACCESS=AC1,
     &                   FORM='UNFORMATTED')

        NKERN1 = 0
        NKERN3 = 0
        IF (SAVEDG .OR. SAVSID) THEN
          IF (DUMMY.EQ.'Y' .OR. DUMMY.EQ.'y') THEN

C ---         Count the number of kernels stored already
            IF (SAVEDG) THEN
5049          READ (UN1,END=5050) ITMP1,ITMP2,ITMP3,ITMP4,
     &                            ITMP5,(EDGCHN(I),I=1,ITMP5)
              NKERN1 = NKERN1 + 1
              GOTO 5049
            ENDIF
5050        CONTINUE

            IF (SAVSID) THEN
5053          READ (UN3,END=5054) ITMP1,ITMP2,ITMP3,ITMP4,
     &                            ITMP5,(EDGCHN(I),I=1,ITMP5)
              NKERN3 = NKERN3 + 1
              GOTO 5053
            ENDIF
5054        CONTINUE

            IF (SAVEDG) WRITE (*,*) NKERN1,' kernels in ',FN1
            IF (SAVSID) WRITE (*,*) NKERN3,' kernels in ',FN3

            ITMP1 = 0
            IF (SAVEDG) ITMP1 = NKERN1
            IF (SAVSID) ITMP1 = NKERN3

            IF ((SAVEDG .AND. NKERN1.NE.ITMP1) .OR.
     &          (SAVSID .AND. NKERN3.NE.ITMP1)) THEN
              WRITE (*,*) ' '
              WRITE (*,*) 'Warning.  Number of kernels in these '//
     &                    'files do not match'
              WRITE (*,*) ' '
            ENDIF

          ENDIF
        ENDIF
```

```
C ***   Initialize the Grinnell
1010    CALL IMINIT ('ERASE')
        CALL GRDOP(1,0)
        CALL GRDSH(1,0)
        CALL GRDTH(1,0,0)
        CALL GRDDG(1,2,,6)
        CALL GRSBFD

        CALL TTYINC('Strike RETURN to take a picture:  ',
     &               DUMMY,IERR,0,' ')

C       Display all image color planes of the picture
        CALL GRDDG(1,1,0,7)
        CALL GRSBFD

C       Ask the operator which view we're looking at
88      CALL TTYINC('What kernel view is this?  (Top/ Side):  ',
     &               DUMMY,IERR,0,'?')
        IF (IERR.NE.0) DUMMY = '?'
        IF (DUMMY.EQ.'T' .OR. DUMMY.EQ.'t') THEN
          TPVIEW = .TRUE.
        ELSE IF (DUMMY.EQ.'S' .OR. DUMMY.EQ.'s') THEN
          TPVIEW = .FALSE.
        ELSE
          TYPE 1,' '
          TYPE 1,' Invalid view.  Only Top and Side views allowed.'
          GOTO 88
        ENDIF

C       Tell op to hold his horses.  That we're working on it.
        TYPE 1,' Copying image into memory...'

C       Copy the image from the Grinnell into our IMAGE array
        CALL IMDISP('READ','INTEGER*2',
     X          IMAGE,NELEM,NELEM,NLINE,0,0,'WHITE')

        TYPE 1,' Calculating histogram...'

C ***   Set the outside edges to white

C       Top row
        DO I = 1,NELEM
          IMAGE(I,    1) = WHITE
        ENDDO

C       Bottom row
        DO I = 1, NELEM
          IMAGE(I,NLINE) = WHITE
        ENDDO

C       Left and right edges
        DO J = 2, NLINE-1
          IMAGE(1, J) = WHITE
          DO I = NELEM+1-RM, NELEM
            IMAGE(I, J) = WHITE
```

```
          ENDDO
        ENDDO

C ***   Copy the portions of the image that we're likely to destroy
        DO J = DELTA, NLINE, DELTA
          DO I = DELTA, NELEM, DELTA
            TIMAGE(I,J) = IMAGE(I,J)
          ENDDO
        ENDDO

C ***   Calculate the histogram

C       Clear the histogram array
        DO I = BLACK, WHITE
          HIST(I) = 0
        ENDDO

C       Count the number of each intensity present in the image
        DO J = 2, NLINE-1
          DO I = 2, NELEM-RM
            HIST(IMAGE(I,J)) = HIST(IMAGE(I,J))+1
          ENDDO
        ENDDO

C ***   Find a good threshold value below first white peak

        THRESH = 0

C       Running sum of bins
        SUM = 0
        DO I = WHITE, WHITE-NBIN+1, -1
          SUM = SUM + HIST(I)
        ENDDO

C       Detect right peak
        DO WHILE (SUM.LE.THR1 .AND. I.GE.0)
          SUM = SUM + HIST(I) - HIST(I+NBIN)
          I = I-1
        ENDDO

C       Detect right side of valley
        DO WHILE (SUM.GE.THR2 .AND. I.GE.0)
          SUM = SUM + HIST(I) - HIST(I+NBIN)
          I = I-1
        ENDDO

C       Find the first place that would catch water (i.e., low spot)
        IF (I.GE.0) THEN
          LSUM = SUM
          SUM = SUM + HIST(I) - HIST(I+NBIN)
          I = I-1
        ENDIF

        DO WHILE (SUM.LE.LSUM .AND. I.GE.0)
          LSUM = SUM
          SUM = SUM + HIST(I) - HIST(I+NBIN)
```

```
          I = I-1
        ENDDO

C       Remember this spot
        CTHRSH = I+NBIN

C       Detect next peak to left
        DO WHILE (SUM.LE.THR1 .AND. I.GE.0)
          SUM = SUM + HIST(I) - HIST(I+NBIN)
          I = I-1
        ENDDO

C       If we couldn't find two peaks, tell him to adjust things
        IF (I.LE.0) THEN
          TYPE 1,' '
          TYPE 1,' Threshold could not be automatically determined.'
          TYPE 1,' Better results may be obtained by adjusting the '//
     &          'camera'
          TYPE 1,' f-stop so that the picture is not washed out or '//
     &          'dark,'
          TYPE 1,' or by changing the camera height so that the '//
     &          'kernel and'
          TYPE 1,' background areas are more the same size.'
          TYPE 1,' '
          CTHRSH = 0
          GOTO 501
        ENDIF

C       Turn back ...

C       Detect left side of valley
        DO WHILE (SUM.GE.THR2 .AND. I+1+NBIN.LE.255)
          I = I+1
          SUM = SUM - HIST(I) + HIST(I+NBIN)
        ENDDO

C       Find the first place that would catch water (i.e., low spot)
        IF (I+1+NBIN.LE.255) THEN
          LSUM = SUM
          I = I+1
          SUM = SUM - HIST(I) + HIST(I+NBIN)
        ENDIF

        DO WHILE (SUM.LE.LSUM .AND. I+1+NBIN.LE.255)
          LSUM = SUM
          I = I+1
          SUM = SUM - HIST(I) + HIST(I+NBIN)
        ENDDO

C       Set thresh to center of valley
        CTHRSH = (CTHRSH + I+1)/2

C ***   Let the user choose the threshold
501     TYPE 3,' Computed threshold:  ',CTHRSH
        TYPE 3,' Previous threshold:  ',THRESH
        TYPE 1,' Enter desired threshold value,'
```

```
      TYPE 1,'   or 1 to plot histogram,'
      CALL TTYINI('   or RETURN to use computed threshold:  ',
     &            I,IERR,0,0)
      IF (IERR.NE.0) I = 0

      IF (I.EQ.0) THEN

C       For RETURN, use the computed threshold
        THRESH = CTHRSH

      ELSE IF (I.EQ.1) THEN

C       Response of 1 asks us to plot the histogram
        CALL PLAWT(HIST,'Histogram')
        GOTO 501

      ELSE

C       Other response is his idea of a good threshold
        THRESH = I

      ENDIF

      TYPE 3,' THRESH is ',THRESH

C *** Search out the wheat kernels

C     Punch holes every DELTA pixels
      DO ICKY = DELTA, NLINE, DELTA
        DO ICKX = DELTA, NELEM, DELTA

C         Search in a different order for side views
          IF (TPVIEW) THEN
            SCANX = ICKX
            SCANY = ICKY
          ELSE
            SCANX = ICKY
            SCANY = ICKX
          ENDIF

C         Test the punched pixels against our threshold
          IF (IMAGE(SCANX,SCANY).LE.THRESH) THEN

C ---       Found a good pixel, so zoom left until find an edge
            ROOTY = SCANY
            ROOTX = SCANX - 1

101         DO WHILE (ROOTX.GE.1 .AND. IMAGE(ROOTX,ROOTY).LE.THRESH)
              ROOTX = ROOTX - 1
            ENDDO
            ROOTX = ROOTX + 1

C ---       Follow the edge, putting directions in the EDGCHN array
            DIRECT = 0
            CHAINX = ROOTX
            CHAINY = ROOTY
```

```
            EDGCNT = 1

100         ITEMP = DIRECT + 5
            DO I = ITEMP, ITEMP+8-1
              DIRECT = MDD(I,8)
              IF (IMAGE(CHAINX+XDELT(DIRECT),CHAINY+YDELT(DIRECT))
     &           .LE. THRESH) THEN
                EDGCHN(EDGCNT) = DIRECT
                CHAINX = CHAINX + XDELT(DIRECT)
                CHAINY = CHAINY + YDELT(DIRECT)

                EDGCNT = EDGCNT + 1
                IF (EDGCNT.LE.EDGMAX) THEN
                  IF (CHAINX.EQ.ROOTX .AND. CHAINY.EQ.ROOTY) GOTD 200
                  GOTO 100
                ELSE
                  TYPE 1,' CHAIN TOD LDNG.  TRACING ABORTED.'
                  GDTO 300
                ENDIF
              ENDIF
            ENDDO
200         CONTINUE
            EDGCNT = EDGCNT - 1
            CALL GRSBFD

C ---       Find the bounds of the edge
            MAXLX = ROOTX
            MINLX = ROOTX
            MAXLY = ROOTY
            MINLY = RODTY

            ICUR = ROOTX
            JCUR = ROOTY
            DO I = 1, EDGCNT
              MINLX  = MIN(MINLX,ICUR)
              MAXLX  = MAX(MAXLX,ICUR)
              MINLY  = MIN(MINLY,JCUR)
              MAXLY  = MAX(MAXLY,JCUR)
              DIRECT = EDGCHN(I)
              ICUR   = ICUR + XDELT(DIRECT)
              JCUR   = JCUR + YDELT(DIRECT)
            ENDDD

C ---       Set up the LIMAGE array so that interior elements are
C           .TRUE.

C           Clear subset of interior membership array
            DD J = MINLY, MAXLY
              DO I = MINLX, MAXLX
                LIMAGE(I,J) = .FALSE.
              ENDDD
            ENDDO

C           Toggle membership until correct
            DD I = 1, EDGCNT
              DIRECT = EDGCHN(I)
```

```
              IF (DIRECT.GE.1 .AND. DIRECT.LE.3) THEN
                DO J = ICUR, MAXLX
                  LIMAGE(J,JCUR) = .NDT. LIMAGE(J,JCUR)
                ENDDO
                ICUR = ICUR + XDELT(DIRECT)
                JCUR = JCUR + YDELT(DIRECT)
              ELSE IF (DIRECT.GE.5 .AND. DIRECT.LE.7) THEN
                ICUR = ICUR + XDELT(DIRECT)
                JCUR = JCUR + YDELT(DIRECT)
                DO J = ICUR, MAXLX
                  LIMAGE(J,JCUR) = .NDT. LIMAGE(J,JCUR)
                ENDDD
              ELSE
                ICUR = ICUR + XDELT(DIRECT)
                JCUR = JCUR + YDELT(DIRECT)
              ENDIF
            ENDDO
C
            Include the border elements as members
            DO I = 1, EDGCNT
              LIMAGE(ICUR,JCUR) = .TRUE.
              DIRECT = EDGCHN(I)
              ICUR   = ICUR + XDELT(DIRECT)
              JCUR   = JCUR + YDELT(DIRECT)
            ENDDO

C ---       Test the contour direction
            TESTX = ROOTX - 1
            IF (TESTX.GE.MINLX .AND. LIMAGE(TESTX,ROOTY)) THEN

C             The point we considered to be outside ended up on
C             the inside, so we've been deceived by some noise.
C             Therefore, we must resume searching for the outside
C             edge of this region on the other side of this noise
C             blotch.

              ROOTX = MINLX
              RDOTY = MINLY
              DO WHILE (.NDT. LIMAGE(RODTX,ROOTY))
                RODTY = RODTY + 1
              ENDDO
              GOTO 101

            ENDIF

C ---       If the chain is long enough and doesn't touch the edges,
C           then process it
            IF (EDGCNT.GE.EDGMIN .AND. MINLX.GT.2 .AND. MINLY.GT.2
     &        .AND. MAXLX.LT.NELEM-RM .AND. MAXLY.LT.NLINE-1) THEN

C ---         Calculate the centroid and area
              XSUM = D
              YSUM = 0
              AREA = 0
              DD J = MINLY, MAXLY
                DD I = MINLX, MAXLX
```

```
                IF (LIMAGE(I,J)) THEN
                  XSUM = XSUM + I
                  YSUM = YSUM + J
                  AREA = AREA + 1
                ENDIF
              ENDDO
            ENDDO
            XCENT = IFIX( FLOAT(XSUM)/FLOAT(AREA) )
            YCENT = IFIX( FLOAT(YSUM)/FLOAT(AREA) )

C ---       If the centroid is not internal, automatically reject
C           this edge
            IF (.NOT. LIMAGE(XCENT,YCENT)) GOTO 2020

C ---       Show the op the boundary that we've found
            DO I = 1, EDGCNT

C             Turn this pixel green
              CALL GRFAR(7,  0,0,1,ICUR-1,NLINE-JCUR,1,1)
              CALL GRFAR(2,255,0,1,ICUR-1,NLINE-JCUR,1,1)
              DIRECT = EDGCHN(I)
              ICUR   = ICUR + XDELT(DIRECT)
              JCUR   = JCUR + YDELT(DIRECT)
            ENDDO
            CALL GRSBFD

C ---       Ask him what he wants to do with this one (point with
C           crosshairs)
2021        CALL GRZCL(1,XCENT-1,NLINE-YCENT)
            CALL GRZCB(1,0)
            CALL GRZCO(1,1)
            CALL GRSBFD
            CALL TTYINC('Enter  Accept, Reject, Change:  ',
     &                 DUMMY,IERR,0,'?')
            IF (IERR.NE.0) DUMMY = '?'
            CALL GRZCO(1,0)
            CALL GRSBFD

            IF (DUMMY.EQ.'A' .OR. DUMMY.EQ.'a') THEN

C ...          The operator has accepted this edge

              IF (SAVEDG .OR. SAVSIO) THEN
                XSPAN = MAXLX - MINLX + 1
                YSPAN = MAXLY - MINLY + 1
                OFSTX = ROOTX - MINLX + 1
                OFSTY = ROOTY - MINLY + 1
                IF (TPVIEW) THEN
                  WRITE (UN1) XSPAN,YSPAN,OFSTX,OFSTY,
     &                        EDGCNT,(EDGCHN(I),I=1,EDGCNT)
                  NKERN1 = NKERN1 + 1
                ELSE
                  WRITE (UN3) XSPAN,YSPAN,OFSTX,OFSTY,
     &                        EDGCNT,(EDGCHN(I),I=1,EDGCNT)
                  NKERN3 = NKERN3 + 1
                ENDIF
```

```
                  ENDIF

                  CALL GRSBFD

                ELSE IF (DUMMY.EQ.'R' .OR. DUMMY.EQ.'r') THEN

C ...             The operator has rejected the edge
                  DO I = 1, EDGCNT

C                   Turn this pixel blue
                    CALL GRFAR(7,  0,0,1,ICUR-1,NLINE-JCUR,1,1)
                    CALL GRFAR(4,255,0,1,ICUR-1,NLINE-JCUR,1,1)
                    DIRECT = EDGCHN(I)
                    ICUR   = ICUR + XDELT(DIRECT)
                    JCUR   = JCUR + YDELT(DIRECT)
                  ENDDO
                  CALL GRSBFD

                ELSE IF (DUMMY.EQ.'C' .OR. DUMMY.EQ.'c') THEN

C ...             The op wants to change something
                  GOTO 1012

                ELSE

C ...             Any other answer is invalid
                  TYPE 1,' Invalid response'
                  GOTO 2021

                ENDIF

              ENDIF

C ---         In any case, don't search in here again (put the white
C             dots in)
2020          KL = MINLX + DELTA-1
              KL = KL - MOD(KL,DELTA)
              KB = MINLY + DELTA-1
              KB = KB - MOD(KB,DELTA)
              IF (KL.LE.MAXLX .AND. KB.LE.MAXLY) THEN
                DO J = KB, MAXLY, DELTA
                  DO I = KL, MAXLX, DELTA
                    IF (LIMAGE(I,J)) IMAGE(I,J) = WHITE
                  ENDDO
                ENDDO
              ENDIF

            ENDIF
          ENDDO
        ENDDO
300     CONTINUE
        CALL GRSBFD

C ***   Restore the parts of the image that we messed up
        DO J = DELTA, NLINE, DELTA
          DO I = DELTA, NELEM, DELTA
```

```
          IMAGE(I,J) = TIMAGE(I,J)
        ENDDO
      ENDDO

C ***  Upper command level.  What does the op want to do?
1012  CALL TTYINC(
     &  'Enter  Thresh, Value, Camera, Number, Range, or Exit:  ',
     &  DUMMY,IERR,0,'?')
      IF (IERR.NE.0) DUMMY = '?'

      IF (DUMMY.EQ.'T' .OR. DUMMY.EQ.'t') THEN

C ---   The op wants to change the threshold values
        CALL TTYINY('Redraw screen? (Y/<N>):  ',
     &              ITMP1,IERR,0,'NO')
        DUMMY = 'N'
        IF (IERR.EQ.0 .AND. ITMP1.EQ.1) DUMMY = 'Y'

        IF (DUMMY.EQ.'Y' .OR. DUMMY.EQ.'y') THEN

C         Restore any damage we may have done
          DO J = DELTA, NLINE, DELTA
            DO I = DELTA, NELEM, DELTA
              IMAGE(I,J) = TIMAGE(I,J)
            ENDDO
          ENDDO

C         Redisplay the image
          CALL IMDISP('WRITE','INTEGER*2',
     X                IMAGE,NELEM,NELEM,NLINE,0,0,'WHITE')

        ENDIF
        GOTO 501

      ELSE IF (DUMMY.EQ.'V' .OR. DUMMY.EQ.'v') THEN

C ---   The op wants to view the intensities of selected pixels
        CALL GRZCB(1,0)
        CALL GRZCD(1,1)
        CALL GRSBFD
        TYPE 1,' Turn cursor select knob to ZOOM and switch'
        TYPE 1,' cursor 1 and FUNA on.'
        TYPE 1,'  '
        TYPE 1,' Move cursor to desired location and press ENTER.'
        TYPE 1,' To exit, turn FUNA off and press ENTER.'
1011    CALL GRZCR(1,XLOC,YLOC,ENTER,FUN1,FUN2)
        ENTER = 0
        DO WHILE (ENTER.EQ.0)
          CALL GRZCW(1,ISTAT)
          CALL GRZCR(1,XLOC,YLOC,ENTER,FUN1,FUN2)
        ENDDO
        IF (FUN1.EQ.1) THEN
          TYPE 3,' Pixel value is',IMAGE(XLOC+1,NLINE-YLOC)
          GOTO 1011
        ENDIF
        CALL GRZCO(1,0)
```

```
           CALL GRSBFD
           GOTO 1012

         ELSE IF (DUMMY.EQ.'C' .OR. DUMMY.EQ.'c') THEN

C ---      The op selected 'camera', so start again from the top
           IF (SAVEDG) WRITE (*,*) NKERN1,' kernels in ',FN1
           IF (SAVSID) WRITE (*,*) NKERN3,' kernels in ',FN3
           GOTO 1010

         ELSE IF (DUMMY.EQ.'E' .OR. DUMMY.EQ.'e') THEN

C ---      The op wants to exit, so close the files and the Grinnell and
C          exit

         ELSE IF (DUMMY.EQ.'N' .OR. DUMMY.EQ.'n') THEN

C ---      The op wants to know how many we have
           IF (SAVEDG) WRITE (*,*) NKERN1,' kernels in ',FN1
           IF (SAVSID) WRITE (*,*) NKERN3,' kernels in ',FN3
           GOTO 1012

         ELSE IF (DUMMY.EQ.'R' .OR. DUMMY.EQ.'r') THEN

C ---      The op wants to color threshold a range of the picture
907        CALL TTYINI('Enter low end of range:  ',
     &                 LRANGE,IERR,0,-1)
           IF (IERR.NE.0) LRANGE = -1
           IF (LRANGE.LT.0 .OR. LRANGE.GT.255) THEN
             TYPE 1,' Invalid pixel intensity'
             GOTO 907
           ENDIF

908        CALL TTYINI('Enter high end of range:  ',
     &                 HRANGE,IERR,0,-1)
           IF (IERR.NE.0) HRANGE = -1
           IF (HRANGE.LT.0 .OR. HRANGE.GT.255) THEN
             TYPE 1,' Invalid pixel intensity'
             GOTO 908
           ENDIF

           IF (HRANGE.LT.LRANGE) THEN
             TYPE 1,' Ranges specified in wrong order.  Range aborted.'
             GOTO 1012
           ENDIF

909        CALL TTYINI('Enter color planes  1=Red, 2=Green, 4=Blue:  ',
     &                 ICOLOR,IERR,0,-1)
           IF (IERR.NE.0) ICOLOR = -1
           IF (ICOLOR.LT.0 .OR. ICOLOR.GT.7) THEN
             TYPE 1,' Invalid color plane combination'
             GOTO 909
           ENDIF

C          Set pixels in the range to the specified color
           DO J = 1, NLINE
```

```
          DO I = 1, NELEM
            ITEMP = IMAGE(I,J)
            IF (MOD(I,DELTA).EQ.0 .AND. MOD(J,DELTA).EQ.0)
     &        ITEMP = TIMAGE(I,J)
            IF (ITEMP.GE.LRANGE .AND. ITEMP.LE.HRANGE) THEN
              CALL GRFAR(     7,  0,0,1,I-1,NLINE-J,1,1)
              CALL GRFAR(ICOLOR,255,0,1,I-1,NLINE-J,1,1)
            ENDIF
          ENDDO
        ENDDO

        CALL GRSBFD
        GOTO 1012

      ELSE

C ---   Any other response is invalid
        TYPE 1,' Invalid response'
        GOTO 1012

      ENDIF

      WRITE (*,*) ' '

      IF (SAVEDG) THEN
        CLOSE(UN1)
        WRITE (*,*) NKERN1,' kernels in ',FN1
      ENDIF

      IF (SAVSID) THEN
        CLOSE(UN3)
        WRITE (*,*) NKERN3,' kernels in ',FN3
      ENDIF

9999  CALL GRSEND

      WRITE (*,*) 'Have a nice day'

1     FORMAT(A)
3     FORMAT(A,I7)

      END
```

```
C*********************************************************************
C
C       PLAWT
C
C       VAX-11 FORTRAN SOURCE FILENAME:         PLAWT.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             OCTOBER 23, 1984        TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               CALL PLAWT(IARRAY,TITLE)
C
C       PURPOSE
C
C               A simple function-plotting routine
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               BELL   - PLOT10 Routine
C               GETUTX - PLOT10 Routine
C               PAXIS  - KSU Plot Routine
C               PCLOSP - KSU Plot Routine
C               PCLRSC - KSU Plot Routine
C               PINIT  - KSU Plot Routine
C               PORIG  - KSU Plot Routine
C               PLINE  - KSU Plot Routine
C               PSCALE - KSU Plot Routine
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE'
C
C               IARRAY - INTEGER array of 256 values to be plotted
C                        vs. an X-axis from 0 to 255.
C               TITLE  - CHARACTER*(*) string to be plotted as the
C                        X-axis label
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C*********************************************************************
C
        SUBROUTINE PLAWT(IARRAY,TITLE)

        IMPLICIT NONE

        INTEGER IARRAY(0:255),IGOT,I
        REAL    RARRAY(0:255)/256*0./,FIRSTX,DELTAX,XARRAY(0:255)
```

```
      REAL    FIRSTY,DELTAY,FIROEL(4),OIVLNX,OIVLNY
      EQUIVALENCE (FIROEL(1),FIRSTX),(FIRDEL(2),OELTAX),
     X            (FIROEL(3),FIRSTY),(FIRDEL(4),DELTAY)
      CHARACTER*(*) TITLE
      CHARACTER*1 ACHAR

C     Convert the y-coordinates to real
      DO I=0,255
        RARRAY(I) = IARRAY(I)
      ENODO

C     Initialize the x-coordinate array
      DO I=0,255
        XARRAY(I) = I
      ENDOO

C     Plot on the Tektronix
      CALL PINIT(4014,' ',1.,'A')
      CALL PSCALE(XARRAY,256,30.,FIRSTX,OELTAX,DIVLNX)
      CALL PSCALE(RARRAY,256,20.,FIRSTY,OELTAY,OIVLNY)
      CALL PORIG(4.,4.)
      CALL PAXIS(0.,0.,TITLE,' ',220,2201,30.,0. ,FIRSTX,OELTAX,
     &           DIVLNX)
      CALL PAXIS(0.,0.,' '  ,' ',120,1201,20.,90.,FIRSTY,OELTAY,
     &           OIVLNY)
      CALL PLINE(XARRAY,RARRAY,256,FIROEL,0,' ',OIVLNX,OIVLNY)

C     Wait for carriage return before continuing
      CALL BELL
      CALL GETUTX(1,' ',1,ACHAR,IGOT)
C     CALL PCLRSC
      CALL PCLOSP
      WRITE (*,*) CHAR(27)//'2 '

      RETURN
      END
```

```
C***********************************************************************
C
C       ORIENT
C
C       VAX-11 FORTRAN SOURCE FILENAME:         ORIENT.FOR
C
C       OEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             NOVEMBER 30, 1984       TERRY E. SCHMALZRIEO
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIEO
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIEO
C
C***********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN ORIENT
C
C       PURPOSE
C
C               This program displays the contours of wheat kernels
C               that were produced by the WHEAT program.  The operator
C               moves a joystick to indicate the germ end of the
C               kernel.  An archive file is created and the file that
C               came from the WHEAT program can then be discarded.
C
C       ROUTINE(S) ACCESSEO OR CALLEO BY THIS ROUTINE
C
C               IMINIT - KSU Olsplay Oevice Initialization Routine
C               GRFAR  - Grinnell Systems Routine
C               GRFVC  - Grinnell Systems Routine
C               GRSBFO - Grinnell Systems Routine
C               GRZCB  - Grinnell Systems Routine
C               GRZCL  - Grinnell Systems Routine
C               GRZCO  - Grinnell Systems Routine
C               GRZCR  - Grinnell Systems Routine
C               GRZCW  - Grinnell Systems Routine
C               SERCH2 - Kernel Radial Measurement Routine
C               TTYINC - Ralph's Character Input Routine
C
C       ARGUMENT(S) REQUIREO FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIEO  TO  THE CALLING ROUTINE
C
C               None
C
C***********************************************************************
C
        IMPLICIT        NONE
        INTEGER         I,J,K,XSPAN,YSPAN,NELEM,NLINE,XCENT,YCENT,
     &                  XLOC,YLOC,NMEAS,ENTER,FUN1,FUN2,ISTAT,XSUM,YSUM,
```

```
     &               AREA,ICUR,JCUR,DIRECT,EDGCNT,ROOTX,RODTY,
     &               UN1,UN2,EDGMAX,ITMP,IERR,
     &               XDELT(D:7)/1, 1, 0,-1,-1,-1,0,1/,
     &               YDELT(D:7)/D,-1,-1,-1, 0, 1,1,1/
      PARAMETER      (NELEM=512,NLINE=512,EDGMAX=2000)
      INTEGER*2      EDGCHN(EDGMAX)
      REAL           X,Y,Y2,DX,DY,DIFANG,MAJANG,MDVANG,PI,IX,IY,IXY,
     &               TIX,TIY,TIXY
      PARAMETER      (PI=3.141592654)
      LOGICAL*1      LIMAGE(NELEM,NLINE)
      CHARACTER      DUMMY*1,FN1*24,FN2*24,FN3*24,SWDRK*80

C     Ask him which file we're examining
      CALL TTYINC('Enter filename of contour data file:  ',
     &            FN1,IERR,0,' ')
      IF (IERR.NE.0 .OR. FN1.EQ.' ') GOTO 9999
      UN1 = 15
      OPEN(UN1,FILE=FN1,ERR=9999,STATUS='DLD',FDRM='UNFDRMATTED')

C     Ask him where to put the archive data
      CALL TTYINC('Enter filename of archive file to be created:  ',
     &            FN2,IERR,0,' ')
      IF (IERR.NE.0 .OR. FN1.EQ.' ') GOTD 9999
      UN2 = 16
      OPEN(UN2,FILE=FN2,ERR=9999,STATUS='NEW',FDRM='UNFDRMATTED')

C     Set up the Grinnell the way we like it
      CALL IMINIT('ERASE')
      CALL GRZCB(1,0)

C     Tell the op what the scoop is:
      TYPE 1,' '
      TYPE 1,' Cursor select ZODM, cursor 1 on, track on'
      TYPE 1,' '
      TYPE 1,' Move cursor toward germ end for each kernel'
      TYPE 1,' '
      TYPE 1,' '

C     Keep track of how many we've measured
      NMEAS = 0

C     Load in a kernel
100   READ (UN1,END=9998) XSPAN,YSPAN,RODTX,RDOTY,
     &                    EDGCNT,(EDGCHN(I),I=1,EDGCNT)

C     Clear the membership array
      DD J = 1, YSPAN
        DD I = 1, XSPAN
          LIMAGE(I,J) = .FALSE.
        ENDDO
      ENDDD

C     Set up the membership array
      ICUR = RODTX
      JCUR = RDDTY
      DD I = 1, EDGCNT
```

```
          DIRECT = EDGCHN(I)
C         Color as we go
          CALL GRFAR(3,255,0,1,ICUR-1,NLINE-JCUR,1,1)
C         This is the fancy toggling
          IF (DIRECT.GE.1 .AND. DIRECT.LE.3) THEN
            DO J = ICUR, XSPAN
              LIMAGE(J,JCUR) = .NOT. LIMAGE(J,JCUR)
            ENDDO
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
          ELSE IF (DIRECT.GE.5 .AND. DIRECT.LE.7) THEN
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
            DO J = ICUR, XSPAN
              LIMAGE(J,JCUR) = .NOT. LIMAGE(J,JCUR)
            ENDDO
          ELSE
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
          ENDIF
        ENDDO      .
C       Make sure that the border elements are members
        DO I = 1, EDGCNT
          LIMAGE(ICUR,JCUR) = .TRUE.
          DIRECT = EDGCHN(I)
          ICUR   = ICUR + XDELT(DIRECT)
          JCUR   = JCUR + YDELT(DIRECT)
        ENDDO
C       Find the centroid
        XSUM = 0
        YSUM = 0
        AREA = 0
        DO J = 1, YSPAN
          DO I = 1, XSPAN
            IF (LIMAGE(I,J)) THEN
              XSUM = XSUM + I
              YSUM = YSUM + J
              AREA = AREA + 1
            ENDIF
          ENDDO
        ENDDO
        XCENT = IFIX( FLOAT(XSUM)/FLOAT(AREA) )
        YCENT = IFIX( FLOAT(YSUM)/FLOAT(AREA) )
C       Calculate the moments of and product of inertia about
C       (XCENT,YCENT)
        IX  = 0.
        IY  = 0.
        IXY = 0.
        DO J = 1, YSPAN
          Y    = J - YCENT
          Y2   = Y*Y
```

```
          TIX  = 0.
          TIY  = 0.
          TIXY = 0.
          OO I = 1, XSPAN
            IF (LIMAGE(I,J)) THEN
              X    = I - XCENT
              TIX  = TIX  + Y2
              TIY  = TIY  + X*X
              TIXY = TIXY + X*Y
            ENDIF
          ENDDO
          IX  = IX  + TIX
          IY  = IY  + TIY
          IXY = IXY + TIXY
        ENDOO
C       Calculate the angle of the major principal axis
110     MAJANG = ATAN2(-IXY*2.,IX-IY)/2. + PI/2.
C       Turn on the cross hairs
        CALL GRZCL(1,XCENT-1,NLINE-YCENT)
        CALL GRZCO(1,1)
        CALL GRSBFO
C       Wait for him to move the zoom cursor toward the germ end
        CALL GRZCR(1,XLOC,YLOC,ENTER,FUN1,FUN2)
        ENTER = 0
        DO WHILE (ENTER.EQ.0)
          CALL GRZCW(1,ISTAT)
          CALL GRZCR(1,XLOC,YLOC,ENTER,FUN1,FUN2)
        ENDOO
C       Turn off the cross hairs
        CALL GRZCO(1,0)
        CALL GRSBFO
C       Calculate the angle of his motion
        MOVANG = ATAN2(FLOAT((NLINE-YLOC)-YCENT),FLOAT((XLOC+1)-XCENT))
C       Adjust MAJANG if necessary
        DIFANG = MOO( ABS(MOVANG-MAJANG), 2.*PI )
        IF (DIFANG .GT. PI    ) OIFANG = 2.*PI - OIFANG
        IF (DIFANG .GT. PI/2.) MAJANG = MAJANG + PI
        OO WHILE (MAJANG.LT.0.0)
          MAJANG = MAJANG + 2.*PI
        ENOOO
        MAJANG = MOO( MAJANG, 2.*PI )
C       Color the positive principal axis red for verification purposes
        CALL SERCH2(LIMAGE,NELEM,NLINE,MAJANG,XCENT,YCENT,I,J,XSPAN,
     &             YSPAN)
        CALL GRFVC(1,255,0,1,XCENT-1,NLINE-YCENT,I-1,NLINE-J)
        CALL GRSBFD
C       Make sure he didn't make a mistake
        WRITE (SWORK,'(I5,A)') NMEAS+1,'  <Accept>, Re-orient:  $'
```

```
      J = INDEX(SWORK,'%') - 1
      CALL TTYINC(SWORK(1:J),DUMMY,IERR,0,'A')
      CALL GRFAR(7,0,0,1,0,NLINE-YSPAN,XSPAN,YSPAN)
      CALL GRSBFD
      IF (DUMMY.EQ.'R' .OR. DUMMY.EQ.'r') THEN

C       He wants us to re-display this kernel
        DO I = 1, EDGCNT
          DIRECT = EDGCHN(I)
          CALL GRFAR(3,255,0,1,ICUR-1,NLINE-JCUR,1,1)
          ICUR   = ICUR + XDELT(DIRECT)
          JCUR   = JCUR + YDELT(DIRECT)
        ENDDO
        GOTO 110
      ENDIF

C     Chalk up one more
      NMEAS = NMEAS + 1

      WRITE (UN2) EDGCNT,(EDGCHN(I),I=1,EDGCNT),MAJANG

C     Go back for another kernel.
      GOTO 100

C-----------------------------------------------------------------------

C     Come here when no more to be read
9998  CLOSE(UN1)
      CLOSE(UN2)

9999  CALL GRSEND

1     FORMAT(A)
2     FORMAT(A,I4,A)
      END
```

```
C*********************************************************************
C
C       SERCH2
C
C       VAX-11 FORTRAN SOURCE FILENAME:         ORIENT.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             NOVEMBER 30, 1984       TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               CALL SERCH2(LIMAGE,NELEM,NLINE,THETA,ISTART,JSTART,
C            &              IEND,JEND,XSPAN,YSPAN)
C
C       PURPOSE
C
C               This subroutine follows a vector, starting at
C               (ISTART,JSTART) at angle THETA, until it meets with the
C               edge.  The final endpoint of the vector is returned as
C               (IEND,JEND).  The edge is considered to be the last
C               .TRUE. element in LIMAGE that the vector encounters as
C               it is grown from the beginning location.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               None
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               LIMAGE - NELEM by NLINE array of LOGICAL*1.  True
C                        values indicate that that pixel is interior
C                        to the current edge contour.
C               NELEM  - INTEGER number of columns in the LIMAGE array
C               NLINE  - INTEGER number of rows in the LIMAGE array
C               THETA  - REAL angle to draw vector
C               ISTART - INTEGER column index of initial point
C               JSTART - INTEGER row index of initial point
C               XSPAN  - INTEGER last valid column in LIMAGE array
C               YSPAN  - INTEGER last valid row in LIMAGE array
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               IEND   - INTEGER column index of edge intersection
C               JEND   - INTEGER row index of edge intersection
C
C*********************************************************************
C
        SUBROUTINE SERCH2(LIMAGE,NELEM,NLINE,THETA,ISTART,JSTART,IEND,
```

```
     &                      JEND,XSPAN,YSPAN)

      IMPLICIT NONE

      INTEGER   ISTART,JSTART,IEND,JEND,I,J,XSPAN,YSPAN,NELEM,NLINE
      LOGICAL*1 LIMAGE(NELEM,NLINE)
      REAL      THETA,X,Y,DX,DY

C     Figure an increment and set up the initial point
      DX   = 0.5*COS(THETA)
      DY   = 0.5*SIN(THETA)
      X    = FLOAT(ISTART)
      Y    = FLOAT(JSTART)
      IEND = NINT(X)
      JEND = NINT(Y)
      I    = NINT(X)
      J    = NINT(Y)

C     Find and return the edge of the contour
      DO WHILE (I.GE.1 .AND. J.GE.1 .AND. I.LE.XSPAN .AND. J.LE.YSPAN
     &          .AND. LIMAGE(I,J))
        IEND = I
        JEND = J
        X = X + DX
        Y = Y + DY
        I = NINT(X)
        J = NINT(Y)
      ENDDO

      RETURN
      END
```

```
C*********************************************************************
C
C       MEASURE
C
C       VAX-11 FORTRAN SOURCE FILENAME:         MEASURE.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             NOVEMBER 30, 1985       TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN MEASURE
C
C       PURPOSE
C
C               This program measures the wheat kernel contours from
C               the archive file and creates a measurement file for
C               use by PLOTSTATS and the classifier programs.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               SERCH2 - Kernel Radial Measurement Routine
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None.
C
C*********************************************************************
C
        IMPLICIT        NONE
        INTEGER         I,J,K,XSPAN,YSPAN,NELEM,NLINE,XCENT,YCENT,
     &                  XLOC,YLOC,NMEAS,ENTER,FUN1,FUN2,ISTAT,XSUM,YSUM,
     &                  AREA,ICUR,JCUR,DIRECT,EDGCNT,ROOTX,ROOTY,
     &                  UN1,UN2,NANGLS,EDGMAX,ITMP,XMAX,XMIN,YMAX,YMIN,
     &                  XDELT(0:7)/1, 1, 0,-1,-1,-1,0,1/,
     &                  YDELT(0:7)/0,-1,-1,-1, 0, 1,1,1/
        PARAMETER       (NELEM=512,NLINE=512,NANGLS=720,EDGMAX=2000)
        INTEGER*2       EDGCHN(EDGMAX)
        REAL            X,Y,Y2,DX,DY,DIFANG,MAJANG,MOVANG,PI,IX,IY,IXY,
     &                  TIX,TIY,TIXY,RADIAL(NANGLS),ANG
        PARAMETER       (PI=3.141592654)
        LOGICAL*1       LIMAGE(NELEM,NLINE)
        CHARACTER       DUMMY*1,FN1*24,FN2*24,FN3*24
```

```
C       Ask him which file we're examining
        TYPE 1,'$Enter filename of archive file:  '
        ACCEPT 1,FN1
        UN1 = 15
        OPEN(UN1,FILE=FN1,ERR=9999,STATUS='OLD',FORM='UNFORMATTED')

C       Ask him where to put the measurements
        TYPE 1,'$Enter name of measurement file to be created:  '
        ACCEPT 1,FN2
        UN2 = 16
        OPEN(UN2,FILE=FN2,ERR=9999,STATUS='NEW',FORM='UNFORMATTED')

C       Keep track of how many we've measured
        NMEAS = 0

C       Load in a kernel
100     READ (UN1,END=9998) EDGCNT,(EDGCHN(I),I=1,EDGCNT),MAJANG

C       Find the extents of the contour
        ICUR = 0
        JCUR = 0
        XMAX = ICUR
        XMIN = ICUR
        YMAX = JCUR
        YMIN = JCUR
        DO I = 1, EDGCNT
          DIRECT = EDGCHN(I)
          ICUR   = ICUR + XDELT(DIRECT)
          JCUR   = JCUR + YDELT(DIRECT)
          XMAX   = MAX(XMAX,ICUR)
          XMIN   = MIN(XMIN,ICUR)
          YMAX   = MAX(YMAX,JCUR)
          YMIN   = MIN(YMIN,JCUR)
        ENDDO
        XSPAN = XMAX - XMIN + 1
        YSPAN = YMAX - YMIN + 1
        ROOTX = 1 - XMIN
        ROOTY = 1 - YMIN

C       Clear the membership array
        DO J = 1, YSPAN
          DO I = 1, XSPAN
            LIMAGE(I,J) = .FALSE.
          ENDDO
        ENDDO

C       Set up the membership array
        ICUR = ROOTX
        JCUR = ROOTY
        DO I = 1, EDGCNT
          DIRECT = EDGCHN(I)

C         This is the fancy toggling
          IF (DIRECT.GE.1 .AND. DIRECT.LE.3) THEN
            DO J = ICUR, XSPAN
              LIMAGE(J,JCUR) = .NOT. LIMAGE(J,JCUR)
```

```
            ENDDO
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
          ELSE IF (DIRECT.GE.5 .AND. DIRECT.LE.7) THEN
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
            DO J = ICUR, XSPAN
              LIMAGE(J,JCUR) = .NOT. LIMAGE(J,JCUR)
            ENDDO
          ELSE
            ICUR = ICUR + XDELT(DIRECT)
            JCUR = JCUR + YDELT(DIRECT)
          ENDIF
        ENDDO

C       Make sure that the border elements are members
        DO I = 1, EDGCNT
          LIMAGE(ICUR,JCUR) = .TRUE.
          DIRECT = EDGCHN(I)
          ICUR   = ICUR + XDELT(DIRECT)
          JCUR   = JCUR + YDELT(DIRECT)
        ENDDO

C       Find the centroid
        XSUM = 0
        YSUM = 0
        AREA = 0
        DO J = 1, YSPAN
          DO I = 1, XSPAN
            IF (LIMAGE(I,J)) THEN
              XSUM = XSUM + I
              YSUM = YSUM + J
              AREA = AREA + 1
            ENDIF
          ENDDO
        ENDDO
        XCENT = IFIX( FLOAT(XSUM)/FLOAT(AREA) )
        YCENT = IFIX( FLOAT(YSUM)/FLOAT(AREA) )

C       Chalk up one more
        NMEAS = NMEAS + 1
        WRITE (*,*) NMEAS

C       Measure the kernel and stash the info in a measurement file
        DO K = 1, NANGLS
          ANG        = FLOAT(K-1)/FLOAT(NANGLS) * 2.0*PI + MAJANG
          CALL SERCH2(LIMAGE,NELEM,NLINE,ANG,XCENT,YCENT,I,J,XSPAN,
     &                YSPAN)
          X          = FLOAT(I-XCENT)
          Y          = FLOAT(J-YCENT)
          RADIAL(K) = SQRT( X*X + Y*Y )
        ENDDO

        WRITE (UN2) XCENT,YCENT,AREA,MAJANG,
     &              NANGLS,(RADIAL(I),I=1,NANGLS)
```

```
C       Go back for another kernel.
        GOTO 100

C-----------------------------------------------------------------------

C       Come here when no more to be read
9998    CLOSE(UN1)
        CLOSE(UN2)

9999    CONTINUE

1       FORMAT(A)
2       FORMAT(A,I4,A)
        END
```

```
C*********************************************************************
C
C       MEASSV
C
C       VAX-11 FORTRAN SOURCE FILENAME:          MEASSV.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                     PROGRAMMER(S)
C       --------        ----                     -------------
C       1.0             JANUARY 15, 1985         TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985            TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN MEASSV
C
C       PURPOSE
C
C               Normalized measurements and writes results to disk.
C               Pre-normalized measurements speed up the classifier
C               programs.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               None
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C*********************************************************************
C
        IMPLICIT  NONE

        INTEGER   NANGMX,VARMX,METHMX,VIEWMX
        PARAMETER (NANGMX=720,VARMX=4,METHMX=3,VIEWMX=2)

        CHARACTER VARIETY(VARMX)*10,VIEW(VIEWMX)*3,METH(3)*1,FN*40
        INTEGER   IANG,IVAR,IMETH,IVIEW,NANGLS,XCENT,YCENT,AREA,I,UN(3)
        REAL      NORM(METHMX),MAJANG,RADIAL(NANGMX,2)

C       These are the I/O channels we'll use
        DO IMETH = 1, METHMX
          UN(IMETH) = 14 + IMETH
        ENDDO

C       The filenames of the contour data
```

```
      VARIETY(1) = 'ARKAN'
      VARIETY(2) = 'ARTHUR'
      VARIETY(3) = 'NUGAINES'
      VARIETY(4) = 'SAGE'

C     The filetypes of the contour data (top and side views)
      VIEW(1) = 'TM'
      VIEW(2) = 'SM'

C     Give the method identifiers
      METH(1) = ' '
      METH(2) = 'L'
      METH(3) = 'W'

C     Process all of the varieties
      DO IVAR = 1, VARMX

        WRITE (*,*) ' '
        WRITE (*,*) 'Processing variety ',VARIETY(IVAR)

        DO IVIEW = 1, VIEWMX

C         Figure out the file names for all views of the same variety
          FN = VARIETY(IVAR) // '.' // VIEW(IVIEW)
          OPEN(UN(1),FILE=FN,ERR=9000,STATUS='OLD',SHARED,
     &         FORM='UNFORMATTED')

          DO IMETH = 2, METHMX
            FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // METH(IMETH)
            OPEN(UN(IMETH),FILE=FN,ERR=9000,STATUS='NEW',SHARED,
     &           FORM='UNFORMATTED')
          ENDDO

C         Do the whole file, one kernel at a time
          DO WHILE (.TRUE.)

C           Get the data for the next kernel
            READ (UN(1),END=200) XCENT,YCENT,AREA,MAJANG,
     &                           NANGLS,(RADIAL(I,1),I=1,NANGLS)

C           Figure the norms
            NORM(2) = 1. / ( RADIAL(          1,1) +
     &                       RADIAL(NANGLS  /2+1,1) )
            NORM(3) = 1. / ( RADIAL(NANGLS  /4+1,1) +
     &                       RADIAL(NANGLS*3/4+1,1) )

C           Calculate the normalized coordinate along this axis
            DO IMETH = 2, METHMX
              DO IANG = 1, NANGLS
                RADIAL(IANG,2) = RADIAL(IANG,1) * NORM(IMETH)
              ENDDO
              WRITE (UN(IMETH)) XCENT,YCENT,AREA,MAJANG,
     &                          NANGLS,(RADIAL(IANG,2),IANG=1,NANGLS)
            ENDDO

          ENDDO
```

```
C           Close the files
200         DO IMETH = 2, METHMX
              CLOSE(UN(IMETH))
            ENDDO

          ENDDO
        ENDDO

9000    CONTINUE
        END
```

```
C*********************************************************************
C
C       PUTMEAN
C
C       VAX-11 FORTRAN SOURCE FILENAME:         PUTMEAN.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             JANUARY 15, 1985        TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN PUTMEAN
C
C       PURPOSE
C
C               Calculates mean curves for each variety and writes them
C               to disk to speed up the classifier programs.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               None
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C*********************************************************************
C
        IMPLICIT  NONE

        INTEGER   NANGMX,VARMX,METHMX,VIEWMX
        PARAMETER (NANGMX=720,VARMX=4,METHMX=3,VIEWMX=2)

        CHARACTER VARIETY(VARMX)*10,VIEW(VIEWMX)*3,FN*40
        INTEGER   UN,IANG,IVAR,IMETH,IVIEW,ICNT,
     &            NANGLS,XCENT,YCENT,AREA,I
        REAL      NORM(METHMX),RCNT,MEAN(NANGMX,VARMX,METHMX,VIEWMX),
     &            MAJANG,RADIAL(NANGMX)

C       The unit number we'll use for disk I/O
        UN = 15

C       These are the filenames of the contour data
        VARIETY(1) = 'ARKAN'
```

```
      VARIETY(2) = 'ARTHUR'
      VARIETY(3) = 'NUGAINES'
      VARIETY(4) = 'SAGE'

C     The filetypes of the contour data (top and side views)
      VIEW(1) = 'TM'
      VIEW(2) = 'SM'

C     Clear all of the mean contours
      OO IVIEW = 1, VIEWMX
        OO IMETH = 1, METHMX
          OO IVAR = 1, VARMX
            OO IANG = 1, NANGMX
              MEAN(IANG,IVAR,IMETH,IVIEW) = 0.
            ENOOO
          ENODO
        ENOOO
      ENOOO

C     Calculate a mean contour for each data file
      NORM(1) = 1.

C     For all views...
      OO IVIEW = 1, VIEWMX

C       And for all varieties...
        DO IVAR  = 1, VARMX

C         Construct the filename
          FN = VARIETY(IVAR) // '.' // VIEW(IVIEW)

C         Open the data file
          OPEN(UN,FILE=FN,ERR=9000,STATUS='OLO',SHAREO,
     &         FORM='UNFORMATTEO')

C         Count the number of kernel samples in the data file
          ICNT = 0

C         Now, for all of the data in the file...
          OO WHILE (.TRUE.)

C           Get the data for one sample
            REAO (UN,ENO=100) XCENT,YCENT,AREA,MAJANG,
     &                        NANGLS,(RAOIAL(I),I=1,NANGLS)

C           Increment count of samples in file
            ICNT = ICNT + 1

C           Some of the measurement methods require normalization
            NORM(2)= 1./(RAOIAL(         1)+RAOIAL(NANGLS/2  +1))
            NORM(3)= 1./(RADIAL(NANGLS/4+1)+RAOIAL(NANGLS*3/4+1))

C           Sum all measurements in the MEAN contour array
            OO IMETH = 1, METHMX
              OO IANG = 1, NANGLS
                MEAN(IANG,IVAR,IMETH,IVIEW) =
```

```
     &              MEAN(IANG,IVAR,IMETH,IVIEW) +
     &              RADIAL(IANG)*NORM(IMETH)
                 ENDDO
               ENDDO
             ENDDO

C            Now, divide by number of samples in file for true means
100          RCNT = FLOAT(ICNT)
             DO IMETH = 1, METHMX
               DO IANG = 1, NANGLS
                 MEAN(IANG,IVAR,IMETH,IVIEW) =
     &             MEAN(IANG,IVAR,IMETH,IVIEW) / RCNT
               ENDDO
             ENDDO

C            Close the data file
             CLOSE(UN)

C            Write the mean curves out to a file
             FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // 'M'
             OPEN(UN,FILE=FN,ERR=9000,STATUS='NEW',SHARED,
     &            FORM='UNFORMATTED')
             DO IMETH = 1, METHMX
               WRITE (UN) (MEAN(IANG,IVAR,IMETH,IVIEW),IANG=1,NANGMX)
             ENDDO
             CLOSE(UN)
           ENDDO
         ENDDO

C        At this time, we have mean contours for each data file.

9000     CONTINUE
         END
```

```
C***********************************************************************
C
C       EXCLASS
C
C       VAX-11 FORTRAN SOURCE FILENAME:          EXCLASS.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                    PROGRAMMER(S)
C       --------        ----                    -------------
C       1.0             JANUARY 15, 1985        TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985           TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C***********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN EXCLASS
C
C       PURPOSE
C
C               Performs an exhaustive search for a small set of
C               features that give good classification results with
C               a minimum distance classifier.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               TTYINC - Ralph's Character Input Routine
C               TTYINI - Ralph's Integer Input Routine
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C***********************************************************************
C
        IMPLICIT  NONE

        INTEGER   NANGMX,VARMX,METHMX,VIEWMX,DIMMX
        PARAMETER (NANGMX=720,VARMX=4,METHMX=3,VIEWMX=2,DIMMX=10)

        CHARACTER VARIETY(VARMX)*10,VIEW(VIEWMX)*3,FN*40,
     &            VSTR*4,MSTR*8,STRING*80,DUMMY*1,METH(METHMX)*1
        INTEGER   UN(METHMX,VIEWMX,VARMX),IANG,IVAR,IMETH,IVIEW,
     &            NANGLS,XCENT,YCENT,AREA,
     &            CORRECT(METHMX,VIEWMX,NANGMX),TOTPOS,
     &            WINNER,CLASSV(DIMMX),CLASSA(DIMMX),CLASSM(DIMMX),
     &            IDIM,IKERN,IMEAN,I,IERR,IVARPT,
     &            VARX,METHX,VIEWX,DIMX,
     &            BEST,BESTA,BESTV,BESTM,J
```

```
      REAL      MEAN(NANGMX,VARMX,METHMX,VIEWMX),
     &          MAJANG,RADIAL(NANGMX,METHMX,VIEWMX),THEME,
     &          DTEMP,ODISTSQ(VARMX,VARMX,100),DISTSQ(VARMX,NANGMX)

C     These need to be set sooner or later
      VARX  = VARMX
      METHX = METHMX
      VIEWX = VIEWMX
      DIMX  = DIMMX

C     Find out what our dimensions will be
79    CALL TTYINI('Enter number of varieties <4>:  ',VARX,IERR,0,4)
      IF (IERR.NE.0) VARX = VARMX
      IF (VARX.GT.VARMX .OR. VARX.LT.2) THEN
        WRITE (*,*) 'Number of varieties must be between 2 and',VARMX
        GOTO 79
      ENDIF
      WRITE (*,*) ' '

C     These are the filenames of the contour data
      IF (VARX.EQ.4) THEN
        VARIETY(1) = 'ARKAN'
        VARIETY(2) = 'ARTHUR'
        VARIETY(3) = 'NUGAINES'
        VARIETY(4) = 'SAGE'
      ELSE
        DO IVAR = 1, VARX
          WRITE (STRING,'(A,I1,A)') 'Enter variety name ',IVAR,':  $'
          I = INDEX(STRING,'$') - 1
          CALL TTYINC(STRING(1:I),VARIETY(IVAR),IERR,0,'?')
        ENDDO
        WRITE (*,*) ' '
      ENDIF

C-------------------------------------------------------------------

C     Here, we specify what measurements are to be used for each
C     dimension

89    CALL TTYINI('Enter number of dimensions:  ',DIMX,IERR,0,2)
      IF (IERR.NE.0) DIMX = 2
      IF (DIMX.GT.DIMMX .OR. DIMX.LT.1) THEN
        WRITE (*,*)'Number of dimensions must be between 1 and',DIMMX
        GOTO 89
      ENDIF
      WRITE (*,*) ' '

      DO IDIM = 1, DIMX

67      CALL TTYINC('Enter view (Top, Side):  ',DUMMY,
     &              IERR,0,'?')
        IF (IERR.NE.0) DUMMY = '?'
        IF (DUMMY.EQ.'T' .OR. DUMMY.EQ.'t') THEN
          CLASSV(IDIM) = 1
        ELSE IF (DUMMY.EQ.'S' .OR. DUMMY.EQ.'s') THEN
          CLASSV(IDIM) = 2
```

```
          ELSE
            WRITE (*,*) 'Invalid view'
            GOTO 67
          ENDIF

77        CALL TTYINC(
     &         'Enter norm method (Absolute, Length, Width):  ',
     &         DUMMY,IERR,0,'?')
          IF (IERR.NE.0) DUMMY = '?'
          IF (DUMMY.EQ.'A' .OR. DUMMY.EQ.'a') THEN
            CLASSM(IDIM) = 1
          ELSE IF (DUMMY.EQ.'L' .OR. DUMMY.EQ.'l') THEN
            CLASSM(IDIM) = 2
          ELSE IF (DUMMY.EQ.'W' .OR. DUMMY.EQ.'w') THEN
            CLASSM(IDIM) = 3
          ELSE
            WRITE (*,*) 'Invalid method'
            GOTO 77
          ENDIF

          WRITE (STRING,'(A,I3,A)')
     &         'Enter measurement index (1-',NANGMX,'):  $'
          I = INDEX(STRING,'$') - 1
87        CALL TTYINI(STRING(1:I),CLASSA(IDIM),IERR,0,0)
          IF (IERR.NE.0) CLASSA(IDIM) = 0
          IF (CLASSA(IDIM).LT.1 .OR. CLASSM(IDIM).GT.NANGMX) THEN
            WRITE (*,*) 'Index must be between 1 and',NANGMX
            GOTO 87
          ENDIF

          WRITE (*,*) ' '

        ENDDO

C       Document the dimensions used
        WRITE (*,*) DIMX,' dimensions used as classifiers:'
        DO IDIM = 1, DIMX

          IF (CLASSV(IDIM).EQ.1) THEN
            VSTR = 'top'
          ELSE
            VSTR = 'side'
          ENDIF

          IF (CLASSM(IDIM).EQ.1) THEN
            MSTR = 'absolute'
          ELSE IF (CLASSM(IDIM).EQ.2) THEN
            MSTR = 'length'
          ELSE
            MSTR = 'width'
          ENDIF

          WRITE (*,102) VSTR,MSTR,CLASSA(IDIM)
102       FORMAT(' view = ',A,', normalization = ',A,', index = ',I3)

        ENDDO
```

```
      WRITE (*,*) ' '
C-------------------------------------------------------------------

C     The filetypes of the contour data (top and side views)
      VIEW(1) = 'TM'
      VIEW(2) = 'SM'

C     The methods that we use
      METH(1) = ' '
      METH(2) = 'L'
      METH(3) = 'W'

C     The unit numbers we'll use for disk I/O
      I = 15
      DO IVAR = 1, VARX
        DO IVIEW = 1, VIEWX
          DO IMETH = 1, METHX
            UN(IMETH,IVIEW,IVAR) = I
            I = I + 1
          ENDDO
        ENDDO
      ENDDO

C     Fill all of the mean contours
      DO IVIEW = 1, VIEWX
        DO IVAR = 1, VARX
          FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // 'M'
          OPEN(UN(1,1,1),FILE=FN,ERR=9000,STATUS='OLD',SHARED,
     &         FORM='UNFORMATTED')
          DO IMETH = 1, METHX
            READ (UN(1,1,1))
     &           (MEAN(IANG,IVAR,IMETH,IVIEW),IANG=1,NANGMX)
          ENDDO
          CLOSE(UN(1,1,1))
        ENDDO
      ENDDO

C     At this time, we have mean contours for each data file

C-------------------------------------------------------------------

C     Now, let's try the classifier:

C     The number of kernels we have to classify
      TOTPOS = VARX * 100

C     Open the data files
      DO IVAR = 1, VARX
        DO IVIEW = 1, VIEWX
          DO IMETH = 1, METHX
            FN = VARIETY(IVAR) // '.' // VIEW(IVIEW)//METH(IMETH)
            OPEN(UN(IMETH,IVIEW,IVAR),FILE=FN,ERR=9000,STATUS='OLD',
     &           SHARED,FORM='UNFORMATTED')
          ENDDO
        ENDDO
```

```
      ENDDO

C     Clear all of the sum of distance-squared values
      DO IKERN = 1, 100
        DO IVAR = 1, VARX
          DO IVARPT = 1, VARX
            ODISTSQ(IVARPT,IVAR,IKERN) = 0.
          ENDDO
        ENDDO
      ENDDO

C     Keep track of our initial correctness
      CORRECT(1,1,1) = 0

C     Calculate the sum of distance-squared values to date
      DO IVAR = 1, VARX

        DO IKERN = 1, 100

          DO IVIEW = 1, VIEWX
            DO IMETH = 1, METHX
              READ (UN(IMETH,IVIEW,IVAR)) XCENT,YCENT,AREA,MAJANG,
     &             NANGLS,(RADIAL(I,IMETH,IVIEW),I=1,NANGLS)
            ENDDO
          ENDDO

          DO IDIM = 1, DIMX
            DO IVARPT = 1, VARX
              DTEMP = RADIAL(CLASSA(IDIM),CLASSM(IDIM),CLASSV(IDIM))
     &          - MEAN(CLASSA(IDIM),IVARPT,CLASSM(IDIM),CLASSV(IDIM))
              ODISTSQ(IVARPT,IVAR,IKERN) =
     &          ODISTSQ(IVARPT,IVAR,IKERN) + DTEMP*DTEMP
            ENDDO
          ENDDO

C         Choose closest mean pt as winner and check its
C         correctness
          WINNER = 1
          DO IVARPT = 2, VARX
            IF (ODISTSQ(IVARPT,IVAR,IKERN) .LT.
     &          ODISTSQ(WINNER,IVAR,IKERN)) WINNER = IVARPT
          ENDDO
          IF (WINNER.EQ.IVAR) CORRECT(1,1,1) = CORRECT(1,1,1) + 1

        ENDDO

C       Rewind the data files
        DO IVIEW = 1, VIEWX
          DO IMETH = 1, METHX
            REWIND(UN(IMETH,IVIEW,IVAR))
          ENDDO
        ENDDO

      ENDDO

      WRITE (*,*) 'Initial',CORRECT(1,1,1),' out of',TOTPOS,' for',
```

```
     &             FLDAT(CORRECT(1,1,1))/FLDAT(TOTPDS)*100.,'%'
        WRITE (*,*) ' '

C----------------

        BEST = CORRECT(1,1,1)
        DIMX = DIMX + 1

        DO WHILE (DIMX.LE.DIMMX)

          BESTA = D

C         Keep track of the number correctly classified
          DO IANG = 1, NANGMX
            DO IVIEW = 1, VIEWX
              DO IMETH = 1, METHX
                CDRRECT(IMETH,IVIEW,IANG) = D
              ENDDO
            ENDDD
          ENDDO

C         Process all of the varieties
          DO IVAR = 1, VARX

            DO IVIEW = 1, VIEWX
              DD IMETH = 1, METHX

C               Do the whole file, one kernel at a time
                DO IKERN = 1, 100

C                 Get the data for all views of the same kernel
                  READ (UN(IMETH,IVIEW,IVAR))XCENT,YCENT,AREA,MAJANG,
     &                 NANGLS,(RADIAL(I,IMETH,IVIEW),I=1,NANGLS)

                  DO IANG = 1, NANGMX

C                   Now, finish the summations
                    DO IVARPT = 1, VARX
                      DTEMP = RADIAL(IANG,IMETH,IVIEW)
     &                 - MEAN(IANG,IVARPT,IMETH,IVIEW)
                      DISTSQ(IVARPT,IANG) =
     &                                DDISTSQ(IVARPT,IVAR,IKERN) +
     &                                DTEMP*DTEMP
                    ENDDD

C                   Choose closest mean pt as winner and check its
C                   correctness
                    WINNER = 1
                    DO IVARPT = 2, VARX
                      IF (DISTSQ(IVARPT,IANG) .LT.
     &                    DISTSQ(WINNER,IANG))
     &                   WINNER=IVARPT
                    ENDDD
                    IF (WINNER.EQ.IVAR)
     &                CDRRECT(IMETH,IVIEW,IANG) =
     &                CORRECT(IMETH,IVIEW,IANG) + 1
```

```
                   ENDDO

                 ENDDO

C                Rewind the file
                 REWIND(UN(IMETH,IVIEW,IVAR))

               ENDDO
             ENDDO

           ENDDO

           DO IANG = 1, NANGMX
             I = 1
             J = 1
             DO IVIEW = 1, VIEWX
               DO IMETH = 1, METHX
                 IF (CORRECT(IMETH,IVIEW,IANG) .GT.
     &               CORRECT(I,J,IANG)) THEN
                   I = IMETH
                   J = IVIEW
                 ENDIF
               ENDDO
             ENDDO

C            WRITE (*,*) 'Angle',IANG,' gives ',
C    &         FLOAT(CORRECT(I,J,IANG))/FLOAT(TOTPOS)*100.,
C    &         '% for',J,I

             IF (CORRECT(I,J,IANG) .GT. BEST) THEN
               BESTV = J
               BESTM = I
               BESTA = IANG
               BEST  = CORRECT(I,J,IANG)
             ENDIF

           ENDDO

           IF (BESTA.NE.0) THEN
             CLASSV(DIMX) = BESTV
             CLASSM(DIMX) = BESTM
             CLASSA(DIMX) = BESTA

C            Update the ODISTSQ array
             DO IVAR = 1, VARX

C              Do the whole file, one kernel at a time
               DO IKERN = 1, 100

C                Get the data for all views of the same kernel
                 READ (UN(BESTM,BESTV,IVAR)) XCENT,YCENT,AREA,MAJANG,
     &                 NANGLS,(RADIAL(I,BESTM,BESTV),I=1,NANGLS)

C                Now, finish the summations
                 DO IVARPT = 1, VARX
```

```
              DTEMP = RADIAL(BESTA,BESTM,BESTV)
     &         - MEAN(BESTA,IVARPT,BESTM,BESTV)
              ODISTSQ(IVARPT,IVAR,IKERN) =
     &         ODISTSQ(IVARPT,IVAR,IKERN) + DTEMP*DTEMP
            ENDDO

          ENDDO

C         Rewind the file
          REWIND(UN(BESTM,BESTV,IVAR))

        ENDDO

        WRITE (*,*) 'Dimension',DIMX,' is:',BESTV,BESTM,BESTA
        WRITE (*,*) 'for',FLOAT(BEST)/FLOAT(TOTPOS)*100.,'%'

        WRITE (FN,'(A,I1,A)') 'DIM',DIMX,'.LOG'
        OPEN  (2,FILE=FN,ERR=9000,STATUS='NEW')
        WRITE (2,*) 'Dimension',DIMX,' is:',BESTV,BESTM,BESTA
        WRITE (2,*) 'for',FLOAT(BEST)/FLOAT(TOTPOS)*100.,'%'
        CLOSE (2)

        DIMX = DIMX + 1
      ELSE
        DIMX = DIMMX + 1
        WRITE (*,*) 'Stop due to no increase in accuracy'
      ENDIF

      ENDDO

      DO IVAR = 1, VARX
        DO IVIEW = 1, VIEWX
          DO IMETH = 1, METHX
            CLOSE(UN(IMETH,IVIEW,IVAR))
          ENDDO
        ENDDO
      ENDDO

9000  CONTINUE
      END
```

```
C***********************************************************************
C
C       MINCLASS
C
C       VAX-11 FORTRAN SOURCE FILENAME:          MINCLASS.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                     PROGRAMMER(S)
C       --------        ----                     -------------
C       1.0             JANUARY 28, 1985         TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985            TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C***********************************************************************
C
C       CALLING SEQUENCE
C
C               RUN MINCLASS
C
C       PURPOSE
C
C               An exhaustive nearest neighbor classifier.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               None
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C***********************************************************************
C
        IMPLICIT  NONE

        INTEGER   NANGMX,VARMX,METHMX,VIEWMX,TOTPOS
        PARAMETER (NANGMX=720,VARMX=4,METHMX=3,VIEWMX=2,TOTPOS=400)

        INTEGER   IVAR,IVIEW,IMETH,IANG,IKERN,I,CMPKRN,SELKRN,IFAVX,
     &            MAXCOR,CORRECT(NANGMX),IFAV,MATCH,BESTCOR,
     &            UN(METHMX,VIEWMX,VARMX),FAVLOG(3,10),XCENT,YCENT,
     &            AREA,NANGLS,ITMP1,ITMP2,IKE,ICK,ICKY,ICKF(4,2)
        REAL      MIND,DIST(TOTPOS),DTEMP,FAV(10,TOTPOS),MAJANG,
     &            RADIAL(TOTPOS,NANGMX),BUFFER(NANGMX)
        CHARACTER VARIETY(VARMX)*10,VIEW(VIEWMX)*3,METH(METHMX)*1,FN*40

        IKE = 100

C       Write a permanent record of it
        WRITE (FN,'(A,I3,A)') 'PART',IKE,'.LOG'
```

```
      IKE = IKE + 1
      OPEN  (3,FILE=FN,STATUS='NEW')

C     These are the filenames of the contour data
      VARIETY(1) = 'ARKAN'
      VARIETY(2) = 'ARTHUR'
      VARIETY(3) = 'NUGAINES'
      VARIETY(4) = 'SAGE'

C     The filetypes of the contour data (top and side views)
      VIEW(1) = 'TM'
      VIEW(2) = 'SM'

C     The methods that we use
      METH(1) = ' '
      METH(2) = 'L'
      METH(3) = 'W'

C     Open all of the files
      I = 15
      DO IVAR = 1, VARMX
        DO IVIEW = 1, VIEWMX
          DO IMETH = 1, METHMX
            FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // METH(IMETH)
            UN(IMETH,IVIEW,IVAR) = I
            OPEN(UN(IMETH,IVIEW,IVAR),FILE=FN,STATUS='OLD',SHARED,
     &           FORM='UNFORMATTED')
            I = I + 1
          ENDDO
        ENDDO
      ENDDO

C     This program takes a very long time to run.  Therefore, it was
C     necessary to run it in several chunks.  These variables that
C     start with ICK are initialized with historical data from
C     previous chunks.
      ICK = 2

      ICKF(1,1) = 2
      ICKF(2,1) = 3
      ICKF(3,1) = 339
      ICKF(4,1) = 259

      ICKF(1,2) = 2
      ICKF(2,2) = 1
      ICKF(3,2) = 536
      ICKF(4,2) = 318

C     The largest number of kernels classifier correctly so far is 0
C     (unless data from a previous chunk says otherwise)
      BESTCOR = 0

      IF (ICK.GT.0) THEN
        DO ICKY = 1, ICK
          DO IVAR = 1, VARMX
            ITMP1 = (IVAR-1)*100+1
```

```
              ITMP2 = ITMP1 + 99
              DO IKERN = ITMP1, ITMP2
                READ (UN(ICKF(2,ICKY),ICKF(1,ICKY),IVAR))
     &              XCENT,YCENT,AREA,MAJANG,
     &              NANGLS,(BUFFER(IANG),IANG=1,NANGLS)
                FAV(ICKY,IKERN) = BUFFER(ICKF(3,ICKY))
              ENDDO
              REWIND(UN(ICKF(2,ICKY),ICKF(1,ICKY),IVAR))
            ENDDO
          ENDDO
          BESTCOR = ICKF(4,ICK)
        ENDIF

C       Start up where we left off
        DO IFAVX = ICK, 9
          MAXCOR = 0
          DO IVIEW = 1, VIEWMX
            DO IMETH = 1, METHMX

C             Fill a large array with statistics
              WRITE (3,*) 'Reading another file'

              DO IVAR = 1, VARMX
                ITMP1 = (IVAR-1)*100+1
                ITMP2 = ITMP1 + 99
                DO IKERN = ITMP1, ITMP2
                  READ(UN(IMETH,IVIEW,IVAR)) XCENT,YCENT,AREA,MAJANG,
     &              NANGLS,(RADIAL(IKERN,IANG),IANG=1,NANGLS)
                ENDDO
                REWIND(UN(IMETH,IVIEW,IVAR))
              ENDDO

              DO IANG = 1, NANGMX
                CORRECT(IANG) = 0
              ENDDO

C             Enter a tight loop of comparisons and calculations
              WRITE (3,*) 'Entering the meat grinder'

              DO IANG = 1, NANGMX
                DO SELKRN = 1, TOTPOS
                  MIND = 1.E30
                  DO CMPKRN = 1, TOTPOS
                    DIST(CMPKRN) = 0.
                  ENDDO
                  IF (IFAVX.GT.0) THEN
                    DO CMPKRN = 1, TOTPOS
                      DO IFAV = 1, IFAVX
                        DTEMP = FAV(IFAV,SELKRN) - FAV(IFAV,CMPKRN)
                        DIST(CMPKRN) = DIST(CMPKRN) + DTEMP*DTEMP
                      ENDDO
                    ENDDO
                  ENDIF
                  DO CMPKRN = 1, TOTPOS
                    DTEMP = RADIAL(SELKRN,IANG) - RADIAL(CMPKRN,IANG)
                    DTEMP = DIST(CMPKRN) + DTEMP*DTEMP
```

```
                IF (DTEMP.LT.MIND .AND. CMPKRN.NE.SELKRN) THEN
                  MIND  = DTEMP
                  MATCH = CMPKRN
                ENDIF
              ENDDO
C
C             Check here for match of variety (match & selkrn)
C             and update correct vector (all iang)
              IF ((MATCH-1)/100 .EQ. (SELKRN-1)/100)
     &          CORRECT(IANG) = CORRECT(IANG) + 1
            ENDDO

            WRITE (3,*) IANG,CORRECT(IANG)
          ENDDO

C         Find max of correct vector and if .gt. maxcor, then
C         update maxcor & remember view & meth & angle
          IANG = 1
          DO I = 2, NANGMX
            IF (CORRECT(I).GT.CORRECT(IANG)) IANG = I
          ENDDO
          IF (CORRECT(IANG).GT.MAXCOR) THEN
            MAXCOR = CORRECT(IANG)
            DO IKERN = 1, TOTPOS
              FAV(IFAVX+1,IKERN) = RADIAL(IKERN,IANG)
            ENDDO
            FAVLOG(1,IFAVX+1) = IVIEW
            FAVLOG(2,IFAVX+1) = IMETH
            FAVLOG(3,IFAVX+1) = IANG
          ENDIF

C         Write a permanent record of it
          CLOSE (3)
          WRITE (FN,'(A,I3,A)') 'PART',IKE,'.LOG'
          IKE = IKE + 1
          OPEN  (3,FILE=FN,STATUS='NEW')

        ENDDO
      ENDDO
      IF (MAXCOR.LT.BESTCOR) THEN
        WRITE (3,*) 'Stop due to no increase in accuracy'
        GOTO 9080
      ELSE
        BESTCOR = MAXCOR

C       Print what we're updating our favorites array with
        WRITE (3,*) 'Favorite update:',(FAVLOG(I,IFAVX+1),I=1,3),
     &              BESTCOR

C       Write a permanent record of it
        WRITE (FN,'(A,I1,A)') 'FAV',IFAVX+1,'.LOG'
        OPEN  (2,FILE=FN,STATUS='NEW')
        WRITE (2,*) 'Favorite update:',(FAVLOG(I,IFAVX+1),I=1,3),
     &              BESTCOR,FLOAT(BESTCOR)/FLOAT(TOTPOS)*100.
        CLOSE (2)
```

```
        ENDIF
      ENDDO

9080  DO IVAR = 1, VARMX
        DO IVIEW = 1, VIEWMX
          DO IMETH = 1, METHMX
            CLOSE(UN(IMETH,IVIEW,IVAR))
          ENDDO
        ENDDO
      ENDDO

      CLOSE (3)

      END
```

```
C*******************************************************************
C
C       SCLASS
C
C       VAX-11 FORTRAN SOURCE FILENAME:          SCLASS.FOR
C
C       DEPARTMENT OF ELECTRICAL ENGINEERING     KANSAS STATE UNIVERSITY
C
C       REVISION        DATE                     PROGRAMMER(S)
C       --------        ----                     -------------
C       1.0             JANUARY 19, 1985         TERRY E. SCHMALZRIED
C       2.0             MARCH 8, 1985            TERRY E. SCHMALZRIED
C
C       COPYRIGHT 1985 TERRY E. SCHMALZRIED
C
C*******************************************************************
C
C       CALLING SEQUENCE
C
C               RUN SCLASS
C
C       PURPOSE
C
C               This is a hybrid, elective minimum distance classifier.
C               A whole vote is given to the winner of each pair-off.
C
C       ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C               None
C
C       ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C               None
C
C       ARGUMENT(S) SUPPLIED  TO  THE CALLING ROUTINE
C
C               None
C
C*******************************************************************
C
        IMPLICIT  NONE

        INTEGER   NANGMX,VARMX,METHMX,VIEWMX
        PARAMETER (NANGMX=720,VARMX=4,METHMX=3,VIEWMX=2)

        CHARACTER VARIETY(VARMX)*10,VIEW(VIEWMX)*3,HARD(VARMX)*1,
     &            METH(METHMX)*1,FN*40
        INTEGER   UN(METHMX,VIEWMX,VARMX),IMETH,IVIEW,IVAR,IANG,
     &            NANGLS,XCENT,YCENT,AREA,CORRECT,TOTPOS,IDIM,
     &            WINNER,CLASSN(6),CLASSV(5,6),CLASSA(5,6),CLASSM(5,6),
     &            ISEL,IKERN,I,J,K,L,M,N,C3,W3
        INTEGER*2 TALLY(4),TSOLN(4,4,4,4,4)
        REAL      MEAN(NANGMX,VARMX,METHMX,VIEWMX),DISTSQ(2),DTEMP,
     &            MAJANG,RADIAL(NANGMX,METHMX,VIEWMX)

C       A couple of unit numbers we'll use for disk I/O
```

```
      I = 15
      DO IVAR = 1, VARMX
        DO IVIEW = 1, VIEWMX
          DO IMETH = 1, METHMX
            UN(IMETH,IVIEW,IVAR) = I
            I = I + 1
          ENDDO
        ENDDO
      ENDDO

C     These are the filenames of the contour data
      VARIETY(1) = 'ARKAN'
      VARIETY(2) = 'ARTHUR'
      VARIETY(3) = 'NUGAINES'
      VARIETY(4) = 'SAGE'

C     The filetypes of the contour data (top and side views)
      VIEW(1) = 'TM'
      VIEW(2) = 'SM'

C     The methods that we use
      METH(1) = ' '
      METH(2) = 'L'
      METH(3) = 'W'

C     Description of the hardness of each of the varieties
      HARD(1) = 'H'
      HARD(2) = 'S'
      HARD(3) = 'S'
      HARD(4) = 'H'

C     Fill all of the mean contours
      DO IVIEW = 1, VIEWMX
        DO IVAR = 1, VARMX
          FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // 'M'
          OPEN(UN(1,1,1),FILE=FN,ERR=9000,STATUS='OLD',SHARED,
     &         FORM='UNFORMATTED')
          DO IMETH = 1, METHMX
            READ (UN(1,1,1))
     &          (MEAN(IANG,IVAR,IMETH,IVIEW),IANG=1,NANGMX)
          ENDDO
          CLOSE(UN(1,1,1))
        ENDDO
      ENDDO

C     At this time, we have mean contours for each data file

C-----------------------------------------------------------------

C     The next step is to do a minimum distance classifier for each
C     combination of 2 varieties.  A classifier will be run for all
C     normalizations and views of each combination and the single
C     best measurement method will be remembered for later use.

C     The comparison matrix is:
C
```

```
C               VARIETY
C        INDEX  1 2 3 4
C        -------------
C          1    * *
C          2    *   *
C          3    *     *
C          4      * *
C          5      *   *
C          6        * *

C 888    We have skipped that step for now and simply have put in the
C        results:

C        The number of dimensions for each comparison
         CLASSN(1) = 4
         CLASSN(2) = 5
         CLASSN(3) = 4
         CLASSN(4) = 2
         CLASSN(5) = 1
         CLASSN(6) = 4

C        The best view for each comparison
         CLASSV(1,1) = 1
         CLASSV(2,1) = 2
         CLASSV(3,1) = 1
         CLASSV(4,1) = 1

         CLASSV(1,2) = 2
         CLASSV(2,2) = 2
         CLASSV(3,2) = 1
         CLASSV(4,2) = 1
         CLASSV(5,2) = 2

         CLASSV(1,3) = 2
         CLASSV(2,3) = 2
         CLASSV(3,3) = 2
         CLASSV(4,3) = 2

         CLASSV(1,4) = 2
         CLASSV(2,4) = 1

         CLASSV(1,5) = 2

         CLASSV(1,6) = 2
         CLASSV(2,6) = 2
         CLASSV(3,6) = 2
         CLASSV(4,6) = 1

C        The best normalization method for each comparison is
         CLASSM(1,1) = 2
         CLASSM(2,1) = 3
         CLASSM(3,1) = 3
         CLASSM(4,1) = 2

         CLASSM(1,2) = 2
         CLASSM(2,2) = 2
```

```
      CLASSM(3,2) = 2
      CLASSM(4,2) = 3
      CLASSM(5,2) = 3

      CLASSM(1,3) = 3
      CLASSM(2,3) = 1
      CLASSM(3,3) = 2
      CLASSM(4,3) = 2

      CLASSM(1,4) = 3
      CLASSM(2,4) = 3

      CLASSM(1,5) = 3

      CLASSM(1,6) = 2
      CLASSM(2,6) = 3
      CLASSM(3,6) = 3
      CLASSM(4,6) = 3
C     The measurement index to do each comparison at is
      CLASSA(1,1) = 144
      CLASSA(2,1) =  65
      CLASSA(3,1) = 111
      CLASSA(4,1) = 380

      CLASSA(1,2) = 720
      CLASSA(2,2) = 276
      CLASSA(3,2) = 695
      CLASSA(4,2) = 434
      CLASSA(5,2) = 199

      CLASSA(1,3) = 344
      CLASSA(2,3) = 474
      CLASSA(3,3) = 154
      CLASSA(4,3) = 650

      CLASSA(1,4) = 697
      CLASSA(2,4) = 401

      CLASSA(1,5) = 713

      CLASSA(1,6) = 379
      CLASSA(2,6) = 235
      CLASSA(3,6) = 193
      CLASSA(4,6) =  95

C-----------------------------------------------------------------

C     Now that we have the knowledge that we think we need, let's
C     try the overall classification and see how smart we really are.

C     Initialize this array
      DO I = 1, 4
        DO J = 1, 4
          DO K = 1, 4
            DO L = 1, 4
```

```
              DO M = 1, 4
                TSOLN(M,L,K,J,I) = 0
              ENDDO
            ENDDO
          ENDDO
        ENDDO
      ENDDO

C     Keep track of how many we tried to classify
      TOTPOS  = 0

C     Open the data files
      DO IVAR = 1, VARMX
        DO IVIEW = 1, VIEWMX
          DO IMETH = 1, METHMX
            FN = VARIETY(IVAR) // '.' // VIEW(IVIEW) // METH(IMETH)
            OPEN(UN(IMETH,IVIEW,IVAR),FILE=FN,ERR=9000,STATUS='OLD',
     &           SHARED,FORM='UNFORMATTED')
          ENDDO
        ENDDO
      ENDDO

C     Keep track of the number correctly classified
      CORRECT = 0
      C3 = 0
      W3 = 0

C     Process all of the varieties
      DO IVAR = 1, 4

C       Tell the operator how we're progressing
        WRITE (*,*) 'Testing variety ',VARIETY(IVAR)

C       Do the whole file, one kernel at a time
        DO IKERN = 1, 100

C         Get all of the data available for this kernel
          DO IVIEW = 1, VIEWMX
            DO IMETH = 1, METHMX
              READ(UN(IMETH,IVIEW,IVAR)) XCENT,YCENT,AREA,MAJANG,
     &             NANGLS,(RADIAL(I,IMETH,IVIEW),I=1,NANGLS)
            ENDDO
          ENDDO

C         Keep an array that tells us how many times each is selected
          DO I = 1, 4
            TALLY(I) = 0
          ENDDO

C         This is the number of varieties we're dealing with
          N = 4

C         This is our index into the selection (or testing) matrix
          ISEL = 0

C         For all possible combinations of two varieties...
```

```
            DO I = 1, N-1
              DO J = I+1, N

C                Figure the selector into our classifier arrays
                 ISEL = ISEL + 1

C                Clear these variables for summation
                 DISTSQ(1) = 0.
                 DISTSQ(2) = 0.

C                Compute distances from here to means
                 DO IDIM = 1, CLASSN(ISEL)
                   DTEMP  = RADIAL(CLASSA(IDIM,ISEL),
     &                          CLASSM(IDIM,ISEL),CLASSV(IDIM,ISEL))
     &                  - MEAN  (CLASSA(IDIM,ISEL),I,
     &                          CLASSM(IDIM,ISEL),CLASSV(IDIM,ISEL))
                   DISTSQ(1) = DISTSQ(1) + DTEMP*DTEMP

                   DTEMP  = RADIAL(CLASSA(IDIM,ISEL),
     &                          CLASSM(IDIM,ISEL),CLASSV(IDIM,ISEL))
     &                  - MEAN  (CLASSA(IDIM,ISEL),J,
     &                          CLASSM(IDIM,ISEL),CLASSV(IDIM,ISEL))
                   DISTSQ(2) = DISTSQ(2) + DTEMP*DTEMP
                 ENDDO

C                Here, add 1.0 to TALLY(?) that is closer
                 IF (DISTSQ(1).LT.DISTSQ(2)) THEN
                   TALLY(I) = TALLY(I) + 1
                   IF (IVAR.EQ.J) WRITE (*,1080) IKERN,' failed ',I,J
1080               FORMAT(' ',I3,A,I1,':',I1)
                 ELSE
                   TALLY(J) = TALLY(J) + 1
                   IF (IVAR.EQ.I) WRITE (*,1080) IKERN,' failed ',I,J
                 ENDIF

              ENDDO
            ENDDO

C           Pick the largest as the winner
            WINNER = 1
            DO I = 2, 4
              IF (TALLY(I).GT.TALLY(WINNER)) WINNER = I
            ENDDO

C           See if we chose the right variety by checking it against
C           the filename
            IF (WINNER.EQ.IVAR) THEN
              CORRECT = CORRECT + 1
              IF (TALLY(WINNER).EQ.3) C3 = C3 + 1
            ELSE
              WRITE (*,1081) IKERN,(TALLY(I),I=1,4)
1081          FORMAT(' ',I3,5X,4I5)
              IF (TALLY(WINNER).EQ.3) W3 = W3 + 1
            ENDIF

C           Keep track of this stuff
```

```
          TSOLN(IVAR,TALLY(1)+1,TALLY(2)+1,TALLY(3)+1,TALLY(4)+1) =
     &    TSOLN(IVAR,TALLY(1)+1,TALLY(2)+1,TALLY(3)+1,TALLY(4)+1) + 1

C         Keep track of the number that we tried to classify
          TOTPOS = TOTPOS + 1

        ENDDO

200     DO IVIEW = 1, VIEWMX
          DO IMETH = 1, METHMX
            CLOSE(UN(IMETH,IVIEW,IVAR))
          ENDDO
        ENDDO

      ENDDO

C     Tell us our stats
      WRITE (*,*) ' '
      DO I = 1, 4
        DO J = 1, 4
          DO K = 1, 4
            DO L = 1, 4
              IF ((I+J+K+L-4).EQ.6) THEN
                WRITE (*,1090) I-1, J-1, K-1, L-1,
     &                         (TSOLN(M,I,J,K,L),M=1,4)
1090            FORMAT(' ',4I2,2X,4I4)
              ENDIF
            ENDDO
          ENDDO
        ENDDO
      ENDDO
      WRITE (*,*) ' '
      WRITE (*,*) CORRECT,' correct of',TOTPOS,' possible'
      WRITE (*,*) 'Percent correct is:',FLOAT(CORRECT)/FLOAT(TOTPOS)
      WRITE (*,*) 'Perfect 3.0s were:',C3,' right',W3,' wrong'

9000  CONTINUE
      END
```

CLASSIFICATION OF WHEAT KERNELS
BY MACHINE-VISION MEASUREMENT

by

TERRY EUGENE SCHMALZRIED

B. S., Kansas State University, 1983

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1985

It is not always possible for a grain inspector to correctly grade a shipment of wheat because many varieties are virtually indistinguishable from one another. But, it is important to the consumer that the wheat be classified correctly because of the vastly different milling and baking characteristics of some of these varieties. This paper describes a machine-vision system that was designed to analyze wheat and find ways to classify it on the kernel level.

The paper is divided into several sections. First, kernel characteristics used by grain inspectors to identify wheat varieties are discussed. Second, algorithms are explained for obtaining measurements of the wheat kernels, using those measurements to classify other wheat kernels, and selecting a minimal set of them to use as discriminators between varieties. Methods described include thresholding, object detection, edge tracking, internal region determination, insuring externality of the edge contour, centroids, and principal axes. Classification methods include minimum-distance, nearest-neighbor, and an elective variant of the minimum-distance classifier. The discriminator set was chosen through exhaustive comparison of each measurement's classification results. Finally, results and conclusions are given.

Four varieties of wheat were used to test the machine-vision system. Two of them, Arkan and Arthur, are often confused by grain inspectors because of their similiar appearance, although one is a hard wheat and the other is a soft wheat. Using a 400-

kernel sample set (100 kernels of each variety) and a multiple-dimensional classifier, 94% of the set was correctly classified by variety and 96.25% by hardness. A four-dimensional classifier properly classified 97.5% of a mixture of only Arkan and Arthur kernels. These percentages were determined using the same sample set to train the classifier and to test it, so the results are probably higher than they would be if a different test set were used. But, if additional features are considered, high classification rates could probably be achieved for even larger numbers of varieties. Additional features to consider are discussed at the end of the report.