Anomalous Shape Recognition Using Computer Vision

Chang-Sik Kim

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://repository.lsu.edu/gradschool_disstheses

Recommended Citation

https://repository.lsu.edu/gradschool_disstheses/8370

This Thesis is brought to you for free and open access by the Graduate School at LSU Scholarly Repository. It has been accepted for inclusion in LSU Historical Dissertations and Theses by an authorized administrator of LSU Scholarly Repository. For more information, please contact gradetd@lsu.edu.
A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanization College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Interdepartmental Program in Engineering Science

by

Chang-Sik Kim
B.S., Kangwon National University, 1994
M.S., Louisiana State University, 1999
May 2000
MANUSCRIPT THESES

Unpublished theses submitted for the Master’s and Doctor’s Degrees and deposited in the Louisiana State University Libraries are available for inspection. Use of any thesis is limited by the rights of the author. Bibliographical references may be noted, but passages may not be copied unless the author has given permission. Credit must be given in subsequent written and published work.

A library which borrows this thesis for use by its clientele is expected to make sure that the borrower is aware of the above restrictions.

LOUISIANA STATE UNIVERSITY LIBRARIES
I wish to express a deep feeling of gratitude to Dr. Malcolm E. Wright, Major Professor and Chairman of my doctoral program committee, for the opportunities, encouragement, support and patience that he provided during the entirety of my study and research program.

A great thanks to my graduate committee members: Dr. Jorge L. Aravena, Dr. Subhash C. Kak, Dr. J. Bush Jones, Dr. Mehdy Sabbaghian, and Dr. Warren N. Waggenspack Jr. A special appreciation to Dr. Mike Cannon, Dr. Christopher A. Clark, Dr. Don R. LaBonte, and Mr. Irv Daniel for the contribution that they provided in inspecting sweet potatoes for my experiments.

I am very grateful to Dr. Lalit R. Verma and all the staff and the faculty members of the Biological and Agricultural Engineering Department for their support. I am also grateful to Dr. Whoa S. Kang for his guidance and support throughout my whole graduate study and for opening the door of the agricultural engineering world for me.

I would like to dedicate this work to my parents and brother, without whose constant spiritual and intellectual support, encouragement and love, its realization would not have been possible.
# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ........................................................................... ii

**ABSTRACT** .............................................................................................. vi

**CHAPTER I. INTRODUCTION** ................................................................. 1

**CHAPTER II. LITERATURE REVIEW** ..................................................... 3

2.1 Shape Analysis Techniques ................................................................. 3

2.2 Application of Pattern Recognition Techniques to Agriculture .......... 12

**CHAPTER III. CURVATURE-ANGULAR DESCRIPTOR (CAD)** .............. 27

3.1 Introduction .......................................................................................... 27

3.2 Method .................................................................................................. 27

3.2.1 Partition of Input Curve Shape into Small Elements ...................... 28

3.2.2 Constructing the Series of Unit Vectors ($V_i$, i=1,...,k-1) and the Reference Vector $V_{ref}$ ................................................................. 30

3.2.3 Obtaining the Magnitude of the Cross Product $|V_i \times V_{ref}|$ and Normalizing the Locations of Unit vectors $V_i$ ............................................. 30

3.2.4 The Linear Regression Model .......................................................... 35

3.3 Relationship between the Linear Regression Model from the CAD and the Input Curve Shape .............................................................. 35

3.3.1 Shape Feature Value Selection ........................................................ 35

3.3.2 Effects of Sine Wave Noise on the Shape Feature Values ($r$, $SSE$, $slope$, and $concavity$) ................................................................. 40

3.4 Summary and Conclusions .................................................................. 50

**CHAPTER IV CLOSED-BOUNDARY SWEET POTATO SHAPE GRADING USING THE CURVATURE-ANGULAR DESCRIPTOR (CAD)** .......... 52

4.1 Introduction .......................................................................................... 52

4.2 Materials and Methods ........................................................................ 53

4.2.1 Image Processing System ................................................................. 53

4.2.2 Sweet Potato Shape Feature Extraction using the CAD ................ 53

4.2.2.1 Segmentation of the Object from the Background ..................... 55

4.2.2.2 The Closed Boundary Tracing of the Segmented Object ............. 59

4.2.2.3 Dividing the Closed-Boundary of a Sweet Potato into Two Curves .......................................................................................... 63

4.2.3 The Application of CAD to Extract Shape Features from Two Curves ......................................................................................... 63

4.2.4 Sweet potato shape feature extraction ............................................. 67

4.2.5 Learning Vector Quantization (LVQ) for Clustering of Feature Vectors ($SSE$, $Slope$, and $Concavity$) ............................................. 71
CHAPTER V SWEET POTATO SHAPE RECOGNITION USING STRUCTURED-LIGHT TECHNIQUES AND THE CURVATURE-ANGULAR DESCRIPTOR

5.1 Introduction
5.2 Materials and Methods
5.2.1 Sensing Environment
5.2.2 Quantitative Analysis of Degree of Ambiguity
5.2.2.1 Assumption
5.2.2.2 Basic formulas of a line in 3D
5.2.2.3 Computation in 3D space
5.2.3 Determination of Camera Standoff (Zc) and Camera Tilt Angle (θ)
5.2.4 Algorithm to Extract Stripe Patterns from Structured-Light Images
5.2.4.1 Thinning Algorithm
5.2.4.2 Thinned Stripe Extraction Algorithm
5.2.5 Selection of Surface Shape Features using the Curvature-Angular Descriptor (CAD) for Sweet Potato Shape Recognition
5.2.6 Exploration of the Relationship between Shape Feature Values and Shapes

5.3 Experimental Evaluations and Results
5.3.1 Separability Criterion
5.3.2 Reduction of the Number of Feature Vectors
5.3.3 Training LVQ Net and Comparison of its Performance with Human Inspectors

5.4 Summary and Conclusions

CHAPTER VI SUMMARY AND FURTHER RESEARCH

REFERENCES

APPENDIX A
Program 1. Main function
Program 2. Storing input image file names into array
Program 3. Retrieving input image from disk
Program 4. Segmentation of object
Program 5. Finding centroid of segmented object
Program 6. Tracing the boundary of object
Program 7. Dividing the extracted closed-boundary into two curves
Program 8. Partitioning input curve into small elements
Program 9. Constructing the series of unit vectors and reference vector and obtaining the length of cross product
Program 10. Normalizing the location of series of unit vectors and creating the linear regression model and extracting the feature vectors (SSE, slope, and concavity) ..................145
Program 11. Finding the point in input curve, which has maximum distance from the baseline ............................................146

APPENDIX B .................................................................................................................................148
Program 1. Main Function ...........................................................................................................148
Program 2. Image Processing Board Initialization ........................................................................151
Program 3. Scaling Down of Input Image .....................................................................................152
Program 4. Segmentation of Stripe Patterns ...............................................................................153
Program 5. Thinning Algorithm ................................................................................................153
Program 6. Computing the Degree of Pixels ..............................................................................156
Program 7. Stripe Extraction ......................................................................................................157
Program 8. Calculating the Shortest Distance between a point and a base line .........................160
Program 9. Partition of Extracted Stripe .....................................................................................160
Program 10. Obtaining the length of Cross Product ....................................................................162
Program 11. Linear Regression Model from CAD .................................................................163

VITA ...............................................................................................................................................165
ABSTRACT

In this research, a new method for anomalous shape recognition, named Curvature-Angular Descriptor (CAD), was written. The CAD generates shape feature vectors from a given curve shape. These vectors can be used to uniquely characterize the shape.

The sweet potato has one of the most irregular shapes of any fruit or vegetable. The Standards for Grades of sweet potato shapes (The Code of Federal Regulation, title 7 Agriculture, 1995) are very subjective. For this reason, the sweet potato was chosen as an anomalous shape for evaluating the CAD. The CAD was applied using two methods for sweet potato shape recognition.

In the first method (closed-boundary sweet potato shape grading), 407 sweet potatoes were randomly selected from several commercial sweet potato farms in Louisiana. These were inspected by four professional human inspectors and used to extract shape feature vectors using CAD. These extracted feature vectors were clustered using a Learning Vector Quantization (Kohonen, 1989) neural network. There was 27.23% average disagreement between inspectors. The method used in this application gave 24.39% average disagreement with each inspector. Based on experiments, this method achieved about the same ability as human inspectors within the subjective limits of human graders.
In the second method (sweet potato grading using low level three dimensional computer vision), 240 random sample sweet potato shapes were selected and inspected by five human inspectors. The LVQ net was trained using extracted shape feature vectors. The trained LVQ net was compared with the human inspectors. The average inspection agreement between each inspector was about 85.8%. On the other hand, the average inspection agreement between the trained LVQ net and inspectors was about 78%.

The anomalous shape description method developed in this study, the Curvature-Angular Descriptor, shows great potential as a method to distinguish irregular or defective sweet potato shapes and regular or non-defective sweet potato shapes. In its present form it can be used for research purposes to help determine the shape characteristics of new varieties and the effects on shape of cultural practices and disease. Further refinement in computation and hardware will lead to rapid commercial sorters.
CHAPTER I
INTRODUCTION

Agricultural products to be sold in fresh markets are to be free from damage and of a uniform shape and size. The sorting and grading of agricultural products by hand is a labor intensive and subjective task. Human sorting involves scanning the item and removing or separating defective or non-uniform items. Human grading is highly variable and difficult to evaluate.

Computer vision analysis has been applied in manufacturing industries for quality control, and is now being used in the food industry for the same purpose. There have been several commercial systems for sorting of agricultural product. An English company, Loctronic Graders Limited, produces a series of sorting machines that use image analysis to size white potatoes, or similar objects, by size or degree of blemishes. In one system, a human operator using a light pen on a video monitor selects the blemished potatoes. In another, a computer algorithm makes the decision. The user with an adjustment knob sets the blemish level required for rejection. The Decco Tiltbelt Pennwalt Company makes a line of packing house equipment that can be configured to color sort into six classes and weight sort into fourteen at 19,800 fruit per hour per lane. Each machine can include up to nine lanes. The decision making and recording electronics can control three machines simultaneously. Because the maximum number of drops is thirty, the full range of six
colors times fourteen weights is not realized. No sorting is done by shape.

The overall goal of this project was to develop a method for anomalous shape recognition. To accomplish this goal,

1) The curved-shape description method, named Curvature-Angular Descriptor (CAD), was developed in this project for anomalous shape recognition.

2) The shape descriptor was applied to closed-boundary sweet potato shape recognition for evaluation of the CAD.

3) The descriptor was also applied to low level 3 dimensional sweet potato shape recognition using a structured-light technique for evaluation of CAD.
2.1 Shape Analysis Techniques

Shape analysis methods play an important role in systems for object recognition, matching, registration, and analysis. This review is concentrated on shape description. Shape description refers to the methods that result in a numeric descriptor of a shape and is a step subsequent to shape representation. A shape description method generates a shape descriptor vector (also called a feature vector) from a given shape. The goal of description is to uniquely characterize the shape using its feature vectors. The required properties of a shape description method are invariance of translation, scale, and rotation. Much research for shape descriptions and algorithms for shape analysis has been reviewed and classified by Pavlidis (1978) and Loncaric (1998). The most frequent methods used in shape recognition are Fourier descriptors (FDs), moments, shape matrices and vectors, medial axis transform, string matching, and the autoregressive model.

- Fourier descriptor

The main idea of this method is to represent the boundary as a function of one variable $\phi(t)$, expand $\phi(t)$ in its Fourier series, and use the coefficients of the series as Fourier descriptors (FDs). A finite number of these FDs can be used to describe the shape.
Suppose we have a clockwise-oriented, simple closed curve $\gamma$ represented by the parameterized function

$$ Z(t) = [x(t), y(t)], \quad 0 \leq t \leq L $$

where $l$ is the arc length along $\gamma$. A point along the curve generates the complex function

$$ u(t) = x(t) + jy(t) $$

which is a function of period, $L$. The Fourier series expansion of $u(t)$ is given by

$$ u(t) = \sum_{n=-\infty}^{\infty} a_n e^{j(2\pi n / L)t} $$

The FDs are the coefficients $\{a_n\}$ defined by

$$ a_n = \frac{1}{L} \int_0^L u(t) e^{-j(2\pi n / L)t} dt. $$

Zahn and Roskies (1972) used this method for the analysis and synthesis of closed curves in a plane. A curve is represented parametrically as a function of arc length by the accumulated change in direction of the curve from the starting point. This function is expanded in a Fourier series and the coefficients are arranged in the amplitude/phase-angle form. Several examples are provided to indicate the usefulness of Fourier descriptors as features for shape discrimination and a number of interesting symmetric curves are generated by computer and plotted out. Persoon and Fu (1977) applied this method in character recognition and machine parts recognition. They assumed that the two-
dimensional shape to be described is represented as a sequence of \( m \) points. From the sequence of points, they define a sequence of unit vectors and a sequence of cumulative differences. The FDs are then defined with these two variables. Tao et al. (1995) used FDs to develop a method for shape grading of potatoes using machine vision for automated inspection. The relationship between object shape and its boundary spectrum values in the Fourier domain was explored for shape extraction. Heinemann et al. (1996) also used FDs to analyze the shape of potatoes with their automated inspection station for machine-vision grading of potatoes. Loncaric (1998) addressed the disadvantage of this method as it is impossible to "sense" local shape features and it is also computationally intensive. The problem of invariance to translation, scale, and rotation is also present in this approach.

\section*{Moments}

Moments were first used in mechanics for purposes other than shape description. The shape of boundary segments can be described quantitatively by using moments. The two-dimensional Cartesian moment \( m_{pq} \) of order \( p + q \) of the function \( f(x,y) \) is defined as

\[ m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) \, dx \, dy. \]

Moments \( m_{pq} \) are uniquely determined by the function \( f(x,y) \) and vice versa the moments \( m_{pq} \) are sufficient to accurately reconstruct the original function \( f(x,y) \). The zeroth order moment \( m_{00} \) is equal to the
shape area assuming that $f(x,y)$ is a silhouette function with value one within the shape and zero outside the shape. First order moments can be used to compute the coordinates of the center of the mass as $x_c = \frac{m_{10}}{m_{00}}$ and $y_c = \frac{m_{01}}{m_{00}}$. Second order moments are called moments of inertia and can be used to determine the principal axes of the shape. Principal axes are axes with respect to which there are maximum and minimum second-order moments. The moments defined by the equations above are not ideal for shape description because they are not invariant to translation, rotation, and scale. To overcome this difficulty, Hu (1962) proposed seven invariant moments. These moments do not depend on the position, origination, or scale of the shape. A generalization of moment transform to other transform kernels is possible by replacing a conventional transform kernel $x^p y^q$ by a more general form $P_p(x) P_q(y)$. Teague (1980) and Teh and Chin (1988) used orthogonal polynomials as moment transform kernels. Some of the orthogonal polynomial systems that have been investigated include Legendre, Zernike, and pseudo-Zernike polynomials. Loncaric (1998) addressed the advantage and disadvantage of the moment method. The advantage of moment methods is that they are mathematically concise. The disadvantage is that it is difficult to correlate high-order moments with shape features and local information and shape features cannot be detected.
Shape Matrices and Vectors

Shape matrix and vector approaches use global shape information to create a numerical (matrix or vector) description of the shape. Goshtasby (1985) used a matrix to represent the pixel values corresponding to a polar raster of coordinates centered in the shape center of a mass. A polar raster of concentric circles and radial lines is positioned in the shape center of the mass. The binary value of the shape is sampled at the intersections of the circles and radial lines. The shape matrix is formed so that the circles correspond to the matrix columns and the radial lines correspond to the matrix rows. This scheme is invariant to translation, rotation, and scaling. The maximum radius of the shape is equal to the radius of the circle centered in the shape of the mass that contains the shape.

Taza and Suen (1989) described shapes by means of shape matrices and a comparison of matrices was performed to classify unknown shapes into one of the known classes. A scheme for weighting matrix entries was developed for more objective comparison. The weighting was based on the fact that sampling density is not constant with the polar sampling raster. Without weighting the innermost shape samples are implicitly given much more importance than peripheral shape pixels because sampling density is much higher in the center of the shape.
Parui et al. (1986) proposed a shape description scheme based on the relative areas of the shape contained in concentric rings located in the shape center of the mass. Let $L$ be the maximum radius of the shape $S$ to be described. Let $T_k$ be the $k$th ring of $n$ concentric rings obtained by sectioning the maximum radius $L$ into $n$ equal segments. Note that

$S \subset \bigcup_{i=1}^{n} T_i$. Let

$$x = \frac{A(S \cap T_i)}{A(T_i)},$$

where $A(\cdot)$ is the function that returns the area of its argument. In other words, $x_i$ is the area of shape $S$ contained in ring $T_i$ relative to the area of the ring itself. The shape descriptor vector is formed as $x = [x_1 \ldots x_n]^T$. The author demonstrated that the shape vector scheme could be used for shape matching.

\section{Medial Axis Transform (MAT)}

The idea of this approach is to represent shape using a graph and hope that the important shape features are preserved in the graph. Blum (1967 and 1973) originally proposed this method. The purpose of the medial axis transform is to extract a skeleton, stick-like figure from the object. This figure can be used to represent and describe the object shape. The formation of a skeleton can be explained by the following example. Let the object interior be composed of a burnable dry grass while the object exterior is composed of unburnable wet grass. If fire is set simultaneously at all points of the shape boundary, it will propagate.
towards the center of the shape. However, at some points the wavefront from one direction will intersect the fire wavefront coming from one direction, thus extinguishing the fire. Points of wavefront collision are called quench points. The skeleton of the figure is defined as the set of quench points. It is possible to reconstruct the shape using its skeleton and a quench function. The quench function $q(x)$ at some point $x$ of the skeleton $S$ has a value equal to the radius of the circle touching at that point. The touching circle at some point has the center in that point and touches the boundary at least two other points. To reconstruct the object, one must position a disk of radius $q(x)$ at location $x$, for each $x \in S$. The union of all disks is equal to the original shape.

The disadvantage of this medial axis transform is that it is very sensitive to noise on the shape boundary. Even small changes in the shape can cause significant changes in the topology of the MAT graph. Much research has been done to overcome these difficulties and improve the MAT (Loncaric, 1998).

**String Matching Techniques**

A number of different algorithms for symbolic string matching have been proposed in the literature. In this section, the algorithm for string edit distance computation applied by Bunke and Buhler (1993) is briefly reviewed. This algorithm computes a measure of similarity, or distance, between two strings in terms of basic edit operations. The considered operations allow changing one symbol of a string into another
symbol, deleting one symbol from a string, or inserting a symbol into a string. There is a function that assigns a non-negative cost \( c(e) \) to each basic edit operation \( e \). The cost \( c(e) \) is usually related to the likelihood of the operation \( e \). The more likely \( e \) is to occur, the smaller is its cost \( c(e) \). Given two strings of symbols \( x = x_1 \ldots x_n \) and \( y = y_1 \ldots y_m \), their distance \( d(x,y) \) is defined as the minimum cost sequence of edit operations needed to transform \( x \) into \( y \). For any two strings we always have \( d(x,y) \geq 0 \). The smaller \( d(x,y) \) is, the greater is the similarity between \( x \) and \( y \).

The algorithm computes the element of a \((n+1) \times (m+1)\) matrix \( D(i,j) \) row by row from left to right for any given two strings \( x = x_1 \ldots x_n \) and \( y = y_1 \ldots y_m \). The first row and first column of \( D(i,j) \) are separately computed in an initial phase. The basic idea of the algorithm is to find a minimum cost path from \( D(0,0) \) to \( D(n,m) \). This path corresponds to the minimum cost sequence of edit operations transforming \( x \) into \( y \). In each element \( D(i,j) \) of the path, the minimum accumulative costs are stored transforming \( x' = x_1 \ldots x_i \) and \( y' = y_1 \ldots y_j \), i.e. \( D(i,j) = d(x',y') \). Hence, the lower right corner of the matrix, \( D(n,m) \), holds the value of \( d(i,j) \). For any path element \( D(i,j) \) there exists three potential processors, namely, \( D(i,j-1) \), \( D(i-1,j-1) \), and \( D(i-1,j) \). Going from \( D(i-1,j-1) \) to \( D(i,j) \) corresponds to the substitution of \( x_i \) and \( y_i \), while a step from \( D(i,j-1) \)
to \( D(i,j) \) represents the insertion of \( y_i \), and a step from \( D(i-1,j) \) to \( D(i,j) \) the insertion of \( x_i \). The pointer can be used to find, in a backward trace starting at \( D(n,m) \) and proceeding to \( D(0,0) \), the actual sequence of edit operations. Hence the algorithm not only gives the distance \( d(i,j) \) but also the way in which the symbols of \( x \) correspond with the symbols of \( y \).

Maes (1990 and 1991) presented this same algorithm and studied the several aspects of the use of string-matching techniques as an approach to the problem of recognizing and classifying polygons.

Ghazanfari and Irudayarai (1996a) applied the principle of cyclic string matching for shape classification of four varieties of pistachio nuts. The "shape" is defined as the external contour of a 2D image, and boundary sequences are used as a symbolic representation of the shape of individual nuts. Each boundary sequence is considered as a string consisting of the lengths of \( N \) angularly equispaced radii extending from the centroid of the shape to the boundary points.

\section*{Autoregressive Model}

Kashyap and Chellappa (1981) proposed the use of circular autoregressive models (CAR) for representation and reconstruction of closed boundaries. The CAR model is characterized by a set of unknown parameters and an independent noise sequence. Because the boundary is closed, the boundary 1-D function \( r_t \) is a periodic function. The particular CAR model that was used is the \( m \)-th order difference
equation: \( r_i = \alpha + \sum_{j=1}^{m} \theta_j r_{i-j} + \sqrt{\beta} \omega_i \), where \( \omega_i \) are independent random noise sources. The parameters \( \{\alpha, \theta_1, \ldots, \theta_m, \beta\} \) are unknown and need to be estimated. The model parameters are estimated by fitting the model to the observed time series \( \{r_1, \ldots, r_n\} \). The method of least squares is used to find the model parameters that minimize the expected value of the square error, \( \beta \). The estimated parameters of \( \theta_i \) are translation, rotation, and scale-invariant. Note that the rotation invariant holds only for angles that are multiples of \( 2\pi/N \). Parameters \( \alpha \) and \( \beta \) are not scale invariant, but the quote \( \frac{\alpha}{\sqrt{\beta}} \) is. Therefore, the vector \( \begin{bmatrix} \theta_1, \ldots, \theta_m, \frac{\alpha}{\sqrt{\beta}} \end{bmatrix}^T \) is used as a shape descriptor.

Dubois and Glanz (1986) used the same autoregressive model. The classification was performed by computing the weighted Euclidean distance between unknown object description and training objects.

2.2 Application of Pattern Recognition Techniques to Agriculture

One of the most exciting recent developments in electronics is the advent of computer vision which has opened a broad range of promising applications in agriculture: image analysis, robotic vision, and inspection. Perhaps the majority of image processing applications in agriculture may be categorized as machine vision inspection. Most involve quality verification, defect removal, and sorting and grading raw and processed food products. Machine vision inspection is already being
applied to a number of foods: *potatoes, corn kernels, apples, peaches, pistachio nuts, tomatoes, oysters, raisins*, and etc.

**Potatoes**

Wright et al (1986) developed an algorithm to calculate volumes and surface areas of sweet potatoes with a vision system. The system resolution was 240 by 320 by 64 gray-levels. Each sweet potato was suspended by its longitudinal axis in front of a camera and rotated in 10-degree increment. The mean difference for the calculated minus the measured volume was −1.92%. The range of differences was from −10.58% to +8.77%.

McClure and Morrow (1987) studied vision inspection of potatoes for size and shape. Various parameters to classify the shape of a potato were explored. The number of boundary points and the root-mean-square (RMS) were considered as the best separation parameters.

Tao et al. (1995) developed a Fourier-based shape separation method for shape grading of potatoes using machine vision for automated inspection. Their method was to extract the boundary of a potato based on radius, and express it in a one-dimensional boundary signature. The boundary signature was treated as a one-dimensional digital signal and translated to the Fourier domain. They defined the shape separator by multiplying the Fourier harmonics by their magnitudes. The separator achieved the transformation of shapes from visual evaluation to quantitative values. Based on their experiments, the
inspection system achieved 89% human agreement for shape grading of potatoes.

Tao et al. (1995) documented techniques for color defect detection, shape classification, and blemish inspection for grading of potatoes. The vision system achieved over 90% accuracy in inspection of potatoes and apples by representing features with hue histograms and applying multivariate discriminant techniques.

Deck et al. (1995) studied the relative strengths and weakness of the backpropagation neural network versus the Fisher discriminant function. Their performance was compared for machine vision inspection of greening, shape, and shatter bruise in two potato cultivars. For inspection of greening and shatter bruise, the hue (i.e., color) histogram was used to extract needed visual information relevant to greening and shatter bruise. For inspection of shape, they used the method of boundary radius and its Fourier transform to the spectrum domain. Based on their performance test, the backpropagation classifier achieved 73% accuracy for shape inspection where as the Fisher classifier did 68% accuracy.

Heinemann et al. (1996) developed a prototype inspection station based on the United States Department of Agriculture (USDA) inspection standards for potato grading. The primary task of their project was the integration of hardware and software. The station (i.e., hardware) consisted of an imaging chamber, conveyor, camera, sorting units, and
personal computer for image acquisition. For software, they used the potato-evaluation algorithm developed and tested by Tao et al. (1995). The classification accuracy ranged from 96%-98% for stationary potatoes, but only 77%-88% for moving potatoes, primarily because of +100% to −400% difference in the shape results between moving and stationary potatoes. Solving this problem was left for future work.

Marchant et al. (1990) developed a computer vision system for grading potatoes into size and shape categories and interfaced it with a grader to work at a speed of up to 40 potatoes per second. Twelve views were analyzed during the time the potatoes were in the field of view. Because all the measurements were inferred from the boundary, only the boundary data was stored and used for subsequent processing. They established an algorithm for splitting complex blobs (i.e. images consisting of two or more touching potatoes) to avoid the singulation of potatoes on the same row. Twelve strips (8 pixel wide) of slightly different parts of the surface were collected into memory. The strips were joined together forming a map of the surface. Colored patches of the map (diseased or green or black, etc) that were not sound tissue color were identified and their sizes (area) and shapes (how closely the shape conformed to an ellipse) measured. Potatoes were graded according to the number of defects, their sizes and shapes. Three processors then processed output from the blob splitting algorithm, each one working on data from one row of produce. After processing, produce attributes (e.g.
position, size, and blemish count) were collected and appropriate control signals sent to the grading mechanism. This multi-processing made it possible for the system to grade 40 potatoes per second. Marchant and Tillet (1994) modified this system increasing the speed of produce grading. They applied a transputer as a multiprocessing element having an ability to communicate with other transputers along high-speed serial links. Their system allowed the processing load to be distributed over a transputer network in an efficient manner to give a projected throughput of 80 objects per second.

Lefbvre et al. (1993) designed an automatic pulp sampling system for potatoes to detect viral disease. The results of the analysis of samples were used for the certification of potatoes to be marked as seed. One of their approaches was 3-D image analysis. This method performs the 3-D localization of a sprout by means of two images. The first one is used for the detection of the external protuberances (the sprout). After selection of the largest sprout, the potato is rotated 90 degrees to bring this sprout in front of the camera, providing the second image. By combining the information from these two images, the 3-D coordinates of the sprout are deduced. For the sprout detection from the images, they used several combined methods: thresholding, morphological opening followed by an exclusive-or (xor) operation, and measuring local variance of gray-levels on images.
**Corn Kernels**

Gunasekaran et al. (1987) developed an image algorithm for detecting stress cracks in corn kernels using a commercial vision system. White light in back-lighting mode with black-coated background with a small aperture for the light provided the best viewing conditions. The kernel images, when processed using the algorithm developed, produced white streaks corresponding to stress cracks. Double stress cracks were the easiest to detect. Careful positioning of the kernel over the lighting aperture was necessary for satisfactory detection of single and multiple stress cracks.

Liao et al. (1993) developed a machine-vision system to identify corn kernel breakage based on kernel shape profiles for automated corn quality inspection. The profile of a corn kernel was sampled into a sequence of one-dimensional digital signals based on its binary image. Shape parameters were selected by analyzing the kernel profile and were sent into a machine learning algorithm to train for a shape membership function of broken versus whole kernels. The system provided successful classifications of 99% and 96% for whole and broken kernels, and 89% and 94% for whole and broken round kernels, respectively.

Ding and Gunasekaran (1994) developed a computer-vision based feature extraction and classification system for corn kernels. For feature extraction, a statistical model-based feature extractor (SMB) and a multi-index active model-based (MAM) feature extractor were developed.
to improve the accuracy of classifications. For classification, a back-propagation neural network was applied as a multi-index classifier and minimum indeterminate zone (MIZ) classifiers were developed to speed up training. The results showed 91% average accuracy when the MAM feature extractor was used in conjunction with the MIZ classifier.

Ni et al. (1997a) developed a machine-vision system to identify different types of crown and shapes of corn kernels. Image processing techniques were used to enhance the object and reduce noise in the acquired image. Corn kernels were classified as convex or dent based on their crown end shape. Dent corn kernels were further classified into smooth dent or non-smooth dent kernels. A one-dimensional line profile analysis was used to obtain the needed three-dimensional information. The system provided an average accuracy of approximately 87% compared to human inspection. The processing time was between 1.5 and 1.8 s/kernel.

Ni et al. (1997b) designed and built a prototype machine vision system for automatically inspecting corn kernels. A strobe light was used to eliminate image blur due to the motion of corn kernels. Kernel singulation and the synchronization of strobe firing with the image acquisition were achieved by using optical sensors. Control circuitry was designed to enable synchronization of strobe firing to the vertical blanking period of the camera. Corn kernels of random orientation were inspected for whole versus broken percentages and on-line tests had
successful classification rates of 91% and 94% for whole and broken kernels, respectively.

Ni et al. (1998) developed an electronic corn kernel size grading system using machine vision. Measurements of kernel length, width, and projected area were obtained independent of kernel orientation. The performance of the size-grading algorithm was compared to the results of mechanical sieving using a precision round-hole seed sizer. Average accuracy from 73% to 90.3% was achieved on seed corn samples that were pre-sized using precision graders.

- **Apples and Peaches**

  Rehkugler and Throop (1986) developed and tested an apple handling and sorting device using machine vision for bruise detection and classification into USDA grades. A rotating cone and wheel mechanism oriented the fruit with the stem calyx axis in the vertical direction. The fruit was rotated 360 degree on a vertical axis spindle and viewed by a 64-pixel line scan camera. A digital image was captured and computer image data represented most of the surface of the apple. Gray level response to bruised tissue was represented by reduced image intensity. Image filtering, differentiating, binary image thresholding and measurement of the shape of the areas representative of bruises determined bruise patterns by using thinness ratios. The algorithm and system for bruise detection appeared to equal the accuracy
of manual bruise detection systems and was a consistent means of apple bruise detection.

Rehkugler and Throop (1989) presented an algorithm for processing gray level images from a line scan camera of near-infrared reflectance from an apple surface for bruised tissue detection. Clusters of black pixels (gray level = 0) in the resulting binary image that were representative of potential bruises were analyzed to determine their size and shape. If the shape of the cluster was nearly circular, it was assumed to be a bruise. From this information, the amount of bruise area on the fruit was determined, and the fruit graded. Gray level images of other defects including scab, bird pecks, russeting, hail damage and cuts were determined. Scab, bird pecks and hail damage could be discriminated in the line scan image by the lower gray level of these areas in the unprocessed image.

Miller and Delwiche (1991) developed a laboratory machine vision system to detect and identify surface defects (scar, cuts, bruise, scale, wormholes, and brown rot) on fresh market peaches. Image analysis algorithms were developed for segmenting defect regions in the peach images, and a classifier identified the segmented regions as specific defect types. Classifier performance in identifying each segmented region as a member of one of eight classes (scar, stem cavity, cut, bruise, scale, wormholes, brown rot, and noise) had a 31% error rate for the near-infrared system and a 40% error rate for the color system.
Yang (1993) proposed a neural network approach to perform classification of apple surface features using machine vision. Three major apple surface features were considered, namely non-defective area, patch-like blemish, and elongated blemish. For each view of an apple, both the image under normal diffuse light and the image with structured light were preprocessed to extract the features as the input to the neural network. The feasibility of the proposed system was examined with experiments using defective apples. An average classification accuracy of 96.6% was achieved for the test samples.

Singh and Delwiche (1994) developed a machine vision system consisting of an illumination chamber, monochrome camera with a near infrared band-pass filter, frame grabber, and microcomputer. Defect segmentation methods were developed using gray level histograms of images that consisted of peach and defect pixels. The feature extraction techniques were simple, fast, and well suited for pipeline image processing hardware because they used raster scans. Tests were conducted to study the performance of the machine vision system at detecting and identifying major defects. The overall classification error in identifying peaches with major defects (cut, scar, bruise, and wormholes) was 28.6%. Errors were primarily due to natural variability in the features.

Yang (1996) presented an image analysis technique for the identification of apple stems and calyxes. Because apple stems and
calyx areas appear as dark patches in images, the analysis focused on the dark patches of fruit surfaces. The patches were first segmented by a flooding algorithm. To distinguish stems and calyx areas from patch-like blemishes, the three-dimensional shape of an apple surface was obtained by a structured light technique. For each patch, the characteristic features were extracted from both the image under normal diffused light and the image with structured light. With these features, back-propagation neural networks were used to classify each patch as stem/calyx or patch-like blemish. The proposed technique tested with sample apples yielded an average identification accuracy of 95%.

Crowe and Delwiche (1996a, b) proposed simultaneous color evaluation and defect detection using three cameras to sense reflectance in the visible region and narrow bands in the near infrared (NIR) region. Information from the visible region was intended for use in color grading. A narrow band centered at 780 nm allowed concavity identification with structured illumination, and a second band centered at 750 nm was used for detection of dark spots under diffuse illumination. Hardware for fruit handling and image acquisition was developed based on these concepts. The system included a single-lane roller conveyor, interface electronics, cameras, lamps, and laser line generators. They also developed and implemented an algorithm to acquire and analyze two combined near infrared (NIR) images of each fruit in real-time with a pipeline image processing system. Information from the structured
illumination portion of each image was used to distinguish between
defects and concavities which both appeared as dark spots in the
diffusely illuminated scene. The total projected area of defects on each
fruit was estimated, and subsequent classification was based on the
defect pixel total. The classification error rates for bruise, crack, and
cut apple classes were 38%, 38%, and 33%, respectively. The error rates
for bruise, scar, and cut peaches were 9%, 3%, and 30%, respectively.

Pistachio Nuts

Ghazanfari and Irudayaraj (1996a) developed a modified cyclic
string-matching algorithm and applied this method to classify four
classes of pistachio nuts, based on their two-dimensional shapes. A
string consisting of N angularly equispaced radii extending from the
centroid to the boundary represented a pistachio nut shape. The
recognition algorithm was developed calculating the cumulative distance
between an unknown and a class prototype. The class of the unknown
was determined by the minimum-distance classification rule. The
algorithm they developed gave an overall accuracy of 90%.

Pearson and Slaughter (1996) used computer vision to detect early
split lesions on the hull of pistachio nuts. Gray scale intensity profiles
were computed across the width of the nut (perpendicular to the suture
along the longitudinal axis). If the profile crossed an early split lesion,
a deep and narrow valley on the profile at the early location was
observed. Profiles were computed every 0.5 mm along the longitudinal
axis of the nut and the number of adjacent profiles with deep and narrow valleys was recorded. Early split nuts contained a significantly higher count of these adjacent profiles than normal nuts. Combining unhulled nut cross-sectional area with the adjacent profile data, 100% of the early split nuts and 99% of the normal nuts were correctly classified of the total of 180 nuts tested.

Ghazanfari et al. (1996b) proposed a multi-structure neural network (MSNN) applied to classify four varieties (classes) of pistachio nuts. The MSNN classifier consisted of four parallel discriminators (one per class), followed by a maximum selector. Each discriminator was a feed-forward neural network with two hidden layers and a single-neuron output layer. The discriminators were individually trained using physical attributes of the nuts extracted from their images as input. The performance of the MSNN classifier was compared with the performance of a multi-layer feed-forward neural network (MLNN) classifier. The average classification accuracy of the MSNN classifier was 95.9%, an increase of over 8.9% of the performance of MLNN.

Other Products (Tomatoes, Oysters, Raisins, and Cantaloupes)

Sarkar and Wolfe (1985a and b) developed an algorithm for digital image analysis and pattern recognition techniques for orientation of fresh market tomatoes and classification based on size, shape, color and surface defects. They developed a machine vision based system for tomatoes with the analysis of stem and blossom end views as features
indicative of fresh market quality. Suitable illumination and orientation
techniques were designed. Evaluation of the complete tomato sorting
system showed an error rate of 3.5%. Further studies demonstrated the
flexibility of this system in implementing various grading schemes
according to the needs of the marketplace. The feasibility of obtaining
the sorting speeds required for commercial application was explored.

Koslav et al. (1989) described the development of a grading and
sorting machine for raw oyster meat. The machine uses a vision system
to measure the projected area and then sort meats into three size
categories based on volume. The machine sorted with an accuracy of
88%. Due to equipment limitations, the machine sorted at a rate of one
oyster per two seconds.

Okamura et al. (1993) developed a machine vision system for
grading raisins. Raisin maturity is mainly based on visual features such
as degree of wrinkles and shape. The features used in image analysis
were wrinkle edge density, average gradient magnitude, angularity, and
elongation. A Bays classifier was used to separate the raisins into three
grades: B or better, C, and substandard. The machine vision system was
evaluated by comparing the grading results with those of the industry
standard air-stream sorter, sight graders, and a panel of sight graders
who graded by consensus. Four lots of raisins were used in the
experimental tests: high quality, medium quality, low quality natural
condition, and low quality reconditioned. Panel sight grading was
assumed to give the "true" grade. The sight grading results were the most accurate in terms of panel sight grading. Machine vision accuracy was comparable to the air-stream sorter accuracy. Air-stream sorting had the lowest variability in grading results, followed in order of increasing variability by machine vision, panel sight grading, and sight grading.

Ozer et al. (1995) presented an approach to automated fruit sorting using information that is acquired from selected sensors which measure and quantify parameters (color, firmness, size, weight) that are indicators of fruit quality. The information was used as input to a recurrent autoassociative memory that classified the fruits into four maturity stages. The method achieved good results in the classification of cantaloupes (85.1% correct classification and 10.1% neighbor class classification), and had the advantages of rapid training using small numbers of samples and classification capability for different cultivars of cantaloupes without additional training.
CHAPTER III
CURVATURE-ANGULAR DESCRIPTOR (CAD)

3.1 Introduction

The objective of the work described in this chapter was to derive the Curvature-Angular Descriptor (CAD) and understand how this method could be used for recognition of anomalous shapes such as sweet potatoes. The CAD is a curved-shape description method. The CAD generates shape feature vectors from a given curve shape. These vectors can be used to uniquely characterize the shape. These feature vectors have properties of translation, scale, and rotation invariance. These properties are required because these three transformations, by definition, do not change the shape of the object.

3.2 Method

Input to this method is an array containing the x and y coordinates making up a curve shape in image coordinates. The input array is shown in figure 3.1. The method has four stages: 1) partitioning of input curve shape into small elements; 2) constructing the series of vectors \( (V_i, i = 1, ..., n-1) \) and a reference vector \( V_{ref.} \); 3) obtaining the length cross product \( |V_i \times V_{ref.}| \) and normalizing the locations of \( V_i \); and 4) constructing the linear regression model with the normalized locations of \( V_i \) (dependent variable) and the lengths of cross products \( |V_i \times V_{ref.}| \) (independent variable).
3.2.1 Partition of Input Curve Shape into Small Elements

The *Iterative Endpoint Fit & Split* algorithm was used for partition of input curves (Haralick and Shapiro, 1993).

Step(1): Draw the base line between first point and last point in the input array as shown in figure 3.1.

Step(2): Obtain the shortest distance between each point in the input array and the base line. The shortest distance can be calculated as follows: Three points, \( P_0 \), \( P_1 \), and \( Q \) are given. Points \( P_0 \) and \( P_1 \) form the base line and the point \( Q \) is one of the points in the input array shown in figure 3.1. A vector \( V \) from \( P_0 \) to \( P_1 \) and an another vector from \( P_0 \) to \( Q \) \( (P_0Q = Q-P_0) \) are defined as in figures 3.2 and 3.3. A parallelogram is defined by these two vectors \( V \) and \( P_0Q \) as in figure 3.3. The height of this parallelogram is equivalent to the desired shortest distance “\( d \)” from the base line to points in the input array. The area of the parallelogram is equal to the length of its base line times its height, i.e. \( |V| \times d \). Using the properties of vectors, it can also be shown that the area of the parallelogram is equal to the magnitude of the cross product of the vectors \( V \) and \( P_0Q \).

\[
\text{Area} = |V \times (Q - P_0)| = |V| |Q - P_0| \sin(\theta) = |V| d
\]

Thus, the distance “\( d \)” can be obtained as

\[
d = \frac{\text{Area}}{\text{base line}} = |Q - P_0| \sin(\theta) = \frac{|V \times (Q - P_0)|}{|V|}.
\]
Figure 3.1 Input array to CAD, consisting of X-Y coordinates. Input Array(k) = (x_k, y_k), where k=1,2,...,n = Input Sequence.

Figure 3.2 P0 and P1 are the first and the last points in the input array sequence and Q can be any point in the input array sequence.
Step(3): Choose the point that has the maximum distance from the base line among the points in the input array sequence as shown in figure 3.4. This splits the curve.

Step(4): Repeat step (2) and (3) to obtain two divided curve elements (see figure 3.5). Repeat steps (2) and (3) until the maximum distance from the base line is less than a prearranged small tolerance $d_t$. This partition process yields the sequence of points, which divide the input curve into small elements represented as a line (figure 3.6).

3.2.2 Constructing the Series of Unit Vectors $(V_i, i = 1, ..., k-1)$ and the Reference Vector $V_{ref}$.

The series of unit vectors $(V_i, i = 1, ..., k-1)$ using the sequences obtained from the previous partition process are obtained from (see figure 3.7)

$$V_i = \frac{(x_{i+1} - x_i, y_{i+1} - y_i)}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}},$$

where $i = 1, ..., k-1$ and the number of out sequences from the partition process is $k$. The reference vector $V_{ref}$ is created using the first and the last points from the input array as (see figure 3.7):

$$V_{ref} = \frac{(x_k - x_1, y_k - y_1)}{\sqrt{(x_k - x_1)^2 + (y_k - y_1)^2}}.$$

3.2.3 Obtaining the Magnitude of the Cross Product $|V_i \times V_{ref}|$ and Normalizing the Locations of $V_i$

Assuming the original point of the x-y coordinates is the initial point of the reference vector $V_{ref}$ and the direction of the reference
Figure 3.3 The parallelogram defined by two vectors, \( \mathbf{V} \) and \( \mathbf{Q-P_0} \).

Figure 3.4 The point which has a maximum distance from the base line splits the input curve into two elements.
vector is the same as the x-axis direction, all of the members of the series of unit vector $V_i$ can be rearranged as shown in figure 3.8 using the concept of a sliding vector, that is, a vector whose initial point can be any point on a straight line that is parallel to the vector.

The length of cross product of $V_{\text{ref.}}$ and $V_i$ is the sine of the angle between two vectors because all the vectors are unit vectors ($|V_{\text{ref.}}|=1$ and $|V_i|=1$).

$$|V_{\text{ref.}} \times V_i| = |V_{\text{ref.}}||V_i|\sin(\theta),$$

where $\theta$ is the angle between $V_{\text{ref.}}$ and $V_i$.

Suppose we have $n$ number of points in an input array as shown in figure 3.6. We also have $k$ number of locations ($\text{location}(i), i = 1, \ldots, k$) dividing the input curve into small segments as shown in figure 3.6. The normalized location of $V_i$ ($\text{normalized \_ \_ location}(j), j = 1, \ldots, k - 1$) is obtained as

$$\text{normalized \_ \_ location}(1) = \left(\frac{\text{location}(1) + \text{location}(2)}{2 \times \frac{n}{2}}\right) \times 2 - 1$$

$$\text{normalized \_ \_ location}(2) = \left(\frac{\text{location}(2) + \text{location}(3)}{2n}\right) \times 2 - 1$$

$$\vdots$$

$$\text{normalized \_ \_ location}(k - 1) = \left(\frac{\text{location}(k - 1) + \text{location}(k)}{2n}\right) \times 2 - 1$$
Figure 3.5 Input curve partition process. The input curve is split until all split curve elements can be represented as line segment.

Input Array

<table>
<thead>
<tr>
<th>(x_1, y_1)</th>
<th>(x_2, y_2)</th>
<th>(x_3, y_3)</th>
<th>...</th>
<th>(x_{n-1}, y_{n-1})</th>
<th>(x_n, y_n)</th>
</tr>
</thead>
</table>

Locations in the Input Array dividing the input curve into small elements

1
4
n-3
n

Figure 3.6 The location dividing the input curve into small segments in Input Array.
\[ \vec{V}_i = \frac{(x_{it1} - x_i, y_{it1} - y_i)}{\sqrt{(x_{it1} - x_i)^2 + (y_{it1} - y_i)^2}} \]

where, \( i = 1,2,3,...,k-2,k-1 \)

Figure 3.7 The series of unit vectors \((V_i, i=1,2,3,...,k-2,k-1)\) and the reference unit vector \(V_{\text{ref}}\).

\[ \vec{V}_{\text{ref}} = \frac{(x_k - x_1, y_k - y_1)}{\sqrt{(x_k - x_1)^2 + (y_k - y_1)^2}} \]

Figure 3.8 The rearrangement of the location of the series of unit vectors \(V_i\) by the concept of sliding vectors.
3.2.4 The Linear Regression Model

The linear regression model with the normalized location of $V_i$ as an independent variable ($x$) and the magnitude of the cross product $|V_{ref} \times V_i|$ as a dependent variable ($y$) is defined as

$$\mu_{y|x} = \beta_0 + \beta_1 x$$

where $\beta_1$ is the slope of the regression line, that is, the change in $y$ corresponding to a unit change in $x$ and $\beta_0$ is the intercept, that is, the value of the line when $x=0$. Note also that $\mu_{y|x}$ is an estimate of the $y$ for any given $x$.

3.3 Relationship between the Linear Regression Model from the CAD and the Input Curve Shape

3.3.1 Shape Feature Value Selection

Seven half-elliptic curves were created using the elliptic equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

with variation of the constant $b$ (40, 35, 30, ..., 20, 15, 10) and a fixed constant $a$ (40) as shown in figure 3.9. The output from CAD and its linear regression model for these curves was plotted (figure 3.10 and figure 3.11).

Four feature values are explored from these outputs as follows:

1. correlation coefficient ($r$)

$$r = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{(x-\bar{x})^2(y-\bar{y})^2}}$$
where \( x, y \) are independent variables and dependent variables, respectively, and \( \bar{x}, \bar{y} \) are mean values of \( x \) and \( y \), respectively. This is the measurement of linear relationship between two variables. A correlation of 1 signifies an exact linear relationship between two variables and the correlation of 0 indicates that no relation exist between the two variables.

2. **error or residual sum of squared (SSE)**

\[
SSE = \sum (y - \mu_{jx})^2 = \sum (y - \beta_0 - \beta_1 x)^2.
\]

This is another criterion to measure the strength of the linear relationship between the two variables. This quantity describes the variation in \( y \) after estimating the linear relationship of \( y \) to \( x \).

3. **slope**

\[
slope = |\beta_1|
\]

This is the absolute value of the estimated slope, which is a measure of the change in the mean of the dependent variable (\( y \)) (the magnitude of the cross product \( |V_{ref. \times V_i}| \)) for a unit change in the independent variable (\( x \)) (the normalized location of \( V_i \)).

4. **concavity**

\[
C(i) = |V_{ref. \times V_{i+1}}| - |V_{ref. \times V_i}|
\]

\[
C(i) = \begin{cases} 
0, & \text{if } C(i) \geq 0 \\
|C(i)|, & \text{if } C(i) < 0 
\end{cases}
\]

\[\text{concavity} = \sum_i C(i)\]
Figure 3.9 Half-elliptic curves created using elliptic equation,
\[
x(t) = \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1
\]

where \(i = 1, 2, 3, \ldots, 79, 80, 81\).
Figure 3.10 CAD output and linear regression model of half-elliptic curves in figure 3.9.
Figure 3.11 CAD output and linear regression model of half-elliptic curves in figure 3.9.
This is a measurement of the smoothness of the input curve shape. A concavity of 0 signifies smoothness of the input curve shape. If the input curve shape is a regular convex shape, the dependent variable \( \left| \mathbf{V}_{\text{ref}} \times \mathbf{V}_i \right| \) increases as a function of independent variable (location of the series of unit vectors, \( V_i \)). On the other hand, if the input curve is concave, the dependent variable decreases in the location of where the concavity exits on the input curve.

These four feature values of the seven half-elliptic curves in figure 3.9 are tabulated in table 3.1 and plotted in figure 3.12. It is noted that:

1) If the curve shape is half-circle (b/a=1), the correlation coefficient (r) and the slope are close to 1 and the residual sum of squares (SSE) is close to 0.

2) As the b/a decreases from 1 to 0.25, the slope decreases, the residual sum of squares (SSE) increases, and the correlation coefficient (r) slightly decreases.

3) All the concavities are zeros.

3.3.2 Effects of Sine Wave Noise on the Shape Feature Values (r, SSE, slope, and concavity)

The relationship between different curve shapes and feature values are explored in this section. The noise-added curve, h(t), is created by adding a sine wave, g(t), to a half-elliptic curve, f(t), as shown in figure 3.13. The different shapes of the noise-added curve, h(t), can be
created by varying the value of $N_{freq}$ and $N_{scale}$, where $N_{freq}$ and $N_{scale}$ correspond to the frequency and the amplitude of noises on the curve, $h(t)$, respectively. The frequency of the noise on the curve, $h(t)$, increases as the value of $N_{freq}$ increases as shown in figure 3.14. The amplitude of the noise also increases as the value of $N_{scale}$ increases as shown in figure 3.15.

The shape feature values ($r$, $SSE$, slope, and concavity) of noise-added curves were calculated as a function of $N_{freq}$ and $N_{scale}$. These four values were plotted in figures 3.16, 3.17, 3.18, and 3.19, respectively. It can be noted that:

1) The correlation coefficient ($r$) is relatively higher than the one with high frequency ($N_{freq}$) and amplitude ($N_{scale}$) of the noise. On the other hand, it is relatively lower than the one with low $N_{freq}$ and $N_{scale}$.

2) The residual sum of squares ($SSE$) is relatively lower than the one with low $N_{freq}$ and $N_{scale}$. On the other hand, it is relatively higher than the one with high $N_{freq}$ and $N_{scale}$.

3) For a noise-free curve shape, the slope decreases as the constant ratio $(b/a)$ in half-elliptic equation decreases. However, as the $N_{freq}$ and $N_{scale}$ increase, the slope decreases for a noise-added curve shape.

4) The concavity is relatively low with low $N_{freq}$ and $N_{scale}$, on the other hand, it is relatively high with high $N_{freq}$ and $N_{scale}$. However, the
Table 3.1 Feature values of seven half-elliptic curves in figure 3.9.

| b/a | corr. coef. | SSE error sum of squares | slope $|\beta_1|$ | concavity $C(i)$ |
|-----|-------------|--------------------------|------------------|-----------------|
| 0.25 | 0.950       | 0.3816                   | 0.6361           | 0.0             |
| 0.375| 0.966       | 0.3803                   | 0.7360           | 0.0             |
| 0.5  | 0.983       | 0.2992                   | 0.8294           | 0.0             |
| 0.625| 0.992       | 0.2157                   | 0.8992           | 0.0             |
| 0.75 | 0.997       | 0.1372                   | 0.9536           | 0.0             |
| 0.875| 0.999       | 0.0653                   | 0.9971           | 0.0             |
| 1.0  | 1.0         | 0.0015                   | 1.0326           | 0.0             |
Figure 3.12 Feature values vs. \( \frac{b}{a} \) for half-elliptic curves, \( \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \).
\[ f(t) = b \sqrt{1 - \frac{(t - a)^2}{a^2}} \]
where \( a = 40 \) and \( b = 30 \)

\[ g(t) = \sin \left( \frac{t}{2a} \cdot 2\pi \cdot N_{\text{freq}} \right) \]
where \( N_{\text{freq}} = 2 \)

\[ h(t) = f(t) + N_{\text{scale}} \cdot f(t) \cdot g(t) \]
where \( N_{\text{scale}} = 0.4 \)

Figure 3.13 f(t): half-elliptic curve, g(t): sine wave, and h(t): noise-added curve.
Figure 3.14 Noise-added curve shapes with fixed $N_{\text{scale}}=0.2$ and (a)-(p) $N_{\text{freq}} = 0, 1, 2, \ldots, 9, 10$.

Figure 3.15 Noise-added curve shapes with fixed $N_{\text{freq}} = 3$ and (a)-(p), $N_{\text{scale}} = 0, 0.05, 0.1, \ldots, 0.45, 0.5$. 
Figure 3.16 Distribution of shape feature value, $r$, as a function of $N_{freq}$ and $N_{scale}$.
Figure 3.17 Distribution of shape feature value, SSE, as a function of \( N_{freq} \) and \( N_{scale} \)
Figure 3.18 Distribution of shape feature value, \textit{slope}, as a function of \( N_{freq} \) and \( N_{scale} \).
Figure 3.19 Distribution of shape feature value, concavity, as a function of $N_{freq}$ and $N_{scale}$.
5) concavity with high $N_{freq}$ and low $N_{scale}$ is relatively smaller than the one with low $N_{freq}$ and high $N_{scale}$.

### 3.4 Summary and Conclusions

The curved-shape descriptor, named Curvature-Angular Descriptor (CAD), was derived in this chapter. Input to this method is an array that contains the x and y coordinates making up a curve shape in image coordinates. First, the input curve shape is partitioned into small elements. Second, the series of vectors $(V_i, i=1,\ldots,n-1)$ and a reference vector $V_{ref}$ are created. Third, the length cross product $|V_i \times V_{ref}|$ is obtained and the locations of $V_i$ are normalized. Finally, the linear regression model with normalized location of $V_i$ (dependent variable) and the length of cross product $|V_i \times V_{ref}|$ (independent variable) is obtained. Four shape feature values are extracted from the linear regression model. These feature vectors can be used to uniquely characterize the shape. These feature vectors have properties of translation, scale, and rotation invariance.

Seven half-elliptic curves were created by constant ratios ($b/a$) from elliptic equations ($\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$). Four feature values were extracted from these curves: correlation coefficient ($r$), residual sum of squares (SSE), slope, and concavity. It was found that these feature values from different shapes (or different constant ratios, $b/a$) have different values.
The effect of noise on curve shape was described. The noise-added curves were created by adding sine waves to the half-elliptic curve. Feature values were also extracted from these shapes. As the frequency and amplitude of noises on curve shape changed, these feature values were also changed.

Therefore, a small number of feature values can be extracted using this Curvature-Angular Descriptor. These feature values are enough to characterize different curve shapes.
CHAPTER IV

CLOSED-BOUNDARY SWEET POTATO SHAPE GRADING USING THE CURVATURE-ANGULAR DESCRIPTOR (CAD)

4.1 Introduction

The sweet potato has one of the most irregular shapes of any fruit or vegetable. Peaches and apples can be sorted for size on the basis of a single measurement such as width. White potatoes are uniform enough to be sorted for size by weight. Either of these two measurements is sufficient to sort many agricultural products, but not sweet potatoes.

The United State Standards for Grades of Sweet potatoes is found in the Code of Federal Regulation, title 7 Agriculture, 1995. These standards are almost totally subjective. *Fairly well shaped* means that a sweet potato is not so curved, crooked, constricted or otherwise misshapen as to materially detract from its individual appearance or the general appearance of the lot. The classifications allow for variance from the specifications within a certain percentage: 10 percent of the sweet potatoes in any lot may fail to meet the requirement of the grades. The tolerance is less stringent and the criteria less objective than for manufacturing industries. In this chapter, the feature extraction method, the Curvature-Angular Descriptor, was applied for grading of sweet potato based on the shape of a single view. The clustering of extracted features was attempted using the Learning Vector Quantization Neural
Network (Kohonen, 1989) and the results were compared with grading by four professional human inspectors.

4.2 Materials and Methods

4.2.1 Image Processing System

The image processing system for this study is shown in figure 4.1. It consisted of a Pulnix TM-34 KAGB CCD video camera for monochrome image signals, a VISIONplus-AT® modular frame grabber (MFG) image processor, a 80486/33MHz computer, lighting, and static object standing. The frame grabber captured monochrome images from the camera, and stored the images into 1 MB (1024 by 1024 by 8 bit) onboard buffers or into part of the RAM installed in computer. The VISIONplus-AT® MFG frame processor has a direct data bus connected to the frame grabber for bypassing the data transfer of the AT bus. ITEX MFG™ Library (Image Technology Incorporated, 1992) software containing basic functions for image acquisition and processing was used to support the VISIONplus-AT® MFG frame grabber. The programming language was C (Microsoft Version 6.0) used for the function development of the system. In this study, all the images were stored in files and transferred to the Sun Workstation with a Unix environment to develop the algorithm using Matlab software.

4.2.2 Sweet Potato Shape Feature Extraction using the CAD

The shape of an object is generally described from its boundary information. The nature of shape perception is to convert the boundary
Figure 4.1 The image processing system.
information to certain defined categories. The extracting of boundary information compresses the massive number of points or pixels to certain shape categories. For inspection of sweet potato shapes, the objective is to map their boundary shapes into categories, which relate to their grades.

Four stages of the process for extracting of shape features were used:

1) Segmentation of a light object from the dark background
2) Closed boundary tracing of the segmented object
3) Dividing the closed boundary into two curves by choosing the two points (proximal and distal ends as shown in figure 4.2)
4) Application of CAD to extract shape features.

4.2.2.1 Segmentation of the Object from the Background

The input image consisted of a light object on dark background in such a way that the object and background pixels have gray levels grouped into two dominant modes. One obvious way to extract objects from the background is to select a threshold (T) that separates these modes. Then, any point for which the gray level is greater than the threshold is called an object point; otherwise, the point is called a background point. The threshold was determined using Pixel Classification in Gray Level and Edge Value Space, magnitude of a digital approximation to the gray level gradient, (Panda and Rosenfeld, 1978).
Figure 4.2 The regular convex elliptic shape of a sweet potato
The input and output of images are shown in figure 4.3. Consider an image containing a bright object, with mean gray level \( s_1 \), surrounding by a darker background, with mean gray level \( s_0 < s_1 \). If the difference between \( s_0 \) and \( s_1 \) is much larger than the noise variance in the background and in the object, then the gray level histogram of the image may be bimodal, and thresholding at the valley bottom of this histogram may segment the image into background and object. But the histogram will not display a valley bottom if the noise variance in the images is large or the size of the object region is small.

If the object boundary in the image is sharp then the gray level will quickly fall off as we move from the interior of the object to the background region. If the object has a blurry boundary then the gray level fall-off from the object region to the background region will be gradual.

Let us consider the joint occurrence of the gray level and the gradient (magnitude of a digital approximation to the gray level gradient) at each point of such an image with a blurry boundary. Qualitatively, most of the points in the boundary region will have gray level between \( s_0 \) and \( s_1 \), and gradients that are relatively high compared to that of most object and background points. If the boundary is ramp like, the boundary points will be concentrated around a specific gradient (the slope \( r \) of the ramp), and will have gray levels between \( s_0 \) and \( s_1 \). As their gray levels approach \( s_0 \) and \( s_1 \) (at the shoulders of the ramp),
Figure 4.3 Input (a) and output (b) images to the segmentation process.
their gradients should decrease. Thus, the boundary points should give rise to a \( \cap \)-shaped cluster in (gray level, gradient) space, with the legs of the \( \cap \) corresponding to low gradients and to gray levels near \( s_0 \) and \( s_1 \), while the arch of the \( \cap \) represents intermediate gray levels and high gradients corresponding to the edge ramp slope. The object and background points, on the other hand, should generate clusters concentrated at relatively low gradient (unless the object and background are very noisy), and at gray levels near \( s_0 \) and \( s_1 \), respectively. These observations of the input image (shown in figure 4.3) are illustrated in figure 4.4.

It is noted from the qualitative description from the above that many of the points with gray levels between \( s_0 \) and \( s_1 \) are located away from the zero value in the joint space, because these points correspond to the boundary region. Similarly, many of the noise points, which cause the gray level variance to increase in the background and the object regions, are also located away from the zero gradient in the joint space. Thus, for the gray level histogram at low gradients, the background points will be concentrated around the gray level \( s_0 \) and the object points will be concentrated around the gray level \( s_1 \).

4.2.2.2 The Closed Boundary Tracing of the Segmented Object

Sweet potato boundaries were extracted with an eight-neighbor tracing algorithm (Tao et al. 1995). The input to this process is a binary (0 or 1) image, the object has a gray level 1 and the background a gray
Figure 4.4 Gray level and gradient space.
level 0 (shown in figure 4.3 (b)). The first boundary point is found by starting at the centroid \((x_c, y_c)\) of the object and moving towards the edge of the object, simultaneously looking for a variation in gray level. The centroid can be calculated from moments by using Green's theorem:

\[
\begin{align*}
    x_c &= \frac{\iint_{x, y \in \text{object}} x g(x, y) \, dx \, dy}{\iint_{x, y \in \text{object}} g(x, y) \, dx \, dy} = \sum_x \sum_y x g(x, y) \\
    y_c &= \frac{\iint_{x, y \in \text{object}} y g(x, y) \, dx \, dy}{\iint_{x, y \in \text{object}} g(x, y) \, dx \, dy} = \sum_x \sum_y y g(x, y)
\end{align*}
\]

where \(g(x, y)\) is the gray level at point \((x, y)\).

Once the first boundary point is located, the rest of the boundary points are detected with an eight-directional compass (figure 4.5). The compass has a current direction (idr), which is set to zero at the start of the boundary tracing. When the first boundary point is found, the starting direction is 0. The next boundary point is found by rotating the idr clockwise until the cell has the gray level of objects and the previous cell has the gray level of background.

As an example, when clockwise rotation is started, the gray level of cell 7 is the first one that matches the previous boundary point's gray level and the gray level of cell 1 is the background point. Note that when cell 7 is found, the current idr direction is 7. The algorithm continues until it comes back to the original starting boundary cell. The output of this process is shown in figure 4.7 (a).
Figure 4.5 Definition of compass: (a) compass placed on the boundary pixel (b) shaded region indicates the object.
4.2.2.3 Dividing the Closed-Boundary of a Sweet Potato into Two Curves

The extracted boundary of a sweet potato is divided into two curves, by finding the two end points (proximal and distal ends as shown in figure 4.2). The distance \( r(k) \) between boundary points to the centroid of the object is:

\[
r(k) = \left( (x_k - x_c)^2 + (y_k - y_c)^2 \right)^{1/2}
\]

\( k = 0,1,2 \ldots n = \text{boundary-sequence} \)

where \( x_k \) and \( y_k \) are the coordinates of boundary pixel \( k \), and \( x_c, y_c \) is the centroid of the object from section 4.3.1.2. This one-dimensional boundary \( r(k) \) shown in figure 4.6 always has two peak points if the closed boundary is an elliptic shape such as a sweet potato. These two peak points correspond to the proximal and distal ends as shown in figure 4.2. Thus, the closed boundary is divided into two curves by choosing these peak points. The input and output of this dividing process are shown in figure 4.7.

4.2.3 The Application of CAD to Extract Shape Features from Two Curves

The two extracted curves are shown in figure 4.7 (b) and used as inputs to CAD. These two curves are divided into small segments as described in section 3.1.1. These divided segments of the two curves are shown in figure 4.8 (a) and (b), respectively. It is noted that each of divided segments can be represented as a line, while preserving the original shape of the input curve.
Figure 4.6 One-dimensional boundary signature.
Figure 4.7 (a) Extracted closed boundary of a sweet potato, (b) curves divided by choosing the two endpoints.
Figure 4.8 Divided segments of curves in figure 4.7
The outputs of CAD of the curves in figures 4.8 (a) and (b) are shown in figure 4.9 and figure 4.10, respectively. It is noted from these two outputs that the irregularity of the convex curve shapes is reflected by destroying the linear relationship of the CAD output. For instance, if the input curve shape to CAD is a regular convex shape, the output of CAD has a linear relationship (location of vector versus length of cross product). But if the input curve shape is not a regular convex shape, the linear relationship from CAD has been destroyed. The degree of the irregularity corresponds to the corruptness of the linear relationship.

4.2.4 Sweet potato shape feature extraction

In this application, three feature values were used: SSE, slope, and concavity. These values were extracted from five groups (groups a, b, c, d, and e) of example shapes of sweet potatoes. Group a consisted of sweet potatoes shaped like (1) and (2) in figure 4.11 and examples of regular elliptic shapes of sweet potato, which are acceptable as product. Group b consisted of sweet potatoes shaped like (3) and (4) in figure 4.11 and examples of shapes with a small amount curve or constriction but which are also acceptable as product. Group c consisted of sweet potatoes shaped like (5) and (6) in figure 4.11 and examples of “baseball” shapes which are not acceptable as product. Group d consisted of sweet potatoes shaped like (7) and (8) in figure 4.11 and examples of constricted shapes which are not acceptable as product. Group e consisted of sweet potatoes shaped like (9) and (10) in
Figure 4.9 The output of CAD with left curve in figure 4.8 (a)
Figure 4.10 The output of CAD with right curve in figure 4.8 (b)
Figure 4.11 Example shapes of sweet potato
Because both correlation coefficient and SSE measure the same characteristics (strength of linear relationship between two variables), either one of feature values could be excluded for application. The comparisons of SSE, slope, and concavity of the example shapes are shown in figure 4.12 with the following notations:

1. Note that group a and group c have relatively smaller values of SSE than the rest of the three groups, because these two groups have regular convex shapes.

2. Note that group c has a relatively higher value of slope than the rest of the four groups, because the constant ratio of a1/a2 (from the ellipse equation in section 3.2) for group c was higher than the rest of the four groups.

3. Note that group d and group e have high concavity and group a and group c have low concavity.

Therefore, it was concluded that these three feature values (SSE, slope, and concavity) were feasible for recognition of closed boundary shapes of sweet potato.

4.2.5 Learning Vector Quantization (LVQ) for Clustering of Feature Vectors (SSE, Slope, and Concavity)

In this case of the application of CAD, Learning Vector Quantization (LVQ) was chosen for clustering of the extracted features.
Table 4.1 Extracted feature values from example shapes in figure 4.11

<table>
<thead>
<tr>
<th>group</th>
<th>Sweet Potato Shape</th>
<th>SSE</th>
<th>Slope</th>
<th>Concavity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0.2234</td>
<td>0.8873</td>
<td>0.2703</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.2649</td>
<td>0.9061</td>
<td>0.3100</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>0.4881</td>
<td>0.7712</td>
<td>0.5450</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.6503</td>
<td>0.7692</td>
<td>0.5887</td>
</tr>
<tr>
<td>c</td>
<td>5</td>
<td>0.2569</td>
<td>1.1517</td>
<td>0.2379</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.2339</td>
<td>1.2753</td>
<td>0.2945</td>
</tr>
<tr>
<td>d</td>
<td>7</td>
<td>0.7773</td>
<td>0.8251</td>
<td>0.8607</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.6910</td>
<td>0.7959</td>
<td>0.9437</td>
</tr>
<tr>
<td>e</td>
<td>9</td>
<td>1.0078</td>
<td>0.5751</td>
<td>1.0418</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.8533</td>
<td>0.7242</td>
<td>1.2371</td>
</tr>
</tbody>
</table>
Figure 4.12 Comparison of feature values between five groups of example shapes shown in figure 4.11
Learning Vector Quantization (Kohonen, 1989) is one of the pattern recognition methods in which each output unit represents a particular class or category. The learning results are achieved in a supervised, decision-controlled teaching process.

This method seeks the smallest distance of the unknown vector from a set of reference vectors. And a fixed number of reference vectors for each class is selected. In training, only the nearest reference vector is updated. Updating of the reference vector is made for both correct and incorrect classification.

As a result, it turns out that the reference vectors in a way approximate the probability density functions of the pattern classes. To be more accurate, the nearest neighbors define decision surfaces between the pattern classes. And in classification, the class boundaries are of primary importance; description of the inside of the density functions is less important.

Let’s consider a process which is supposed to optimally allocate K reference vectors $m_i \in \mathbb{R}^n, i = 1, 2, ..., K$ to the space of n-dimensional pattern vectors $x \in \mathbb{R}^n$ such that the local point density of the $m_i$ (i.e., the number of $m_i$ falling in a small volume of $\mathbb{R}^n$ centered at point $x$) can be used to describe the probability density function $P(x)$. For the initial values of the $m_i, i = 1, 2, ..., K$, the first K samples of $x$ can be chosen. The next samples of $x$ are then used for training. In the case that only a limited number of samples are available, they must be applied iteratively.
in the learning process, e.g., picking them randomly from the training set. Let $m_c$ be that vector among the $m_i$ from which $x$ has the smallest distance $\|x-m_c\|$ (in arbitrary metric). This vector is then updated so that $\|x-m_c\|$ is decreased or increased for the case of correct or incorrect classification, respectively.

The algorithm is initialized with the first samples of $x$ which are identified with the $m_i = m_i(0)$. Thereafter these vectors are labeled using another set of calibration samples of $x$ with known classification. Distribution of the calibration samples to the various classes, as well as the relative numbers of the $m_i$ assigned to these classes must comply with the a priori probabilities $P(S_k)$ of the classes $S_k$. Each calibration sample is assigned to that $m_i$ to which it is closest. Each $m_i$ is then labeled according to the majority of classes represented among those samples which have been assigned to $m_i$.

Let the training vector $x(t)$ belong to class $S_r$. Assume that the closest reference vector $m_c$ is labeled according to class $S_s$. The supervised learning algorithm that rewards correct classifications and punishes incorrect ones is then defined as,

$$m_c(t+1)=m_c(t)+\alpha(t)[x(t)-m_c(t)] \quad \text{if} \quad S_s=S_r,$$

$$m_c(t+1)=m_c(t)-\alpha(t)[x(t)-m_c(t)] \quad \text{if} \quad S_s \neq S_r,$$

$$m_i(t+1)=m_i(t) \quad \text{if} \quad i \neq c,$$
where $\alpha(t)$ is a monotonically decreasing scalar gain factor (preferably $\alpha << 1$). Note that the updating is described in the discrete-time formulation ($t=1,2,...$). Only the closest of the vectors $m_i$ is modified at a time, but the direction of the correction depends on the correctness of classification.

The primary effect of the rule applied in this method is to minimize the number of misclassifications. At the same time, the reference vectors $m_i$ are pulled away from the zones where misclassifications occur.

### 4.3 Experimental Evaluation and Results

Four hundred seven (407) sweet potatoes were randomly selected from three places: (1) The LSU Burden Research Farm, Baton Rouge, LA, (2) three commercial sweet potato farms, and (3) The Sweet Potato Research Center, Chase, LA. The sweet potatoes were separated into two categories (acceptable and unacceptable) by each of four professional sweet potato inspectors. In other words, each sample sweet potato was inspected four times. The inspection results for four human graders are shown in table 4.2.

These samples were separated into 5 groups. The first group was the samples that were selected as unacceptable product by all four inspectors. The second group was the samples that were selected as unacceptable product by three inspectors and selected as acceptable product by one inspector. The third group was the samples that were
Table 4.2 Sample shapes of sweet potatoes separated into 5 groups

<table>
<thead>
<tr>
<th>Group</th>
<th>number of polls as unacceptable shape</th>
<th>number of polls as acceptable shape</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>4</td>
<td>0</td>
<td>91 (22.36%)</td>
</tr>
<tr>
<td>Group 2</td>
<td>3</td>
<td>1</td>
<td>59 (14.56%)</td>
</tr>
<tr>
<td>Group 3</td>
<td>2</td>
<td>2</td>
<td>71 (17.46%)</td>
</tr>
<tr>
<td>Group 4</td>
<td>1</td>
<td>3</td>
<td>68 (16.71%)</td>
</tr>
<tr>
<td>Group 5</td>
<td>0</td>
<td>4</td>
<td>118 (28.99%)</td>
</tr>
</tbody>
</table>
selected as unacceptable product by two inspectors and selected as acceptable product by the other two inspectors. The fourth group was the samples that were selected as unacceptable product by one inspector and selected as acceptable product by the other three inspectors. And the fifth group was the samples that were selected as acceptable product by all four inspectors.

Next two sets of samples were chosen from the 407 total potatoes as follows:

1) The first set consisted of the first group as having unacceptable shape and fifth group as acceptable (see table 4.2). The number of potatoes was 209.

2) The second set consisted of first and second groups as unacceptable shape and fourth and fifth groups as acceptable shape. The number of potatoes was 336.

The three feature values (SSE, slope, and concavity) from the first set of samples are plotted in figure 4.13. Note that ‘x’ represents unacceptable shape (first group) and ‘o’ represents acceptable shape (fifth group). The three feature values (SSE, slope, and concavity) from the second set of samples are plotted in figure 4.14. Note that ‘x’ represents unacceptable shape (first & second group) and ‘o’ represents acceptable shape (fourth & fifth group).

The classification of sweet potato shapes into two categories (acceptable and unacceptable shapes) was attempted using the LVQ
Figure 4.13 Feature vector space of first set of samples
Figure 4.14 Feature vector space of second set of samples
neural network with feature vectors from the two sample sets. The error rate versus training cycle (Epoch) with two sample sets is shown in figure 4.15. Note that the error rate is the percentage of misclassification.

After 9730 epochs, the LVQ net yielded a 6.70% error with the first sample set and 14.88% error with the second sample set. This result indicates that the LVQ net showed 93.3% memorization (the ability to recall perfectly patterns that have been learned) with the first sample set and 85.12% memorization with the second sample set.

The trained LVQ net with the first sample set was tested with 407 total samples and the classification results were compared with each inspector. The result is shown in table 4.3. They show 24.39% (2.93 standard deviation) average disagreement between the LVQ net and inspectors. However, there was 27.23% (7.08 standard deviation) average disagreement between each inspector. Comparison of the disagreement between each inspector is also shown in table 4.2.

Another attempt was made to test the generalization (the ability of a neural net to produce reasonable responses to input patterns that are similar, but not identical, to training patterns) of the LVQ net with the first sample set. Fixed numbers (50, 70, and 90) of sample potatoes were randomly selected as testing data for generalization and the rest of the unselected samples were used to train the LVQ net. This test was repeated 10 times with each case. The training of LVQ net was stopped
First set: group1 (unacceptable), group5 (acceptable)
Second set: group1&2 (unacceptable), group4&5 (acceptable)

Figure 4.15 The error convergence rate of LVQ net training.
Table 4.3: Disagreement between human inspectors and the trained LVQ net

<table>
<thead>
<tr>
<th></th>
<th>Inspector1</th>
<th>Inspector2</th>
<th>Inspector3</th>
<th>Inspector4</th>
<th>LVQ net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspector1</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0/407)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspector2</td>
<td>24.82%</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(101/407)</td>
<td>(0/407)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspector3</td>
<td>31.20%</td>
<td>16.71%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(127/407)</td>
<td>(68/407)</td>
<td>(0/407)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inspector4</td>
<td>23.34%</td>
<td>30.47%</td>
<td>36.86%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(95/407)</td>
<td>(124/407)</td>
<td>(150/407)</td>
<td>(0/407)</td>
<td></td>
</tr>
<tr>
<td>LVQ net</td>
<td>20.39%</td>
<td>24.08%</td>
<td>27.03%</td>
<td>26.04%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Table 4.4 Results of the generalization of the LVQ net with test data from 50, 70 and 90 sample potatoes.

<table>
<thead>
<tr>
<th>Repetition</th>
<th>50 test data</th>
<th>70 test data</th>
<th>90 test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.571</td>
<td>20.0</td>
<td>18.889</td>
</tr>
<tr>
<td>2</td>
<td>11.429</td>
<td>10.0</td>
<td>13.333</td>
</tr>
<tr>
<td>3</td>
<td>11.429</td>
<td>10.0</td>
<td>10.0</td>
</tr>
<tr>
<td>4</td>
<td>14.286</td>
<td>18.0</td>
<td>12.222</td>
</tr>
<tr>
<td>5</td>
<td>20.0</td>
<td>20.0</td>
<td>12.222</td>
</tr>
<tr>
<td>6</td>
<td>8.571</td>
<td>16.0</td>
<td>15.556</td>
</tr>
<tr>
<td>7</td>
<td>18.571</td>
<td>8.0</td>
<td>7.778</td>
</tr>
<tr>
<td>8</td>
<td>10.0</td>
<td>22.0</td>
<td>17.778</td>
</tr>
<tr>
<td>9</td>
<td>15.714</td>
<td>6.0</td>
<td>21.111</td>
</tr>
<tr>
<td>10</td>
<td>18.571</td>
<td>10.0</td>
<td>15.556</td>
</tr>
<tr>
<td>Average</td>
<td>14.714</td>
<td>14.0</td>
<td>14.444</td>
</tr>
</tbody>
</table>
if the misclassification error was less than 7%. The results were 14.71% average error (4.15 standard deviation), 14.0% average error (5.81 standard deviation), and 14.4% average error (4.12 standard deviation) with 50, 70, and 90 training samples, respectively. These results are shown in Table 4.4.

4.4 Summary and Conclusions

The Curvature-Angular Descriptor (CAD) was applied to closed-boundary sweet potato shape grading using machine vision. Shape information was extracted from the closed boundaries of sweet potatoes and compressed into three feature values \(\{\text{SSE, slope, and concavity}\}\). It was found that these features are good enough for sweet potato recognition. Using 407 sweet potatoes, previously inspected by four professional inspectors, these feature values were extracted and clustered using a Learning Vector Quantization neural network (Kohonen, 1989).

The inspection results of the human graders showed 27.23% (7.08 standard deviation) average disagreement between each inspector. The method used in this study gave 24.39% (2.93 standard deviation) average disagreement with each inspector. Therefore, based on these experiments, this method achieved about the same ability as human inspectors within the subjective limits of human graders.
CHAPTER V

SWEET POTATO SHAPE RECOGNITION USING STRUCTURED-LIGHT TECHNIQUES AND THE CURVATURE-ANGULAR DESCRIPTOR

5.1 Introduction

An overhead view of a sweet potato in its natural rest position on a flat surface will, in most cases, be the view representing the worst shape. However, in an actual inspection line, it would be best to include a surface inspection to ensure that the worst-shape view can be determined. Three types of sweet potato root surface defects are reported (Huaman, 1987); lenticels, constrictions, and grooves. The structured-light technique was applied to solve this problem by extracting three-dimensional surface shape information of a sweet potato. A technique, called light stripping (Haralick and Shapiro, 1993), was used for structured light, in which the projected light planes and a camera make up an active stereo system so that three-dimensional geometric information about an object surface can be derived from optical triangulation. The stripes on different parts of the surface have different shapes. The stripes on convex surfaces are continuous and parabolic, and their curvature directions maintained. Stripes on nearly flat surfaces are almost parallel lines. Stripes can touch adjacent stripes due to sharp changes in depth and appear broken due to occlusion. In the case of continuous stripes, their curvatures change signs when the surface changes from convex to concave. Therefore, the correspondence
between the shape of the stripes and the shape of surface parts provides necessary three-dimensional surface information and this can be obtained from the analysis of the different shape patterns of curved stripes. Thus, the objective of this study was to develop a method to recognize sweet potato surface shapes. To accomplish this objective:

1. The sensing environment to extract the surface information of the object was created using the structured-light technique.

2. Quantitative analysis of the degree of ambiguity of the system was done to explore the relationship between camera tilt angle, camera standoff from the worktable, and stripe spacing.

3. An algorithm was applied to extract the curved stripes containing surface information from the image obtained from the structured-light sensing environment.

4. The Curvature-Angular Descriptor (CAD) was applied as a method for analysis of the extracted curved stripes.

5. The feature vectors for sweet potato surface shape classification were explored.

5.2 Materials and Methods

5.2.1 Sensing Environment

Figure 5.1 gives a sketch of the sensing environment used. The object holding plate was V-shaped valley channel, that was about 127mm x 254mm x 26.4mm (5in. x 10in. x 1in.). A striped pattern of light was projected vertically downward onto the scene, which was viewed by a
camera whose axis was about 45 degrees offset from the projector axis. The standoff of the camera from the scene was about 500mm (≈20.0 in.) and the stripe spacing was about 20mm.

5.2.2 Quantitative Analysis of Degree of Ambiguity

Hu and Stockman (1989) proposed a quantitative analysis on the degree of ambiguity, which refers to the number of surface solutions. The question is how many solutions may be possible without any knowledge about the objects in the scene except the projection and imaging geometry. Figure 5.2 illustrates the problem. The number of possible solutions depends on several factors, for instance, light stripe spacing, number of stripes, and the distance from the camera to the scene. In addition, a camera ray and a projector ray (both are lines in 3D) may not intersect in space due to measurement and computation inaccuracy even though the projector ray does create that very point in the 2D image. But it is assumed that a camera ray and a projector ray do intersect in 3D space if the distance between the two rays is less than a predetermined tolerance T. Taking this factor into consideration, the degree of ambiguity is also an increasing function of T. This quantitative analysis shows that the degree of ambiguity is not arbitrarily large, even though infinitely many projected light stripes are assumed.
Figure 5.1 Sensing environment for structured-light
Figure 5.2 Ambiguity of location in 3D space
5.2.2.1 Assumption

Rays from the projector are assumed to be orthographic or parallel projections. That is, all stripes are parallel to each other. Camera imaging is assumed perspective. It is desirable to examine how many locations are possible in 3D space for a stripe on the x-y plane (worktable). The projection model is illustrated in Figure 5.3.

5.2.2.2 Basic formulas of a line in 3D

The equation of the line through two fixed points \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\) is given by

\[
\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}
\]

(1)

The direction numbers of a line

\[
Ax + By + Cz + D = 0 \\
ax + by + cz + d = 0
\]

are \(l, m, n\) given by

\[
l = \begin{vmatrix} B & C \\ b & c \end{vmatrix}, \quad m = \begin{vmatrix} C & A \\ c & a \end{vmatrix}, \quad n = \begin{vmatrix} A & B \\ a & b \end{vmatrix}
\]

The distance between two lines is

\[
d = \frac{|(x_1 - x_2)L + (y_1 - y_2)M + (z_1 - z_2)N|}{\sqrt{L^2 + M^2 + N^2}}
\]

(2)

where \((x_1, y_1, z_1)\) is an arbitrary single point on line and \((x_2, y_2, z_2)\) is another arbitrary point on the same line, and

\[
L = \begin{vmatrix} m_1 & n_1 \\ m_2 & n_2 \end{vmatrix}, \quad M = \begin{vmatrix} n_1 & l_1 \\ n_2 & l_2 \end{vmatrix}, \quad N = \begin{vmatrix} l_1 & m_1 \\ l_2 & m_2 \end{vmatrix}
\]
with \( l_1, m_1, n_1 \) the direction numbers of the first line, and \( l_2, m_2, n_2 \) the direction numbers of the second line.

### 5.2.2.3 Computation in 3D space

Let stripe spacing be \( t \) and the two vertices of one stripe be \((x_g t, y_g t)\) and \((x_g t, (y_g + 1) t)\). Let also the optical axis of the camera be from the camera \((0, y_c, z_c)\) to the midpoint of the stripe, i.e. the coordinates of the camera are

\[
\left[0, \left(y_g + \frac{1}{2} \right), \frac{x_g}{\tan \theta}\right].
\]

Suppose that a camera ray created with the two points, \([0, (y_g + \frac{1}{2}), \frac{x_g}{\tan \theta}]\) and \((x_g t, y_g t)\) is called CL1 and another camera ray created with the two points, \([0, (y_g + \frac{1}{2}), \frac{x_g}{\tan \theta}]\) and \((x_g t, (y_g + 1)t)\) is called CL2.

CL1 has the equation

\[
CL_1 : \begin{cases}
x + 2x_g y - (2y_g + 1)x_g t = 0 \\
2x_g y - \tan \theta z - 2x_g y_g t = 0
\end{cases}
\]

Thus, the direction numbers for camera ray \( CL_1 \) are:

\[
cl_1 = \begin{vmatrix} 2x_g & 0 \\ 2x_g & -\tan \theta \end{vmatrix} = -2x_g \tan \theta
\]

\[
cm_1 = \begin{vmatrix} 0 & 1 \\ -\tan \theta & 0 \end{vmatrix} = \tan \theta
\]

\[
cn_1 = \begin{vmatrix} 1 & 2x_g \\ 0 & 2x_g \end{vmatrix} = 2x_g
\]
Figure 5.3 Optical triangulation with camera and parallel stripe-projector

\[ Z_e = \frac{x_g t}{\tan \theta} \]
Suppose \( k \) locations of a single stripe exist which are obtained by shifting the stripe toward the camera along its optical axis by \( k \) X-stripes, then the corresponding equation of the projector ray \( L_1 \) at \( (x_g - k, y_g t, 0) \) is given by

\[
L_1 = \begin{cases} 
  x - (x_g - k) = 0 \\
  y - y_g t = 0
\end{cases}
\]

and the direction numbers are

\[
l_1 = \begin{vmatrix} 0 & 0 \\ 1 & 0 \end{vmatrix} = 0, \quad m_1 = \begin{vmatrix} 0 & 1 \\ 0 & 0 \end{vmatrix} = 0, \quad n_1 = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1
\]

Hence, the distance \( D_1 \) between the camera ray \( CL_1 \) and the projector ray \( L_1 \) (as shown in figure 5.4) is computed according to formula (2):

\[
D_1 = \left( x_g - k \right) \tan \theta + \frac{y_g t - \left( y_g + \frac{1}{2} t \right)}{\sqrt{\tan^2 \theta + 4x_g^2 \tan^2 \theta}} + 0 = \frac{kt}{\sqrt{1 + 4x_g^2}}
\]

The distance \( D_2 \), the distance between the camera ray \( CL_2 \) and the projection ray \( L_2 \) at \( (x_g - k, (y_g + 1)t, 0) \) can be obtained in a similar way.

\[
D_2 = \frac{kt}{\sqrt{1 + 4x_g^2}}.
\]

Making these distance \( D_1 \& D_2 \) less than a prearranged tolerance \( T \), we have

\[
k \leq \sqrt{1 + 4x_g^2} T \approx \frac{2x_g}{t} T \approx \frac{2z_c}{t^2} T \tan \theta
\]

(3)
Figure 5.4 Zoom-in view of figure 5.3 and distance $D_1$ & $D_2$. 
Table 5.1 Number of possible locations in 3D for a striped square as a function of camera standoff (Zc) and camera tilt angle with fixed stripe spacing (t)=20 mm and tolerance (T)=3.0mm

<table>
<thead>
<tr>
<th>Camera tilt angle</th>
<th>Camera standoff, Zc (mm)</th>
<th>200</th>
<th>250</th>
<th>300</th>
<th>350</th>
<th>400</th>
<th>450</th>
<th>500</th>
<th>550</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>20°</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>25°</td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>30°</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>35°</td>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>40°</td>
<td></td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>45°</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>50°</td>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>55°</td>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>60°</td>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
Table 5.2 Number of possible locations in 3D for a stripe as a function of camera standoff (Zc) and stripe spacing (t) with fixed camera tilt angle (θ)=45° and tolerance (T)=3.0mm

<table>
<thead>
<tr>
<th>Camera standoff (mm)</th>
<th>Stripe spacing (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>200</td>
<td>11</td>
</tr>
<tr>
<td>250</td>
<td>14</td>
</tr>
<tr>
<td>300</td>
<td>17</td>
</tr>
<tr>
<td>350</td>
<td>20</td>
</tr>
<tr>
<td>400</td>
<td>23</td>
</tr>
<tr>
<td>450</td>
<td>26</td>
</tr>
<tr>
<td>500</td>
<td>29</td>
</tr>
<tr>
<td>550</td>
<td>32</td>
</tr>
<tr>
<td>600</td>
<td>35</td>
</tr>
</tbody>
</table>
Hence, we conclude that the number $k$ (possible locations of a stripe in 3D space) is proportional to the camera standoff (determined by $x_g$) and the tolerance $T$, and inversely proportional to square of stripe spacing ($r^2$). The number $k$ was calculated using equation (3) and tabulated in table 5.1 and 5.2.

5.2.3 Determination of Camera Standoff ($Z_c$) and Camera Tilt Angle ($\theta$)

Several assumptions are made as follows:

1. The object is a perfect ellipsoid. Thus, the cross section of the object is an ellipse as shown figure 5.5.

2. The geometry of the object is created using the ellipse equation, 
\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \]
   The a/b ratio is set to 3 to 1, because the elliptic shape of sweet potato is defined such that the a/b ratio should not be more than 3 to 1 (CFR, title 7, 1995).

3. One of the projection rays intersects the centroid of the object as shown in figure 5.5.

4. The centroid of the object is placed on the point where the optical axis intersects the worktable as shown in figure 5.5.

5. The point O is origin of x-y coordinates.

$\theta_1$ and $\theta_2$ are defined as
\[
\theta_1 = \angle OCP_1 = \sin^{-1}\left(\frac{\overrightarrow{PC} \times \overrightarrow{OC}}{\|\overrightarrow{PC}\| \|\overrightarrow{OC}\|}\right)
\]
Zc: Camera Standoff  
Xc: Zc \cdot \tan \theta  
\theta: Camera Tilt Angle  
t: Stripe Spacing  
C: Camera Location  
P_1: Farthest Projection Point on Object from Point O  
P_2: 2^{nd} Farthest Projection Point on Object from Point O  

Figure 5.5 Determination of camera tilt angle
Figure 5.6 $\theta_1 - \theta_2$ as a function of $Zc$ with $Xc=500\text{mm}$.

- $\circ$: $a=50\text{mm}$, $b=16\text{mm}$
- $\times$: $a=70\text{mm}$, $b=23\text{mm}$
- $\times$: $a=90\text{mm}$, $b=30\text{mm}$
- $\times$: $a=110\text{mm}$, $b=36\text{mm}$
- $\times$: $a=130\text{mm}$, $b=43\text{mm}$

$\theta_1$, $\theta_2$, $Zc$, $Xc$, $a$, $b$, $\text{mm}$.
The x-y coordinates of points O, C, P₁, and P₂ are:

\[ O = (0, 0) \]
\[ C = (0, Z_c) \]
\[ P₁ = (X_c + t₁, b + tt₁) \]
\[ P₂ = (X_c + t₂, b + tt₂) \]

where, \( num = \text{quotient of } a / t \), \( t₁ = num \times t \), \( t₂ = (num - 1) \times t \), \( tt₁ = \frac{b \sqrt{1 - t₁^2}}{a^2} \), and \( tt₂ = \frac{b \sqrt{1 - t₂^2}}{a^2} \). If \( \theta₁ - \theta₂ > 0 \), every stripe on the object is visible to camera. However, if \( \theta₁ - \theta₂ < 0 \), the farthest stripes on the object from camera point will not be visible. Camera standoff and camera tilt angle can be determined where all of stripes are visible to camera. The difference between \( \theta₁ \) and \( \theta₂ \) is calculated as a function of \( Z_c \) with fixed \( X_c = 500 \text{ mm} \) for five different size of objects as shown in figure 5.6. Thus, the stripe spacing was chosen to be about 20 mm, camera standoff was 500mm (\( \equiv 20 \text{ in.} \)), and camera tilt angle was 45° in this study.

5.2.4 Algorithm to Extract Stripe Patterns from Structured-Light Images

The input image to this algorithm consisted of light stripe patterns on the object and dark background as shown in figure 5.7. The stripe patterns were segmented out using the thresholding method explained in chapter 4, section 4.3.1 as shown in figure 5.8. These segmented stripes
Figure 5.7 Input stripe patterns on ellipsoid object image.
Figure 5.8 Segmented stripe patterns on ellipsoid object.
Figure 5.9 Thinned stripe patterns
were thinned to thickness of 1 as shown in figure 5.9. Then the thinned stripes were extracted for shape analysis using the Curvature-Angular Descriptor. The output of this process contains the x-y coordinates, which makes up the stripe patterns. The flow chart of algorithm is as follows:

![Flow chart of algorithm](chart.png)

### 5.2.4.1 Thinning Algorithm

The thinning algorithm, presented by Gonzalez and Woods (1992), was used. Region pixels were assumed to have value 1 and background pixels to have value 0. The method consists of successive passes of two basic steps applied to contour pixels of the given region, where a contour pixel is any pixel with value 1 and having at least one 8-neighborhood valued 0. The 8-neighborhood definition is shown in figure 5.10.

Step 1: Flags a contour pixel p for deletion if the following conditions are satisfied.

(a) \(2 \leq N(p) \leq 6\);
where \( N(P_i) \) is the number of nonzero neighbors of \( P_i \) and \( S(P_i) \) is the number of 0-1 transactions in the ordered sequences of \( P_2, P_3, \ldots, P_8, P_9, P_2 \).

Step 2: Conditions (a) and (b) of step 1 remain the same, but conditions (c) and (d) are changed to:

(c') \( P_2 \times P_4 \times P_8 = 0 \);

(d') \( P_2 \times P_6 \times P_8 = 0 \).

Step 1 is applied to every border pixel in the binary region under consideration. If one or more of conditions (a)-(d) are violated, the value of the pixel in question is not changed. If all conditions are satisfied the point is flagged for deletion. After step 1 has been applied to all border pixels, those that were flagged are deleted (changed to 0). Then, step 2 is applied to the resulting data in exactly the same manner as step 1.

Therefore, one iteration of the thinning algorithm consists of (1) applying step 1 to flag border pixels for deletion; (2) deleting the flagged pixels; (3) applying step 2 to flag the remaining border pixels for deletion; and (4) deleting the flagged pixels. This procedure is applied iteratively until no further pixels are deleted yielding the skeleton of the region.
Figure 5.10 8-neighborhood arrangement used by the thinning algorithm
5.2.4.2 Thinned Stripe Extraction Algorithm

The input to this algorithm is the output image from the thinning algorithm and the output of this algorithm is the lists containing the coordinates of the pixels that make up the thinned stripe pattern on the image. Later, the Curvature-Angular Descriptor will be applied to these extracted stripes to evaluate features for surface shape recognition.

The procedure of the algorithm is as follows:

1. For each bright pixel, compute its degree (of value 1 or 2) by counting the number of bright pixels in its 8-neighbors (as shown in figure 5.10). Pixels of degree 1 are stripe endpoints. Pixels of degree 2 are connecting points on stripes.

2. Extract the network $n_j$ by tracking the stripe points. $V_j \leftarrow \{\text{endpoints tracked}\}$. $E_j \leftarrow \{(p_{ik}, p_{il}), p_{ik}, p_{il} \in V_j \text{ and connected by pixels of degree 2}\}$. $n_j \leftarrow \{V_j, E_j\}$. Reset all stripe points tracked to zero.

3. Repeat (2) until all pixels in the image are zero.

For step (2) and (3), stripe points occupy only a small fraction of the whole image, and each stripe point is examined once. Hence, the time complexity is linear in the number of stripe pixels.

5.2.5 Selection of Surface Shape Features using the Curvature-Angular Descriptor (CAD) for Sweet Potato Shape Recognition

Let us assume that there are $n$ stripes on an object as shown in figure 5.11. Five feature values were selected for surface shape recognition as follows:
Figure 5.11 Stripes on ellipsoid object.
Figure 5.12 Unit vectors $\mathbf{V}_i$ ($i=1,2,\ldots,n$) created by using midpoints on based lines.
Figure 5.13 Length of base lines, $l_i$, of each curve's length, $L$, between midpoints of first and last base lines
1. **SSE (Sum of error square)**

\[
SSE = \frac{1}{n} \sum_{i=1}^{n} sse_i
\]

where \( sse_i \) is the sum of error squares obtained using the CAD for each stripe on the object.

2. **Slope**

\[
Slope = \frac{1}{n} \sum_{i=1}^{n} slope_i
\]

where \( slope_i \) is the slope obtained using CAD for each stripe on the object.

3. **Concavity**

\[
Concavity = \frac{1}{n} \sum_{i=1}^{n} concavity_i
\]

where \( concavity_i \) is the concavity obtained using CAD for each stripe on the object.

4. **Core**

\[
Core = \frac{1}{(n-1)} \sum_{i=2}^{n} \overrightarrow{V_i} \cdot \overrightarrow{V_i}
\]

\[i = 2, 3, \ldots, n\]

where unit vectors \( \overrightarrow{V_i} \) \( (i=1, 2, \ldots, n) \) are created using midpoints on base line of each of the curves as shown in figure 5.12.

5. **Elongation**

\[
elongation = \frac{1}{n} \sum_{i=1}^{n} \frac{l_i}{L}
\]
where \( l_i \) is the length of the base lines of each of the curves and \( L \) is the length between midpoints of first and last base lines as shown in figure 5.13.

5.2.6 Exploration of the Relationship between Shape Feature Values and Shapes

Two sizes of ellipsoids (object1 and object2), one elliptic cylinder (object3), and one curved-shape sweet potato (object 4) were used (shown in figure 5.14) for evaluating the relationship between shape feature values and shapes. The shape feature values from these four shapes are tabulated in table 5.3.

It can be found (by comparing the shape feature values between these four objects) that:

1) The cross sections of objects 1&2 are circles and the cross section of object 3 is an ellipse. If we compare the SSE and slope of these objects, \( SSE(object\ 1&2) < SSE(object\ 3) \) and \( slope(object\ 1&2) > slope(object\ 3) \).

2) The “backbones” of the object 1&2 are close to a straight line and the backbone of object 4 is curved. If we compared core values of these objects, \( core(object\ 1&2) > core(object\ 4) \).

3) The overall shape of object 2 is a baseball-like shape and that of objects 1&3 is a long football-like shape. If we compared core values of these objects, \( elongation(object\ 2) > elongation(object\ 1&3) \).
4) The surfaces of objects 1, 2, & 3 are free of bumps and concavities. On the other hand, the surfaces of object 4 are not free of bumps and concavities. Therefore, the concavity value of these objects is zero and that of object 4 is slightly greater than zero.

5.3 Experimental Evaluations and Results

240 sample shapes of sweet potato were inspected by five human inspectors into two categories: “good” and “bad”. These samples can be grouped into 6 based on the inspection results as shown in Table 5.4. These samples were used to train the LVQ net and compare the result with human inspectors.

The separability criterion was developed to quantify the degree of how these two patterns (“good” and “bad” shapes) were separated. Using the criterion, described in the next section, the reduction of the number of feature vectors was made. The LVQ net was then trained with a reduced number of feature vectors from the sample shapes graded by five human inspectors and the performance of the trained LVQ net was compared with the human inspectors.

5.3.1 Separability Criterion

The covariance matrix is defined by

\[
\Sigma = E\{ (X - M)(X - M)^T \} = E\left\{ \begin{bmatrix}
    x_1 - m_1 \\
    \vdots \\
    x_n - m_n
\end{bmatrix} \begin{bmatrix}
    x_1 - m_1 \\
    \vdots \\
    x_n - m_n
\end{bmatrix} \right\}
\]
Table 5.3 Feature values of four different shapes (object 1, 2, 3, & 4)

<table>
<thead>
<tr>
<th>Object</th>
<th>Shape</th>
<th>Concavity</th>
<th>Core</th>
<th>Elongation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.9944</td>
<td>0.0000</td>
<td>2.4476</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.9090</td>
<td>0.0000</td>
<td>5.2395</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.7071</td>
<td>0.0000</td>
<td>2.5482</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.6477</td>
<td>0.0000</td>
<td>2.0768</td>
</tr>
</tbody>
</table>

Figure 5.14 Four different shapes: object 1, 2, 3, & 4

*object 1:* \( l_1 = 1.13', l_2 = 3.0' \)
*object 2:* \( l_1 = 1.50', l_2 = 1.75' \)
*object 3:* \( l_3 = 0.82', l_4 = 1.29', l_5 = 5.6' \)
*object 4:* curved - shape sweet potato

Figure 5.14 Four different shapes: object 1, 2, 3, & 4
Table 5.3 Feature values of four different shapes (object 1, 2, 3, & 4)

<table>
<thead>
<tr>
<th></th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSE</td>
<td>0.0669</td>
<td>0.8944</td>
<td>0.0000</td>
<td>0.9940</td>
<td>2.4476</td>
</tr>
<tr>
<td>Slope</td>
<td>0.0664</td>
<td>0.9101</td>
<td>0.0000</td>
<td>0.9995</td>
<td>5.2395</td>
</tr>
<tr>
<td>Concavity</td>
<td>0.1590</td>
<td>0.7644</td>
<td>0.0000</td>
<td>0.9749</td>
<td>2.5482</td>
</tr>
<tr>
<td>Core</td>
<td>0.3236</td>
<td>0.8129</td>
<td>0.2807</td>
<td>0.7938</td>
<td>2.0765</td>
</tr>
<tr>
<td>Elongation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Object 1 | Object 2 | Object 3 | Object 4 |

Table 5.4 Sample shapes separated into 6 groups based on inspection result

<table>
<thead>
<tr>
<th></th>
<th>Number of polls as acceptable shape</th>
<th>Number of polls as unacceptable shape</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>5</td>
<td>0</td>
<td>161</td>
</tr>
<tr>
<td>Group 2</td>
<td>4</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Group 3</td>
<td>3</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Group 4</td>
<td>2</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>Group 5</td>
<td>1</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Group 6</td>
<td>0</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>
\[ E \left\{ \begin{bmatrix} (x_1 - m_1)(x_1 - m_1) & \cdots & (x_1 - m_1)(x_n - m_n) \\ \vdots & \ddots & \vdots \\ (x_n - m_n)(x_1 - m_1) & \cdots & (x_n - m_n)(x_n - m_n) \end{bmatrix} \right\} = E\left\{ \begin{bmatrix} E\{(x_1 - m_1)(x_1 - m_1)\} & \cdots & E\{(x_1 - m_1)(x_n - m_n)\} \\ \vdots & \ddots & \vdots \\ E\{(x_n - m_n)(x_1 - m_1)\} & \cdots & E\{(x_n - m_n)(x_n - m_n)\} \end{bmatrix} \right\} \]

where \( \mathbf{X} \) is a random vector characterized by its distribution or density function and \( \mathbf{M} \) is the expected vector of a random vector \( \mathbf{X} \). The diagonal components of the covariance matrix are the variances of individual random variables. Also, it is noted that all covariance matrices are symmetric matrices. The trace of covariance is the summation of all the eigenvalues or diagonal components of the matrix. Therefore, this trace of the covariance matrix can be a parameter which indicates the dispersion of the distribution.

Fukunaga (1972) defined two matrices for discriminant analysis (with within-class and between-class scatter matrices), that are used to formulate criteria of class separability.

A within-class scatter matrix shows the scatter of samples around their class expected vector, and is expressed by

\[
S_w = \sum_{i=1}^{2} P(\omega_i)E\left\{ (X - M_i)(X - M_i)^T \mid \omega_i \right\} = \sum_{i=1}^{2} P(\omega_i)\Sigma_i
\]

On the other hand, a between-class scatter matrix can be defined by

\[
S_b = E\left\{ (X^{(1)} - X^{(2)})^T(X^{(1)} - X^{(2)}) \right\} \\
= E\left\{ X^{(1)}X^{(1)^T} \right\} + E\left\{ X^{(2)}X^{(2)^T} \right\} - 2E\left\{ X^{(1)}X^{(2)^T} \right\} \\
= \Sigma_1 + \Sigma_2 + (M_1 - M_2)(M_1 - M_2)^T
\]
where \( X^{(i)} \)'s are the samples from class \( i (i=1,2) \), and \( X^{(1)} \) and \( X^{(2)} \) are assumed to be independent.

In order to formulate a criterion for class separability, we have to calculate a number from these matrices. The number should be larger when the between-class scatter is larger or the within-class scatter is smaller. Therefore, one possible criterion is:

\[
J = \frac{\text{tr} S_b}{\text{tr} S_w}
\]

where \( J \) is the separability criterion and \( \text{tr} \) is the trace of the matrix.

5.3.2 Reduction of the Number of Feature Vectors

Five feature values were extracted from three groups of sample shapes. The first group was 57 sample shapes that were from a box of potatoes especially selected to compete in a product fair for "best in class". The second group was 161 sample shapes that were inspected as "good" shapes by five human inspectors from 240 random samples. And the third group is 11 sample shapes, which were inspected as "bad" shapes by five human inspectors.

Let us define the terminology of \( J_{f_1,f_2,\ldots,f_n}(c_1,c_2) \) as the separability between two patterns, \( c_1 \) and \( c_2 \), with \( n \) feature values (\( f_1, f_2, \ldots, f_n \)). The separability between the three groups of potatoes is tabulated in Table 5.5. It is noted that the separability between the first and third groups increased as the number of feature vectors decreased from 5 (\( f_1, f_2, f_3, f_4, f_5 \)) to 2 (\( f_1, f_4 \)). Also, the separability between the second and third groups increased as the number of feature vectors decreased.
Table 5.5 Separability between groups with reduction of the number of feature vectors, where c1: 1\textsuperscript{st} group sample, c2: 2\textsuperscript{nd} group sample, c3: 3\textsuperscript{rd} group sample, f1: SSE, f2: slope, f3: concavity, f4: core, and f5: elongation

<table>
<thead>
<tr>
<th></th>
<th>$J(c_1,c_3)$</th>
<th>$J(c_2,c_3)$</th>
<th>$J(c_1,c_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{f_1,f_2,f_3,f_4,f_5}$</td>
<td>3.5156</td>
<td>2.6547</td>
<td>1.8989</td>
</tr>
<tr>
<td>$J_{f_1,f_2,f_3,f_4}$</td>
<td>8.6737</td>
<td>6.0265</td>
<td>1.6924</td>
</tr>
<tr>
<td>$J_{f_1,f_2,f_4}$</td>
<td>9.3913</td>
<td>6.2407</td>
<td>1.6539</td>
</tr>
<tr>
<td>$J_{f_1,f_4}$</td>
<td>9.9494</td>
<td>6.4845</td>
<td>1.5891</td>
</tr>
</tbody>
</table>

Table 5.6 Inspection agreement between inspectors (%)

<table>
<thead>
<tr>
<th></th>
<th>Wright</th>
<th>Clark</th>
<th>Labonte</th>
<th>Hammond</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kang</td>
<td>82.92</td>
<td>84.58</td>
<td>84.58</td>
<td>82.08</td>
</tr>
<tr>
<td>Wright</td>
<td>91.67</td>
<td>88.33</td>
<td>87.5</td>
<td></td>
</tr>
<tr>
<td>Clark</td>
<td></td>
<td>87.5</td>
<td>88.33</td>
<td></td>
</tr>
<tr>
<td>Labonte</td>
<td></td>
<td></td>
<td>80.83</td>
<td></td>
</tr>
</tbody>
</table>
On the other hand, the separability between first and second groups decreased as the number of feature vectors decreased. Thus, it can be concluded that by decreasing the number of feature vectors from \((f_1, f_2, f_3, f_4, f_5)\) to \((f_1, f_4)\), as shown in Table 5.5, the separability could be increased between "good" and "bad" shapes.

### 5.3.3 Training LVQ Net and Comparison of its Performance with Human Inspectors

The comparison between five human inspection results is tabulated in Table 5.6 with the percentage scale of agreement on 240 random samples. It showed they agreed with an average of 85.8\% (3.38\% standard deviation). The grading results of three inspectors (Wright, Clark, and Hammond) were selected to train the LVQ net, because they had the best agreement among each other.

In this section, three groups of samples were used to show the scatter plot of feature values. These groups were selected as: 1) special grading as cited in the previous section, 2) 187 samples inspected by all three selected inspectors as "good" from the 240 random sample lot, and 3) 14 samples inspected by all three inspectors as "bad" from the 204 random sample lot. Scatter plots of feature values from these three groups are presented in figure 5.15, 5.16, and 5.17. Note that only two feature values (SSE and core) were used to train the LVQ net in scatter plots.

The LVQ net was trained with a second set of "good" shape patterns and a third set of "bad" shape patterns. The LVQ net was
Figure 5.15 Scatter plot of feature values (first set versus second set), where $o = \text{first set}$ & $\times = \text{second set}$. 
Figure 5.16 Scatter plot of feature values (first set versus third set), where $o =$ first set & $x =$ third set.
Figure 5.17 Scatter plot of feature values (second set versus third set), where \( o = \text{second set} \) \& \( x = \text{second set} \).
Table 5.7 Inspection agreement between trained LVQ and human inspectors (%)

<table>
<thead>
<tr>
<th></th>
<th>Kang</th>
<th>Wright</th>
<th>Clark</th>
<th>Labonte</th>
<th>Hammond</th>
<th>avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVQ (epoch=1000)</td>
<td>72.5</td>
<td>82.08</td>
<td>80.42</td>
<td>74.58</td>
<td>77.92</td>
<td>77.5</td>
</tr>
<tr>
<td>LVQ (epoch=5000)</td>
<td>74.17</td>
<td>80.42</td>
<td>79.58</td>
<td>74.58</td>
<td>78.75</td>
<td>77.5</td>
</tr>
<tr>
<td>LVQ (epoch=10000)</td>
<td>72.92</td>
<td>83.33</td>
<td>81.67</td>
<td>76.67</td>
<td>76.67</td>
<td>78.25</td>
</tr>
<tr>
<td>LVQ (epoch=15000)</td>
<td>75.83</td>
<td>82.08</td>
<td>81.25</td>
<td>76.25</td>
<td>77.08</td>
<td>78.5</td>
</tr>
</tbody>
</table>

Table 5.8 Number of misgrading between trained LVQ net and human inspectors

<table>
<thead>
<tr>
<th></th>
<th>Kang</th>
<th>Wright</th>
<th>Clark</th>
<th>Labonte</th>
<th>Hammond</th>
<th>Total misgrading</th>
</tr>
</thead>
<tbody>
<tr>
<td># of samples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>selected as &quot;good&quot;</td>
<td>184</td>
<td>215</td>
<td>209</td>
<td>193</td>
<td>201</td>
<td>124</td>
</tr>
<tr>
<td># of samples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>selected as &quot;bad&quot;</td>
<td>56</td>
<td>25</td>
<td>31</td>
<td>47</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>Total # of Samples</td>
<td>240</td>
<td>240</td>
<td>240</td>
<td>240</td>
<td>240</td>
<td>240</td>
</tr>
<tr>
<td># of down-grading</td>
<td>21</td>
<td>29</td>
<td>27</td>
<td>25</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td># of up-grading</td>
<td>37</td>
<td>14</td>
<td>18</td>
<td>32</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Total misgrading</td>
<td>58</td>
<td>43</td>
<td>45</td>
<td>57</td>
<td>55</td>
<td></td>
</tr>
</tbody>
</table>
trained through 1000, 5000, 10000, and 15000 cycles. The average agreement between the trained LVQ net and the five human inspectors was about 78% and is tabulated in Table 5.7.

Let “down-grading” be defined when the inspector grade is “good” and trained LVQ net grade is “bad” with same sample; and “up-grading” be defined as when the inspector grade is “bad” and trained LVQ net grade is “good” with same sample. The number of misgradings (“down-grading” and “up-grading”) is tabulated in Table 5.8.

5.4 Summary and Conclusions

In this chapter, a structured light approach to 3-D sensing and low level 3-D vision was presented using the Curvature-Angular Descriptor (CAD), established in this study to extract the feature values from the stripe pattern. The sensing environment using structured-light techniques was established using a slide projector and camera. The algorithm to extract stripe patterns from an object was developed. Five feature values \((SSE, slope, concavity, core, \text{ and } elongation)\) were selected for shape classification in this application.

240 random sample shapes were selected and inspected by five human inspectors into two classes: “good” and “bad” shapes. Another set of 57 sample shapes was specially selected to compete in a product fair for “best in class”. Five feature values were extracted from sample shapes using developed algorithm. Dimensions of the feature vectors were reduced from 5 \((SSE, slope, concavity, core, \text{ elongation})\) to 2 \((SSE,\)
using a separability criterion. The results of visual grading by three inspectors out of five were selected for comparison with the LVQ net based on the inspection agreement among the five inspectors. The LVQ net was trained using 161 samples, from a total of 240, which all inspectors agreed upon. The trained LVQ net was compared with human inspections. The average inspection agreement between inspectors was about 85.8%. On the other hand, the average inspection agreement between the trained LVQ net and inspectors is about 78%.
Shape analysis is one of the important roles in systems for object recognition, matching, registration, and analysis. It is also one of the most difficult problems to solve with computer vision. In this research, a new method for anomalous shape recognition, named Curvature-Angular Descriptor (CAD), was written. It is scale, rotation, and translation invariant, properties necessary to shape recognition methods. This method was applied to closed boundary sweet potato shape recognition and low level 3 dimensional computer vision using a structured light technique. The results of the application of closed boundary shape recognition were compared to four human professional inspectors. The results showed that the CAD could do the inspection job as well as human inspectors within the subjective limits of human graders. It was also shown, based on experiments, that the CAD can be easily applied to low level 3 dimensional computer vision for the recognition of sweet potatoes using the structured light technique. Therefore, it can be concluded that the Curvature-Angular Descriptor has a great potential for anomalous shape recognition based on these two applications.

However, the refinement of second application is necessary because the result of the second application showed lower performance than the first application. There are several possible ways to improve
the performance: 1) The feature values from all stripes on an object were averaged in this project. If the feature value is selected from “worst shaped” stripe among the stripes on object, the separability of feature vectors between two classes could increase. 2) The edge data from the longitudinal center line sweet potato from structured-light could be build by connecting the midpoints on the stripes of object. This edge data could be used as a third edge combined with edge data from the first application. 3) Decreasing of stripe spacing on the structured light could increase the extractability of shape information from stripe patterns on an object. 4) The optimal feature value set ($SSE$ and $core$) was extracted from five feature values ($SSE$, $slope$, $concavity$, $core$, and $elongation$) by comparing the separability with only four combinations of feature values (see table 5.5). However, the total number of possible combination of feature value sets out of five feature values is $153 (= \sum_{n=1}^{5} n!)$. More complete testing of separability has to be done to find the combination of feature value sets that has the highest separability value.

The relationship between sine wave noise and feature values was investigated in chapter 3. As the magnitude and the frequency of the noise increases, the feature values vary. Each feature value varies differently depending on the characteristics of each feature value. The change of tolerance (defined in section 3.2.1) also has an effect on variation of the feature values in CAD. For example, if the tolerance is large, the noise that has small magnitude won’t give an effect to
variation of the feature values. However, the relationship between the
tolerance and the magnitude of the noise has not been investigated in
this study. This is left as an another future study.
REFERENCES


Program 1. Main function

%---------------------------------------------------------------
% Main function for 'Closed-boundary shape recognition', which can be executable on 'MATLAB' computing environment.

% The functions are called:
% 1) re();
% 2) step1();
% 3) seg1();
% 4) centroid();
% 5) bound();
% 6) core();
% 7) shape();
% 8) ang();
% 9) reg();

%---------------------------------------------------------------

%---------------------------------------------------------------
% Reading input image file names and storing into array

[file, r, c] = read;

for a = 1:r

%---------------------------------------------------------------
% Retrieving input images from disk

    int = step( file, a, c );

%---------------------------------------------------------------
% Segmentation of objects

    out = seg( int, 105 );

%---------------------------------------------------------------
% Finding centroid of segmented object
[x, y] = centroid(out);

% Tracing the boundary of object
bd = bound(out, x, y);

% Dividing the extracted boundary into two curves
[bd1, bd2] = core(out, bd, x, y);

% partitioning two input curve into small elements
i1 = shape(bd1, 1);
i2 = shape(bd2, 1);

% constructing the series unit vectors reference vector and obtaining the length of cross product
out1 = ang(bd1, i1);
out1 = out1 * -1;
out2 = ang(bd2, i2);

% normalizing the location of unit vectors and creating the linear regression model and extracting the feature vectors (SSE, slope, and concavity)
[ro1, coef1(a,:)] = reg(out1, i1);
[ro2, coef2(a,:)] = reg(out2, i2);

end

Program 2. Storing input image file names into array

function [file, r, c] = read()
% The purpose of this function is to storing the input file
names into array.
%
% Input variable: None.
% Output: 1) file: array storing input image names
% 2) r: number of input images
% 3) c: length of input file name
%
file = ['e21_l ';'e26_l ';'e34_l ';'e52_l ';'e93_l ';'e95_l ']
[r,c] = size(file);

Program 3. Retrieving input image from disk

function img_file = step( file, a , c )

% The purpose of this function is to retrieving input images from disk.
%
% Input: 1) file: file: array storing input image names
% 2) r: number of input images
% 3) c: length of input file name
%
% Output: 1) img_file: input image retrieved from disk
%
if file(a,c-1) == ' '
    fname = file(a,1:end-2);
elseif file(a,c) == ' '
    fname = file(a,1:end-1);
else
    fname = file(a,:);
end
Program 4. Segmentation of object

function out = seg( int , thres )
% The purpose of this function is to segment the light object from
% dark background in input image using 'thresholding' methods.
% Input: 1) int: input image retrieved from disk
% 2) thres: thresholding gray level
% Output: 1) out: output image, which is consisted pixels
% having gray level of 0 (background) and 1 (object).

[m,n] = size( int );

out = zeros(m,n);

for a = 1 : m
    for b = 1 : n
        if int(a,b) > thres
            out(a,b) = 1;
        else
            out(a,b) = 0;
        end
    end
end

Program 5. Finding centroid of segmented object

function [x_cen, y_cen] = centroid( int )
% The purpose of this function is to find the centroid of
% segmented object.
% Input: 1) int: image storing segmented objects
% Output: 1) x_cen: x coordinates of centroid in image coordinate
% 2) y_cen: y coordinates of centroid in image coordinate

[m,n] = size( int );
Program 6. Tracing the boundary of object

function out = bound( int, x, y )

% The purpose of this function is to tracing the closed boundary shape of segmented out object (sweet potato).

% Input: 1) int: image storing segmented objects
% 2) x: x coordinates of centroid in image coordinate
% 3) y: y coordinates of centroid in image coordinate
% Output: 1) out: closed-boundary shape information, containing x and y coordinates of the boundary.

[m,n] = size( int );

x = round( x );
y = round( y );

for a = y : n
    if (int(x,a)==1) & (int(x,a+1)==0)
        out(1,1) = x;
        out(1,2) = a;
        break;
    end
end
idr = 0;
p = 1;
position = 1;
c_d = 1;

while position == 1
    for a = 0:7
        dirt = a + idr;
pdirt = a + idr - 1;
        if dirt > 7
            dirt = dirt - 8;
        end
        if dirt < 0
            dirt = dirt + 8;
        end
        if pdirt > 7
            pdirt = pdirt - 8;
        end
        if pdirt < 0
            pdirt = pdirt + 8;
        end

        [x_cur, y_cur] = deci( out(p,1), out(p,2), dirt );
        [x_pre, y_pre] = deci( out(p,1), out(p,2), pdirt );

        x_cur = round( x_cur );
y_cur = round( y_cur );
x_pre = round( x_pre );
y_pre = round( y_pre );

        if p > 2
            x_d = out(p-1,1) - x_cur;
y_d = out(p-1,2) - y_cur;
c_d = sqrt( x_d*x_d + y_d*y_d );
        end
        if c_d < 0.1
            idr = idr + 1;
        elseif ((int(x_cur,y_cur)== 1 )&(int(x_pre,y_pre)==0))
            idr = dirt;
p = p + 1;
endif

endif
out(p,1) = x_cur; out(p,2) = y_cur;

break;
end

end

dist=sqrt((out(p,1)-out(1,1))*(out(p,1)-out(1,1))+(out(p,2)-
out(1,2))*(out(p,2)-out(1,2)));

if dist < 0.1
    position = 0;
end

end

Program 7. Dividing the extracted closed-boundary into two curves

function [bd1,bd2] = core(int,bd,x,y)

% The purpose of this function is to divide the closed-boundary
% sweet potato shape into two curved shapes.
%
% Input: 1) int: image storing segmented objects
% 2) bd: closed-boundary shape information, containing
% x and y coordinates of the boundary.
% 3) x: x-coordinates of centroid in image coordinate
% 4) y: y-coordinates of centroid in image coordinate
%
% Output: 1) bd1: one of divided curved shapes
% 2) bd2: one of divided curved shapes

[m,n] = size(int);
[l,k] = size(bd);

div = 20;

for a = 1:l
    rad(a,1) = sqrt( (bd(a,1)-x)*(bd(a,1)-x) + (bd(a,2)-y)*(bd(a,2)-y) );
end

for a = 2:l-1
    rad(a,1) = (rad(a-1,1)+rad(a,1)+rad(a+1,1))/3;
end
for a = 1:(l/div)
    ind(a,1) = a;
end

for a = 1:(l/div)
    st = (ind(a,1)-1)*div + 1;
    ed = ind(a,1) * div;

    [mx, tmp] = max( rad(st:ed, 1) );
    pos(a,1) = tmp + st - 1;
end

n_r = 1;
for a = 1:(l/div)-1
    for b = a+1:(l/div)
        s_p = pos(a,1); e_p = pos(b,1);
        x_d = bd(s_p,1) - bd(e_p,1);
        y_d = bd(s_p,2) - bd(e_p,2);

        crit( n_r,1 ) = sqrt( x_d*x_d + y_d*y_d );
        crit( n_r,2 ) = pos(a);
        crit( n_r,3 ) = pos(b);
        n_r = n_r + 1;
    end
end

[mx, tmp] = max( crit(:,1) );
i1 = crit(tmp,2);
i2 = crit(tmp,3);

num1 = 0;
for a = i1:i2
    num1 = num1 + 1;
    bd1(num1,1) = bd(a,1);
    bd1(num1,2) = bd(a,2);
end

num2 = 0;
for a = i1 : -1 : 1
num2 = num2 + 1;
bd2(num2,1) = bd(a,1);
bd2(num2,2) = bd(a,2);
end

for a = 1 : -1 : i2
num2 = num2 + 1;
bd2(num2,1) = bd(a,1);
bd2(num2,2) = bd(a,2);
end

Program 8. Partitioning input curve into small elements

function index = shape(bd, s_crit)
% The purpose of this function is to partition input curved shape into small elements.
% Input: 1) bd: input curved shape
% 2) s_crit: prearranged small tolerance
% Output: 1) index: locations in input array dividing input curve into small elements

index = zeros( 1,max(size(bd)) );
a_index = zeros( 1 , max(size(bd)) );
tmp = zeros( 1 , max(size(bd)) );

index(1) = 1;
index(2) = max(size(bd));

d_crit = 0;
ni = 2;

while d_crit == 0
num = 0;
for a=1:ni-1
if (index(a+1)-index(a)) > 1
[s,N] = fmidp( index(a), index(a+1), bd );
if s>s_crit

a_index(a) = N + index(a);
num = num + 1;
else
    a_index(a) = 0;
end
else
    a_index(a) = 0;
end
end

if num == 0
    d_crit = 1;
else
    tmp(1) = index(1);
t2 = 1;
for b=1:ni-1
    if a_index(b) > 0
        t2 = t2 +1;
tmp(t2) = a_index(b);
t2 = t2 +1;
tmp(t2) = index(1+b);
ni = ni + 1;
else
    t2 = t2 + 1;
tmp(t2) = index(1+b);
end
end
if d_crit == 0
    index = tmp;
end
end

Program 9. Constructing the series of unit vectors and reference vector and obtaining the length of cross product

function out = ang( bd , ind )

% The purpose of this function is to constructing the series of unit vectors and reference vector and obtaining the length of cross product.
%
% Input: 1) bd: input curved shape
% 2) ind: locations in input array dividing input curve into small elements (see page 38)
% Output: 1) out: the length of cross product

ed = max(size(ind));

vec = bd( ind(2:ed) , :) - bd( ind(1:ed-1) , : );

vr = bd( ind(ed) , :) - bd( ind(1) , : );
s = sqrt( vr(1,1).^2 + vr(1,2).^2 );
vr(1,1) = vr(1,1) / s;
vr(1,2) = vr(1,2) / s;

for a=1:max(size(vec))
    s = sqrt( vec(a,1).^2 + vec(a,2).^2 );
    vec(a,1) = vec(a,1) / s;
    vec(a,2) = vec(a,2) / s;
    out(a) = vec(a,1) * vr(1,2) - vec(a,2) * vr(1,1);
end

Program 10. Normalizing the location of series of unit vectors and creating the linear regression model and extracting the feature vectors (SSE, slope, and concavity)

function [out, outl] = reg( int , ind )

% The purpose of this function is to normalize the location of series of unit vectors and creating the linear regression model and extracting the feature vectors (SSE, slope, and concavity).

% Input: 1) int: length of cross product
% 2) ind: locations in input array dividing input curve into small elements (see page 38)
% Output: 1) out(:,1): normalized location of series of unit vectors
% 2) out(:,2): estimates of the length of cross product for an given out(:,1) (normalized location of series of unit vectors)
% 3) outl(1): correlation coefficient
% 4) outl(2): slope
% 5) outl(3): intercept
% 6) outl(4): sum of squared residuals
% 7) outl(5): concavity

145
ed = max(size(ind));

x_val = ((ind(1:ed-1)+ind(2:ed))./2) ./ ((ind(ed) + ind(ed))./2);
x_val = (x_val .* 2) - 1;

y_val = int;

x_mean = mean( x_val );
y_mean = mean( y_val );

numer = sum( (x_val - x_mean) .* (y_val - y_mean) );
den1 = sum( (x_val - x_mean) .^ 2 );
den2 = sum( (y_val - y_mean) .^ 2 );
denom = sqrt( den1 * den2 );

R = numer / denom;
B = numer / den1;
B0 = y_mean - B*x_mean;

den = sqrt(sum( ((y_val-(B0+B.*x_val)).^2) )); % .* sqrt( 1-abs(x_val).^1.5 ) ));

out(:,1) = x_val';
out(:,2) = B0 + B.*x_val';

out1(1) = R;
out1(2) = B;
out1(3) = -1*B0/B;
out1(4) = mse;

ed = max(size(y_val));

tmp = y_val(2:ed) - y_val(1:ed-1);

[out1(5),i] = min( tmp );
out1(5) = abs(out1(5));

Program 11. Finding the point in input curve, which has maximum distance from the baseline.

function [size, N] = fmidp( n1, n2, bd )

% The purpose of this function is to find the point in input curve, which has maximum distance from the base line (see page 36).
% Input: 1) nl: start point in input curve array
% 2) n2: end point in input curve array
% 3) bd: input curved shape
% Output: 1) size: the size of maximum distance from the base line
% 2) N: location of point in input array, which has
% maximum distance from base line.

v = bd(n2,:) - bd(nl,:);

sv = sqrt( v(1,1).^2 + v(1,2).^2 );

qp(:,1) = bd(n1+1:n2-1,1) - bd(n1,1);
qp(:,2) = bd(n1+1:n2-1,2) - bd(n1,2);

vqp = v(1,1) .* qp(:,2) - v(1,2) .* qp(:,1);

d = abs( vqp ./ sv );

[size,N] = max( d );
APPENDIX B

COMPUTER PROGRAMS FOR SHAPE INSPECTION USING STRUCTURED-LIGHT

Program 1. Main Function

#include "globals.h"
#include <mfggaoi.h>
#include "function.h"

/* This program is the main function. */

void main (void)
{
    /* Variable Declaration */
    GAOI *aoiptr;
    short gaoi,oaoi,paoi;
    char fname[15], fname_inp[15];
    div_t dresx,dresy;
    int i,j,k,ratio,Dx,Dy,tx,ty,thres,cnt,
        Vx[100],Vy[100],Ex[200],Ey[200],ind[100],
        Xs[20],Ys[20],Xe[20],Ye[20],dummy;
    float Dmax,out_ang[100],res[10],Rc[20],Sp[20],
        See[20],Con[20];
    FILE *fp;

    /* Initialization of Image Processing Board */
    gaoi = init_mfg();

    printf("\n Input filename which storing a image data = ");
    scanf("%s" , &fname_inp);
    printf("\n Enter file name to save = ");
    scanf( "%s" , &fname );
printf("\n Enter threshold value = ");
scanf("%d", &i);
/* Store the image data on allocated memory on system. */
mfg_im__restore( gaoi, fname_tmp );

/* Divide the allocated memory into 4 by 4. */
ratio = 2;
mfg_gaoi_getarea( gaoi, &tx, &ty, &Dx, &Dy);
dresx = div( Dx, ratio );
dresy = div( Dy, ratio );
oaoi = mfg_gaoi_fbcreate(CURRENT_F, 0, 0, dresx.quot, dresy.quot);
paoi = mfg_aoi_fbcreate(CURRENT_F, dresx.quot + 1, 0, dresx.quot * 2, dresy.quot);

/* Size down the 640 by 480 image into 320 by 240 image */
sdown( &oaoi, &gaoi, 2 );

/* Segmentation of stripes */
seg( &oaoi, &oaoi, i );

/* Thinning Process */
do{
cnt = thin( &oaoi, &oaoi, 0, 255 );
} while( cnt > 0 );

/* Obtaining the Degree of Pixels */
degree(&paoi, &oaoi, 1, Vx, Vy);
dummy = 0;
for(k=1; k<=*(Vx+0) ;k++) {
if( *(Vx+k) != 0) && *(Vy+k) != 0 ) {

/* -------------------------------
Stripe Extracting
----------------------------- */

StripeExtract(&paoi, k, Vx, Vy, Ex, Ey);

/* --------------------------------
Curve Partitioning
--------------------------------- */

Dmax = 1.5;
divide(&paoi,Ex, Ey, Dmax, ind);
if( *(ind+0) > 3 ) {

dummy = dummy + 1;

/* --------------------------------------
Obtaining the length of Cross Product
---------------------------------------- */

ang(Ex,Ey,ind,*(ind+0) ,out_ang);

/* ----------------------------------------
Normalize the location of Vectors and
Obtain the linear regression model
------------------------------------------ */

reg(out_ang, ind,*(ind+0),res,Ex,Ey);

*(Xs+dummy) = *(Ex + 1);
*(Ys+dummy) = *(Ey + 1);
*(Xe+dummy) = *(Ex + *(Ex+0) );
*(Ye+dummy) = *(Ey + *(Ex+0) );

*(Rc+dummy) = *(res + 1);
*(Sp+dummy) = *(res + 2);
*(See+dummy) = *(res + 4);
*(Con+dummy) = *(res + 5);
for(k=1;k<=dummy;k++) {
cnt = 0;
    for(j = 1 ;j <=dummy ;j++)
        if(j !=k) {
            if( (*(Xs+k) - *(Xs+j)) < 0 )
                cnt = cnt + 1;
        }
    *(ind+k) = dummy - cnt;
}

/*
Arrangement of stripes into a set in order
----------------------------------------------- */

#include "globals.h"

#include "globals.h"

/*
This function initialize the image processing board
----------------------------------------------- */

short init_mfg( void )
{
    short gaoi;

    mfg_loadcnf ("");
    mfg_err_level(2);
    mfg_init();
    mfg_wipe(0);
    mfg_setgframe(G);
    mfg_dacmode(PSEUDO_8_G);
    mfg_initluts();
gaoi = mfg_gaoi_fbcreate(CURRENT_F,0,0,Size_Dx,Size_Dy);
    return gaoi;
}

Program 3. Scaling Down of Input Image

#include "globals.h"

/*
   This function scales down the size (X by Y) of
   input image into (X/ratio) by (Y/ratio)
   Note that ration is given number into this function.
-------------------------------------------------------------------*/

void sdown( short *oaoi, short *gaoi, int ratio )
{
    WORD pix[Size_Dx+Size_Dy], wpix[Size_Dx+Size_Dy];
    int x, y, vpix, tx, ty, Dx, Dy, i, j;
    div_t dresx, dresy;

    mfg_gaoi_getarea( *gaoi, &tx, &ty, &Dx, &Dy );

    dresx = div( Dx, ratio );
    dresy = div( Dy, ratio );

    for(y=0; y<=dresy.quot; y++) {
        for(x=0; x<=dresx.quot; x++) *(wpix+x) = 0;
        for(j=y*ratio; j<y*ratio+ratio; j++) {
            mfg_rhline(*gaoi, 0, j, Dx, pix);
            for(x=0; x<=dresx.quot; x++) {
                vpix=0;
                for(i=x*ratio; i<x*ratio+ratio; i++)
                    vpix = *(pix+i) + vpix;
                *(wpix+x) = vpix;
            }
        }
    }

    for(x=0; x<dresx.quot-1; x++)
        *(wpix+x) = *(wpix+x) / (ratio*ratio);

    mfg_whline( *oaoi, 0, y, dresx.quot, wpix );
}
Program 4. Segmentation of Stripe Patterns

#include "globals.h"

/*
This function extracts out the stripes from the input image using
thresholding methods.
*/

void seg( short *oaoi, short *gaoi, int thres )
{
    WORD pix[Size_Dx+10];
    int x,y,tx,ty,Dx,Dy;

    mfg_gaoi_getarea( *gaoi, &tx, &ty, &Dx, &Dy );

    for(y=1;y<=Dy;y++) {
        mfg_rhiine( *gaoi, 0,y,Dx, pix );
        for(x=1;x<=Dx;x++) {
            if( *(pix+x) > thres )
                *(pix+x) = 255;
            else
                *(pix+x) = 0;
        }
        mfg_whline( *oaoi, 1,y,Dx, pix );
    }
}

Program 5. Thinning Algorithm

#include "globals.h"

/*
The purpose of this function is thinning the stripes in image after
segmentation by thresholding.
*/

int thin( short *oaoi, short *gaoi, int lst, int hst )
{
    WORD pix1[Size_Dx+10],
        pix2[Size_Dx+10],
        pix3[Size_Dx+10],


```c
wpix[Size_Dx+10];
int x,y,tx,ty,Dx,Dy,i,N,S,p[15], crit, cnt;

mfg_gaoi_getarea( *gaoi, &tx, &ty, &Dx, &Dy );
cnt = 0;

/* -------------------------------

  STEP 1

------------------------------- */

for(y=2;y<Dy-1;y++) {

  mfg_rhline( *gaoi, 1, y-1, Dx, pix1);
  mfg_rhline( *gaoi, 1, y , Dx, pix2 );
  mfg_rhline( *gaoi, 1, y+1, Dx, pix3 );

  for(x=2;x<Dx-1;x++) {

    p[1] = *(pix2+x);
    p[2] = *(pix1+x);
    p[3] = *(pix1+x+1);
    p[4] = *(pix2+x+1);
    p[5] = *(pix3+x+1);
    p[6] = *(pix3+x);
    p[7] = *(pix3+x-1);
    p[8] = *(pix2+x-1);
    p[9] = *(pix1+x-1);

    N = 0;
    for(i=2;i<=9;i++)
      N = N + p[i];

    S = 0;
    for(i=2;i<9;i++)
      if( (-l*p[i]+p[i+l])==hst ) S = S + 1;
    if( (N>=2*hst) 
      && (N<=6*hst) 
      && (p[1] == hst) 
      ) { 
      *(wpix+x) = lst;
    }

  }
}
```

154
cnt = cnt + 1;
}
else
    *(wpix+x) = *(pix2+x);
}
mfg_whline(*oaoi, 1, y, Dx, wpix);
}
printf("\n number of erased = %d ", cnt);

/*  ---------------------------
STEP 2
--------------------------- */

for(y=2;y<Dy-1;y++) {
    mfg_rhline(*oaoi, 1, y-1, Dx, pixl);
    mfg_rhline(*oaoi, 1, y, Dx, pix2);
    mfg_rhline(*oaoi, 1, y+1, Dx, pix3);
    for(x=2;x<Dx-1;x++) {
        p[1] = *(pix2+x);
        p[2] = *(pix1+x);
        p[3] = *(pix1+x+1);
        p[4] = *(pix2+x+1);
        p[5] = *(pix3+x+1);
        p[6] = *(pix3+x);
        p[7] = *(pix3+x-1);
        p[8] = *(pix2+x-1);
        p[9] = *(pix1+x-1);

        N = 0;
        for(i=2;i<=9;i++) N = N + p[i];

        S = 0;
        for(i=2;i<9;i++)
            if( (-1*p[i]+p[i+1])==hst ) S = S + 1;
            if( (-1*p[9]+p[2])==hst ) S = S + 1;

        if( (N>=2*hst) 
            && (N<=6*hst) 
            && (S==1) 
            155
& & (p[1] == hst)
) {
  *(wpix+x) = lst;
cnt = cnt + 1;
}
else
  *(wpix+x) = *(pix2+x);
}

mfg_whline(*gaoi, 1, y, Dx, wpix);

*oaoi = *gaoi;
printf("\n number of erased = %d ", cnt);
return cnt;
}

Program 6. Computing the Degree of Pixels

#include "globals.h"

/*----------------------------------------*/
The purpose of this function is to compute the degree of pixels
(of value of 1 or 2) by counting the number of bright pixels in its
3 x 3 neighborhood.
----------------------------------------*/

void degree(short *out, short *inp, int rad, int *Vx, int *Vy) 
{
  WORD wpix[Size_Dx+10], cpix[Size_Dx+10], al[12];
  int x,y,tx,ty,Dx,Dy,i,cnt,ttx,tty,tDx,tDy, num_v;
  mfg_gaoi_getarea(*inp, &tx, &ty, &Dx, &Dy);
  mfg_gaoi_getarea(*out, &ttx, &tty, &tDx, &tDy);
  num_v = 0;
  for(y=rad;y<=Dy-rad;y++) 
  {

mfg_rhline( *inp, 0, y, Dx, cpix );

for(x=rad;x<=Dx-rad;x++) {
    if( *(cpix+x) == 0 ) *(wpix+x) = 225;
    if( *(cpix+x) == 255 ) {
        *(al+1) = mfg_rpixel( *inp, x-1,y-1);
        *(al+2) = mfg_rpixel( *inp, x ,y-1);
        *(al+3) = mfg_rpixel( *inp, x+1,y-1);
        *(al+4) = mfg_rpixel( *inp, x+1,y );
        *(al+5) = mfg_rpixel( *inp, x+1,y+1);
        *(al+6) = mfg_rpixel( *inp, x ,y+1);
        *(al+7) = mfg_rpixel( *inp, x-1,y+1);
        *(al+8) = mfg_rpixel( *inp, x-1,y );
        *(al+9) = mfg_rpixel( *inp, x-1,y-1);

        cnt = 0;
        for(i=1;i<9;i++)
            if( abs(-1*al[i]+al[i+1]) == 255 )
                cnt = cnt + 1;

        if( (cnt==4)||(cnt==2) ) *(wpix+x) = cnt;

        if( cnt==2 ) {
            num_v = num_v + 1;
            *(Vx+ num_v) = x;
            *(Vy+ num_v) = y;
        }
    }
}

mfg_whline( *out, 0, y, Dx, wpix );

*(Vx+0) = num_v;

Program 7. Stripe Extraction

#include "globals.h"

/*-------------------------------------------*/
The purpose of this function is to extract the stripe from the image after thinning process.

```c
void Stripe_Extract( short *inp, int i, int *Vx, int *Vy, int *Ex, int *Ey) {
    WORD nh;
    int tindex, j, ecrit, ex[10], ey[10];
    float td;

    *(Ex+l) = *(Vx+i);
    *(Ey+l) = *(Vy+i);
    tindex = 1;
    ecrit = 0;

    while( ecrit == 0 ) {
        *(ex+1) = *(Ex+tindex)-1;
        *(ex+2) = *(Ex+tindex);
        *(ex+3) = *(Ex+tindex)+1;
        *(ex+4) = *(Ex+tindex)+1;
        *(ex+5) = *(Ex+tindex)+1;
        *(ex+6) = *(Ex+tindex);
        *(ex+7) = *(Ex+tindex)-1;
        *(ex+8) = *(Ex+tindex)-1;

        *(ey+1) = *(Ey+tindex)-1;
        *(ey+2) = *(Ey+tindex)-1;
        *(ey+3) = *(Ey+tindex)-1;
        *(ey+4) = *(Ey+tindex);
        *(ey+5) = *(Ey+tindex)+1;
        *(ey+6) = *(Ey+tindex)+1;
        *(ey+7) = *(Ey+tindex)+1;
        *(ey+8) = *(Ey+tindex);

        td = 0.0;
        for( j = l; j <= 8; j++ ) {
            nh = mfg_rpixel( *inp, *(ex+j), *(ey+j) );
            if( nh == 4 ) {
                td = td + 1.0;
                tindex = tindex + 1;
                *(Ex+tindex) = *(ex+j);
                *(Ey+tindex) = *(ey+j);
            }
        }
    }
}
```
mfg_wpixel( *inp, *(ex+j), *(ey+j), 254 );
break;
}

if( (nh == 2)
    && ( (abs(*((Ex+1)- *(ex+j)))
        +abs(*((Ey+1)- *(ey+j))) >0 )
    )
{ 
    td = td + 1.0;
    tindex = tindex + 1;
    *(Ex+tindex) = *(ex+j);
    *(Ey+tindex) = *(ey+j);

ecrit = 1;
break;
}

}

if( td == 0.0 ) {
mfg_wpixel( *inp, *(Ex+tindex),
            *(Ey+tindex), 255 );
tindex = tindex - 1;
}

} /* End of while */

for(j=i+1;j<= *(Vx+0); j++) {
    td = abs( *(Ex+tindex) - *(Vx+j) ) +
         abs( *(Ey+tindex) - *(Vy+j) );
    if( td <=2 ) {
        tindex = tindex + 1;
        *(Ex+tindex) = *(Vx+j);
        *(Ey+tindex) = *(Vy+j);
        *(Vx+j) = 0;
        *(Vy+j) = 0;
        break;
    }
}

*(Ex+0) = tindex;
}
Program 8. Calculating the Shortest Distance between a point and a base line

#include "globals.h"

/*
  The purpose of this function is obtain the shortest distance between a point and a line.
*/

void cdist( int *Ex, int *Ey, int sp, int ep, float *max, int *loc)
{
    int i,j;
    float dist[Size_Dy],Vx,Vy,QPx,QPy;

    Vx = (float)(*(Ex+ep) - *(Ex+sp));
    Vy = (float)(*(Ey+ep) - *(Ey+sp));

    for(i=(sp+l);i<=(ep-l);i++) {
        QPx = (float)(*(Ex+i) - *(Ex+sp));
        QPy = (float)(*(Ey+i) - *(Ey+sp));
        *(dist+i) =sqrt((Vx*QPy-Vy*QPx)*(Vx*QPy-Vy*QPx))/sqrt(Vx*Vx+Vy*Vy);
    }

    *max = 0;

    for(i=(sp+1);i<=(ep-1);i++) {
        if( *(dist+i) > *max ) {
            *max = *(dist+i);
            *loc = i;
        }
    }
}

Program 9. Partition of Extracted Stripe

/*
  The purpose of this function is to split the stripe into small segmets, which can be represented as line.
*/

void divide(short *inp,int *Ex,int *Ey,float merit, int *ind)
{
int tindex, i, j, k, l, loc, num, crit, t_ind[100], tn, tcr, merit, etn, ex[10], ey[10];
float MAX, tr, tnm1, tnm2, td, tdm, tt;
FILE *fp;

tindex = *(Ex+0);

num = 2;
*(ind+1) = 1;
*(ind+2) = tindex;
crit = 0;
*(t_ind+1) = *(ind+1);

while( crit == 0 ) {
tcr = 0;

for(j=1;j<num;j++) {
dist( Ex, Ey, *(ind+j), *(ind+j+1), &MAX, &loc );

if( MAX > merit ) {
    tn = tn + 1;
    *(t_ind+tn) = loc;
    tn = tn + 1;
    *(t_ind+tn) = *(ind+j+1);

tcr = tcr + 1;
}

if( MAX <= merit ) {
    tn = tn + 1;
    *(t_ind+tn) = *(ind+j+1);
}
}

for(j=1;j<=tn;j++) *(ind+j) = *(t_ind+j);

num = tn;
*(ind+0) = num;

if(tcr > 0) crit = 0;
if(tcr == 0) crit = 1;
for( j=2; j<num; j++)
  mfg_wpixel( *inp, Ex[*(ind+j)], Ey[*(ind+j)], 0);
}

Program 10. Obtaining the length of Cross Product

#include "globals.h"

/*
The purpose of this function is to construct the series of unit vectors
(V_i, i=1,2,3,...,n-2,n-1) and the reference unit vectors V_ref. and
obtain the length of cross product of V_ref and V_i

---------------------------------------------------------------*/

void ang( int *Ex, int *Ey, int *ind, int num, float *out )
{
  int i,i1,i2;
  float vecx,vecy, vrx,vry, size;

  i1 = *(ind+num);
  i2 = *(ind+1);
  vrx = (float)(* (Ex+i1) - *(Ex+i2));
  vry = (float)(* (Ey+i1) - *(Ey+i2));
  size = sqrt( vrx*vrx + vry*vry );

  vrx = vrx / size;
  vry = vry / size;

  for(i=1;i<num;i++)
  {
    i1 = *(ind+i+1);
    i2 = *(ind+i);

    vecx = (float)(* (Ex+i1) - *(Ex+i2));
    vecy = (float)(* (Ey+i1) - *(Ey+i2));
    size = sqrt( vecx*vecx + vecy*vecy );

    vecx = vecx / size;
    vecy = vecy / size;

    *(out + i) = vecx * vry - vecy * vr;
  }
void reg( float *ang, int *ind, int num, float *out, int *Ex, int *Ey )
{
    int i;
    float x_val[100],con[100],x_mean,y_mean
         ,numer,den1,den2,mse,tx,ty;
    FILE *fp;
    for(i=1;i<num;i++)
        *(x_val+i) = (float)( *(ind+i)+ *(ind+i+1) )/2 ;
    for(i=1;i<num;i++)
        *(x_val+i) = *(x_val+i) / (float)( *(ind+num) );
    for(i=1;i<num;i++)
        *(x_val+i) = *(x_val+i) * 2 - 1;
    x_mean = 0.0;
    y_mean = 0.0;
    for(i=1;i<num;i++) {
        x_mean = x_mean + *(x_val+i);
        y_mean = y_mean + *(ang+i);
    }
    x_mean = x_mean / (float)(num-1);
    y_mean = y_mean / (float)(num-1);
    numer = 0.0;
    for(i=1;i<num;i++) {
        tx = *(x_val+i);
        ty = *(ang+i);
        numer = numer + (tx - x_mean) * (ty - y_mean);
    }
}
den1 = 0.0;
for(i=1;i<num;i++) {
    tx = *(x_val+i);
    den1 = den1 + (tx - x_mean) * (tx - x_mean);
}

den2 = 0.0;
for(i=1;i<num;i++) {
    ty = *(ang+i);
    den2 = den2 + (ty - y_mean) * (ty - y_mean);
}

*(out+1) = numer / sqrt( den1 * den2 ); /* R */
*(out+1) = pow( *(out+1) , 2 );
*(out+2) = numer / den1; /* B */
*(out+3) = y_mean - B * x_mean; /* BO */

mse = 0.0;
for(i=1;i<num;i++) {
    tx = *(x_val+i);
    ty = *(ang+i);
    mse = mse +
}

*(out+4) = mse;

for(i=1;i<num-1;i++) *(con+i) = *(ang+i+1) - *(ang+i);

*(out+5) = 2.0;
for(i=1;i<num-1;i++)
    if( *(out+5) > *(con+i) ) {
        *(out+5) = *(con+i);
    }
}
VITA

Chang-Sik Kim was born in Chunchon, South Korea, on June 3, 1970. He graduated from Chunchon High School in 1990. He attended the Kangwon National University and received a Bachelor of Engineering in mechanical engineering in February 1994. In March 1994, he was accepted as a graduate student to the Kangwon National University in agricultural engineering. In August 1995, he transferred to the Louisiana State University as a graduate student. He received a Master of Science in biological & agricultural engineering in August 1999. He continued his study pursuing a Doctor of Philosophy degree with a major in engineering science and minors in electrical engineering and mechanical engineering. He will receive the degree of Doctor of Philosophy in May 2000.