THE DEVELOPMENT OF AN IMPROVED LOW COST MACHINE
VISION SYSTEM FOR ROBOTIC GUIDANCE AND MANIPULATION OF
RANDOMLY ORIENTED, STRAIGHT EDGED OBJECTS,

A Thesis Presented to
The Faculty of the College of Engineering and Technology
of Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

by
Michael E. Miller
November 1989
ABSTRACT

An improved feature geometry based, low-cost, machine vision software system was developed by conducting a statistical reliability study of a previously developed system, evaluating alternative methods of completing the machine vision process, developing more effective edge linking and corner detection algorithms and implementing these new algorithms into a more efficient and reliable version of the existing machine vision software. The resulting system utilizes the same machine vision hardware that was utilized by the previous machine vision software system. This hardware includes an Apple IIe microcomputer, that serves as the host computer for the machine vision system; a 4K frame buffer, that is used to capture designated areas of the image obtained by the image sensor; a General Electric TN2500 Digital Camera, that serves as the sensor; a monitor and a Microbot robotic arm that is used to pick up and move an object for demonstration purposes. The resulting system is capable of recognizing a flat surfaced, randomly positioned rectangular object that contrasts well with its background within about 11 seconds with an overall reliability of about 80%. This system is equipped with an interactive user interface that allows the system to be operated and maintained by a person with very little knowledge of computers and machine vision. The interface gives the user the capability of changing parameters that affect how the machine vision system operates. These features combined with the system's ability to interactively output the data that is generated during the system's operation makes this system a valuable teaching tool to demonstrate the application of machine vision to students.
ACKNOWLEDGEMENT

I would like to thank the many instructors, friends and relatives who have provided me with the knowledge and support that has made the successful completion of this thesis project possible. Although it is not possible to mention each of these special people by name, I would like to offer thanks to my parents, Earl and Marie Miller, without whose love, encouragement and understanding I could have never attained this goal. I would like to thank my advisor, Professor Helmut Zwahlen, for his guidance and encouragement for expediency throughout this project, as well as, for the support, knowledge, and experience that he has extended to me throughout the last few years of my collegiate career. I would like to thank Sharon Stanners for graciously proofreading and helping to radically reduce the number of typing and grammatical errors that exist in this document. Finally, I would like to thank Kay Reeves, Jing Yu and Xiong Shaolin for the years of support and encouragement that they have been kind enough to provide and more importantly for their help in submitting this document from a distant location.
# TABLE OF CONTENTS

## CHAPTER 1 INTRODUCTION
1.1 Machine Vision in Education .......................... 6
1.2 Objectives ........................................... 9

## CHAPTER 2 DESCRIPTION OF THE EXISTING SYSTEM ...... 10
2.1 Hardware .............................................. 10
2.2 General Software Package ............................. 14
2.2.1 Machine Vision Process ............................. 14
2.2.1.1 Object Detection .................................. 15
2.2.1.2 Object Location .................................. 17
2.2.1.3 Edge Detection .................................... 19
2.2.1.4 Corner Detection .................................. 26
2.2.1.5 Object Identification ............................. 28
2.2.2 Set Parameters ..................................... 30
2.2.3 Object Memorization ................................. 31
2.2.4 View and Delete Objects ............................ 32
2.3 System Reliability ..................................... 32

## CHAPTER 3 SYSTEM CONSTRAINTS AND REQUIREMENTS .. 37
3.1 System Constraints ..................................... 37
3.2 System Requirements ................................... 40

## CHAPTER 4 REVIEW OF EDGE DETECTION ALGORITHMS ... 43
4.1 Mero-Vassey Operator .................................. 48
4.2 Roberts Operator ...................................... 49
4.3 Prewitt Operator ...................................... 52
4.4 Sobel Operator ....................................... 57
4.5 Laplacian .............................................. 59
4.6 Absorption Detector ................................... 63
4.7 Urkowitz's Averaging Technique ......................... 65
4.8 Comments on Other Edge Enhancement Methods ....... 67

## CHAPTER 5 REVIEW OF EDGE POINT SELECTION AND LINKING ALGORITHMS
5.1 Thresholding .......................................... 68
5.2 Classification by Direction ............................. 76
5.3 Combining Magnitude and Direction ..................... 77
5.4 Probability Relaxation .................................. 79
5.5 Line Thinning .......................................... 80
5.6 Contour Following and Edge Linking .................... 90

## CHAPTER 6 REVIEW OF CORNER DETECTION ALGORITHMS .. 93

## CHAPTER 7 DISCUSSION AND SELECTION OF ALTERNATIVES 107
7.1 Edge Enhancement ...................................... 107
7.2 Edge Point Determination and Linking ................ 109
7.3 Corner Detection ....................................... 112

## CHAPTER 8 DESCRIPTION OF DEVELOPED SYSTEM ........ 122
8.1 Machine Vision Process ................................ 123
8.1.1 Edge Linking ....................................... 123
8.1.2 Corner Detection .................................... 137
8.1.3 Linear Regression ................................... 142
CHAPTER 1
INTRODUCTION

Machine or computer vision is used to extract descriptive information about a scene from one or more images of that scene. These images are usually obtained from an optical sensor that analyzes the intensity of the illumination reflected or obscured by an object. These optical sensors convert the image data they receive to digital form by sampling the energy they receive from the image at a regularly spaced grid of points and assigning a value to each coordinate in this grid. This representation is called a digital image and each element in this grid is called a picture element or pixel. Depending upon the sensor that is used, each pixel may be quantititized into any number of levels. Commonly used optical sensors quantitize the light intensity into either 2 levels (0 or 1), in which case they are called binary cameras, or into 256, 8-bit integers (0-255) called gray levels. Once an image is obtained, the goal of a machine vision system is to extract the critical information from this digital image. However, there is no single machine vision process, instead, many different approaches have been used to extract different types of information from a scene.

Technological advances in the fields of machine vision and electronic information processing are making the use of machine vision more acceptable in today's
manufacturing environment. Meyfarth (1987) reports on a survey conducted by the University of Michigan's Industrial Development Division of the Institute of Science and Technology. This survey projected that the market for machine vision products will increase from $58 million dollars in 1985 to $457 million in 1990 (values in 1985 dollars). An increase in the use of machine vision systems has been predicted by Bader (1988), who has projected that the annual sales of complete machine vision systems will increase in volume from 285 in 1986, when the cumulative cost of systems and components was $285 million, to 2684 total systems in 1991, when the total cost of systems and components sold will grow to more than $2.7 billion.

These systems are used for a number of purposes including visual inspection, object identification and robot guidance. Machine vision systems used in industry today are used primarily for quality control. The initial investment for machine vision systems that perform visual inspection tasks are often substantial, however, they are cost effective in many situations that require almost 100 percent of the parts to be inspected, since their operating costs are relatively small compared to the cost of human inspection. Machine vision systems also perform visual inspection tasks more effectively than humans. Bader (1988) stated that automated inspection is 97% effective in eliminating defective parts while human inspection under optimum conditions is only
about 78% effective in eliminating defective parts. The superiority of using machine vision over human inspection is particularly evident when it is necessary to detect small defects or when the failure rate is very small. This is due to the fact that humans suffer a vigilance decrement when the defects are not easily detected or the ratio of defective to quality parts is small (see Wickens 1985, p. 34).

As technology in the area of machine vision improves, machine vision is becoming more feasible in flexible manufacturing workstations. In this situation the machine vision systems become the "eyes" of the workstation, allowing the workstation to sense the information it needs to adapt to different parts that are presented to it. These systems may be required to identify if a part is defective, identify the part and/or determine how the part is oriented. This information can be used by the workstation to determine the operations that must be performed to process this part.

Machine vision systems that are currently produced and used in today's industrial environment often compare parameters such as the area, perimeter, centroid and first and second moments of inertia of an object, that is viewed by a camera or other sensory device, to equivalent parameters of an ideal object, that are stored in the machine vision system's memory. Another common approach is to compare the gray scale values of the image to the gray scale values for an ideal image stored
in a computer's memory through the use of cross correlation or other matching algorithms. These approaches may be used to uniquely define an object with a known orientation. Methods such as those mentioned above, have been successfully implemented for inspection, identification and x-y location of objects. However, many of these parameters can not be effectively used to inspect or identify an object with a random orientation and therefore many currently produced commercial vision systems require the object being viewed to be presented at a known orientation.

Some research has been conducted on methods of machine vision that allow the machine vision system to determine the orientation of an object. One method involves the use of feature marks, as discussed by Chang et al. (1983). In this approach a set of high contrast marks are placed on an object in known locations. When this object is viewed by the machine vision system, it is only necessary to determine the centroids of these feature marks in order to determine the orientation of the object. It should be noted that it is necessary to use special fixturing to give the object a known orientation before the feature marks are added to the object.

A more general approach of determining the orientation and location of an object is to use feature geometry algorithms. These algorithms are designed to find the edges, corners or other geometric features of an object and describe the object based upon the location
of these features. One of the original machine vision systems employed feature geometry algorithms to attempt to describe three dimensional toy blocks and is described by Roberts (1965). More advanced applications of feature geometry algorithms have been discussed by other authors, including Perkins (1978). The machine vision system described by Perkins (1978) was able to find the position and orientation of complex objects with curved and straight edges. This machine vision system was able to determine the location and orientation of 24 steering knuckles with average errors in position and orientation of .66 cm and 1.8 degrees. However, this system required about 20 seconds of CPU time on an IBM 370/168 to process each image.

Computationally less expensive feature geometry algorithms have been described in the literature. Pingle (1969) has briefly described a system developed at Stanford University for tracing the external edge of an object and defining the position of the object for manipulation by a robotic arm. A significant amount of research has been conducted at the Stanford Research Institute (SRI) to develop a low cost vision system based upon feature geometry algorithms for industrial use. However, even these simplified algorithms have been implemented on general purpose minicomputers or special purpose machine vision processors and are relatively expensive to build and maintain.

In general, it appears that the vast majority of
machine vision research is geared towards developing complex, general machine vision algorithms that are flexible, suited to a number of tasks and implemented on expensive hardware. In reality, machine vision systems may not require this flexibility since many real world applications require a machine vision system to perform a very limited number of tasks. If in fact a number of constraints can be placed upon the tasks that the machine vision system is expected to be able to complete, the algorithms could be simpler and implemented on much less expensive hardware. One area where constrained, low cost machine vision systems have been implemented is the educational environment.

1.1 MACHINE VISION IN EDUCATION

Demonstration machine vision systems have been established at a number of universities to give students some idea of the possible applications of machine vision and possibly some hands on experience at operating a machine vision system. Due to the prohibitive cost of expensive industrial machine vision systems, the systems used in education are often very simple. For example, Douchette and Nazemetz (1985) describe a machine vision system used in the educational system at Oklahoma State University that determines the area of a viewed object and identifies the object based upon this area. A second system used at the University of Central Florida and discussed by Murat and Biegel (1985) is capable of
locating a spherical white object on a black background and commanding a robot to retrieve this object. Since this object has a predetermined size and is spherical, the machine vision system only needs to determine the centroid of the object in order to command the robot to retrieve the object.

Machine vision research has been conducted in the Industrial and Systems Engineering Department at Ohio University to develop a low-cost, microcomputer based system that may be used to expose students to machine vision and some of the applications it may have in industry. This research is discussed in two masters theses: Karr (1985) and Straumann (1987). Karr (1985) discusses the hardware that was assimilated to make machine vision possible and a simple machine vision system that could locate, count the number of dots on a die and command a robot to move the die to a predefined location. This system was limited to this particular application and restrictions were placed on the location and orientation of the dice in the viewing area.

Straumann (1987) used feature geometry algorithms to establish a microcomputer based, low cost, flexible machine vision software system that could identify, locate and determine the orientation of a number of simple straight edged, flat surfaced, randomly oriented objects that contrasted well with their background. The resulting machine vision system is much more versatile than the system developed by Karr (1985) since no
restrictions were placed on the location, orientation or position of the object. The primary goal of Straumann's thesis was to establish algorithms to show that a basic general machine vision system could be created using low-cost, mass produced, current day microprocessors. Theoretically this low-cost microcomputer-based machine vision system is capable of performing many of the tasks for demonstration purposes, that have been performed by more sophisticated commercial machine vision systems. Several of these tasks include: detection, recognition, location and determination of the orientation of simple objects. Unfortunately, general observation and preliminary reliability studies have indicated that the system, established by Straumann (1987), has a low reliability. One reliability study included 500 trial runs during which a simple white square object was placed perpendicular to the camera on a black background in one of five locations within the viewing area of the camera. During this study the machine vision system completed its task successfully only 36.4 percent of the time. This study indicated that errors occurred in the edge linking algorithm in 220 out of 500 trials, the corner detection algorithm in 39 out of the remaining 280 trials, regression failed due to division by zero in 3 out of the remaining 241 trials, errors due to lack of memory occurred in 5 out of the remaining 238 trials and errors due to misrecognition of the object occurred in 51 out of 283 remaining trials. Further, the operation of the
system seemed to be somewhat dependent upon the orientation and location of the object in the viewing area.

1.2 OBJECTIVES

In general, the objective of this thesis is to improve the reliability of the machine vision system software developed by Straumann (1987) so that a system reliability of at least 95 percent is attained for the detection, location and identification of white, flat-surfaced, rectangular objects on a dark background. Specifically, the objective is to conduct a literature review of algorithms that have been developed for machine vision that might be used by this system, design/develop new algorithms to perform edge following and corner detection, and to develop an improved machine vision system that is significantly more reliable than the machine vision system developed by Straumann (1987). In addition, the improved system should be made more user friendly by providing the user with understandable menus and more output options to aid in the understanding of the processing that is conducted by the machine vision system. The resulting system will describe only outside shapes consisting of straight lines. The development of this system is described in this thesis.
CHAPTER 2

DESCRIPTION OF THE EXISTING SYSTEM

This chapter discusses the machine vision system that has been developed by Straumann (1987) so that the design goals of this thesis may be more fully understood. This chapter has been divided into three sections. In the first section the hardware that was utilized by Straumann (1987) will be discussed. The second section explains the machine vision software system and algorithms, that were used by Straumann, and the third section describes a more thorough reliability study of the system developed by Straumann (1987). The results of this reliability study and the implications these results have upon the redesign of the existing system will also be discussed.

2.1 HARDWARE

The existing system utilizes an Apple IIe microcomputer, a General Electric TN2500 solid state digitizing camera, a 4K frame buffer and a monitor. A Teachmover Microbot is used in conjunction with the machine vision system and the Apple IIe microcomputer to pick up, move and place objects with parallel sides in a designated area for demonstration purposes.

The General Electric camera has a pixel array of 244 rows by 248 columns and is capable of detecting and transmitting up to 256 gray scale values (0-255) for each pixel. Each of these pixels has a dimension of 1.4
x $10^{-7}$ inches in the vertical direction and $1.8 \times 10^{-7}$ inches in the horizontal direction (notice that the pixels are not square). This camera has a fixed position that is 22 inches above the viewing area and views an area of 7.7 inches (18.2 degrees) by 9.9 inches (23.3 degrees).

The 4K frame buffer, that was developed by the Welding Engineering Department of the Ohio State University in Columbus Ohio, receives a digital image that is output by the TN2500 digitizing camera, stores it and then transmits it to the microcomputer. This buffer was used in the system since the camera outputs the gray scale values at a faster speed than the microcomputer can receive them. This buffer is not capable of storing more than 4 kilobytes (K) of information. Since the camera outputs an 8 bit gray-scale value for each of the 244 x 248 pixels (60,512 8-bit bytes), only about 4 thousand of the approximately 59 thousand pixel values that may be output by the camera, for any image, can be stored by the frame buffer and transmitted to the computer. However, this buffer allows the user to select the rows and columns of pixels from which data will be stored through software commands. The ability to select certain rows and columns of pixels for storage and analysis is called windowing.

The Apple IIe microcomputer is used to process the information stored by the frame buffer. This microcomputer has a 6502 microprocessor with a 1 Megahertz clock.
speed, 64K of Random Access Memory (RAM) and 16K of Read Only Memory (ROM). Like the earlier version Apple II computers (Apple II and Apple II PLUS) the Apple IIe has a 16 bit program counter. The 16 bit program counter allows one to access \(2^{16}\) or 64K of memory at one time. Figure 1 shows a memory map of the Apple IIe microcomputer. This figure shows that the Apple IIe is capable of accessing 48K of RAM (shown as hexadecimal values 0000 through C000) and the additional 16K of addresses access ROM (hexadecimal values C000 through FFFF). An additional 16K of RAM can be accessed but only through a process called bank switching. The primary disadvantage of bank switched memory is that not all of the memory can be accessed simultaneously. In the Apple IIe when one accesses the additional bank switched memory, the ROM, which contains the basic interpreter and other necessary system information, can not be accessed. However, if the programs do not require the use of the information stored in the upper 12K of ROM, as much as 16K of additional bank switched memory can be accessed. This bank switched memory is composed of two, 4K banks (bank 1 and bank 2 in Figure 1) of bank switched memory that are located between hexadecimal address D000 and DFFF. An additional 8K of bank switched memory, located from hexadecimal address E000 through FFFF can also be accessed. In the existing machine vision system the bank switched memory is not used, although practically all of the direct access RAM is and much of the data
Figure 1 Depiction of the memory layout of the Apple IIe microcomputer.
generated and used during the machine vision process is overwritten in order to avoid exceeding the available direct access RAM.

2.2 GENERAL SOFTWARE PACKAGE

The software package allows the user to execute the machine vision process, set parameters, store coordinates of new objects into memory, or view (and if necessary delete) coordinates of objects already stored in memory. This software utilizes an interactive user interface that offers a choice of these four options.

2.2.1 Machine Vision Process

The machine vision process encompasses five phases including object detection, object location, edge detection, edge following, corner detection and object identification. Each process is complex and requires the system to perform large numbers of calculations. Since the Apple IIe has a relatively low processing speed (1 MHz) and limited storage capacity (64K RAM), Straumann (1987) found it necessary to formulate constraints to reduce the processing time of the vision system. The following constraints were placed upon objects in the viewing area: 1) only one object can be present in the viewing area at any time, 2) the contrast ratio between the object and the background (dark) should be as high as possible, 3) the smallest dimension of the object has to be greater than 1 degree of the visual angle, 4) the
largest dimension of the object has to be less than 16 degrees of the visual angle, 5) the ratio between the largest dimension and smallest dimension within an object should not be greater than 7:1, 6) all edges must be straight lines.

2.2.1.1 Object Detection

A locating algorithm using a simplified version of the edge detection operator presented by Roberts (1965) was utilized to determine the location of an object's edges. In this edge detection algorithm, an element in a matrix of gray scale values is chosen. If this matrix element is in row i and column j, the Roberts gradient R(i, j) for the element (i, j) may be calculated from the following equation:

\[ R(i, j) = |G(i, j) - G(i+1, j+1)| + |G(i, j+1) - G(i+1, j)| \]

where G(i, j) represents the gray scale value for element (i, j).

To locate the object, the frame buffer initially accepts every 4th row and 4th column of gray scale values so that the entire viewing area can be represented and stored in a low resolution matrix (57 pixels x 57 pixels) which requires 3249 bytes of memory. This matrix would be similar to the matrix shown in Figure 2. Beginning at the upper left hand corner and sweeping to the right and down each row of the matrix, the algorithm calculates the Roberts gradient for each pixel in the entire matrix and a new matrix is generated (an example
of such a matrix is shown in Figure 3). Next, each value in the newly generated gradient matrix is compared to a threshold value, which is set by the machine vision operator. When two consecutive values higher than the threshold are found, it is assumed that an object is present in the viewing area and object location is attempted. However, if two consecutive values larger than the threshold are not found then it is assumed that there is no object present in the viewing area and a new set of gray scale values for the viewing area are acquired. Notice that the gradient values near the edge of the object are much larger than the gradient values that are not near the edge of the object. Again, it should be noted that in this first digital representation of the image obtained by the machine vision camera the matrix will contain a value for every fourth pixel of the camera. For this reason, the digital image is lower in resolution than the best representation the camera is capable of producing. Objects that are commonly viewed by the machine vision system are characteristically represented by about 16 to 20 pixel values.

2.2.1.2 Object Location

Once an object has been detected in the viewing area, it is necessary to determine its approximate location and dimensions. When the computer determines that an object is present, the row number where the object was initially detected is stored. The computer then
Figure 3 A matrix of Roberts Gradient values calculated from the gray scale matrix shown in Figure 1.
begins at the bottom of the first column and scans up each column until two additional consecutive values larger than the threshold value are found. This is illustrated as search direction 2, in Figure 4, which depicts the object location process. The column number of the matrix is stored. The computer then proceeds to the last column and begins scanning left across each row of the matrix, beginning with the last row in the matrix and advancing towards the top of the matrix (depicted as search direction 3 in Figure 4). When two additional consecutive values are found that are larger than the threshold value, the row number in which they are found is stored. The computer then scans up each column in the matrix, advancing towards the beginning of the matrix until it finds a fourth pair of consecutive values that are larger than the threshold value (search direction 4 in Figure 4). The column number, in which the first of the two values larger than the threshold are found, is stored.

When this process is complete, two row and two column coordinates have been stored. Consecutive horizontal and vertical coordinates are then paired to yield the coordinates of four corner points. These four corner points roughly correspond to the corner points of a rectangle that encloses the object.

2.2.1.3 Edge Detection

After the object has been located, a new window of
Figure 4 A pictorial representation of the search directions utilized to locate the object in the original viewing area.
gray scale values is recorded to include a boundary of 2 pixel values around the four points that were found in the last step. This is illustrated by the box shown in Figure 3. Recording only the gray scale values enclosed within the area where the object is located allows the machine vision system to acquire a higher resolution image of the object than was acquired during the object location process. Figure 5 shows a typical matrix of gray scale values for such a higher resolution image. The program then calculates the Roberts gradient for each element in the matrix. A matrix of Roberts gradient values calculated from the gray scale values in Figure 5 is shown in Figure 6. In order to reduce the amount of data to be stored, as the Roberts Gradient for each element is calculated, the computer compares each of these values to a threshold that is set by the machine vision system operator. Only the positions of the matrix elements whose values are larger than the threshold value are stored as part of a list of points that might lie on the edge of the object. This list of edge points is termed a "candidate edge list". The program then attempts to link the stored location values together to determine the shape of the object, and to filter out local disturbances that have been enhanced by the Roberts Gradient to the point that the Roberts Gradient Values are greater than the threshold.
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 |   |   |   |   |   |   |   |   | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 6 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 7 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 8 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 9 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 10 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 12 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 13 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 14 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 15 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 16 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 17 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 18 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 19 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 20 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 21 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 22 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 23 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 24 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 25 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 26 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 27 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 28 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Figure 5 A typical matrix of gray scale values for a square in a reduced image area.
Figure 6  A matrix of Roberts Gradient values calculated from the gray scale matrix shown in Figure 5.
2.2.1.3 Edge Following

The algorithm developed by Straumann (1987) to follow the object's edges always begins by sorting through the candidate edge list and storing the point in the candidate edge list that is nearest the upper left hand corner of the viewing area into a new edge list called the "linked edge list". The linked edge list is an ordered list of edge points that represent a chain of points along an edge or contrast boundary of the object. After the starting point has been determined, this point may be thought to occupy the center position (position p) in the 3x3 matrix shown in the top left corner of Figure 7. From this position the algorithm checks the position diagonally towards the lower right that is depicted as position 1 in Figure 7. If the coordinates of this point are not stored in the candidate edge list then the algorithm checks positions 2, 3, 4, 5, and 6 respectively. When the coordinates of one of these positions are found in the list of candidate edge points this position is stored in the linked edge list. As the coordinates of a point are stored in the linked edge list the point's coordinates are removed from the candidate edge list so that the algorithm can not back step over this position. When a point is stored in the linked edge list, it becomes the new center position (position p) in the 3x3 matrix and the point that previously occupied position p assumes the blackened position in this 3x3 matrix. The algorithm then follows the edges of the
Figure 7 The priority that the edge linking procedure uses to search for a candidate neighbor of point p when point p and the blackened point have been selected as edge points.

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>P</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
object rotating the 3x3 matrix such that position 1 is always in a straight line with the two previously stored positions. This process is continued until the edge following algorithm follows the edge of the object back to within 2 pixel positions of its original starting point. If the algorithm is unable to follow the edges back to the starting point due to a missing adjacent pixel in the edge candidate list then the process is terminated and the error message "no link found in edge following" is displayed.

2.2.1.4 Corner Detection

It is not only necessary to follow the edges around the object but to segment one straight edge from another straight edge. In the system developed by Straumann (1987), the first point that is stored in the linked edge list is considered a corner point. Successive corner points are found by searching for points in the edge list where the x or y components of the edge direction occurs, with respect to the last corner. As each point is linked around the edge, the algorithm assumes a corner if either of the following two inequalities are satisfied.

\[
| (X_n - X_{1c})*3 - (X_n - X_{n-3})*n | > 1.5 * n + 2
\]

\[
| (Y_n - Y_{1c})*3 - (Y_n - Y_{n-3})*n | > 1.5 * n + 2
\]

Where \( X_n \) represents the current x position, \( X_{1c} \)
represents the x position of the last corner, $n$ represents the number of pixels stored since the last corner, $Y_n$ represents the current y position and $Y_{1c}$ represents the y position of the last corner.

The function of these equations may be more easily understood if they are expressed as:

$$\left| \frac{(X_n - X_{1c}) \times 3 - (X_n - X_{n-3}) \times n}{1.5n} \right| \leq \frac{2}{1.5n}$$

and

$$\left| \frac{(Y_n - Y_{1c}) \times 3 - (Y_n - Y_{n-3}) \times n}{1.5n} \right| \leq \frac{2}{1.5n}$$

From this form of the equations, it can be seen that these equations detect when the x or y direction, indicated by the current point and a point three places earlier in the linked edge list, is significantly different than the x or y direction indicated by the current point and the first point in the current edge. In order for these equations to classify the current point as a corner point these changes must be greater than a threshold. This threshold changes with the length of the edge (i.e. a point is more easily classified as a corner point as the edge becomes longer).

After the exterior edge of the object has been stored in memory the algorithm removes the coordinates of the pixel positions that are within two positions of the pixel positions that remain in the linked edge list. This was designed to remove the coordinates of all the pixel positions, whose gradient values were affected by the exterior edge of the object, from the edge list.
The algorithm then assumes that there is a subshape present on the object if there are more than three pixels remaining in the edge list (A subshape is an area within the exterior border of the object that returns a level of illumination back to the sensor that is different from the illumination that is returned to the camera by the surface of the object inside the edge of the object. This might be caused by things such as holes in the object.) In the presence of a subshape, the algorithm attempts to link its edges in the same manner as it did the exterior edges of the object. After the subshape's edges have been linked and stored in the linked edge list, the algorithm checks for additional subshapes. When all of the edges of all of the subshapes have been placed into an edge list, a linear least squares regression calculation is performed upon each set of edge points enclosed by a pair of corner points. The slopes and intercepts obtained for each edge are then utilized to determine new corner points. By using a pair of linear regression equations to calculate the position of each corner, the location of the corner is based upon all of the points along the edge and is likely to be more representative of the actual corner of the object than the originally detected corner point.

2.2.1.5 Object Identification

The object can be identified after the corner points of the object have been calculated. The object is iden-
tified by comparing the coordinates of the object with the coordinates of objects that have been stored in memory. However, the object may have an infinite number of possible positions and orientations within the viewing area giving the object an infinite number of corner coordinates, therefore, the corners of the object are calculated with respect to an object centered coordinate system. To accomplish this, each side's length is computed using the object's corner coordinates. Next, the lengths are ranked and the point of origin is found by using the following five rules: 1) If all sides have the same length, and the object has no subshapes, then the origin can be chosen at any corner; 2) If all sides have the same length and the object has a subshape (for example a square hole), then the origin is chosen at the corner with the shortest distance to the subshape; 3) Select the longest side as the base; 4) Select the origin such that the shortest side is closest to the y-axis subject to rule 3; 5) If the two sides adjacent to the base are of equal length, the origin is chosen such that the side counter clockwise to the base is closest to the y-axis.

When the origin has been found, the corner points are calculated with respect to this origin. These values are then compared to the coordinates of the object in the machine vision system's memory. If the coordinates of the object being viewed do not vary by more than a user specified tolerance from the coordinates of
the object in memory, the object in the viewing area is identified as this object. Notice that with this algorithm the object is not only identified as a certain object but its dimensions are also checked. One of three situations can occur at this stage: 1) The object is identified, it has been designated as a retrievable object during object memorization and it has two edges with approximately equal slopes (i.e. it has parallel sides). In this case the steps for each of the 6 motors in the robot are calculated and the robot is instructed to retrieve the object and place it in a designated location with a designated orientation. 2) The object is identified, has not been designated as a retrievable object or does not have parallel sides. In this case the object can not be retrieved by the robot's parallel grippers, however, the user is informed of the object's identity. 3) The object can not be identified. In which case the user is notified of the condition. Regardless of which situation occurs, the user is given the option of repeating the vision process.

2.2.2 Set Parameters

The set parameters module that is included in the machine vision system developed by Straumann (1987) allows a novice user to interactively manipulate parameters of the machine vision system. These parameters include the four corner points of the original window (indicated by pixel numbers), the number of pixels to be
skipped in the horizontal and vertical directions for the first picture taken by the camera, the threshold gradient values for both the low and high resolution pictures, and the object recognition tolerance. Unfortunately this module does virtually no error checking and a user may input a value that the software is incapable of handling. Such an error may cause the system to fail repeatedly, giving error messages that are not likely to be associated with the poorly chosen parameter. If this occurs, it is likely that a user may never associate the system's failure with the altered parameter and therefore, the user may never be able to recover from this error - resulting in a system that is unusable.

2.2.3 Object Memorization

The object memorization module allows the machine vision system to memorize what an object looks like. This module is executed very easily and is very interactive. The module executes the same algorithms as were discussed in the machine vision process (section 2.1), however, the new image is not compared with images already stored in the computer's memory. Instead the user is asked to input a title for the object and to indicate if robot retrieval is possible. If the entire process is conducted without failure, the new object's corner coordinates are found by the machine vision system, and stored with the object's title. This enables
the user to continually update the knowledge base of the machine vision system.

2.2.4 View and Delete Objects

The view and delete objects module of the machine vision system allows the user to interactively view objects titles and the corner coordinates of the objects that have been previously stored in the machine vision system's memory and to delete any of the objects from the computer's memory. This process allows the machine vision system operator to delete any objects that are no longer needed.

2.3 System Reliability

A reliability study was conducted by the author to determine the reliability of the present machine vision system and the errors that occur during this system's operation. This study was designed to determine the system's reliability in locating and recognizing two simple, white, flat surfaced objects against a dark background. These objects had a depth of .75 inches and presented a surface to the image sensor that was a .75 inch flat surfaced, square for one object and a .75 by 1.5 inch rectangular for the remaining object. These objects were placed with one of six angles (0, 2, 4, 8, 32 or 45 degrees with respect to the digital camera) and three positions (see Figure 8). The study included 20 observations per cell for a total of 720 observations.
This experimental design includes 3 Positions x 2 Objects x 6 Angles with 20 observations per cell for a total of 720 observations.

<table>
<thead>
<tr>
<th>Positions</th>
<th>Objects</th>
<th>Angles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Square</td>
<td>0° 2° 4° 8° 32° 45°</td>
</tr>
<tr>
<td>2</td>
<td>Rectangle</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Square</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rectangle</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8 A depiction of the experimental design and the layout of the viewing area for a reliability study to determine the reliability of and the errors that occur in the machine vision system created by Straumann (1987).
(20 observations x 2 objects x 6 angles x 3 positions). Uniform lighting conditions were maintained throughout the study, the threshold was set at a constant of 30 and the tolerance for object recognition was set at ±.1 inch.

The data obtained from this study are shown in Table 1. The results indicate that the present machine vision system was able to identify the white square correctly 33 times out of 360 trials and the rectangle 4 times out of 360 trials for an overall system reliability of 5.1%. Looking at Table 1, it can be seen that six types of errors were observed during reliability testing. The most common error, and the error that occurs earliest in the machine vision process, occurred when the edge linking procedure was unable to link the edge points in the candidate edge list into a fully connected edge list (termed "No Link" in Table 1). This error was observed 190 times for the square and 205 times for the rectangle, indicating the machine vision process will be terminated 54.9% of the time due to this error. The second error occurred when the corner detection algorithm detected more than four corner points. This error occurred 94 times for the square and 107 times for the rectangle. This error accounted for 27.9% of the system error rate. The next common error occurred when the object was not recognized. This occurred 27 times for the square and 30 times for the rectangle. The system was unable to recognize an object during 7.9% of the
Table 1  The errors that occurred and the success rate of the Machine Vision Software System Developed by Straumann (1987) for two objects, placed in one of three positions, with one of six angles based upon the reliability study.

<table>
<thead>
<tr>
<th>Location 1</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>13 18 17 17 7 18</td>
<td>13 18 16 17 8 11</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>1 0 0 3 6 1</td>
<td>1 2 4 2 9 7</td>
</tr>
<tr>
<td>Err. in Lin. Regr.</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Div. by Zero</td>
<td>0 0 0 0 0 0</td>
<td>1 0 0 0 0 0</td>
</tr>
<tr>
<td>Out of Memory</td>
<td>1 0 0 0 1 0</td>
<td>2 0 0 1 1 0</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>0 1 2 0 6 1</td>
<td>2 0 0 0 2 2</td>
</tr>
<tr>
<td>Success</td>
<td>5 1 1 0 0 0</td>
<td>1 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location 2</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>3 9 9 10 4 10</td>
<td>7 11 15 15 5 6</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>1 5 7 9 12 9</td>
<td>1 4 4 5 13 13</td>
</tr>
<tr>
<td>Err. in Lin. Regr.</td>
<td>0 0 0 0 0 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Div. by Zero</td>
<td>0 1 1 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Out of Memory</td>
<td>2 2 1 1 0 0</td>
<td>4 1 0 0 0 1</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>2 2 1 0 4 0</td>
<td>8 4 1 0 2 0</td>
</tr>
<tr>
<td>Success</td>
<td>12 1 1 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location 3</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>4 11 15 18 5 2</td>
<td>6 8 12 20 9 8</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>0 7 4 2 13 14</td>
<td>3 9 8 0 11 11</td>
</tr>
<tr>
<td>Err. in Lin. Regr.</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Div. by Zero</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Out of Memory</td>
<td>1 0 1 0 2 1</td>
<td>0 2 0 0 0 1</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>5 0 0 0 0 3</td>
<td>9 0 0 0 0 0</td>
</tr>
<tr>
<td>Success</td>
<td>10 2 0 0 0 0</td>
<td>2 1 0 0 0 0</td>
</tr>
</tbody>
</table>
trials. Three other errors were observed. These errors were, "error in linear regression", "division by zero", and "out of memory". These three errors accounted for 4.1% of the errors.

The initial object detection and window location algorithms developed by Straumann (1987) performed the function for which they were designed without failure. It should be noted that in this case a white object was used that contrasted well with the "black" background and this high target/background contrast could have been critical to the successful performance of this algorithm, however, under the conditions investigated in this thesis this part of the algorithm performed admirably. Unfortunately the remainder of the system did not display this exemplarily performance. Based upon the data collected in the reliability study, it would appear that the greatest improvements to the system can be made by altering the edge following and corner detection algorithms since these algorithms are utilized early in the machine vision process and account for a very large percentage of the errors that occur.
CHAPTER 3
SYSTEM CONSTRAINTS AND REQUIREMENTS

The reliability study, of the machine vision system developed by Straumann (1987), showed that on the average the existing system can be expected to correctly complete the machine vision process only about 5 percent of the time when trying to identify a square or a rectangle with different orientations and positions in the viewing area. This low system reliability is unacceptable. As was stated in Chapter 1 the objective of this thesis is to develop an improved machine vision software system that will be reliable enough to serve as an effective demonstration system. The first step in this development process is to define the system by establishing the constraints that have been placed upon the system, as well as, the goals and requirements of the system.

3.1 SYSTEM CONSTRAINTS

The reader should note that this thesis details the development of a software system. The hardware that is to be used by the system will be identical to the hardware that was used by Straumann (1987). Therefore, this system is constrained to using the General Electric TN 2500 digital camera as an image sensor, the 4K buffer developed by the Welding Engineering Department at the Ohio State University as an interface instrument and the Apple IIe as the processing unit. In addition, the Microbot robotic arm will be used in conjunction with the machine vision system to manipulate objects. The use of this hardware insures that the final system hardware could
be replicated for less than $10,000. However, this hardware imposes the following constraints upon the development of the system:

1) The camera can not resolve the viewing area into more than 244 by 248 pixels. To clarify this statement, assuming that the camera is set such that it views an area of 7.7 by 9.9 inches, a one inch square block whose edges are parallel to the camera's lateral plane will be viewed by about 25 pixels in one direction and by about 32 pixels in the perpendicular direction. The number of pixels that view each edge is dependant upon the orientation of the object since the pixels represent a rectangular area of the viewing area.

2) No more than 4000 pixel values can be stored from a single image by the buffer and transferred to the computer to be analyzed.

3) Not more than 48K of data and programs may be resident in the computer's random access memory and allow the system to access read only memory. If access to read only memory is eliminated the system may access up to 60K of random access memory at one time, however, when operating the system under these conditions any call to a routine in read only memory will result in a fatal error.

4) The microprocessor can not process the information presented to it faster than its 1 Mhz clock speed will allow.

5) Objects that do not have parallel sides and/or sides that are not less than two inches apart can not be manipulated.
by the robotic arm.

In order to reduce the complexity of the machine vision task, it was necessary to place the following additional constraints upon the viewing area and the objects to be placed in the viewing area:

1) The viewing distance from the camera to the top of an object must be about 22 inches.

2) Constant room lighting conditions must be maintained.

3) Only one object may be present in the camera's viewing area at any given point in time.

4) The smallest dimension of an object must be larger than one degree of visual angle (about 3/8 of an inch when the TN 2500 digital camera is positioned 22 inches above the surface of the object). The General Electric TN2500 digital camera represents this one degree of visual angle with about 5 pixels. This smallest dimension was chosen since it is unlikely that one could determine the direction of an edge with any accuracy if it is represented by less than 5 pixels.

5) The largest dimension of an object must not exceed 16 degrees of visual angle (about 6 1/3 inches when the TN 2500 digital camera is positioned 22 inches above the surface of the object). This corresponds to a square object that is viewed by 4000 pixels when one ninth of the pixels are being analyzed.

6) The ratio of the largest dimension of an object to the smallest dimension of an object must be no larger than
7) The object being viewed must have straight edges.
8) The object being viewed must contrast well with the background.
9) The surface of an object presented to the camera must be flat.
10) The object being viewed must be stationary.
11) The object being viewed must not extend beyond the boundaries of the viewing area.
12) The surface of the object being viewed will not contain areas along its external edge that may be perceived as a gap in the edge of the object due to varying luminance contrast from inconsistent lighting, color variation, etc.

3.2 SYSTEM REQUIREMENTS

Despite these constraints the machine vision system that is to be developed as a result of this thesis must fulfill the following requirements:
1) The system must be able to effectively detect, locate and identify a flat surfaced, simple object in less than one minute.
2) The system must be able to recognize, locate and command a robot to manipulate a flat surfaced, rectangular, white object that lies on a dark background with a reliability of at least 95 percent.
3) The algorithms that are used and the data that is generated must not require more storage than RAM available in the Apple IIe microcomputer (i.e. the system must not
access diskette drives or any other external storage
device that would significantly decrement the system's
speed during the machine vision process).

4) The algorithms that are used must be general enough that
any simple, straight edged, flat-surfaced object might be
identified (i.e. additional apriori information can not be
utilized). However, one should expect that the reliabil-
ity of the system will decline as the object/background
luminance contrast ratio or the sharpness of the corners
of the object being viewed is decreased (i.e. the system
is not expected to perform as reliably when viewing dark
gray octagons on a black background as it is expected to
perform when it is viewing white triangles on a black
background).

5) The system should allow the user to interactively adjust
arbitrarily established parameters, including the number
of images acquired by the digital camera, thresholds for
the viewing area and the window and the recognition toler-
ance.

6) The system should allow the user to interactively obtain
printouts of the gray scale matrices for each of the
images analyzed, any transformed matrices calculated dur-
ing the machine vision process, coordinates of the corners
of the window, graphical display of the points that may
qualify as edge points, coordinates and corner code for
the points included in the edge, calculated corner coordi-
nates, and coordinates of corners for the closest match in
the computer's memory. The user should be given the
option of printing any or all of the values mentioned after the vision process is complete.

7) The system should allow the user to interactively add, review and delete objects from the computer's memory.

8) The software must be menu based, user friendly, well documented and should not require the user to have any prior programming knowledge in order to operate the system.

9) The software should be modular to minimize the repeatability of routines used by different programs and to allow a person knowledgeable of the apple IIe assembly language to easily integrate new algorithms into the system.

The objective of this thesis is to investigate different alternative methods that these system requirements might be met, choose what appears to be the most viable method of meeting these requirements and test this method. This objective will be met by conducting a review of relevant literature, evaluating and discussing different alternatives that are identified from the literature, designing a set of machine vision algorithms that might fulfill the requirements and finally coding these algorithms into a software system that should fulfill these requirements.
CHAPTER 4

REVIEW OF ALTERNATIVE EDGE DETECTION ALGORITHMS

Straumann (1987) has shown that constrained machine vision can be conducted on an inexpensive microcomputer using simple feature geometry style algorithms. Therefore, it would appear that other simple feature geometry algorithms might be found that could be used to conduct less constrained machine vision activities on a microcomputer. Since a large number of feature geometry style machine vision algorithms have been described in the literature, it was decided that the literature review and identification of alternatives would be composed of only algorithms that could be classified as feature geometry algorithms. It appears that implementation of this type of algorithm on a microcomputer shows more promise than other types of algorithms since feature geometry algorithms have been used for a number of years and were implemented on a computer as early as 1965 (see Roberts 1965). Since these algorithms were originally developed during a time when computing power was relatively expensive, many of these algorithms are computationally less expensive than many of the algorithms that are being developed today.

The basis for feature geometry machine vision algorithms was eluded to even before 1965. A study of the visual cues used by human subjects to describe an image was conducted by Attneave (1954). After observing human subjects' responses to different images Attneave (1954)
It is evident that redundant visual stimulation results from either a) an area of homogeneous color ("color" is used in the broad sense here, and includes brightness), or b) a contour of homogeneous direction or slope. In other words, information is concentrated along the contours (i.e., regions where color changes abruptly), and is further concentrated at those points on a contour at which its direction changes most rapidly (i.e., at angles or peaks of curvature).

Therefore, feature geometry algorithms are not only valuable because they are able to economically represent the information that is present in an image by representing the image in terms of edges and corners of the objects in the image, but also because they utilize the same cues that humans rely on when they process a visual image. In turn, the output of these algorithms is easily understood by human operators. Additionally, images that are represented by the edges and corners of the objects in the image bear a strong resemblance to engineering drawings. This representation will allow engineering students to better understand how the machine vision systems might be used in the manufacturing environment and how they might be easily integrated into computer aided design and manufacturing systems.

Since the importance of describing images based upon edges and corners has been recognized for many years, many algorithms have been developed that might be used in a machine vision system to find and describe these edges and corners. The algorithms that are described in the literature do not start with a raw digital representation
of an image and yield a quantitative description of the objects in the image, instead, this process is segmented into various parts with each algorithm performing some necessary, predefined segment of the machine vision process. These algorithms can be separated into a number of categories based upon the function that they were designed to fulfill. Generally one might break the machine vision process into image enhancement, where the digital representation is altered to reduce random fluctuations in the gray levels; edge detection, in which the points along the edges of the objects in the image are chosen through some form of nonmaximum suppression; line thinning, where the edges are reduced to a set of edge points that are exactly one pixel wide; edge following, where the edge points are ordered; corner detection, where the edge is segmented into regions with like curvature and edge description, where each of the edge regions with like curvature are mathematically modeled. It is not necessary that every geometric feature based machine vision system contain modules that perform each of these functions, however, many of these functions are normally conducted by feature geometry based machine vision systems.

Due to the abundance of algorithms that have been described in the literature, it is not practical to attempt to discuss all of the various applicable machine vision algorithms in a single chapter of this thesis. Therefore, the feature geometry, machine vision algorithms discussed in this thesis have been separated into three
general classes and these algorithms are discussed in this and the following two chapters. These three classes include image enhancement, edge point determination and linking, and corner detection.

The purpose of this chapter is to discuss the algorithms that have been developed and may be used as alternative image or edge enhancement algorithms in this thesis project. This chapter will not discuss methods that require excessive memory or storage capacity to be considered practical for processing digital images on current day microprocessor based computers.

Image enhancement has been defined by Ekstrom (1984) as the processing of images to increase their usefulness. This is a very broad definition, however, it is an appropriate definition for the many image enhancement techniques that have been devised. One of the most common methods of image enhancement involves some type of spatial filtering technique. Spatial filters may be classified as low-pass, band-pass or high-pass filters.

Low-pass filters preserve the low frequency components of an image while eliminating the high frequency components. When a digital image is processed through this type of filter the digital noise in an image is reduced. However, the use of low-pass filters, such as averaging techniques, tend to smooth and blur the edges of the object. It is possible to design low-pass filters which cause less blurring than averaging. One such filter is the median filter which is usually implemented as a window that
slides from one pixel to the next in the digital representation of the image, replacing the value in the center of a predefined window with the median intensity value of the pixels within the window. Although this filter does not tend to blur the edges as much as averaging, the edges still become wider and less distinct.

Band-pass filters are used to eliminate both the high and low frequency components of an image, leaving only components with a certain band of frequencies. Since the edges are usually represented by the low frequency components and the digital noise is usually represented by the high frequency components, band-pass filters tend to eliminate the useful frequency information in an image. Therefore, this class of filters are not very useful when one wants to enhance the edges of an object in an image.

High-pass filtering emphasizes the high frequency components of a signal while reducing the low frequency components. Since edges or fine details in an image are usually represented by low frequency components in an image, high-pass filtering will tend to enhance the digital noise in an image while eliminating low frequency components of an image. One might implement a high-pass filter by subtracting a local moving average intensity value from each pixel and adjusting the output intensity values to lie about the midpoint of the digital gray scale (for a more detailed description see Green (1983), p. 65).

Many image enhancement methods have been designed specifically to enhance the edges of an object in a digi-
tal image (often called edge enhancement or edge detection methods) and have been discussed in the literature. These methods normally model a two dimensional continuous gradient operator. Their objective is to accent areas where the magnitude of the gray scale values undergo a significant change. Common local edge enhancement methods (commonly referred to as edge detection methods) often output a magnitude and direction of each point along an edge, however, the direction of the edge is often time consuming to compute and is often not very accurate. Therefore, the direction output by local edge enhancement techniques is not commonly used. The remainder of this section discusses a number of commonly used edge enhancement (or edge detection) operators.

4.1 Mero-Vassey Operator

One of the simplest edge enhancement techniques is the Mero-Vassey Operator that is discussed by Shaw (1979). This operator may be defined as:

\[ G(x,y) = f(x,y) + f(x+1,y) - f(x,y+1) - f(x+1,y+1) \]

where \( f(x,y) \) represents the gray scale value at matrix position \((x,y)\) and \( G(x,y) \) represents the value output by the Mero-Vassey operator. This operator may be used to detect edges perpendicular to the \( y \) direction. This operator may also be expressed as:

\[ G(x,y) = f(x,y) + f(x,y+1) - f(x+1,y) - f(x+1,y+1) \]

to detect or enhance edges perpendicular to the \( x \) direction.
Figure 9 shows the absolute values of a matrix generated by computing the Mero-Vassey operator for each entry in the gray scale matrix shown in Figure 5. In this particular example the Mero-Vassey Operator has been implemented such that it will enhance the vertical edges in the image. From this figure, one can see that the horizontal and vertical edges in the object are not equally enhanced. Instead, the edges that are approximately vertical are represented by a band of relatively large numbers (edge values in the neighborhood of 55 to 160), while the edges that are approximately horizontal are represented by smaller numbers (edge values range from about 10 to 70). If the Mero-Vassey operator is used to enhance the image in both the horizontal and vertical directions and the two resulting images are summed together then both the horizontal and vertical edges are enhanced as shown in Figure 10. However, this requires more than twice the number of arithmetic operations that are indicated by the form of the Mero-Vassey operator shown above, as the Mero-Vassey operator must be calculated twice for each point in the gray scale matrix and the two resulting values must be summed together.

4.2 Roberts Operator

The Roberts Operator was originally presented by Roberts (1965). This operator is commonly expressed as:

\[ G(x,y) = \sqrt{[f(x,y) - f(x+1,y+1)]^2 + [f(x,y+1) - f(x+1,y)]^2} \]

or in a computationally simpler form as:
Figure 9 A matrix of values acquired by computing the horizontal form of the Mero-Vassey Gradient operator across the matrix of gray scale matrix shown in Figure 5.
Figure 10 A matrix of values acquired by computing the horizontal and vertical forms of the Mero-Vassey Gradient across the matrix of gray scale matrix shown in Figure 5 and summing the two resulting matrices.
\[ G(x,y) = |f(x,y)-f(x+1,y+1)| + |f(x,y+1)-f(x+1,y)| \]

According to Levialdi (1983) it may also be expressed in a maximum version such as:
\[ G(x,y) = \max[|f(x,y)-f(x+1,y+1)|,|f(x+1,y)-f(x,y+1)|] \]

Yet another representation of this operator has been proposed by Herskovits and Binford (1971) and is termed coincidences predicates. In this representation a point is classified as an edge point if \([f(x,y)-f(x+1,y)] \) and \([f(x,y+1)-f(x+1,y+1)]\) are both large and of the same sign.

Figure 11 shows a matrix of Roberts Gradient values calculated from the gray scale values shown in Figure 5, using the square root version of the Roberts Gradient. A matrix of values that were calculated from the values shown in Figure 5 with the simpler form of the Roberts Gradient was shown earlier in Figure 6 and a matrix of values calculated using the max version discussed by Levaldi (1983) is shown in Figure 12. Unlike the Merox-Vassey operator, which requires one set of calculations to detect or enhance horizontal edges and another set of calculations to detect or enhance vertical edges, the Roberts Gradient Operator enhances both horizontal, vertical and to a lesser degree diagonal edges with a single pass through the matrix. Therefore, one is able to enhance the edges in an image by conducting only three arithmetic operations for each element in the gray scale matrix.

4.3 Prewitt Operator

The Prewitt Operator is shown on page 120 of Levaldi
Figure 11 A matrix of values acquired by computing the original form of the Roberts Gradient Operator discussed in Roberts (1965) for the matrix of gray scale matrix shown in Figure 5.
Figure 12 A matrix of values acquired by computing the max version of the Roberts Gradient operator for the matrix of gray scale matrix shown in Figure 5.
and may be expressed as:

\[ G(x,y) = \left\{ \begin{array}{c} [f(x-1,y-1) + f(x-1,y) + f(x-1,y+1)] - \\
[f(x+1,y-1) + f(x+1,y) + f(x+1,y+1)] - \\
[f(x-1,y-1) + f(x,y-1) + f(x+1,y-1)] - \\
[f(x-1,y+1) + f(x,y+1) + f(x+1,y+1)] \end{array} \right\} \]

This operator may also be expressed in the masks:

\[
\begin{array}{ccc}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

Figure 13 shows the results of executing the Prewitt Operator on the gray scale matrix shown in Figure 5. This operator is very similar to the Roberts Operator, with the exception that it uses nine pixel values (a 3x3 window) instead of four pixel values (a 2x2 window) to obtain the resulting gradient value. For this reason this operator filters out more of the local digital noise than the Roberts Gradient Operator. However, since the Prewitt Operator relies upon a greater number of points for information, it performs more smoothing. As a result the edges of an object in a matrix output by the Prewitt Operator will appear wider than the edges in the matrix output by the Roberts Gradient Operator. This increased smoothing not only results in a wider band of possible edge candidate points but also increases the computing capacity necessary to perform this operation (the Prewitt Operator may require one to conduct as many as 15 arithmetic operations for every matrix element). The fact that the Prewitt Operator performs more smoothing and is able to filter out more of the local noise in an image makes this
Figure 13 A matrix of values acquired by computing the Prewitt Gradient operator for the matrix of gray scale matrix shown in Figure 5.
operator superior to the Roberts Gradient Operator when the target/background contrast is small.

4.4 Sobel Operator

According to Pratt (1978) the Sobel Operator may be expressed as:

\[
G(x,y) = \{[f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)] - \\
[f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)] + \\
[f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)] - \\
[f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)]\}
\]

and the masks for the Sobel Operator may be expressed as:

\[
\begin{pmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{pmatrix}
\quad
\begin{pmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{pmatrix}
\]

Figure 14 shows a matrix of Sobel Operator values calculated from the gray scale matrix shown in Figure 5. The Sobel operator is similar to the Prewitt operator, the difference being that the gray scale values closer to the center of the Sobel Operator are weighted more heavily than the gray scale values in the corners of the gradient window. For this reason the result is more dependent upon the pixel values that are geometrically closest to the center of the mask.

Notice that the band of edge values that are enhanced by this operator is again wider than the band of edge values that is enhanced by the Roberts Gradient Operator. Further, this operator requires more basic mathematical operations to complete than the Prewitt Operator requires.
Figure 14 A matrix of values acquired by computing the Sobel Operator for the matrix of gray scale shown in Figure 5.
(The Sobel Operator requires about 18 arithmetic operations for every matrix element while the Prewitt Operator requires about 15 arithmetic operations for every pixel element). However, notice that the values output by the Sobel Operator for the edge points are higher than the values output for edge points by the Prewitt Operator while the values output by the Sobel Operator for the off edge points are actually lower than the values output by the Prewitt Operator for the off edge points. This increased power for differentiating between points that are on and off the edge of an object indicates that under conditions of low target/background contrast the Sobel Operator would likely perform better than the Prewitt Operator.

4.5 Laplacian

The Laplacian or isotropic second derivative may be represented by the continuous function

\[ G(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

A discrete representation of this for a neighborhood of four pixels is:

\[ G(x, y) = f(x-1, y) + f(x+1, y) + f(x, y-1) + f(x, y+1) - 4f(x, y) \]

This gradient operator may also be extended to eight neighbors, in which case it may be represented as:

\[ G(x, y) = f(x-1, y-1) + f(x-1, y) + f(x-1, y+1) + f(x, y-1) \\
+ f(x, y+1) + f(x+1, y-1) + f(x+1, y) + f(x+1, y+1) - 8f(x, y) \]
The mask representations of these operators are:

\[
\begin{array}{ccc}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{array}
\quad \quad \quad \quad
\begin{array}{ccc}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

(four neighbors) \quad (eight neighbors)

Matrices of 4 and 8 neighbor Laplacian values calculated from the matrix of gray scale values shown in Figure 5 are shown in Figures 15 and 16, respectively. Notice that the magnitude of the values given for pixels that lie near the edge of the object in the image are much larger than the magnitudes of the values that do not lie near the edge of the object. However, the magnitude of the pixels that lie near the edge of the object are higher for the eight neighbor Laplacian than they are for the four neighbor Laplacian. Looking at Figure 15, it can be seen that even for the relatively high target/background contrast image represented by Figure 5, the 4 neighbor Laplacian method produces edge values that are only slightly higher than the values that are produced for the off edge pixel values (the magnitude of off edge pixel values is often as high as 12 while on edge pixel values are often times less than 20). Looking at Figure 16, the edge enhancement that is performed by the 8 neighbor Laplacian appears to be superior to the edge enhancement that was performed by the 4 neighbor Laplacian, however, the differentiation between on and off edge points is not as great as the differentiation obtained with the gradient operators that were discussed earlier.
Figure 15 A matrix of values acquired by computing a four-neighbor version of the Laplacian for the matrix of gray scale matrix shown in Figure 5.
Figure 16 A matrix of values acquired by computing an eight-neighbor version of the Laplacian for the matrix of gray scale matrix shown in Figure 5.
4.6 Absorption Detector

Kittler and Paler (1983) have proposed a local edge detection operator that they term an absorption detector. This operator may be expressed in a mask such as:

\[
\begin{array}{ccc}
1 & K & 1 \\
1 & K & 1 \\
1 & 0 & 1 \\
1 & -K & -1 \\
-K & -1 \\
\end{array}
\]

Where \( K \) is a "free" parameter and can be chosen to achieve the desired performance. Figure 17 shows a matrix of values generated for the Absorption Detector where \( K \) is assigned a value of 2.

From Figure 17, one can see that this edge enhancement (detection) method enhances the edges in the image that are almost vertical to values that are much higher than the background values (edge values about 500, background values about 10). However, these edges appear to be very wide and the edges that are almost horizontal are only enhanced to values that are about half as large as the enhanced values obtained for the almost vertical edges. In fact the best performance with this operator may be achieved with edges that are not necessarily horizontal or vertical, but with edges that lie parallel to a line that is drawn through the row of \( k \) values. It would appear that given an image with edges in only one direction, this method would enhance the edges of an object to an acceptable level even when the target/background contrast ratio is very low. However, for images where edges travel in different directions the performance of this enhancement
Figure 17 A matrix of values acquired using the edge enhancement technique discussed by Kittler and Paler (1983) with $k=2$ for the gray scale matrix shown in Figure 5.
technique is drastically reduced. Looking at Figure 17, one can see that the center of the edges of the object that lie more vertical are enhanced to values that range from about 280 to about 560 while the edges that lie perpendicular to these edges are enhanced to a range of about 100 to 300. One can see that if a threshold, which was independent of the direction of the edge, was applied to this image, the number of edge points that would be above the threshold would be much larger for the edges that lie almost vertical than for the edges that lie almost horizontal.

4.7 Urkowitz's Averaging Technique

Urkowitz (1970) has criticized edge detection or enhancement methods relying totally upon differencing operations, stating that these edge enhancement methods are highly sensitive to noise. For this reason, Urkowitz (1970) proposed that some of the effects of noise might be eliminated if one were to use differences of running averages for single dimensional data or differences of averages of nonoverlapping two-dimensional neighborhoods at each point for the two dimensional case. Figure 18 shows the differences of two non-overlapping 2x2 neighborhoods of pixels. The output of this algorithm appears impressive, yielding only very small values for the pixels that are not on the edge of the object (0, 1 or 2) and higher values along the edges of the object. However, this implementation of this method appears to be very sensitive
A matrix of values obtained by using an averaging technique proposed by Urkowitz (1970) where the difference of the average of two, 2x2 neighborhoods is found for the gray scale matrix shown in Figure 5.
to the direction of the edge. This can be seen by looking at the right edge of the block in the image. This edge contains a gap up to five pixels in length for which no high values were calculated. If one were to use this method to enhance images with edges that travel in any direction then the position of the neighborhoods to be averaged would have to be chosen very carefully. Perhaps the most viable way of utilizing the recommendations made by Urkowitz (1970) is to perform averaging upon an image before applying a gradient operator.

4.8 Comments on Other Edge Enhancement Methods

It should be noted that this list of edge enhancement methods (or edge detectors) is far from being all inclusive. However, other algorithms that were found in the literature appeared to be significantly more costly in computation time and therefore will not be discussed. However, a few of these deserved to be mentioned. These include local methods employed by Kirsh and the Compass gradient (see Levialdi (1983), p. 122), regional operators including the Hummel and Hueckel operators (see Shaw 1979), methods that use different size masks including the Rosenfeld operator (see Davis (1975)), techniques that utilize a model of the edge from a least squares fit (see Nalwa and Binford (1986), techniques using a statistical classifier (see Kundu and Mitra (1987)) and techniques employing the difference of two gaussians (see Marr (1982), Canny (1983) and Huertas and Medioni (1985)).
CHAPTER 5
REVIEW OF EDGE POINT DETERMINATION AND LINKING ALGORITHMS

The edge enhancement techniques described in chapter 4 may be used to enhance the points that lie along an edge of an object in a digitized image. The algorithms discussed in this chapter attempt to separate points in the enhanced digital representation of the image that appear to represent an edge of an object (pixels that are viewing the edge of an object) from points that do not represent the edge of an object (pixels that are viewing either the interior of the object or the background).

5.1 Thresholding

Thresholding is a special case of pattern classification in which a one-dimensional feature space is used to separate two or more regions that have significantly different characteristics. Traditionally, thresholding has been performed on gray scale values to separate regions with different gray levels or illumination intensities in the image space. However, thresholds may also be used at other stages in the machine vision process, including separating the edge points from non-edge points in an image after an edge enhancement algorithm has been applied.

The concept of thresholding is very simple. A value is chosen called the threshold value. Each intensity value for the image being analyzed is compared to the threshold value. All pixels whose intensity values are below the
threshold value are assigned to one category and all pixels whose intensity values are above or equal to the threshold value are assigned to a second category. The problem that occurs during the implementation of thresholding is that it is often difficult to select an appropriate threshold value. This value must be chosen such that the image values are classified into meaningful categories. For example, one may wish to classify all of the gray scale values for an image into two categories, one that represents all pixels viewing an object and a second category representing all of the pixels not viewing an edge. Four methods of determining this threshold value were found in the literature.

The first of these methods of threshold selection assumes that the image can be described by a mixture of Gaussian distributions. If the means and standard deviations of these distributions were known then a threshold could be analytically chosen that minimizes the classification error. However, the means and standard deviations of these distributions are not normally known. In this case, cluster analysis may be performed on the feature space in order to classify the pixels. When analyzing a matrix of gray scale values that represent an image, the number of pixels that are represented by a gray level can be plotted in a histogram and each cluster will appear as a peak on the histogram. When analyzing a gray scale matrix, this peak corresponds to a dense population of near equal gray scale values. Therefore, thresholds may be
chosen at the bottom of each of the valleys in the histogram. However, in order for a threshold to be chosen based upon a histogram of gray scale values, the distribution represented in the histogram must be bi-modal. This assumption implies: 1) the image must be composed of objects and a background, each with a separate unimodal gray-level population (that is the gray level should not significantly fluctuate over the surface of the object or background), 2) the means of the separate populations should be far enough apart to produce separate peaks, 3) the standard deviations of the distributions should be small so that the distributions do not overlap, and 4) all of the distributions should be comparable in size.

Threshold selection techniques of this type have been discussed in a number of papers, most notably Weszka and Rosenfeld (1979), Weszka (1978) and Nagel and Rosenfeld (1974). When using these methods a histogram of the gray scale values is developed. The threshold is then selected in the valley between separate peaks in the distribution. If the distributions were smooth the threshold might be chosen at the bottom of the valley between the gaussian distributions by computing the second difference of the histogram and specifying zero crossings of an arbitrary magnitude as valleys in the original distribution. However, if random peaks and valleys do exist in the frequency distribution then this approach may classify the noise as an additional distribution that does not exist.

An approach that attempts to filter out random peaks
and valleys in the frequency distribution is discussed on page 267 of Rosenfeld and Kak (1976). In this approach the two highest local maxima of the distribution that are some minimum distance apart are found (The specification that these peaks must be some minimum distance apart is added to prevent local irregularities near the top of a single major peak from being detected as a separate major peaks.) The lowest point that occurs in the distribution between these two maxima may be used as the threshold. The ratio of the magnitude of each of the maxima to the magnitude of the minima may be used to assess the flatness of the histogram. However, if this technique is to be used, the number of gaussian distributions that add to the overall distribution must be known.

Determining a threshold by assuming that an image is composed of a number of gaussian distributions is normally discussed when setting a threshold to separate pixels that view an object from pixels that view the background based upon differences in gray levels. It may be feasible to use this method to separate pixels that view the object's edges from pixels that do not view the object's edges. This might be accomplished by applying this threshold selection technique after the edges of an object in an image have been enhanced. Figures 19a and 19b show histograms of the magnitudes of the enhanced image values that were output by each of the edge enhancement techniques discussed in the previous chapter. From these figures, one can see that none of the histograms of the enhanced edge
LEGEND
\(\bar{X}\) - Average Gradient Value
S - Standard Deviation of the Gradient Values within the image
MAX - Maximum Gradient Value

\[ \begin{align*}
\bar{X} &= 12.0 \\
S &= 21.0 \\
MAX &= 162
\end{align*} \]

\[ \begin{align*}
\bar{X} &= 10.0 \\
S &= 20.0 \\
MAX &= 162
\end{align*} \]

\[ \begin{align*}
\bar{X} &= 17.0 \\
S &= 26.0 \\
MAX &= 166
\end{align*} \]

\[ \begin{align*}
\bar{X} &= 9.5 \\
S &= 16.0 \\
MAX &= 115
\end{align*} \]

\[ \begin{align*}
\bar{X} &= 8.3 \\
S &= 13.0 \\
MAX &= 83
\end{align*} \]

Figure 19a Frequency distributions obtained by classifying the outputs of five of the edge enhancement methods discussed in section 4.1 into 25 equidistant classes.
Figure 19b Frequency distributions obtained by classifying the outputs of the remaining six edge enhancement methods discussed in section 4.1 into 25 equidistant classes.
values are clearly bi-modal. Since one of the basic assumptions of this type of threshold selection technique is that the image must be multi-modal, it would appear that none of the edge enhancement techniques provide a clear enough distinction between edge points and nonedge points to provide a clearly bimodal histogram in the frequency domain. For this reason, if one is to use one of the edge enhancement methods discussed in the last section to segregate edge points from non-edge points, it is not reasonable to expect useful results from this method of threshold selection.

A second method of threshold selection sets a threshold based upon the mean and standard deviation of an image. When using this method to select a threshold for edge detection, it is assumed that the major contributions to the edge strength distribution occur due to noise, with actual edges making a smaller contribution at the extremities of the distribution. In this case the edge strength distribution is assumed to be approximately normal. As an example of how this method is used, one might choose a threshold at a value equal to the average value of an image matrix plus two standard deviations. All pixels with a gradient value larger than this threshold are then classified as edge points. Assuming that the distribution of the gradient values were accurately described by a standard normal distribution, establishing the threshold at the mean plus and minus two standard deviations would imply that the designer expects about 4.6 percent of all
pixels to view an edge (see Hines and Montgomery 1980, p. 592-593 or any standard normal table). Stating that a given mean and standard deviation may be used to establish a threshold implies that one already knows the ratio of the number of pixels that lie on the edge of the object to the number of pixels that do not lie on the edge of the object. The fact that this proportion is chosen somewhat arbitrarily by the machine vision system designer decreases the likelihood that the machine vision system will be able to effectively adapt to different viewing conditions. If the vision system is being designed to view different objects with different length perimeters, where differing numbers of pixels would view the edge of the object, the assumption that a given proportion of the pixels view an edge is not valid. This incorrect assumption may result in the selection of a threshold that is perfectly acceptable for some objects and very unacceptable for other objects.

A third method of setting a threshold for edge detection is to set a threshold such that a given proportion of the edge map is retained as edge points. For example, a threshold might be set such that 5 percent of the pixels would be classified as edge points. This is basically the same approach as was just discussed except the fact that the cutoff value is arrived at by a different method. This method would require the machine vision system to perform some sort of ranking for each of the pixel values in an image to find the edge points that were within the top x
percent of the distribution and could be identified as edge points. Once again, the assumption made by this threshold selection technique limits the machine vision system's ability to adapt to objects with different length perimeters.

A fourth method of threshold selection has been discussed by Haddon (1988). In this approach a threshold is chosen based upon a model of the noise in the image that may be attributed to the image sensor. In this approach, two images are taken of the same scene and the differences in gray scale values between the two images are calculated. These differences are then used to calculate the mean and standard deviation of the noise. Since Haddon (1988) has attempted to set a threshold for edge detection, a formula is derived to allow one to set a threshold chosen on the basis of the proportion of edges due to noise that can be tolerated in the thresholded image. However, it should be noted that in order for this threshold to be of any use the lighting conditions, background and all other parameters that might attribute noise to a digital representation of an image must remain constant for all images.

5.2 Classification by Direction

The direction output by the edge detector may also be used to separate the points that are to be classified as edge points from the points that are not classified as edge points. Robinson (1976) has discussed a method in
which edge points were generated by an edge detector that provides eight compass directions. These eight directions were then examined in 3x3 blocks. If the directions were organized in any of the orientations shown in Figure 20 then the block at the center of the 3x3 matrix was considered a valid edge point.

5.3 Combining Magnitude and Direction

An algorithm that is based upon edge magnitude and direction has been described by Nevatia and Babu (1980). In their approach Nevatia and Babu (1980) relied upon six 5x5 edge masks to perform edge detection and to determine the direction of the edge for each pixel in the image. A point is said to be in the edge list if it satisfies the three following conditions: 1) The edge magnitude of the candidate pixel is larger than the edge magnitudes of its two neighbors in a direction normal to the direction of this edge (the normal to a 30 degree edge is approximated by the diagonals on a 3x3 grid); 2) the edge directions of the two neighboring pixels are within one unit (30 degrees) of that of the central pixel and 3) the edge magnitude of the central pixel exceeds a fixed threshold. If conditions 1 and 2 are satisfied, the two neighboring pixels are disqualified as being candidates for edge points. It should be noted that according to the authors, the list that is formed has a high degree of connectivity but is likely to be more than one pixel wide and further edge linking rules are required if the resulting list of
Figure 20  Compass Directions for Acceptance of Points as Valid Edge Points Given by Robinson (1976), p. 545.
edge points is to be used to describe the perimeter of an object.

5.4 Probability Relaxation

Probability relaxation is another method of determining edge points from non-edge points. Like the method discussed by Nevatia and Babu (1980), this method uses more information from a gradient matrix than is used when analyzing an image based upon just thresholds or simple gradient directions. Rosenfeld et al. (1976) has defined relaxation as "a method of using contextual information as an aid in classifying a set of interdependent objects, by allowing interactions among the possible classifications of related objects". Relaxation can be based upon probabilities or through an optimization approach. Relaxation techniques have been developed to aid edge detection and image segmentation. Algorithms to aid edge detection have been discussed by Schachter et al. (1977), Prager (1980) and Zucker et al. (1977). Relaxation Algorithms for image segmentation have been discussed by Rosenfeld and Smith (1981), Rosenfeld et al. (1976), Peleg (1980) and Bhanu and Faugeras (1982). However, the general approach that is taken is similar regardless of where it is applied.

In probability relaxation one can assume that $A_1 \ldots A_n$ is a set of objects (such as pixels) that must be placed into one of the classes $C_1 \ldots C_m$. As an initial condition the probability that each object $i$ belongs to class $j$ is assumed to be greater than or equal to zero. It is
assumed that the classes are mutually exclusive and exhaustive such that for each object \( i \) the sum of the probabilities that this object belongs to one of the \( m \) classes is equal to one. The relaxation process then iteratively adjusts the probability that each element \( i \) belongs to class \( j \) based upon a set of conditions. A survey of relaxation methods has been conducted by Davis and Rosenfeld (1981). Rosenfeld (1983) stated, "Relaxation algorithms involve a substantial amount of computation, but since this consists primarily of iterated local operations, such algorithms are very well suited for implementation on special hardware". Based upon this statement and observations made of the relaxation techniques discussed by a few of the before mentioned authors, it appears that methods of this type require too much computation to be reasonably considered as a component in a microcomputer based machine vision system.

5.5 LINE THINNING

After a list of possible edge points has been determined, it may be desirable to obtain a list of edge points that is exactly one pixel in width. The lists of edge candidate points generated by the previously discussed techniques (especially thresholding) will likely yield a list of pixels that are more than one pixel wide. Thinning algorithms have been developed to alter a list of edge points or pixels that are more than one pixel wide to yield a list of edge points that are exactly one pixel
These algorithms further reduce the amount of information that is available for later analysis. However, they preserve the general shape and central position of the object or the object's edges (depending on how they are applied). The primary reasons for applying thinning algorithms against digital representations of an object's edge are to insure that the output of later linking processes are near the center of the edge and to assist later processes to determine the number of edges in a given image.

It is important that the reader understand some of the terminology in this area. Thinning and Shrinking are irreversible operations that seek to reduce a connected region of pixels with a given property to a set of smaller size. Line thinning algorithms reduce a region to a minimal cross-sectional width while shrinking algorithms reduce the region to a single pixel. Skeletonization is a related operation that transforms a region to a stick figure representation. Line thinning algorithms may be used to skeletonize an object by thinning an object and finding a minimally connected chain of pixels. The resulting skeleton or stick figure is often sufficient to express the structural relationship of complex objects in a scene. These algorithms have been most commonly used in character recognition to reduce the representation of a character (figure) to a set of matrix elements that are exactly one element wide and lie at the center of the thresholded figure. These algorithms might also be applied in feature geometry algorithms to reduce the width
of the edges in an image to an edge that is exactly one pixel wide.

Line thinning and shrinking algorithms have been discussed by a number of authors; including Rutovitz (1966), Hilditch (1969), Rosenfeld (1970), Stefanelli and Rosenfeld (1971) and Deutsch (1972). Tamura (1978) has compared the outputs of these early algorithms. A more current survey and general comparison of line thinning algorithms has been published by Smith (1987). In order to use these methods the digital image must be divided into points that belong to the figure and points that belong to the background (or points that represent points on the edge of an object and points that represent points off of the edge of the object). These algorithms have been formulated such that a skeleton of thin lines is formed by removing points that lie on the figure until only the points that are necessary to maintain a connected figure remain. The algorithm that is used may be defined such that it retains either four or eight connectivity (that is the edge can be followed by moving in one of four directions or one of eight directions).

Hilditch (1969) has stated that a resulting skeleton should have the following properties: 1) the skeleton should consist of thin lines that are only one pixel wide, 2) the resulting thin lines should lie along the centers of the line-like parts of the subset, 3) the process should not alter the connectivity of the subset, and 4) as soon as a skeleton is obtained this skeleton should not be
eroded away by subsequent passes of the algorithm.

Rutovitz (1966) originally proposed the use of an iterative line thinning algorithm for the skeletonization of a digital image, however, this algorithm does not fulfill the requirements set forth by Hilditch (1969) since the end points of the skeleton are eliminated with each pass of the algorithm. Hilditch (1969) has established a number of criterion for the removal of a point from a "thick" figure. When considering whether a point should be removed or not, the point being considered (a1) is assumed to be surrounded by eight neighboring pixels (n1...n8) such as:

\[ a_1 \quad n_1 \quad n_2 \quad n_3 \quad n_4 \quad n_5 \quad n_6 \quad n_7 \quad n_8 \]

According to Hilditch (1969) a point may be removed if it satisfies all of the following conditions: 1) the point belongs to the "thick" region, 2) the point lies on the edge of the "thick" region and at least one of its axially adjacent neighbors (n1, n3, n5 or n7) does not belong to the "thick" region (i.e. \( \mu(p) = n_1 + n_2 + n_3 + n_4 \neq 1 \)), 3) the point is not the last remaining point of a subset (i.e. \( \Sigma n_1 \neq 1 \)), 4) the point is not the tip of the line (i.e. \( \Sigma (1-n_1) \neq 2 \)), 5) removal of the point does not alter connectivity (i.e. \( \Sigma b_i = 1 \) where \( b_i = 1 \) if \( n_{2i-1} = 1 \) and either \( n_{2i} \) or \( n_{2i+1} \) equals zero, \( b_i = 0 \) otherwise), 6) removal of the point in conjunction with any one of its neighbors that has already been removed does not alter connectivity.
(In this context the connectivity of the edge of an object is a property that describes the continuity of the edge points along the edge being thinned. Connectivity is reduced when gaps are left between pixels in the list of edge points that are being analyzed.)

A similar thinning algorithm to preserve the four neighbor connectivity of a rectangular array has been presented by Deutsch (1972). In this algorithm a point that is part of the representation of the figure is represented as a zero and a point that is part of the background is represented as a one. A crossing number \( (X) \) is then defined as:

\[
X = \sum |n_{k+1} - n_k|
\]

A point is removed if all of the following conditions hold.

1) \( X = 0, 2 \) or \( 4 \)

2) \( \Sigma n_k \neq 1 \)

3) \( n_1 \) or \( n_3 \) or \( n_5 = 0 \)

4) \( n_1 \) or \( n_3 \) or \( n_7 = 0 \)

5) If \( X=4 \) then in addition either conditions a, b and c or conditions d, e and f must hold.

a) \( n_1 \) and \( n_7 = 1 \)

b) \( n_2 \) or \( n_6 = 1 \)

c) \( n_3 \) and \( n_4 \) and \( n_5 \) and \( n_8 = 0 \)

d) \( n_1 \) and \( n_3 = 1 \)

e) \( n_4 \) or \( n_8 = 1 \)

f) \( n_2 \) and \( n_5 \) and \( n_6 \) and \( n_7 = 0 \)

This algorithm will remove the points from one side of the edge, biasing the result towards one side of the original representation of the figure. Therefore, after one iteration of this algorithm rules 3, 4 and 5 are replaced by
the following rules:

6) \( n_3 \) or \( n_5 \) or \( n_7 = 0 \)

7) \( n_5 \) or \( n_7 \) or \( n_1 = 0 \)

8) If \( X=4 \) then in addition either conditions a, b and c or conditions d, e and f must hold.
   a) \( n_5 \) and \( n_3 = 1 \)
   b) \( n_6 \) or \( n_2 = 1 \)
   c) \( n_1 \) and \( n_4 \) and \( n_7 \) and \( n_8 = 0 \)
   d) \( n_7 \) and \( n_5 = 1 \)
   e) \( n_8 \) or \( n_4 = 1 \)
   f) \( n_1 \) and \( n_6 \) and \( n_3 \) and \( n_2 = 0 \)

This second set of rules are simply the first set of rules rotated by 180 degrees. Therefore when applying this algorithm, one might use rules 1, 2, 3, 4 and 5 on a first pass; rules 1, 2, 6, 7 and 8 on a second pass. This process is then repeated, applying rules 1, 2, 3, 4 and 5 on the third pass and so on. If a point is removed from the figure then it is treated as though it never was a part of the figure. The process is iterated until the application of both sets of rules does not remove any additional points from the "thick" region. (It should be noted that rules 3, 4, 5c, 5f, 6, 7 and 8f are not exactly the same as the rules presented by Deutsch (1972), however, this representation appears to be syntactically more correct).

A BASIC program was written and executed on an IBM AT personal computer to determine the value of this line thinning algorithm and a modified 4-connected version of this algorithm. These programs are shown in Appendix A. Figure 21 shows three figures that have been thinned using the line thinning algorithm just discussed. From this figure one can see that each of the images have been
Figure 21 Illustration of the pixels removed and the pixels retained in the edge lists of three objects by a 4-neighbor thinning algorithm proposed by Deutsch (1969).
reduced to a representation that is only one pixel wide. One might also note that this implementation does not remove a single edge point that is not connected to any other points. Looking at Figure 21 again, one can also see that when a point has two adjacent neighbors then one of the neighbors are removed to retain an edge that is only one pixel wide. In the case that these three pixels represent the corner of a rectangle-like object that is perpendicular to the camera, the corner point between any two perpendicular edges will be removed. A third feature of this algorithm that may be seen in Figure 21 is when an edge list is analyzed that contains more than one edge then each edge is preserved (provided that they do not merge) and are thinned to separate edges that are exactly one pixel wide.

This algorithm may be modified to give an 8-connected figure simply by removing conditions 5b, 5e, 8b and 8e. Figure 22 shows the output of the modified 8-connected version of this line thinning algorithm for the same three edge lists shown in Figure 21. From Figure 22, one can see that the output of these two algorithms are very similar, except for the fact that the points in the rectangular corners are not removed by the 4-connected version and are removed by the 8-connected version.

These edge thinning algorithms may require a rather substantial amount of time and memory to process an edge list. In an early implementation, a search algorithm was used that required the system to search through the entire
Figure 22 Illustration of the pixels removed and the pixels retained in the edge lists of three objects by an 8-neighbor thinning algorithm.
list of edge candidate points to find the neighboring points of each of the points in the edge list. Although this approach required very little memory (only enough to hold the edge list) it required extensive amounts of processing time (an IBM AT equipped with a 80286 microprocessor operating at 10 Mhz required as much as 30 minutes to execute the interpreted IBM BASIC program for one edge list that contained 40 pixels). The implementation shown in the appendix requires the computer to store the edge list, a list of codes and a binary matrix that represents the pixels off and on the edge of the object. With this implementation the IBM AT still requires in the neighborhood of 30 seconds to execute the algorithm contained in the program, however, the computer is required to store a binary matrix equivalent in size to the original gray scale matrix that was stored by the computer.

A paper by Govindan and Shivaprasad (1987) presents a more current line thinning algorithm. In this algorithm a pixel is deleted from a figure if the following are true: 1) the crossing number (X) is equal to 1, 2) the point being examined has more than one neighbor ($\Sigma n_i > 1$), 3) the point being examined has less than 7 neighbors ($\Sigma n_i < 7$), and 4) the pixel was not declared part of the skeleton in the last pass. If rules one and two are both false then the point is declared as part of the edge for all subsequent passes. The crossing number (X) is arrived at by following the neighbors of the point being examined and counting the number of transactions from a pixel that
is present in the edge list to a pixel that is not present in the edge list.

This algorithm was also programmed in BASIC (the program listing is also exhibited in Appendix A) and used to evaluate the same three edge lists that were examined using the edge thinning algorithm proposed by Deutsch (1969). Figure 23 shows a graphical representation of the edge lists of the objects analyzed by this algorithm. The outputs of this algorithm are practically identical to the outputs of the 4-connected version of the algorithm proposed by Deutsch (1969). Although it was hoped that this algorithm would require substantially less processing time, this algorithm also required about 30 seconds to execute on the IBM AT personal computer for a list that contained 40 points.

5.6 CONTOUR FOLLOWING AND EDGE LINKING

Contour following is normally used to connect the discrete points in a digital image into a continuous sequence of boundary points that represent an edge or a line in an image. Contour following (also called bug following by Pratt (1978)) is usually applied to a list candidate edge points. This type of edge linking procedure is very simple. Beginning at a point that has been determined to be an edge point by an earlier procedure one looks for a neighboring edge point, if one is found then this point is added to the list of connected edge points and one searches for a neighbor to the new point. This is
Figure 23  Illustration of the pixels removed and the pixels retained in the edge lists of three objects by a 4-neighbor thinning algorithm proposed by Govindan and Shivaprasad (1987).
continued until there are no more neighboring points. At this point the algorithm will terminate or look for another edge candidate point that was not linked and begin following the edge again from this point. This procedure may be used to link a list on which line thinning has been performed or it can be used in conjunction with a set of rules to find a linked list of edge points from a list of edges that are more than one pixel wide.
CHAPTER 6
REVIEW OF CORNER DETECTION ALGORITHMS

Once a linked list of edge candidate points or pixels has been completed, it may be desirable to determine where corners exist based upon information in this list of linked edge candidate points (the linked list of linked edge candidates is often termed a chain-code). Corner detection algorithms allow the machine vision system to segment the linked edge list of a linear edged object into line segments. However, before one can determine the corners in an edge list, it is necessary to define a corner. In a world where all objects are composed of straight lines, one might say that corner is formed at the junction of any two intersecting straight lines with different slopes. This definition allows one to determine the position of corners from a continuous representation of a number of lines very easily. However, when dealing with a digital representation of the edges of an object, random variation is often present along the edge of an object that makes it difficult to separate a true change in the slope of a digitally represented line from a change in the slope of a digitally represented line that can be contributed solely to random variation in the position of points in the linked edge list.

Many attempts to solve this problem have been made. One of the earliest attempts to tackle this problem is described by Roberts (1965). In order to detect a corner, Roberts (1965) began by calculating the slope for the
first three points in the edge list. The slope was then calculated for these three points plus an additional point. If the calculated slope changed by more than a predetermined amount due to the addition of this one point then the point was classified as a candidate corner point. When every point had been analyzed in this fashion then the slope of each of the line segments was calculated. If the angle formed by any two line segments was within the tolerance of 180 degrees plus and minus 15 degrees then these two line segments were redefined as a single line segment. However, since it is necessary to perform many regressions to complete this process, it is very time consuming.

A survey of corner detection algorithms was conducted by Rutkowski and Rosenfeld (1978). The results of this survey indicate that of the five corner detection methods evaluated, two methods, one proposed by Rosenfeld and Johnston (1973) and one proposed by Ramer (1972), provided results that were more similar to results obtained from human subjects' judgement than the results produced by the other three methods.

The method of detecting corner points in a list of edge points that has been proposed by Rosenfeld and Johnston (1973) attempts to detect changes in the rate of change of slope. In the real continuous Euclidean plane the rate of change of slope may be expressed in terms of derivatives as \( \frac{d^2y}{dx^2}/(1+(dy/dx)^2)^{3/2} \). When dealing with a discrete representation of integer coordinate points
$P_1 \ldots P_n$ where $P_{i+1}$ is a neighbor of $P_i$, the curvature at $P_i$ can be defined by replacing the derivatives in the continuous form by differences (eg. $dx$ may be approximated by $|x_i - x_{i-1}|$ and $dy$ may be approximated by $|y_i - y_{i-1}|$). A problem arises with this approximation since successive slope angles on the digital curve could only differ by a multiple of 45 degrees. Therefore, any change in the slope, regardless of whether it was due to an actual change in the direction of the edge or to noise in the digital representation edge, would have to be interpreted as a corner. The solution proposed by Rosenfeld and Johnson (1973) is to define the slope at $P_i$ as $(y_{i+k} - y_i)/(x_{i+k} - x_i)$ for some $k>1$. The authors then continue to describe a method of defining significant curvature maxima using a variable degree of curvature. They define the k-vectors at $P_i$ as:

$$a_k = (x_i - x_{i+k}, y_i - y_{i+k})$$
$$b_k = (x_i - x_{i-k}, y_i - y_{i-k})$$

where $x_i$ and $y_i$ are the x and y coordinates of $P_i$. The cosine of the angle between $a_k$ and $b_k$ is then defined as:

$$c_k = (a_k \cdot b_k) / |a_k||b_k|$$

Figure 24 graphically displays a sequence of edge points that this corner magnitude estimator may be applied against. Looking at this figure, one can see that $c_k$ represents the cosine of the angle $\theta$ and that the distance between $P_i$ and the vector between $P_{i+k}$ and $P_{i-k}$ increases, so will the angle $\theta$. The cosine of $\theta$ ($c_k$) will always
Figure 24 Illustration of the corner detection algorithm discussed by Rosenfeld and Johnson (1973).
have a value between -1 and 1, with a value close to -1 if $a_{ik}$ and $b_{ik}$ make an angle near 0 degrees and $c_{ik}$ will have a value close to 1 if $a_{ik}$ and $b_{ik}$ form an angle near 180 degrees. When using this corner detection method $c_{ik}$ is calculated for a chosen number of $k$ (Rosenfeld & Johnson have recommended that $k$ be set equal to each number between 1 and the number of edge points divided by 10) and the maximum $c_{ik}$ for every $i$ is chosen and used to determine the location of the object's corners.

In order to evaluate the effectiveness of this algorithm for edge lists commonly generated by this system, the linked edge lists of three rectangular objects whose corners were incorrectly identified by the algorithms developed by Straumann (1987) were selected. A graphical representation of these objects are shown in Figure 25. This figure also shows the observed corner points or the points that were judged by the experimenter to be the corner points of each of the objects. These corner points were chosen based upon the object's orientation in the viewing area and the general appearance of the plotted edge lists. The algorithm proposed by Rosenfeld and Johnston (1973) was coded in Microsoft BASIC and executed on an Apple Macintosh computer with each of the three data sets shown in Figure 25. The BASIC program is shown in Appendix B. Figure 26 shows the output that was obtained from this algorithm for each of the objects shown in Figure 25. In this figure the arrows above the graph symbolize the position of the corner points that were selected
Figure 25  A graphical depiction of three edge lists obtained by the previous machine vision system and corner points subjectively selected by the author for three typical rectangular objects.
Figure 26 Graphs depicting the cornererity of each of the points in the edge lists depicted in Figure 24, calculated using a corner detection method discussed by Rosenfeld and Johnson (1973).
by the experimenter and shown in Figure 25. The curves that are shown symbolize the output of the corner detector. From this figure one can see that the maxima obtained in the output of the algorithm proposed by Rosenfeld and Johnston (1973) are within 4 pixel positions of the observed corner points for each of the three objects. Further, there are very few false maxima that approach the magnitude of the observed corner point maxima. However, a false peak that is larger than one of the peaks that exists due to the existence of a corner does exist in the center graph.

The corner detection method discussed by Ramer (1972) has been termed the arc-chord distance method. In this method a distance $D_i$ is associated with each point $P_i$ and initially set to zero. Where the chain code is defined as $P_1 ... P_n$, for each $i$ between 0 and $n-1$, a line is constructed through $P_i$ and $P_{i+h}$. For each $j$ between $i+1$ and $i+h-1$, the distance between $P_j$ and the above line, measured perpendicular to the above line, is then computed. If this distance is greater than the current value of $d_j$, $d_j$ is reset to the present distance. Corner points are then determined by performing some type of non-maximal suppression upon the distance array to separate corner points from non-corner points.

Looking at Figure 27, the distance value that is computed based upon this method may be arrived at in the following manner. For all points between $P_1$ and $P_n$, an imaginary line is drawn from the current point $P_i$ and some
Figure 27  Illustration of the distance measurement calculated in the corner detection algorithm discussed by Ramer (1972)
point \( P_{i+h} \) that is some chosen number of points \( h \) ahead of the current point. The distance \( D \) from each point \( P_j \) between \( P_i \) and \( P_{i+h} \) is then calculated and the maximum value is stored as the corner strength of the current point. In order to calculate this distance a second imaginary line (line \( H \)) is drawn from \( P_i \) to each \( P_j \) (this line is shown as the dashed line in Figure 27). The angle subtended by the two imaginary lines may then be calculated by calculating the arc cosine of \( \Theta \) from:

\[
\frac{(X_j-X_i)(X_i+h-X_i)+(Y_j-Y_i)(Y_i+h-Y_i)}{\sqrt{(X_j-X_i)^2+(Y_j-Y_i)^2}(X_i+h-X_i)^2+(Y_i+h-Y_i)^2}}
\]

(Notice that the definition of \( \Theta \) given here is equivalent to the definition of \( \Theta \) given by Rosenfeld and Johnson (1973). Where \( X \) and \( Y \) are the coordinates of point \( P \).

The distance \( D \) can be calculated for each point by multiplying the sine of the angle obtained in the previous equation by the distance from \( P_i \) to \( P_j \) (Call this distance \( J \)). The length of line segment \( J \) may be calculated from:

\[
J = \sqrt{(X_j-X_i)^2+(Y_j-Y_i)^2}
\]

Once again, a BASIC program was written on the Apple Macintosh Computer to demonstrate the usefulness of this corner detection technique. This program is shown in Appendix B. Figure 28 graphically shows the outputs that were obtained from this corner detection technique for the edge lists that were shown in Figure 25. From Figure 28, one can see that the curve obtained for each of the edge lists has four clearly defined peaks that roughly correspond to the observed corner points. One can also see that
Figure 28 Graphs depicting the cornererity of each of the points in the edge lists depicted in Figure 24, calculated using a corner detection method discussed by Ramer (1972).
the random variation within each edge list does not contribute more power to any of the curves than is generated by any of the corners in the other edge lists. Based upon these outputs, it would appear that it is possible to set a threshold that could separate the corner points from noncorner points after this corner detection algorithm had been applied to an edge list.

A corner detection method that was not discussed in the review by Rutkowski and Rosenfeld (1978) has been published by Freeman and Davis (1977). In this method it is assumed that you are given a simple closed n-link chain $A^n=C_1-1a_1=a_1a_2...a_n$. A subchain spanned by $L_j$ is then given by $L_j=C_{i-j-1}+a_i$ where $j=1,2,...n$. The $x$ and $y$ components of $L_j$ are given by $X_j=a_{ix}$ and $Y_j=a_{iy}$. The length of $L_j$ is thus given by $|L_j|=\sqrt{(X_j)^2+(Y_j)^2}$. The angle that $L_j$ makes with the $x$ axis is then given by $\Theta_j=\tan^{-1}(Y_j/X_j)$ if $|X_j|>|Y_j|$ or if $|X_j|<|Y_j|$ then $\Theta_j=\cot^{-1}(X_j/Y_j)$ (Although $\tan^{-1}(Y/X)=\cot^{-1}(X/Y)$ for all $X$ and $Y$ not equal to zero, stating this equality in this fashion eliminates any computational errors due to division by zero. An equivalent statement is to say that if $X$ is not equal to zero then theta is equal to $\tan^{-1}(Y_j/X_j)$ and theta is equal to $\tan^{-1}(0)$ for all $X$ equal to zero). The incremental curvature can then be defined as: $\delta_j=2[\Theta_{j+1}-\Theta_j+(\Theta_j-\Theta_{j-1})]/2=\Theta_{j+1}-\Theta_{j-1}$. Theoretically, an ideal corner could be easily detected since $\delta_j$ should be zero on either side of the corner and have a value not equal to zero at the corner.
A BASIC program was written and used to compute the incremental curvature as defined by Freeman and Davis (1977) for the three objects shown in Figure 25. This program can be found in Appendix B. A graphical depiction of the output from this program for the three objects depicted in Figure 25 is shown in Figure 29. From this figure one can see that a number of minima and maxima exist for each of the objects. Further one can see that significant minima and maxima do not occur for each of the observed corner points.

Freeman and Davis (1977) suggest that for a corner to be detected by this method then two runs of points at which the mean slope is below a threshold should exist and these runs should be separated by a run of appropriate length (approximately equal to the amount of smoothing used to compute the mean slope) over which the mean slope is above some threshold. Based upon the results that are shown in Figure 29, this method appears to perform significantly poorer than the method described by Rosenfeld and Johnson (1973) and the method discussed by Ramer (1972).
Figure 29 Graphs depicting the cornererity of each of the points in the edge lists depicted in Figure 24, calculated using a corner detection method discussed by Freeman and Davis (1977)
CHAPTER 7

DISCUSSION AND SELECTION OF ALTERNATIVES

Given the discussion of the various algorithms in the three previous chapters that might be used in this machine vision system, it was the goal of this thesis to define a machine vision program that fulfilled the requirements stated in Chapter 3. Since the reliability study of the previous machine vision system that was presented in Chapter 2 demonstrated that the machine vision algorithms utilized by Straumann (1987) are able to detect and place a window around simple white objects on a semi-black background with a high reliability, it was decided to retain this part of the algorithm in the present system and concentrate on improving the remaining, less reliable algorithms that made up the remainder of the previous machine vision software package. In addition, since the requirements stated the apriori conclusion that all objects would have straight edges, it was assumed that a linear regression would be utilized to model each of the object's edges. For this reason, the machine vision algorithms that are to be designed in this thesis, are expected to begin with a digital representation of a reasonably, high resolution image of an object and end when they have generated segmented lists of points that represent lists of pixels that represent the object's edges.

7.1 EDGE ENHANCEMENT (DETECTION)

When selecting an edge enhancement (detection) method a
few general criteria may be stated: 1) the enhancement method should enhance all edges regardless of their direction, 2) the enhanced edges should not contain gaps where a segment of an edge is not enhanced, 3) the pixels that view the edge of the object should be highly differentiated from pixels that do not view the edge of the object, 4) the bands of enhanced pixels should lie as close as possible to the actual position of the edge in the image, 5) the band of enhanced pixels that represent the object's edge should be as narrow as possible, 6) the enhancement technique should require a minimal amount of processing time and storage. Looking back at Chapter 4, it can be seen that the Roberts, Prewitt and Sobel operators yielded output matrices that fulfilled these first three criteria better than the other methods that were discussed (i.e. they produced outputs that contained clearly enhanced edges, without gaps and without producing edge strength values that were clearly dependent upon the direction of the edge). The bands of enhanced edge points produced by the Roberts operator were narrower than the bands of enhanced edge points that were produced by the Prewitt or Sobel operators and the Roberts operator required far fewer calculations to complete than either the Sobel or Prewitt operator. However, the matrices produced by the Prewitt and Sobel operators appeared to differentiate edge points from non-edge points more clearly that the Roberts Gradient operator. This increased differentiation becomes important when the target to background luminance contrast ratio is low, how-
ever, for white objects on a black background this increased differentiation between edge and non-edge points does not appear to be critical. Therefore, it was decided to retain the use of the Roberts Gradient in the improved system. Further, the simplified version of the Roberts Gradient operator that was used by Straumann (1987), was chosen since it requires fewer calculations to complete than any of the other versions of the Roberts Gradient operator. It should be noted that if this system were expected to perform under conditions where the target, background luminance contrast ratio was small it is expected that some other edge enhancement (detection) method would be required.

7.2 EDGE POINT DETERMINATION AND LINKING

Chapter 5 discussed various algorithms that may be used during the process of determining edge points from non-edge points and linking these points in a continuous chain that represents the object's edge. Based upon this discussion the four following feasible alternatives might be stated:

Alternative 1:
Use a revised bug following algorithm similar to the one used by Straumann (1987).

Alternative 2:
Use an adaptive thresholding algorithm in conjunction with a revised contour following algorithm.
Alternative 3:
Use an adaptive thresholding technique and edge thinning algorithm to determine edge points from non-edge points and a contour following algorithm to link the edge points into a continuous chain.

Alternative 4:
Use of a contour following technique that relies upon the direction as well as the magnitude that is output by the edge enhancement technique.

Notice that these three alternatives are very similar, with the second and third alternatives varying from the first only because an additional algorithm was added to the list of processes to be completed. Discussion of Probability Relaxation algorithms from Chapter 5 are not continued here due to the extreme amount of processing that these algorithms appear to require.

It was shown in Chapter 5 that the threshold selection techniques that were found in the literature either did not appear to be applicable to separating edge points from non-edge points or required the designer to make assumptions that extremely limited the ability of the machine vision system to adapt to different images. Therefore, it would appear that the use of one of these adaptive thresholding algorithms might be useful only to approximate the threshold value. It is not reasonable to assume that the inclu-
sion of an adaptive thresholding algorithm of this type would significantly improve the system's ability to select a threshold such that a linked, minimally thin edge list would be guaranteed. Since it was assumed that the illumination conditions would remain about constant for the machine vision system, it is expected that the magnitude of the gray scale values and the threshold value that is used to segment this representation of the image would vary by only a small amount. Therefore, it is expected that the use of an adaptive thresholding algorithm would produce threshold values that were approximately constant over time and therefore this type of algorithm would produce very little benefit. For this reason it was decided that an adaptive thresholding algorithm would not be incorporated in the system, instead the contour following algorithm would be designed to follow an edge that could be a few pixels wide.

Chapter 5 showed that the inclusion of edge thinning algorithm into the system would reduce the representation of an edge to a band of pixels that is only one pixel wide. Since this is done in a fashion such that the resulting edge points lie as close to the center of the original band of edge points as is possible, the use of an edge thinning algorithm would likely yield a list of edge points that contains less variation than the bug following algorithm utilized by Straumann (1987). This reduced variation would likely be of some value when attempting to determine the position of corners from an object's edge. The inclusion of edge thinning algorithms in this system would also allow
one to easily separate edge points that belong to one edge of an object from edge points that belong to a second edge that might appear in the digital image (a subshape or second object in the image). However, it was shown that the edge thinning algorithms require a large number of processor steps to complete and would likely increase the time that the machine vision system requires to analyze an image to in excess of one minute. Therefore, it was decided to avoid the use of this type of algorithm in the design of the present machine vision system.

In designing the edge point determination and linking algorithm, it was decided to modify the bug following technique utilized by Straumann (1987), such that the algorithm would not be susceptible to error caused by a gradient value of an edge point that was lower than expected or to error that was caused by a threshold value that was set to high. In order to do this a bridging algorithm would be incorporated into the bug following routine that would allow it to skip over an edge point whose gradient value was lower than expected and a threshold adjustment scheme would be incorporated that would allow the system to adjust the threshold value if it appeared to be set to high.

7.3 CORNER DETECTION

Based upon the discussion of Corner Detection Techniques in chapter 6, it would appear that either the corner detection method discussed by Rosenfeld and Johnston (1973)
or the corner detection method discussed by Ramer (1972) would produce outputs that might be very useful in differentiating corner points from non-corner points. Unfortunately, each of these methods require a substantial number of processing steps, actually requiring a number of computations to determine the best value at each point in the edge list.

Obvious reductions in the number of calculations required by the method described by Rosenfeld and Johnston (1973) do not appear to exist as performing this calculation for multiple values of \( k \) for each edge element is essential to the performance of the algorithm. However, the number of calculations required by the method discussed by Ramer (1972) might be reduced by calculating the distance between the point \( P_{i+h/2} \) and the imaginary line (shown in Figure 27) that was shown as extending between \( P_i \) and \( P_{i+h} \). This might be formulated in the following fashion.

Given the coordinates \((X_m,Y_m)\) of a midpoint \((P_{i+h/2})\) of a line between points \(P_i\) with coordinates \((X_i,Y_i)\) and point \(P_h\) with coordinates \((X_h,Y_h)\) are:

\[
X_m = \frac{(X_i + (X_h-X_i)/2)}{2} \\
Y_m = \frac{(Y_i + (Y_h-Y_i)/2)}{2}
\]

(This point is the midpoint of the line segment between points \(P_i\) and \(P_{i+h}\) in Figure 27.) The distance between \(P_m\) and \(P_j\) can be calculated from the equation:

\[
D = \sqrt{(X_m-(X_i+(X_h-X_i)/2))^2 + (Y_m-(Y_i+(Y_h-Y_i)/2))^2}
\]

After simplifying the resulting calculation that is to be
performed by the system is:
\[ D = \sqrt{((X_m + 0.5X_h - 0.5X_i)^2 + (Y_m + 0.5Y_h - 0.5Y_i)^2)} \]

This formulation reduces the complexity of the calculations that were shown for Ramer (1972) in the previous chapter. Further, calculating a distance for only the center point instead of all of the points between \( P_i \) and \( P_h \) reduces the number of calculations from \( h-2 \) to 1, for every \( j \).

However, this gross simplification of the algorithms discussed by Ramer (1972) has the potential of dramatically decreasing the effectiveness of the algorithm. Figure 30 shows the cornererity values that were obtained by calculating this distance for each of the objects shown in Figure 25. From this figure one can see that even after this simplification of the corner detection method discussed by Ramer (1972), the revised method continues to produce peak values that correspond to the observed corner points.

This simplification drastically reduces the number of processing steps that are required to complete this algorithm. However, as was the case with the other corner detection techniques, this corner detection technique cannot be easily implemented in the integer assembly language that is available on the Apple IIe due to the existence of the square root function and a multiplicative factor of less than one. However, it is not necessary to obtain the exact distance value since only magnitude differences in this distance value are of interest. Therefore, this corner detector might be simplified to the following equation:

\[ D_n = (2X_m - X_i + X_h)^2 + (2Y_m - Y_i + Y_h)^2 \]
Figure 30 Graphs depicting the cornererity of the points in the edge lists depicted in Figure 24, calculated using a simplified version of Ramer (1972).
Although this new function is not linearly related to the original function, the two functions are highly correlated and given equivalent input functions the resulting outputs of these equations will have corresponding maxima and minima.

Figure 31 shows the output that was generated using this modified algorithm to calculate the corner strength of the edge lists depicted in Figure 25. Looking at this figure, one can see that maxima are still produced that correspond to the observed corner points for the edge lists shown in Figure 25. Since it would appear that this modified version of the corner detection algorithm discussed by Ramer (1972) produces results that are comparable to the algorithms discussed by Rosenfeld and Johnston (1973) and Ramer (1972) with significantly fewer processing steps, it was decided that this modified algorithm would be applied in the machine vision system.
Figure 31 Graphs depicting the cornererity of the points in the edge lists depicted in Figure 24, calculated using a version of Ramer (1972) simplified to allow the algorithm to be programmed in integer assembly language.
Chapter 8
DESCRIPTION OF DEVELOPED SYSTEM

The purpose of this chapter is to discuss the machine vision software package that was created, based upon the general algorithms chosen in the previous chapter. The flow chart shown in Figure 32 represents an overview of the machine vision system. The actual implementation of the edge following and corner detection algorithms are represented in more detailed flow charts later in this chapter. Figure 32 shows the five processes that a user of the system is able to perform. These include: 1) execute the machine vision process, 2) store a new object in the machine vision system's memory for later reference, 3) set or view the parameters used by the machine vision system, 4) delete an object from the machine vision system's memory, or 5) exit the machine vision system and enter the disk operating system. The process of conducting the machine vision process is the primary function of this software package and improvement of this module was the primary focus of this thesis. The remaining processes provide support to the machine vision system. The primary purpose of these processes is to enable a user with few computer skills and very little knowledge of the machine vision system to maintain the system and to allow the system to recognize different objects. However, if this system is to be used successfully the usability and dependability of these modules is crucial. Figure 32 gives a brief overview of each of these pro-
Figure 32a An Overall Flow Chart of the Machine Vision Software Developed in this Thesis.
Corner points are determined
Sums for linear regression are calculated
Slope and intercept of each edge are calculated
Corner points are calculated for each pair of intersecting edges
Calculated corner points are compared to the corner points of the objects stored in memory
Are the calculated corners within the specified tolerance of the corner points stored in memory? [YES, GO TO C; NO, GO TO D]
D
Report "Object not recognized"

Edge points of the object are found by the edge linking process
Corner points are determined
Sums for linear regression are calculated
Slope and intercept of each edge are calculated
Corner points are calculated for each pair of intersecting edges
Has the algorithm identified the same number of corners as were entered by the user? [NO, GO TO B; YES, GO TO I]
I
Did the user indicate that the machine vision process is correct? [YES, GO TO J; NO, GO TO M]
M
Ask user to verify the number of corners that the object has.
Figure 32c Flow Chart Continued from Figure 31b

L ) (M

Notify the user that this object has already been stored in memory.

Have five sets of corner points been stored yet?

Notify the user that this object has already been stored in memory.

Calculate an average set of corner points based upon the five stored sets of corner points.

Are these corner points within a specified tolerance of any of the objects already stored in memory?

Compare the average corner point values to the corner points stored in the system's memory.

Run machine vision system.

Robot picks up object and moves it out of the viewing area.

Can the robot manipulate the object?

Would the user prefer to return to main menu, obtain hardcopies, or run machine vision system?

Would the user prefer to return to main menu, obtain another hardcopy, or run machine vision system again?

User is given the option of printing one of several system generated outputs, returning to main menu, or running vision process again.

Obtain Hardcopy

Obtain Hardecky

Return to main menu

Return to main menu

Another Hardecky

Main Menu
Determine if the object is retrievable with the robot

User inputs a title of the object

Store the corner coordinates of the object, object title and robot retrieval code to diskette

Does the user wish to train the machine vision system to learn a new shape?

Figure 32d Flow Chart Continued from Figure 31c
cesses, however, they will not be discussed in more detail until after the algorithms used by the machine vision system have been fully described.

8.1 MACHINE VISION PROCESS

The machine vision process presented here is similar to the machine vision process designed and implemented by Straumann (1987) in many ways. As was stated in the previous chapter, the method of determining a restricted viewing window for a higher resolution image of the viewing area was adopted from Straumann's programs. The only changes that were made to these modules were programming changes that reduced the completion time and memory requirements of these algorithms. Since these algorithms have already been discussed in Chapter 2, they will not be discussed again in this chapter. The primary focus of this thesis project was the development of the edge linking and corner detection algorithms. In addition, minor changes were made to the later processing steps to eliminate errors that were recorded during the reliability study of Straumann's system.

8.1.1 EDGE LINKING

The objective of the edge linking procedure is to begin with a higher resolution image (minimum 5 pixels per edge), determine if an object is present in this image, and follow the external edge of the object. This procedure outputs a list of ordered, connected coordi-
nates of edge points that encompass the edge of an object in the image from some arbitrary beginning point back to this same beginning point. The algorithm that was developed to complete this process is shown in the flow chart in Figure 33.

The first step of the edge linking algorithm is to determine whether an object is actually present in the viewing area. This is done by starting in one of the corners of the matrix that digitally represents the luminance radiated from discrete points in the image. The Roberts Gradient is then calculated for successive elements of this matrix. As gradient values are calculated, they are compared to a user definable threshold. If the calculated Roberts Gradient value for a matrix element is larger than this threshold then the Roberts gradient value is calculated for this element's neighbors. If a neighboring element is found whose Roberts gradient value is also greater than the threshold then it is assumed that these two points lie on the edge of an object in the viewing area. If two adjacent points with Roberts Gradient Values larger than the threshold are not found in the image then it is assumed that no object is present in the viewing area. This result is reported to the user by the machine vision system and the process is terminated. Two adjacent matrix elements are required before it is assumed that these points lie on the edge of an object since the gradient value of a single matrix element could be above a threshold due to random variation. It is
Figure 33a Flow Chart of the Edge Linking Algorithm.
Move to a neighboring pixel stored, according to the search algorithm.

The diagram shows a flow chart of the Edge Linking Algorithm. The chart begins with a decision point: Is this calculated value greater than the threshold? If not, the process continues with storing the two points as the first two points in the edge list. It is assumed that the two pixels found lie along the edge of the object. Advance in the direction indicated by these two pixels. If the pixel position actually exists (or is it outside of the image?), move to the next step. If not, continue with the search. If yes, calculate the Roberts Gradient for this pixel. If the Roberts Gradient for this pixel is greater than the threshold, proceed. If not, return to the previous step. This process is continued until all relevant pixels are processed.

Figure 33b Flow Chart of the Edge Linking Algorithm (Continued from Figure 32a).
Move to the next neighboring pixel that qualifies as an edge point according to the right search algorithm.

Calculate the Roberts Gradient for this pixel.

Is the Gradient for this pixel greater than the threshold?

HAVE all of the neighbors been tested yet?

YES

Move to the next neighboring pixel that qualifies as an edge point according to the left search algorithm.

Calculate the Roberts Gradient for this pixel.

Is the Gradient for this pixel greater than the threshold?

HAVE all of the neighbors been tested yet?

YES

Figure 33c Flow Chart of the Edge Linking Algorithm (Continued from Figure 32b).
Calculate the Roberts Gradient for this Pixel

Is the Gradient Value for this pixel greater than the threshold

Have all directions been tried?

Move to the next position according to the search algorithm

Decrease the current threshold value by 5

Subtract 5/16 of the average of the last five stored pixel's Roberts Gradient value from the threshold.

RETURN TO THE ASSEMBLY DRIVER

Figure 33d Flow Chart of the Edge Linking Algorithm (Continued from Figure 32c).
unlikely that two adjacent points with gradient values above an appropriate threshold will occur only because of random variation.

Once two adjacent points with Roberts Gradient Values greater than the threshold are found, these two points are used as starting points for the edge linking procedure. From these two points, the edge linking procedure attempts to move to a third adjacent point whose Roberts Gradient value is above the threshold. It is assumed that the first two points lie on the edge of an object in the image and a line passing through these two points is parallel to the actual edge. Therefore, the Roberts Gradient value is calculated for the matrix point that lies directly ahead of, and in the direction indicated by these two points. If this point's Roberts Gradient value is not above the threshold either one, or a combination of the three following scenarios have occurred:

Scenario 1 - the first two points do not lie parallel to the edge of the object and therefore the third point along this line does not fall near enough to the edge to elicit a large enough response in the gradient space for the gradient value to be larger than the threshold.

Scenario 2 - these two points are parallel to the edge and the threshold is higher than the Roberts Gradient of an edge point due to ran-
dom variation in the gray scale values. This variation may be caused by an external source such as uneven illumination, variability in the response to an illumination level between the pixel elements, etc.

Scenario 3 - these two points are parallel to and lie on the edge and the threshold has been set to high to allow the algorithm to properly detect the edge.

The first of these three scenarios is the simplest to test. This is done by calculating the Roberts Gradient for each of the neighbors of the last point that was added to the linked edge list and comparing it to the threshold. If the Roberts Gradient for one of these points is greater than the threshold then it is assumed to be on the edge. Figure 34a shows the order that the neighbors of this point are investigated. Looking at Figure 33a the last point that was found along the edge (at this point the second point found) is assumed to be in the center of the 3x3 matrix (position p) and the next to the last point that was found (in this case the first point that was found) is assumed to be in the blackened position. As was stated earlier, the algorithm first checks the position that falls on a straight line with the two previously found points (this is shown as position 1 in Figure 34a). Assuming that the Roberts Gradient of the matrix element in position 1 is less than the
Figure 34a The priority used by the edge linking procedure to link a point to its neighbor on a right rotation.

Figure 34b The priority used by the edge linking procedure to link a point to its neighbor on a left rotation.

Figure 34c The priority used by the edge linking procedure to link a point to a point when no neighboring point is found with an acceptable gradient value.
threshold; the algorithm calculates the Roberts Gradient Value for position 2 and compares it against the threshold. If the Roberts Gradient Value is less than the threshold then the algorithm conducts this same process for each of the neighboring pixels in the order shown in Figure 34a. The algorithm calculates the Roberts Gradient value for each of the remaining neighbors until a point is found whose Roberts Gradient value is greater than the threshold.

If a Roberts Gradient value for one of the neighboring matrix elements is higher than the threshold then this new point is stored in the edge list. The algorithm advances the search matrix to this new point, the new point assumes the center position (position p) in Figure 34b and the pixel whose coordinates were previously represented by position p becomes the blackened position in Figure 34b. The linking algorithm continues following the edge by calculating the Roberts Gradient for the point directly ahead of the last two points that were stored and if this value is higher than the threshold moving forward to this position or if the value is not higher than the threshold, looking for another point that is adjacent to the last point stored whose Roberts Gradient value is greater than the threshold and then moving to the first adjacent point that fulfills this condition. However, each time that the algorithm has to advance to a matrix element that is not in position 1 in the three by three matrix then the algorithm rotates back and forth
between the search positions shown in Figure 34a and the search positions shown in Figure 34b. This is designed to reduce the likelihood that the algorithm will inadvertently revisit a point along the object's edge when it enters a region of the matrix where there is a large number of matrix positions whose Roberts Gradient values are above the threshold. As an additional deterrent to circling, the coordinates of a matrix element are compared to the coordinates of the last five matrix elements that were stored in the edge list before it is stored as a new point in the linked edge list. If the coordinates of this point have already been stored in the edge list then the matrix element is not stored and the algorithm begins looking for another pixel position that is a viable edge point.

If at any point in this process the algorithm is unable to find a consecutive point whose gradient is larger than the threshold then it is assumed that the Roberts Gradient was incapable of enhancing an edge point sufficiently due to a random fluctuation in the gray scale values (Scenario 2). To overcome this deficiency the range of search is expanded. The search positions for this condition are shown in Figure 34c. Again the matrix is oriented such that the last point that was stored in the edge list is represented by the center of the matrix (position P) and the second to the last point stored in the edge list is represented by the blackened position directly above and to the left of position p. The edge
following algorithm then calculates the Roberts Gradient for the matrix position that is represented by position 1 in the 5x5 matrix shown in Figure 34c. This value is compared to the threshold. If this value is larger than the threshold value then the point that lies between the center of the matrix and position 1 is stored in the edge list and the coordinates of the point in position 1 are stored in the edge list. Storing the point between the center of the 5x5 matrix and the point whose Roberts Gradient is larger than the threshold insures that the edge list generated by this procedure is connected.

If the Roberts Gradient value for the point in the gray scale matrix represented by position 1 is not larger than the threshold, the algorithm calculates the Roberts Gradient values for the gray scale matrix positions represented by remaining numbered matrix positions in Figure 34c. When doing this the algorithm begins with position 2 and advances toward position 15 until the Roberts Gradient value for one of these points is higher than the threshold. When a point is found whose Roberts Gradient value is larger than the threshold, the coordinates of the point that connects it to the center point of the 5x5 matrix as indicated by the arrows in Figure 34c are stored in the linked edge list. However, before they are stored their coordinates are compared to the coordinates of the last five points that have been stored in the edge list. If either of these points have already been stored in the linked edge list then they are both
considered to be invalid points and a new point whose Roberts Gradient value is higher than the threshold is sought.

In the case that the algorithm is unable to link the edge by investigating this extended number of neighboring points then it is assumed that the threshold has been set to high (Scenario 3). The algorithm then reduces the threshold by a proportion of the Roberts Gradient value of the last few points. More explicitly the Roberts Gradient values for the last five points in the edge list are summed together and the resulting binary representation of this number is shifted to the right three times, effectively dividing the resulting value by 16. The resulting value is then subtracted from the threshold. If five consecutive points have not yet been stored in the edge list then the threshold is simply reduced by a constant of 5. The entire edge linking procedure is then began again from scratch. Another option would have been to reduce the threshold and then try to begin from where the algorithm had left off during edge following, however, it is possible that the edge following routine may have actually deviated from the actual edge and entered a subtour caused by random variations in the gray scale matrix. If this did occur then the edge following algorithm would likely find the end of the subtour and this branch would likely not rejoin the edge regardless of the threshold value. Therefore the edge linking process is began again with the updated threshold value acting
against the same matrix of gray scale values from the same "high" resolution image that the prior edge linking process was conducted on. Depending upon the threshold value that was set and the particular matrix of gray scale values (which varies due to a number of factors including: illumination conditions, random variations in the pixels' response, etc.) that are being analyzed all or parts of this process may be conducted a number of times for each image.

The edge linking process is terminated only if two adjacent pixels whose Roberts Gradient values greater than the threshold are not found, the threshold is reduced to a minimum threshold value (this value is a constant 15 in the current algorithm) or if the algorithm finds a viable edge point that is adjacent to the first point that was stored in the edge linking process. When the algorithm is able to link a point directly to the first point that was stored then the machine vision process is considered successful. When this occurs it is expected that the algorithm will have generated a list of coordinates that can be analyzed by later machine vision processes. This list of coordinate points for adjacent matrix elements represent pixel elements that viewed the edges of the object. The positions of these pixels can be used to approximate the positions of the edges of the object.
8.1.2 CORNER DETECTION

The corner detector that was chosen to be used as part of this system was discussed in each of the two previous chapters. The basic premise of this algorithm is based upon the equation that the corner strength (CS) for a point in the edge list may be calculated from:

$$CS(X_m, Y_m) = (2X_m - X_1 - X_2)^2 + (2Y_m - Y_1 - Y_2)^2$$

where $X_m$ and $Y_m$ represent the coordinates of the point for which the corner value is being calculated, $X_1$ and $Y_1$ are the coordinates for a point that is some number of points before the point whose corner strength is being calculated, and $X_2$ and $Y_2$ are the coordinates for a point that is an equal number of points in advance of the point whose corner strength is being calculated. As was shown earlier, when the values generated by this equation are plotted against the number of the points in the edge list, the corners roughly correspond to the peaks in the graph. What has not been discussed is how does one decide whether a peak in this graph is attributable to an actual corner or whether it is simply caused by random variations in the edge list that have been caused by some prior process. Therefore, it was necessary to devise an algorithm capable of making this decision.

The algorithm that was devised to decide whether a point should be considered a corner point or not is shown in the flow chart in Figure 35. Basically, the corner detection algorithm begins at the first point stored in the edge list and calculates the corner strength of each
Begin at the first point in the linked edge list for which a value can be calculated.

Calculate the corner strength.

Is this value less than the corner threshold?

NO

Advance to the next point in the linked edge list.

YES

Store 2 as the corner code to serve as a parity check for the sums of linear regression program.

Is this the last point in the linked edge list to be assigned a code?

NO

Has the end of the edge list been reached yet?

NO

Calculate the corner strength for this point.

Is this value greater than the corner threshold?

NO

Terminate the program. (no corner found)

Reset the counter to begin at the beginning of the linked edge list.
Figure 35b Flow Chart of the Corner Detection Algorithm (Continued from Figure 34a).
Set corner flag to indicate that the algorithm is approaching a corner.

Is the corner strength less than the corner strength of the last point?

YES

Is the corner strength less than the corner threshold?

YES

Store the point as a corner point

Reset the corner flag to indicate that the algorithm is not nearing a corner

NO

NO

NO

Does the corner flag indicate that the algorithm is approaching a corner?

NO

Is the corner strength greater than the corner threshold?

YES

Set corner flag to indicate that the algorithm is approaching a corner

Set the corner flag to indicate that algorithm may be approaching a dual peaked corner

Has the algorithm found the beginning of a second peak?

NO

YES

Figure 35c Flow Chart of the Corner Detection Algorithm (Continued from Figure 34b).
Figure 35d Flow Chart of the Corner Detection Algorithm (Continued from Figure 34c).
pixel until the corner strength value climbs from a low value to a value that is higher than a threshold. When this occurs, it is assumed that the corner detector is approaching a corner. From this point on, each time the corner strength of a point is calculated, the coordinates of the point are stored in a final edge list with a flag that indicates whether this point is to be considered a corner point or not. Once the algorithm finds a point whose corner strength is greater than the threshold then the corner strength of the following point is compared to this point. When the corner strength begins to decline then the peak is said to be a corner. However, if the corner strength does not continue to decline to a value that is lower than the threshold value before another peak is found then the first designated corner point is restored as a noncorner point and the point that lies directly between the two peaks is stored as a corner. The algorithm continues in this manner until it has designated each edge point as a corner point or non-corner point.

2.1.3 LINEAR REGRESSION

Once each edge point has been designated as either a corner point or non-corner point, a linear regression is performed for each edge of the object and an average corner value is obtained from calculating the intersections of these linear equations. The basic idea of modeling each edge (which are assumed to be straight lines)
through the use of linear regression is equivalent to the method used by Straumann. However, an error often occurred in Straumann's system due to the form of linear regression that was used. This set of linear regression equations allowed the denominator of the x coefficient to be equal to zero for a perfectly vertical line. This problem has been solved in this system.

Assuming that the basic equation of a line is:

\[ y = ax + b \]

the slope of the line (a) is:

\[ a = \frac{(n\Sigma xy) - (\Sigma x \Sigma y)}{(n\Sigma x^2) - (\Sigma x)^2} \]

and the intercept of the line (b) is calculated as:

\[ b = \frac{\Sigma y}{n} - \frac{a \Sigma x}{n} \]

where \( n \) is the number of points between and including each of the pairs of successive corner points, \( \Sigma x \) is the sum of the x coordinates of the points between and including each of the corner points and \( \Sigma y \) is the sum of the y coordinates of the points between and including each of the corner points.

The x coordinates of the smoothed corner points are then calculated from:

\[ X = \frac{(b_f - b_c)}{(a_c - a_f)} \]

and the y coordinate is calculated from:

\[ Y = b_f + a_f X \]

Where \( b_c \) is the y-intercept of the current point, \( b_f \) is the y-intercept of the following point, \( a_c \) is the slope of the current point and \( a_f \) is the slope of the following point.
This method of smoothed corner calculation fails when the denominator of the slope equation for one of the lines is equal to zero, indicating that the line has an infinite slope. In the revised system the point where the line intersects the x axis is calculated from:

\[ X = \frac{\Sigma x}{n} \]

and only the y coordinate of the points of intersection of this line with its neighboring lines needs to be calculated. This coordinate is calculated from:

\[ Y = b_e + a_eX \]

These two special equations are invoked each time that the denominator of the slope equation is zero. Therefore, avoiding the errors in linear regression that occurred in Straumann's machine vision system.

The remainder of the machine vision process appeared to perform the task that it was required to do quite admirably and therefore it was decided to maintain and refine the code that was written by Straumann (1987). Therefore, the remainder of the machine vision process has not been altered. After the corner points are calculated they are transformed from matrix coordinates to real coordinates of the viewing area through a pair of equations that have been documented by Straumann (1987), the objects corners are then reoriented based upon the rules given in Chapter 2 and the corner coordinates of the object are compared to corner coordinates of the objects that the computer has been trained to recognize. If the object is recognized then the data base is checked
to see if the object is retrievable by the robot. If it is then the robot is ordered to retrieve it. If the object is not retrievable then the user is informed that the object can not be retrieved. The user is then given the option of printing out data that was obtained or calculated during the machine vision process, running the machine vision process again or returning to the main menu.

If the user wishes to print out data that was obtained or calculated during the machine vision process then he or she is given the option of printing out the gray scale matrix for the viewing area, the Roberts Gradient matrix for the viewing area, the gray scale matrix for the window, the Roberts Gradient matrix for the window, a diagram of all of the points in the image above a specified threshold (defaults to the threshold used by the system), the edge list, and/or the coordinates of the objects corner points returned by the corner detection algorithm. (Note that neither the Roberts Gradient matrix for the entire viewing area or the Roberts Gradient matrix for the "high" resolution window are actually calculated during the machine vision process but are generated upon command through special assembly programs to be printed.) These printouts are designed to give the user most of the information that is used at different stages of the machine vision process and to aid a novice user's understanding of the processes that are conducted by the machine vision system. These outputs also serve as a
helpful aid to an experienced user when he or she is attempting to understand errors that might occur during the machine vision process. In addition to these options, the user is given the option of returning to the main menu or running the machine vision process again.

8.2 ADDING A NEW OBJECT TO MEMORY

This module is designed to allow a novice user to add a new object to the computer's memory. The approach that was taken to this process by Straumann (1987) was to run the machine vision process and store the object's corner coordinates, an identifier that was input by the user, and a code that indicated whether the object could be retrieved by the robot into a data file after the data had been reviewed by the user. This method works as long as the system obtained a true picture of the object on this one trial or the user is knowledgeable enough to recognize when the system has correctly analyzed the image. However, it is possible that the machine vision system might not correctly analyze the image, giving the object an extreme set of corner points, an incorrect number of corner points, or a misrepresentation of the slope of the edges and a novice user may not recognize these inconsistencies in the data before storing it. Therefore, this module was redesigned to attempt to control these possible errors.

During the system training process the user is asked to place an object in the viewing area and to indicate to
the system when this task has been completed. The user is also asked to indicate the number of corners that this object has. The system then performs much of the machine vision process as it was discussed in the previous section of this chapter, including finding a window, following the edges, finding corner points, performing the linear regression, calculating smoothed corner points, transforming these corner points to real corner points in the viewing area and reorienting the corner points. Upon completing this process the algorithm compares the number of corner points that were indicated by the user to the number of corner points found by the machine vision system. If these are not equal then the system prompts the user to verify that the object in fact has the number of corners that the user indicated previously. If the user indicates that the object does have this number of corners then the system conducts the machine vision process again. If the number of corners indicated by the user and the number of corners found by the machine vision system are equal then the corner values are stored, the user is asked to give the object another orientation and the machine vision process is conducted again. The process is conducted as many times as necessary to obtain 5 sets of corner coordinates that have the correct number of corner points. This operation was performed to obtain five sets of corner coordinates since according to the Central Limit Theorem (Hines and Montgomery 1980, p. 183) when collecting well behaved data from a normal distrib-
uted population, the average of four or more samples can be expected to approach the population mean. Averaging over five sets of corner coordinates reduces the likelihood that an "extreme" set of corner coordinates will be stored for any given object. Once these five sets of corner coordinates have been found, an average set of corner coordinates are calculated, the user is asked to verify that these corner coordinates are feasible, and these corner coordinates are stored. The user is also asked to enter an object identifier. The system attempts to find a pair of edges whose slopes differ by less than .05 (indicating edges that are approximately parallel) and are close enough to allow robot retrieval. If an appropriate pair of edges are found, the user is asked to indicate whether the object should be retrieved by the robot. If the user indicates that the robot should retrieve the object then this information is also stored on the machine vision system diskettes for later reference by the machine vision system. As many as 20 objects may be stored in the machine vision system's memory at any one time.

8.3 DELETE AN OBJECT

This module is designed to allow an operator to interactively remove the data that was stored for an object from the data base of objects that the machine vision system is able to recognize. When the user selects this module, he or she is given the option of viewing the
data that is stored for the different objects or deleting
an object from the machine vision system's memory. Regardless
of which of these modules are chosen the user is shown the
title, corner coordinates and robot retrieval information that is
stored for each of the objects, one at a time. If the user has
chosen only to view the objects then the information for one object
is displayed and the user is given control of advancing to
successive objects at his or her own pace. When the user
is in this mode, data can not be erroneously removed from
the data base while it is being viewed. When in delete
mode, the identical information is shown on the display
for each object in the data base. When in this mode the
user advances to the next screen by responding to the
question "Do you wish to delete this object?". If the
user gives a positive indication to this question then he
or she is asked to confirm his or her intention, after
which the object may be removed from the data base. If
the response is negative, the system simply displays the
information for the next object in the database.

8.4 SET PARAMETERS

It was expected that the first three modules may be
executed easily by a user who has only a very basic
knowledge of machine vision and practically no computer
experience. It is expected that the user of this fourth
module will have at least some knowledge about the
machine vision system's operation since altering the sys-
tem parameters will have a profound effect upon the machine vision system's ability to operate. However, proven default values are recommended by the system to give the user a recovery path in the event that a disastrous parameter value is selected. In addition, if the user attempts to input a value that is not feasible, the system will inform the user that the value they have just input is not reasonable and ask the user to input another value.
CHAPTER 9

RESULTS AND CONCLUSIONS

A reliability study identical to the one discussed in Chapter 2 and graphically represented in Figure 8 was conducted for the improved machine vision system. The results that were obtained from this reliability study are shown in Table 2. These results indicate that the system was able to successfully recognize simple rectangular objects with different orientations and positions within the viewing area in 580 out of the 720 trials that were conducted. This translates to an overall system reliability of about 81%.

The most prevalent errors include the system's inability to match the corner coordinates of the object to within .1 inch of the corner coordinates of the object that are stored in memory. This error occurs about 7.5% of the time. Corner detection continues to produce errors. This error occurs about 6% of the time and is most prevalent when the object is oriented with a 32 degree angle of rotation with respect to the viewing window (21 of the 43 corner detection errors occurred when the object was placed at this orientation while the object was only given this orientation in 120 out of the 720 trials). Errors also occur in the edge linking procedure. These errors account for about 4 percent of the total errors.

Reductions and modifications in the software that has been used from the previous system, as well as, elimination of the need to compute gradient values for entire matrices of gray scale values have reduced the time required by the
Table 2 The number of errors and successes that were recorded during the reliability study for the improved machine vision software system for two objects placed in one of three positions, with one of six angles.

<table>
<thead>
<tr>
<th>Location 1</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>0 1 4 1 1 0</td>
<td>1 2 4 3 0 0</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>1 1 1 1 3 1</td>
<td>1 0 1 1 1 0</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>2 4 5 0 0 0</td>
<td>0 2 0 0 0 0</td>
</tr>
<tr>
<td>Misc.</td>
<td>1 1 1 0 0 1</td>
<td>1 1 1 0 1 1</td>
</tr>
<tr>
<td>Success</td>
<td>16 13 9 18 16 18</td>
<td>17 15 14 16 18 19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location 2</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>0 1 1 0 0 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>0 0 0 1 6 2</td>
<td>1 0 0 0 4 0</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>3 1 3 3 0 0</td>
<td>2 5 6 1 1 0</td>
</tr>
<tr>
<td>Misc.</td>
<td>1 1 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Success</td>
<td>16 17 16 16 14 17</td>
<td>17 15 14 19 15 20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location 3</th>
<th>Square</th>
<th>Rectangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle (DEG.)</td>
<td>0 2 4 8 32 45</td>
<td>0 2 4 8 32 45</td>
</tr>
<tr>
<td>Linking</td>
<td>1 1 0 2 1 0</td>
<td>0 0 3 0 1 0</td>
</tr>
<tr>
<td>Corner Detection</td>
<td>1 0 2 1 3 1</td>
<td>0 2 1 0 4 2</td>
</tr>
<tr>
<td>Not Recognized</td>
<td>2 5 3 1 0 1</td>
<td>2 0 1 0 1 0</td>
</tr>
<tr>
<td>Misc.</td>
<td>1 1 0 0 0 0</td>
<td>0 0 0 0 0 1</td>
</tr>
<tr>
<td>Success</td>
<td>15 13 15 16 16 18</td>
<td>18 18 15 20 14 17</td>
</tr>
</tbody>
</table>
system to process an image of a 3/4 inch square from about 15 seconds to about 11 seconds. However, due to the iterative threshold adjusting scheme that is used in the improved system the time it requires to process an image varies from about 9 to about 13 seconds.

Although significant improvements in the system's reliability and speed have been made further modifications could be made. Errors that occur due to the system's inability to recognize an object or the system's inability to correctly identify the corners of the object might be reduced by modifying earlier processes such that the variability in the position of points along the edge list would be reduced. This might be accomplished through the use of an algorithm, such as the edge thinning algorithms, that insure that the edge points in the linked edge list are as near the center of the actual edge of the object as is possible. Improvements of this type might also reduce errors due to the system's inability to match the corners of the object being observed to within .1 inches of the object stored in memory since variability of the corner positions should be reduced.

In conclusion, a statistical evaluation of the machine vision software that was developed by Straumann (1987) and comprehensive review of existing feature geometry machine vision algorithms were conducted and revised edge linking and corner detection algorithms were developed. As a result of implementing these revised edge linking and corner detection algorithms into the existing machine vision
software and providing additional smaller improvements to the existing machine vision software, an improved machine vision system has been developed. This improved machine vision software system is capable of recognizing rectangular, flat surfaced, objects that contrast well with their background an average of 80.6% of the time. This system reliability has been greatly improved over the 5% recognition reliability of the prior system. This improved system is capable of completing this process in about 11 seconds on an Apple IIe microcomputer. In addition, an improved user interface has been developed that allows the user to conduct and support the machine vision system, as well as, obtain outputs of data that is generated during the machine vision process. This expanded capability improves the usefulness of the system to demonstrate to students how a machine vision system operates and how it might be applied in an industrial setting.
REFERENCES


Freeman, H. and L.S. Davis, "A Corner Finding Algorithm for Chain Coded Curves", IEEE Transactions on Comput-


APPENDIX A

LINE THINNING PROGRAM LISTINGS
A2

10 REM A TEST PROGRAM TO DETERMINE THE SPEED AND USEFULNESS
20 REM OF A THINNING ALGORITHM PROPOSED BY E.S. DEUTSCH IN
30 REM A PAPER ENTITLED "THINNING ALGORITHMS ON
40 REM RECTANGULAR, HEXAGONAL AND TRIANGULAR ARRAYS",
50 REM PUBLISHED IN THE COMMUNICATIONS OF THE ACM, SEPT.
60 REM 1972
70 REM
80 REM CREATE ARRAYS
90 DIM X(100),Y(100),SQUARE(9),MAT(100,100),CODE(100)
100 REM INITIALIZE ARRAYS
110 FOR I=1 TO 100
120 FOR J=1 TO 100
130 MAT(I,J)=0
140 NEXT J
150 NEXT I
160 FOR I=1 TO 100
170 CODE(I)=0
180 NEXT I
190 BEGTS$=TIME$
200 REM
210 REM INPUT THE COORDINATES OF THE POINT THAT LIE ON THE
215 REM FIGURE
220 OPEN "C:\VW3\NEW\DATA" FOR INPUT AS #1
230 OPEN "C:\VW3\NEW\DEUTSCH4.DAT" FOR OUTPUT AS #2
240 I=1
250 INPUT #1,X(I)
260 IF X(I)=999 THEN GOTO 330
270 INPUT #1, Y(I)
280 CX=X(I)
290 CY=Y(I)
300 MAT(CX,CY)=1
310 I=I+1
320 GOTO 250
330 IFINAL=I-1
340 REM
350 REM
360 REM ********** BEGIN ALGORITHM **********
370 CYCLE=0
380 REM BEGIN CYCLE
390 CHANGE=0
400 FOR I=1 TO IFINAL
410 IF CODE(I)=1 THEN GOTO 690
420 REM CALL SUBROUTINE TO DETERMINE THE STATUS OF
430 REM SURROUNDS
440 GOSUB 1200
450 CX=X(I)
460 CY=Y(I)
470 REM CALCULATE X AND THE NO. OF SURROUNDS
480 X=0
490 NOPT=0
500 FOR K=1 TO 8
510 X=X+ABS(SQUARE(K+1)-SQUARE(K))
520 NEXT K
530 REM BEGIN CHECKING THE CONDITIONS
540 IF X<>0 AND X<>2 AND X<>4 THEN GOTO 690
550 IF NOPT<2 THEN GOTO 690
560 IF SQUARE(1)=1 AND SQUARE(3)=1 AND SQUARE(5)=1 THEN GOTO 690
570 IF SQUARE(1)=1 AND SQUARE(3)=1 AND SQUARE(7)=1 THEN GOTO 690
580 IF X=0 OR X=2 THEN
   CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 690
590 IF SQUARE(1)=0 OR SQUARE(7) = 0 THEN GOTO 640
600 IF SQUARE(2)=0 AND SQUARE(6)=0 THEN GOTO 640
610 IF SQUARE(3)=1 OR SQUARE(4)=1 THEN GOTO 640
620 IF SQUARE(5)=1 OR SQUARE(8)=1 THEN GOTO 640
630 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 690
640 IF SQUARE(1)=0 OR SQUARE(3)=0 THEN GOTO 690
650 IF SQUARE(4)=0 AND SQUARE(8)=0 THEN GOTO 690
660 IF SQUARE(2)=1 OR SQUARE(5)=1 THEN GOTO 690
670 IF SQUARE(6)=1 OR SQUARE(7)=1 THEN GOTO 690
680 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 690
690 NEXT I
700 FOR I=1 TO IFINAL
710 IF CODE(I)=1 THEN GOTO 970
720 REM CALL SUBROUTINE TO DETERMINE STATUS OF SURROUNDS
730 GOSUB 1200
740 CX=X(I)
750 CY=Y(I)
760 X=0
770 NOPT=0
780 FOR K=1 TO 8
790 X=X+ABS(SQUARE(K+1)-SQUARE(K))
800 NOPT=NOPT+SQUARE(K)
810 NEXT K
820 IF X<>0 AND X<>4 AND X<>2 THEN GOTO 970
830 IF NOPT<2 THEN GOTO 970
840 IF SQUARE(3)=1 AND SQUARE(5)=1 AND SQUARE(7)=1 THEN GOTO 970
850 IF SQUARE(5)=1 AND SQUARE(7)=1 AND SQUARE(1)=1 THEN GOTO 970
860 IF X=0 OR X=2 THEN
   CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 970
870 IF SQUARE(5)=0 OR SQUARE(3)=0 THEN GOTO 920
880 IF SQUARE(6)=0 AND SQUARE(2)=0 THEN GOTO 920
890 IF SQUARE(1)=1 OR SQUARE(4)=1 THEN GOTO 920
900 IF SQUARE(7)=1 OR SQUARE(8)=1 THEN GOTO 920
910 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 970
920 IF SQUARE(7)=0 OR SQUARE(5)=0 THEN GOTO 970
930 IF SQUARE(8)=0 AND SQUARE(4)=0 THEN GOTO 970
940 IF SQUARE(1)=1 OR SQUARE(6)=1 THEN GOTO 970
950 IF SQUARE(3)=1 OR SQUARE(2)=1 THEN GOTO 970
960 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1
970 REM CHECK NEXT POINT
980 NEXT I
990 REM COUNT THE NUMBER OF CYCLES THE ALGORITHM GOES THROUGH
1000 CYCLE=CYCLE+1
1010 IF CHANGE>0 THEN GOTO 380
1020 FINT$=TIME$
1030 REM PRINT RESULTS
1040 PRINT #2,"," PROCEDURE BEGAN AT ";BEGT$;" AND
1050 PRINT #2,"," PROCEDURE ENDED AT ";FINT$
1060 PRINT #2,"," NUMBER OF CYCLES = ";CYCLE
1070 PRINT #2,"," X ","Y","CODE"
1080 FLIN=48
1090 START=1
1100 IF IFINAL<=FLIN THEN FLIN=IFINAL
1110 FOR I=START TO FLIN
1120 PRINT #2,"," ;X(I),Y(I),CODE(I)
1130 NEXT I
1140 REMAIN=IFINAL-FLIN
1150 IF REMAIN<=0 THEN END
1160 START=START+FLIN
1170 IF REMAIN>54 THEN FLIN=FLIN+54
1180 IF REMAIN<=54 THEN FLIN=FLIN+REMAIN
1190 GOTO 1110
1200 REM ********** SET SQUARE POSITIONS **********
1210 CX=X(I)+1
1220 CY=Y(I)
1230 SQUARE(1)=MAT(CX,CY)
1240 CY=Y(I)+1
1250 SQUARE(2)=MAT(CX,CY)
1260 CX=X(I)
1270 SQUARE(3)=MAT(CX,CY)
1280 CX=X(I)-1
1290 SQUARE(4)=MAT(CX,CY)
1300 CY=Y(I)
1310 SQUARE(5)=MAT(CX,CY)
1320 CY=Y(I)-1
1330 SQUARE(6)=MAT(CX,CY)
1340 CX=X(I)
1350 SQUARE(7)=MAT(CX,CY)
1360 CX=X(I)+1
1370 SQUARE(8)=MAT(CX,CY)
1380 CY=Y(I)
1390 SQUARE(9)=MAT(CX,CY)
1400 RETURN
10 REM A TEST PROGRAM TO DETERMINE THE SPEED AND USEFULNESS
20 REM OF A THINNING ALGORITHM PROPOSED BY E.S. DEUTSCH IN
30 REM A PAPER ENTITLED "THINNING ALGORITHMS ON
40 REM RECTANGULAR HEXAGONAL AND TRIANGULAR ARRAYS",
50 REM PUBLISHED IN THE COMMUNICATIONS OF THE ACM, SEPT.
60 REM 1972
70 REM REVISED VERSION FOR 8-CONNECTEDNESS
80 REM
90 REM CREATE ARRAYS
100 DIM X(100),Y(100),SQUARE(9),MAT(100,100),CODE(100)
110 REM INITIALIZE ARRAYS
120 FOR I=1 TO 100
130 FOR J=1 TO 100
140 MAT(I,J)=0
150 NEXT J 160 NEXT I
170 FOR I=1 TO 100
180 CODE(I)=0 190 NEXT I
200 
210 REM 220 REM INPUT THE COORDINATES OF THE POINT THAT LIE
ON THE FIGURE
230 OPEN "C:\VW3\NEW\DATA" FOR INPUT AS #1
240 OPEN "C:\VW3\NEW\DEUTSCH8.DAT" FOR OUTPUT AS #2
250 I=1 260 INPUT #l,X(I)
270 IF X(I)=999 THEN GOTO 340
280 INPUT #1, Y(I)
290 CX=X(I) 300 CY=Y(I)
310 MAT(CX,CY)=1
320 I=I+1 330 GOTO 260
340 IFINAL=I-1
350 REM 360 REM
370 REM ********** BEGIN ALGORITHM **********
380 CYCLE=O 390 REM BEGIN CYCLE
400 CHANGE=O 410 FOR I=1 TO IFINAL
420 IF CODE(I)=1 THEN GOTO 680
430 REM CALL SUBROUTINE TO DETERMINE THE STATUS OF
SURROUNDS
440 GOSUB 1230
450 CX=X(I) 460 CY=Y(I)
470 REM CALCULATE X AND THE NO. OF SURROUNDS
480 X=O 490 NOPT=O
500 FOR K=l TO 8
510 X=X+ABS(SQUARE(K+1)-SQUARE(K))
520 NOPT=NOPT+SQUARE(K)
530 NEXT K 540 REM BEGIN CHECKING THE CONDITIONS
550 IF X<>O AND X<>2 AND X<>4 THEN GOTO 680
560 IF NOPT<2 THEN GOTO 680
570 IF SQUARE(l)=1 AND SQUARE(3)=1 AND SQUARE(5)=1 THEN
GOTO 680
580 IF SQUARE(l)=l AND SQUARE(3)=1 AND SQUARE(7)=1 THEN
GOTO 680
590 IF X=0 OR X=2 THEN
CODE(I)=l:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 680
600 IF SQUARE(l)=0 OR SQUARE(7) = 0 THEN GOTO 640
610 IF SQUARE(3)=1 OR SQUARE(4)=1 THEN GOTO 640
620 IF SQUARE(5)=1 OR SQUARE(8)=1 THEN GOTO 640
630 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 680
640 IF SQUARE(1)=0 OR SQUARE(3)=0 THEN GOTO 680
650 IF SQUARE(2)=1 OR SQUARE(5)=1 THEN GOTO 680
660 IF SQUARE(6)=1 OR SQUARE(7)=1 THEN GOTO 680
670 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 680
680 NEXT I 690 FOR I=1 TO IFINAL
700 IF CODE(I)=1 THEN GOTO 940
710 REM CALL SUBROUTINE TO DETERMINE STATUS OF SURROUNDS
720 GOSUB 1230
730 CX=X(I) 740 CY=Y(I)
750 X=0 760 NOPT=0
770 FOR K=1 TO 8
780 X=X+ABS(SQUARE(K+1)-SQUARE(K))
790 NOPT=NOPT+SQUARE(K)
800 NEXT K 810 IF X<>0 AND X<>4 AND X<>2 THEN GOTO 940
820 IF NOPT<2 THEN GOTO 940
830 IF SQUARE(3)=1 AND SQUARE(5)=1 AND SQUARE(7)=1 THEN GOTO 940
840 IF SQUARE(5)=1 AND SQUARE(7)=1 AND SQUARE(1)=1 THEN GOTO 940
850 IF X=0 OR X=2 THEN
860 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 940
870 IF SQUARE(5)=0 OR SQUARE(3)=0 THEN GOTO 900
880 IF SQUARE(1)=1 OR SQUARE(4)=1 THEN GOTO 900
890 IF SQUARE(7)=1 OR SQUARE(8)=1 THEN GOTO 900
900 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1:GOTO 940
910 IF SQUARE(7)=0 OR SQUARE(5)=0 THEN GOTO 940
920 IF SQUARE(3)=1 OR SQUARE(2)=1 THEN GOTO 940
930 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1
940 REM CHECK NEXT POINT
950 NEXT I 960 REM COUNT THE NUMBER OF CYCLES THE ALGORITHM GOES THROUGH
970 CYCLE=CYCLE+1
980 IF CHANGE>0 THEN GOTO 390
990 FINT$=TIME$
1000 REM PRINT RESULTS
1040 PRINT #2," PROCEDURE BEGAN AT ";BEGT$;" AND ENDED AT ";FINT$
1050 PRINT #2," NUMBER OF CYCLES = ";CYCLE
1060 PRINT #2, " 
1070 PRINT #2, " X ","Y","CODE" 
1080 FLIN=48 1090 START=1
1100 IF IFINAL<=FLIN THEN FLIN=IFINAL
1110 FOR I=START TO FLIN
1120 PRINT #2, ";X(I),Y(I),CODE(I) 
1130 NEXT I 1170 REMAIN=IFINAL-FLIN
1180 IF REMAIN<0 THEN END
1190 START=START+FLIN
1200 IF REMAIN>54 THEN FLIN=FLIN+54
1210 IF REMAIN<=54 THEN FLIN=FLIN+REMAIN
1220 GOTO 1110
1230 REM ********** SET SQUARE POSITIONS **********
1240 CX=X(I)+1
1250 CY=Y(I) 1260 SQUARE(1)=MAT(CX,CY)
1270 CY = Y(I) + 1
1280 SQUARE(2) = MAT(CX, CY)
1290 CX = X(I) 1300 SQUARE(3) = MAT(CX, CY)
1310 CX = X(I) - 1
1320 SQUARE(4) = MAT(CX, CY)
1330 CY = Y(I) 1340 SQUARE(5) = MAT(CX, CY)
1350 CY = Y(I) - 1
1360 SQUARE(6) = MAT(CX, CY)
1370 CX = X(I) 1380 SQUARE(7) = MAT(CX, CY)
1390 CX = X(I) + 1
1400 SQUARE(8) = MAT(CX, CY)
1410 CY = Y(I) 1420 SQUARE(9) = MAT(CX, CY)
1430 RETURN
A TEST PROGRAM TO DETERMINE THE SPEED AND USEFULNESS
OF A THINNING ALGORITHM PROPOSED BY GOVINDAN &
SHIVAPRASAD PAPER ENTITLED "A PATTERN ADAPTIVE
THINNING ALGORITHM" PUBLISHED IN PATTERN
RECOGNITION, VOL. 20, NO. 6, PP 623-637, 1987
REM
REM CREATE ARRAYS
DIM X(100), Y(100), SQUARE(9), MAT(100,100), CODE(100)
REM INITIALIZE ARRAYS
FOR I=1 TO 100
FOR J=1 TO 100
MAT(I,J)=0
NEXT J
NEXT I
FOR I=1 TO 100
CODE(I)=0
NEXT I
BEGIN$=TIME$
REM INPUT THE COORDINATES OF THE POINT THAT LIE ON THE
FIGURE
OPEN "C:\VW3\NEW\DATA" FOR INPUT AS #1
OPEN "C:\VW3\NEW\GOVIN4.DAT" FOR OUTPUT AS #2
I=1
INPUT #1, X(I)
IF X(I)=999 THEN GOTO 330
INPUT #1, Y(I)
CX=X(I)
CY=Y(I)
MAT(CX,CY)=1
I=I+1
GOTO 250
FINAL=I-1
REM ********** BEGIN ALGORITHM **********
CYCLE=0
REM BEGIN CYCLE
CHANGE=0
FOR I=1 TO FINAL
IF CODE(I)=1 THEN GOTO 600
CALL SUBROUTINE TO DETERMINE THE STATUS OF
SURROUNDS
GOSUB 850
CX=X(I)
CY=Y(I)
REM CALCULATE X AND THE NO. OF SURROUNDS
X=0
BP=0
FOR K=1 TO 8
IF SQUARE(K)=1 AND SQUARE(K+1)=0 THEN X=X+1
NEXT K
BP=BP+SQUARE(K)
540 NEXT K
550 REM BEGIN CHECKING THE CONDITIONS
560 IF BP <=1 THEN GOTO 600
570 IF BP >=7 THEN GOTO 600
580 IF X <>1 THEN GOTO 600
590 CODE(I)=1:MAT(CX,CY)=0:CHANGE=CHANGE+1
600 NEXT I
610 REM COUNT THE NUMBER OF CYCLES THE ALGORITHM GOES THROUGH
620 CYCLE=CYCLE+1
630 IF CHANGE>0 THEN GOTO 380
640 FINT$=TIME$
650 REM PRINT RESULTS
660 PRINT #2,"PROCEDURE BEGAN AT ";BEGT$;" AND ENDED AT ";FINT$
670 PRINT #2,"NUMBER OF CYCLES = ";CYCLE
680 PRINT #2,"X ","Y","CODE"
700 FLIN=48
710 START=1
720 IF IFINAL<=FLIN THEN FLIN=IFINAL
730 FOR I=START TO FLIN
740 PRINT #2,"X(I),Y(I),CODE(I)
750 NEXT I
760 FOR I=1 TO 12
770 LPRINT
780 NEXT I
790 REMAIN=IFINAL-FLIN
800 IF REMAIN<=0 THEN END
810 START=START+FLIN
820 IF REMAIN>54 THEN FLIN=FLIN+54
830 IF REMAIN<=54 THEN FLIN=FLIN+REMAIN
840 GOTO 730
850 REM ********** SET SQUARE POSITIONS **********
860 CX=X(I)+1
870 CY=Y(I)
880 SQUARE(1)=MAT(CX,CY)
890 CY=Y(I)+1
900 SQUARE(2)=MAT(CX,CY)
910 CX=X(I)
920 SQUARE(3)=MAT(CX,CY)
930 CX=X(I)-1
940 SQUARE(4)=MAT(CX,CY)
950 CY=Y(I)
960 SQUARE(5)=MAT(CX,CY)
970 CY=Y(I)-1
980 SQUARE(6)=MAT(CX,CY)
990 CX=X(I)
1000 SQUARE(7)=MAT(CX,CY)
1010 CX=X(I)+1
1020 SQUARE(8)=MAT(CX,CY)
1030 CY=Y(I)
1040 SQUARE(9)=MAT(CX,CY)
1050 RETURN
APPENDIX B

CORNER DETECTION PROGRAM LISTINGS
REM PROGRAM TO OUTPUT VALUES CALCULATED FROM A CORNER
REM DETECTION METHOD DISCUSSED BY ROSENFELD AND
JOHNSON
REM (1973)
DIM
X(100), Y(100), AX(100,20), AY(100,20), BX(100,20), BY(100,20), C
(100,20), MAXC(100)
REM Open Data Files
OPEN "SHAPE" FOR INPUT AS#1
OPEN "CURVE" FOR OUTPUT AS#2
N=1
REM Read in Data
WHILE NOT EOF(1)
N=N+1
INPUT #1, X(N), Y(N)
LOOP: WEND
CLOSE #1
FOR J=1 TO N
C(J,1)=-1
NEXT J
REM N is the No. of Data points.
REM Setting M to N/10 is recommended on p. 875
M=INT(N/10)
REM Values are calculated for different gradient values from 2
to M
FOR K=2 TO M
REM The cosine of the angle between a(k) and b(k) is calculated
REM for every possible corner point for each value of k
FOR J=(K+1) TO (N-K-1)
AX(J,K)=X(J)-X(J+K)
AY(J,K)=Y(J)-Y(J+K)
BX(J,K)=X(J)-X(J-K)
BY(J,K)=Y(J)-Y(J-K)
C(J,K)=(AX(J,K)*BX(J,K)+AY(J,K)*BY(J,K))/(SQR(AX(J,K)^2+AY(J,
K)^2)*SQR(BX(J,K)^2+BY(J,K)^2))
NEXT J
NEXT K
REM The maximum cosine value is found for each point (Max
value over all k)
FOR J=M TO (N-K)
FOR K=M TO 2 STEP -1
IF C(J,K)>C(J,K-1) THEN MAXC(J)=C(J,K): GOTO LOOP1
NEXT K
LOOP1: NEXT J
FOR J=1 TO (M-1)
PRINT #2, 0
NEXT J
REM This final result is output to a data file
FOR J=M TO (N-K)
PRINT #2, MAXC(J)
NEXT J
END
REM PROGRAM TO OUTPUT VALUES CALCULATED FROM A CORNER DETECTION METHOD
REM DESCRIBED IN RAMER (1972)
DIM
VALX(100),VALY(100),DX(100),DY(100),SUMX(100),SUMY(100),CORN(100)
OPEN "SHAPE" FOR INPUT AS#1
OPEN "CURVE" FOR OUTPUT AS#2
I=0
WHILE NOT EOF(1)
I=I+1
INPUT #1, VALX(I),VALY(I)
LOOP: WEND
CLOSE #1
FOR J=1 TO 1-8
CORN(J)=0
FOR K=J+1 TO J+8
H=SQR((VALX(K)-VALX(J))^2+(VALY(K)-VALY(J))^2)
CSA=((VALX(K)-VALX(J))*(VALX(J+9)-VALX(J))+(VALY(K)-VALY(J))*(VALY(J+9)-VALY(J)))/(SQR((VALX(K)-VALX(J))^2+(VALY(K)-VALY(J))^2)*SQR((VALX(J+9)-VALX(J))^2+(VALY(J+9)-VALY(J))^2))
X=H*CSA
DIST=SQR(H^2-X^2)
IF DIST>CORN(J) THEN CORN(J)=DIST
NEXT K
NEXT J
FOR J=1 TO 3
PRINT#2, 0
NEXT J
FOR J=2 TO 1-9
PRINT#2,CORN(J)
NEXT J
END
REM PROGRAM TO OUTPUT VALUES CALCULATED FROM A CORNER DETECTION METHOD
REM DESCRIBED IN FREEMAN, H. AND DAVIS, L, "A CORNER-FINDING ALGORITHM FOR REM CHAIN CODED CURVES", IEEE TRANSACTIONS ON COMPUTERS, C-26, 1977, PP. 297 TO 303.
DIM VALX(100),VALY(100),DX(100),DY(100),SUMX(100), SUMY(100),THETA(100),GAMMA(100)
OPEN "SHAPE" FOR INPUT AS#1
OPEN "CURVE" FOR OUTPUT AS#2
I=0
WHILE NOT EOF(1)
I=I+1
INPUT #1, VALX(I),VALY(I)
LOOP: WEND
CLOSE #1
FOR K=1 TO I
SUMX(K)=0
SUMY(K)=0
NEXT K
FOR J=2 TO I
DX(J)=VALX(J)-VALX(J-1)
DY(J)=VALY(J)-VALY(J-1)
NEXT J
FOR J=2 TO (I-6)
FOR K=J TO (J+6)
SUMX(J)=SUMX(J)+DX(K)
SUMY(J)=SUMY(J)+DY(K)
NEXT K
NEXT J
FOR J=2 TO (I-6)
IF ABS(SUMX(J))=0 THEN GOTO LOOP2
THETA(J)=ATN(SUMY(J)/SUMX(J))
GOTO LOOP3
LOOP2: THETA(J)=(ATN(0))
LOOP3: PRINT THETA(J)
NEXT J
FOR J=3 TO (I-7)
GAMMA(J)=THETA(J+1)-THETA(J-1)
NEXT J
FOR J=1 TO 6
PRINT#2, 0
NEXT J
FOR J=3 TO 1-7
PRINT#2,GAMMA(J)
NEXT J
END
APPENDIX C

USER'S GUIDE
MACHINE VISION SYSTEM USER'S GUIDE

The machine vision system software that has been developed in this thesis is interactive and requires the user to know relatively little about the system, however, a novice user may find that more information is needed to operate the machine vision system than the information that is displayed by the machine vision system software. It is the objective of this user's guide to make this information available to the user.

USING THE SYSTEM

In order to use the machine vision system the user must turn on the digital camera, the scene monitor, the memory buffer and the robot. Once these have been turned on, the floppy diskette labeled "Machine Vision System Disk A" should be placed into the default diskette drive on the Apple IIe computer (as the system is currently set up the default diskette drive is the drive on the left). The floppy diskette labeled "Machine Vision System Disk B" should be placed into the remaining diskette drive. Turn on the Apple IIe microcomputer.

You can sit back and relax for a few seconds as the computer loads the appropriate programs into memory and executes the software that is necessary to run the machine vision system. When the computer is ready a title screen is displayed that shows you the name of the system and other information. When you are done looking at this, simply press the key labeled "Return" on the right side of
the keyboard (This key will be referred to as the return key throughout this user's guide).

After a second or two the computer displays a second screen of information that explains how you can change an incorrect response that you have input. Remember that any response you input can be changed before you press the return key by: 1) using the key with the arrow that points to the left to move the cursor over top of the incorrect response, 2) typing the correct response and 3) pressing the return key. Anytime that the computer expects you to do something it will display the information as highlighted text. Any reply that the computer expects you to enter will also be displayed as highlighted text. In the second screen you will notice that the following text is highlighted: "DO YOU WISH TO CONTINUE? (Y/N)". This phrase attempts to supply you with two pieces of information, first the question that the computer would like a response to and second the possible replies that you can give to the computer. In this case you can answer the question as yes or no by simply pressing the key "Y" or the key "N" and then pressing the enter key. Since the computer expects you to answer the question by pressing the "Y" or "N" key, any other key you press will not be accepted by the computer. Now press the key "Y" and press return.

Once this key has been pressed the computer screen will show a line of text for a short period of time that says "PLEASE WAIT". You will see this message intermit-
tently when running the machine vision system. When this message appears, simply sit back and relax for a few seconds while the computer is busy working. When the computer is ready for more information from you it will ask for it. Do not press any keys when this message is displayed, as you may interrupt the work that the computer is doing.

When the "PLEASE WAIT" screen disappears, a list of options will be displayed. You may choose the option that you want by pressing the key on the keyboard that corresponds to the number shown to the left of the desired option and then pressing return.

The system will continue to give you instructions and ask you questions while it is running. There are four types of screens that the computer may show you. If you have been following these instructions then you have already seen three of these four types of screens. These four types of screens are described in the following paragraphs.

The first screen is the information screen. This type of screen is similar to the first two screens presented by the machine vision system. The information screen presents you with some information or instructions. Once you are done reading the information you must indicate to the computer that you have read the information by answering a question or pressing a key.

The second type of screen is the wait screen. This type of screen displays the words "PLEASE WAIT" in high-
lighted letters. This type of screen may also present you with some general information about why you are waiting. When this type of screen is being displayed, you should not input anything into the computer. Just simply relax and wait for the computer to ask you for more information.

The third type of screen is the selection screen. This type of screen displays a number of options from which you must choose one and only one option. You should read through these options carefully, choose the one that you would like the computer to carry out, type the number that appears to the left of the desired selection and press the return key. Once the return key has been pressed the computer will carry out the action that you have indicated.

The final type of screen is the input screen. Unlike the other screens this type of screen does not give you a number of discrete options from which to choose, instead it asks you to type in a number or some text and then press return. When the number or text that you are entering can adversely affect the operation of the machine vision system, the system will suggest appropriate numbers. Screens of this type only exist in the change parameters and add another object to memory modules.

WHAT TO DO IF THE SCREEN GOES BLANK

When the screen goes blank for more than a few seconds one of two things might have happened. You may have inadvertently stopped the machine vision system and exited
to the Apple Disk Operating System or an error might have occurred in the machine vision system software. If you have exited the machine vision system software then you can execute it again by typing "RUN", pressing the space bar (the long horizontal key at the bottom of the keyboard) and then typing "HELLO,D1". When the return key is pressed the computer should return you to the first screen of the machine vision system programs. If this does not work then turn the computer off, wait a few seconds and turn it back on again. After a brief period of time the computer should display the first screen of the machine vision system software package.

WHAT IF THE SOFTWARE REPEATEDLY DOES NOT RECOGNIZE THE OBJECT

If the system does not recognize the object repeatedly then a number of things may have happened. First the lighting conditions may have changed. The surface that the camera views should always be evenly lit. This problem can also occur if the parameters have been changed. To correct this problem go to the first options screen and choose set parameters. Be sure that all of the parameters are set to their desired values. If you are not sure what these parameters should be then it is safe to use the default parameters that are displayed by the machine vision system. The third possible problem is that the object has not been stored into the computer's memory. This may be done by running the option entitled "ADD AN
OBJECT TO MEMORY" in the main option screen. If you are not sure whether this object has been stored or not you may view the objects that have been stored by selecting the option "VIEW AND/OR DELETE OBJECTS FROM MEMORY" and answering the questions shown on the computer screen. If this object has already been stored then the system should not allow you to store it again.
APPENDIX D

PROGRAM DOCUMENTATION
# ASSEMBLY PROGRAMS

<table>
<thead>
<tr>
<th>PROG. NAME</th>
<th>PROG. ID.</th>
<th>LOCATION</th>
<th>LINES</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTR2.OBJ0</td>
<td>$6A00-$6A9B</td>
<td>149</td>
<td>Executes all of the assembly programs used during the machine vision process.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ1 TRAXY</td>
<td>$7D70-$7D8B</td>
<td>26</td>
<td>Transfers Data from a base address specified by the variable LADX to a base address specified by the variable LADY.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ2 FINDOB</td>
<td>$7850-$7D6C</td>
<td>534</td>
<td>Determines the approximate position of an object in a low resolution window of data.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ3 ROCED</td>
<td>$77C0-$781D</td>
<td>68</td>
<td>Determines the Roberts Gradient Value at a given point in a matrix.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ4 MULTPL</td>
<td>$77A0-$781D</td>
<td>24</td>
<td>Multiplies.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ5 DIVID</td>
<td>$7770-$779E</td>
<td>28</td>
<td>Divides.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ6 PRMAT</td>
<td>$9030-$90C8</td>
<td>68</td>
<td>Moves a four kilobyte block of data from bank switched memory to lower memory.</td>
<td></td>
</tr>
<tr>
<td>CONTR2.OBJ7 OTMSG1</td>
<td>$7640-$76FB</td>
<td>131</td>
<td>Prints messages from assembly.</td>
<td></td>
</tr>
<tr>
<td>NVOS.OBJ0 NVOS</td>
<td>$7D90-$7FF5</td>
<td>433</td>
<td>Device driver for the machine vision camera interface.</td>
<td></td>
</tr>
<tr>
<td>LNKLST.OBJ0 LNKLST</td>
<td>$6B00-$75CC</td>
<td>1355</td>
<td>Finds the position of an object in the high resolution screen and follows the edge of the object.</td>
<td></td>
</tr>
<tr>
<td>Procedure Object</td>
<td>Procedure</td>
<td>Address Range</td>
<td>Line</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>CORDET.OBJ0</td>
<td>CORDET</td>
<td>$9300-$95A5</td>
<td>349</td>
<td>Finds the object's corners.</td>
</tr>
<tr>
<td>CORDET.OBJ1</td>
<td>TRANED</td>
<td>$90D0-$9147</td>
<td>65</td>
<td>Transfers edge list from bank switched to lower memory.</td>
</tr>
<tr>
<td>LINREG.OBJ0</td>
<td>LINREG</td>
<td>$9150-$925B</td>
<td>140</td>
<td>Computes sums for linear regression.</td>
</tr>
</tbody>
</table>
### BASIC PROGRAMS

<table>
<thead>
<tr>
<th>PROG. NAME</th>
<th>LINES</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELLO</td>
<td>42</td>
<td>Displays Startup Screen, General Instructions Screen and Executes the Selection Program.</td>
</tr>
<tr>
<td>SEINPROG</td>
<td>155</td>
<td>Allows the User to Select the Process to Execute, Loads the Proper Programs into Memory, Moves the Robot if Necessary and Executes the Appropriate Program.</td>
</tr>
<tr>
<td>VISNPROG</td>
<td>315</td>
<td>Executes the Assembly Machine Vision Programs, Calculates Linear Regression, Orders Edges, Calculates Corner Coordinates, Commands the Robot, Calls Printing Program.</td>
</tr>
<tr>
<td>PRROT</td>
<td>260</td>
<td>Allows the User to Printout Gray Scale and Gradient Matrices, Edge Lists, Corner Strengths and Points, and Final Corner Coordinates.</td>
</tr>
<tr>
<td>DISPOBJ</td>
<td>90</td>
<td>Allows the User to View and/or Delete Objects from Memory.</td>
</tr>
<tr>
<td>SETPARAM</td>
<td>395</td>
<td>Allows the User to Set the Parameters Used by the Machine Vision System.</td>
</tr>
<tr>
<td>SHPSTOR</td>
<td>365</td>
<td>Allows the User to Store New Objects into the Computer's Memory, Executes Assembly Machine Vision Programs, Computes Linear Regression, Orders Edges, Computes Corner Coordinates, Compares Coordinates to Objects in Memory, Stores New Coordinates.</td>
</tr>
</tbody>
</table>
APPLE IIe MEMORY MAP

$0000-$06FF
DESIGNATED FOR APPLE SYSTEM USE

$0700-$57FF
BASIC PROGRAMS AND DATA

$6A00-$6AA2
ASSEMBLY PROCESS CONTROL PROGRAM

$6800-$7630
EDGE LINKING PROGRAM

$7640-$776C
PROGRAM TO DISPLAY INFORMATION FROM THE ASSEMBLY PROCESS TO THE DISPLAY SCREEN

$7770-$779E
DIVISION PROGRAM

$77A0-$77BF
MULTIPLICATION PROGRAM

$77C0-$7741
PROGRAM TO CALCULATE A ROBERTS GRADIENT VALUE FOR A MATRIX ELEMENT

$7850-$7D6B
WINDOWING PROGRAM

$7D70-$7D8B
PROGRAM TO MOVE A BLOCK OF DATA IN MEMORY

$7D90-$7FFF
VISION OPERATING SYSTEM

$8000-$901F
RESERVED FOR STORAGE OF TEMPORARY DATA

$9020-$90C7
VISION SYSTEM PARAMETERS

$9030-$90C7
PROGRAM TO CREATE COMPLETE ROBERTS GRADIENT MATRICES FOR PRINTING

$90D0-$9147
CORDET.OBJ1

$9150-$925C
PROGRAM TO COMPUTE SUMS FOR LINEAR REGRESSION

$9300-$95A2
CORNER DETECTION PROGRAM

$9600-$9CFF
RESERVED FOR APPLE SYSTEM

$D000-$DFFF B1
GRAY SCALE VALUES FOR THE VIEWING AREA

$D000-$DFFF B2
GRAY SCALE VALUES FOR THE WINDOW

NOTE: $ Indicates a Hexidecimal Value
B1 Indicates Bank 1 of Bank Switched Memory
B2 Indicates Bank 2 of Bank Switched Memory
LIST OF VARIABLES

The following is a list of variables used by the BASIC programs. The variables used by the assembly programs are defined at the top of the listing of each program.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A$</td>
<td>Input Variable</td>
</tr>
<tr>
<td>AAC</td>
<td>Arc cosine of the angle between the base of the object and the coordinate system</td>
</tr>
<tr>
<td>AC</td>
<td>A coefficient for line/absolute coordinate in ordered list</td>
</tr>
<tr>
<td>A0</td>
<td>Shape initial angle</td>
</tr>
<tr>
<td>BBC</td>
<td>Arc cosine of the shape orientation angle</td>
</tr>
<tr>
<td>BC</td>
<td>B coefficient for line/absolute coordinate in ordered list</td>
</tr>
<tr>
<td>BO</td>
<td>Shape orientation angle</td>
</tr>
<tr>
<td>CC( , )</td>
<td>Corner coordinates</td>
</tr>
<tr>
<td>CD( )</td>
<td>Corner distance</td>
</tr>
<tr>
<td>CO( ,7)</td>
<td>Origin coordinates; X0,Y0,A0,PS,NSHO,PLIN,RD</td>
</tr>
<tr>
<td>ED</td>
<td>Error distance for recognition of object</td>
</tr>
<tr>
<td>EDCO%</td>
<td>Edge code</td>
</tr>
<tr>
<td>ER%</td>
<td>Error flag</td>
</tr>
<tr>
<td>ET</td>
<td>Error threshold for corner comparison</td>
</tr>
<tr>
<td>FCOL</td>
<td>First column of data to be printed on this page</td>
</tr>
<tr>
<td>HSKP</td>
<td>Horizontal number of pixels skipped in image</td>
</tr>
<tr>
<td>GP</td>
<td>Gripper distance to open</td>
</tr>
<tr>
<td>IDS( )</td>
<td>Shape identifier</td>
</tr>
<tr>
<td>L</td>
<td>Robot, shoulder to elbow &amp; elbow to wrist</td>
</tr>
<tr>
<td>LCOL</td>
<td>Last column of data to be printed on this page</td>
</tr>
<tr>
<td>LL</td>
<td>Robot, wrist to fingertip</td>
</tr>
<tr>
<td>LWIN</td>
<td>Left corner of camera window</td>
</tr>
<tr>
<td>NC%</td>
<td>Number of corners</td>
</tr>
<tr>
<td>NCOL</td>
<td>Number of columns of data to be printed</td>
</tr>
<tr>
<td>NE%</td>
<td>Number of elements in linear regression</td>
</tr>
<tr>
<td>NLIN</td>
<td>Number of lines of data to be printed</td>
</tr>
<tr>
<td>OA%</td>
<td>Rank of edge after longest edge</td>
</tr>
<tr>
<td>OB%</td>
<td>Rank of edge before longest edge</td>
</tr>
<tr>
<td>OD%</td>
<td>Difference in rank</td>
</tr>
<tr>
<td>OE%</td>
<td>Difference in rank for comparison</td>
</tr>
<tr>
<td>OP%</td>
<td>Order Pointer</td>
</tr>
<tr>
<td>PI</td>
<td>Constant 3.14159</td>
</tr>
<tr>
<td>PO</td>
<td>Memory location to poke data in when setting parameters</td>
</tr>
<tr>
<td>R</td>
<td>Roll angle for the gripper</td>
</tr>
<tr>
<td>RB( , )</td>
<td>Robot coordinates matrix (joint coordinates)</td>
</tr>
<tr>
<td>RK%( )</td>
<td>Length rank of CD</td>
</tr>
<tr>
<td>R1</td>
<td>Roll vertical cartesian frame</td>
</tr>
<tr>
<td>RW%</td>
<td>Data read/write flag</td>
</tr>
<tr>
<td>S</td>
<td>Robot speed</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>SN%</td>
<td>Shape number detected</td>
</tr>
<tr>
<td>ST%</td>
<td>Number of Stored Shapes</td>
</tr>
<tr>
<td>SX</td>
<td>Sum of X-coordinates for linear regression</td>
</tr>
<tr>
<td>SY</td>
<td>Sum of Y-coordinates for linear regression</td>
</tr>
<tr>
<td>TREDG%</td>
<td>Threshold for edge detection</td>
</tr>
<tr>
<td>TWIN</td>
<td>Top corner of camera window</td>
</tr>
<tr>
<td>VA%</td>
<td>Parameter value input by user</td>
</tr>
<tr>
<td>VSKP</td>
<td>Vertical number of pixels skipped in image</td>
</tr>
<tr>
<td>X</td>
<td>Robot cartesian X-coordinate</td>
</tr>
<tr>
<td>XC( )</td>
<td>X-coordinate relative and absolute</td>
</tr>
<tr>
<td>XY</td>
<td>Sum X*Y coordinates for regression</td>
</tr>
<tr>
<td>XO</td>
<td>Shape position X-coordinate</td>
</tr>
<tr>
<td>Xl</td>
<td>Shape base X-coordinate</td>
</tr>
<tr>
<td>Y</td>
<td>Robot cartesian Y-coordinate</td>
</tr>
<tr>
<td>YC( )</td>
<td>Y coordinates of corners (relative and absolute)</td>
</tr>
<tr>
<td>Y0</td>
<td>Shape position Y-coordinate</td>
</tr>
<tr>
<td>Y1</td>
<td>Shape base Y-coordinate</td>
</tr>
<tr>
<td>Z</td>
<td>Robot cartesian Z-coordinate</td>
</tr>
<tr>
<td>ZX</td>
<td>Sum X^2 coordinates</td>
</tr>
</tbody>
</table>
APPENDIX E

BASIC PROGRAM LISTINGS
10 D$ = CHR$(4)
20 REM HELLO FOR VISION SYSTEM 12/1/88
30 REM DISPLAY TITLE PAGE
40 TEXT = HOME
50 VTAB(4): HTAB(6): PRINT "AN IMPROVED, LOW COST MACHINE"
60 VTAB(5): HTAB(6): PRINT "VISION SOFTWARE PACKAGE FOR"
70 VTAB(6): HTAB(6): PRINT "LOCATION AND IDENTIFICATION"
80 VTAB(7): HTAB(6): PRINT "OF RANDOMLY ORIENTED, STRAIGHT"
90 VTAB(8): HTAB(14): PRINT "EDGED OBJECTS"
100 VTAB(12): HTAB(6): PRINT "Copyright Ohio University 1989"
110 VTAB(16): HTAB(2): PRINT "Project Director: Dr. Helmut T. Zwahlen"
120 VTAB(17): HTAB(2): PRINT "Programmer: Michael E. Miller"
130 VTAB(19): HTAB(2): PRINT "Programmed in: Assembly and"
140 VTAB(20): HTAB(20): PRINT "Applesoft BASIC"
150 VTAB(23): HTAB(8): INVERSE :: PRINT "PRESS ANY KEY TO CONTINUE";; NORMAL : GET A$
160 HOME
170 HTAB(15): INVERSE :: PRINT "PLEASE NOTE:";; NORMAL
180 VTAB(2): PRINT "Throughout this program, when you are"
190 PRINT "asked to input a response, you should"
200 PRINT "always input your response, check it,"
210 PRINT "and then press the return key. If you"
220 PRINT "key in a response and realize that it"
230 PRINT "is not correct before you press the"
240 PRINT "return key, then you may correct your"
250 PRINT "response by using the left arrow key"
260 PRINT "to backspace over the incorrect"
270 PRINT "response. A new response may then be"
280 PRINT "typed over the incorrect response."
290 PRINT "(The left arrow key is located near"
300 PRINT "the bottom right corner of the"
310 PRINT "Apple IIe keyboard.)"
315 PRINT
320 INVERSE :: PRINT "CAUTION:";; NORMAL : PRINT "Be sure you ha ve entered the"
330 HTAB(10): PRINT "correct response before you"
340 HTAB(10): PRINT "press the return key. A"
350 HTAB(10): PRINT "response can not be changed"
360 HTAB(10): PRINT "after the return key has"
370 HTAB(10): PRINT "been pressed."
380 VTAB(24): HTAB(5): INVERSE :: INPUT "DO YOU WISH TO CONTINUE? (Y/N) ";A$;; NORMAL
390 IF A$ < > "Y" THEN HOME: END
400 HOME
410 VTAB(12): HTAB(15): INVERSE :: PRINT "PLEASE WAIT";; NORMAL
420 PRINT D$ + "RUN SELNPROG.D1"
10 HOME
15 D$ = CHR$(4)
16 PI = 3.141592654
20 REM ALLOW USER TO SELECT A MODULE
30 VTAB (1); PRINT "Machine Vision System Version 2";
40 VTAB (6); HTAB (2); PRINT "SELECT ONE OF THE FOLLOWING OPTIONS:
50 VTAB (9); HTAB (5); INVERSE; PRINT "1": NORMAL
60 PRINT "Begin the Machine Vision Process"
70 VTAB (11); HTAB (5); INVERSE; PRINT "2": NORMAL
80 PRINT "Add a New Object to Memory"
90 VTAB (13); HTAB (5); INVERSE; PRINT "3": NORMAL
100 PRINT "Display and/or Delete an Object"
110 VTAB (15); HTAB (5); INVERSE; PRINT "4": NORMAL
120 PRINT "Display and/or Change Parameters"
130 VTAB (17); HTAB (5); INVERSE; PRINT "5": NORMAL
140 PRINT "Exit the System"
150 VTAB (21); HTAB (2); PRINT "PLEASE ENTER THE NUMBER OF YOUR"
160 VTAB (22); HTAB (2); INPUT "SELECTION AND PRESS RETURN "; A$
170 IF A$ = "1" THEN GOTO 500
180 IF A$ = "2" THEN GOTO 1000
190 IF A$ = "3" THEN GOTO 1500
200 IF A$ = "4" THEN GOTO 2000
210 IF A$ = "5" THEN HOME: END
220 PRINT CHR$(7)
230 HOME
240 VTAB (10); HTAB (2); PRINT "YOU MUST ENTER A NUMBER BETWEEN 1 AND 5"
250 VTAB (13); HTAB (13); INPUT "PLEASE TRY AGAIN"
260 FOR I = 1 TO 2000
270 NEXT I
280 GOTO 10
300 HOME
500 VTAB (2); HTAB (3); PRINT "MACHINE VISION PROCESS MODULE"
520 VTAB (9); HTAB (2); PRINT "PLACE THE ROBOT IN ITS INITIAL"
530 VTAB (10); HTAB (2); PRINT "POSITION BEFORE CONTINUING"
540 VTAB (17); HTAB (2); INPUT "DO YOU WISH TO CONTINUE? (Y/N) "; A$
550 IF A$ < > "Y" THEN GOTO 10
560 HOME
570 VTAB (2); HTAB (3); PRINT "MACHINE VISION SYSTEM PROCESS MOD
580 VTAB (9); HTAB (15); INVERSE; PRINT "PLEASE WAIT": NORMAL
590 VTAB (12); HTAB (5); PRINT "The programs and data needed to"
600 VTAB (13); HTAB (7); PRINT "complete the machine vision"
610 VTAB (14); HTAB (6); PRINT "process are being loaded into"
620 VTAB (15); HTAB (10); PRINT "the computer's memory"
630 REM LOAD PROCESS INTO MEMORY
635 PRINT D$;"LOAD NVOS.OBJ0,02"
640 PRINT D$;"BLOAD CONTR2.OBJ0,D2"
650 PRINT D$;"BLOAD CONTR2.OBJ1,D2"
660 PRINT D$;"BLOAD CONTR2.OBJ2,D2"
670 PRINT D$;"BLOAD CONTR2.OBJ3,D2"
680 PRINT D$;"BLOAD CONTR2.OBJ4,D2"
690 PRINT D$;"BLOAD CONTR2.OBJ5,D2"
700 PRINT D$;"BLOAD CONTR2.OBJ6,D2"
710 PRINT D$;"BLOAD CONTR2.OBJ7,D2"
720 PRINT D$;"BLOAD LINKLIST.OBJ0,D2"
730 PRINT D$;"BLOAD CORDET.OBJ0,D2"
740 PRINT D$;"BLOAD CORDET.OBJ1,D2"
750 PRINT D$;"BLOAD LINREG.OBJ0,D2"
760 PRINT D$;"BLOAD PARAMETERS,O2"
770 HOME
780 VTAB (9): HTAB (15): INVERSE : PRINT "PLEASE WAIT": NORMAL
790 VTAB (12): HTAB (7): PRINT "The robot is being moved out"
800 VTAB (13): HTAB (11): PRINT "of the viewing area."
810 GOSUB 3000: REM MOVE THE ROBOT OUT OF THE VIEWING AREA
820 REM BEGIN MACHINE VISION PROCESS
830 PRINT D$ + "RUN VISNPROG,D1"
840 HOME
850 VTAB (2): HTAB (8): PRINT "OBJECT MEMORIZATION MODULE"
860 VTAB (9): HTAB (4): PRINT 'PLEASE CLEAR EVERYTHING OUT OF THE"
870 VTAB (10): HTAB (5): PRINT "CAMERA'S VIEWING AREA EXCEPT THE"
880 VTAB (11): HTAB (5): PRINT "OBJECT YOU WOULD LIKE THE MACHINE"
890 VTAB (12): HTAB (5): PRINT "VISION SYSTEM TO MEMORIZE BEFORE"
900 VTAB (13): HTAB (16): PRINT "CONTINUING"
910 VTAB (17): HTAB (7): INPUT "DO YOU WISH TO CONTINUE? (Y/N)"
920 IF A$ < > "Y" THEN GOTO 10
930 HOME
940 VTAB (2): HTAB (8): PRINT "OBJECT MEMORIZATION MODULE"
950 VTAB (9): HTAB (15): INVERSE : PRINT "PLEASE WAIT": NORMAL
960 VTAB (12): HTAB (5): PRINT "The programs and data needed to"
970 VTAB (13): HTAB (5): PRINT "complete the object memorization"
980 VTAB (14): HTAB (6): PRINT "process are being loaded into"
990 VTAB (15): HTAB (10): PRINT "the computer's memory"
1000 REM LOAD PROCESS INTO MEMORY
1005 HIMEM: 27136: REM $6A00
1010 PRINT D$;"BLOAD CONTR2.OBJ0,D2"
1015 PRINT D$;"BLOAD NVS.OBJ0,D2"
1020 PRINT D$;"BLOAD CONTR2.OBJ1,D2"
1025 PRINT D$;"BLOAD CONTR2.OBJ2,D2"
1030 PRINT D$;"BLOAD CONTR2.OBJ3,D2"
1210 PRINT D$:*"BLOAD CONTR2.OBJ4,D2"
1220 PRINT D$:*"BLOAD CONTR2.OBJ5,D2"
1230 PRINT D$:*"BLOAD CONTR2.OBJ6,D2"
1240 PRINT D$:*"BLOAD CONTR2.OBJ7,D2"
1250 PRINT D$:*"BLOAD LNKLIST.OBJ0,D2"
1260 PRINT D$:*"BLOAD CORDER.OBJ0,D2"
1270 PRINT D$:*"BLOAD CORDER.OBJ1,D2"
1280 PRINT D$:*"BLOAD LINREG.OBJ0,D2"
1290 PRINT D$:*"BLOAD PARAMETERS,D1"
1310 PRINT CHR$(4) + "RUN SHPSTOR,D1"
1320 END
1500 HOME
1510 VTAB (9): HTAB (15): INVERSE : PRINT "PLEASE WAIT": NORMAL
1520 VTAB (12): HTAB (7): PRINT "The display object module is"
1530 VTAB (13): HTAB (15): PRINT "being loaded."
1540 PRINT D$:*"RUN DISPOBJ,D1"
2000 HOME
2010 VTAB (9): HTAB (15): INVERSE : PRINT "PLEASE WAIT": NORMAL
2020 VTAB (12): HTAB (5): PRINT "The Parameter Selection Module is"
2040 PRINT D$:*"RUN SETPARAM,D1"
3000 REM INITIALIZE THE ROBOT AND MOVE IT OUT OF THE VIEWING AREA
3010 PRINT D$:*"PR#2"
3020 PRINT D$:*"INH2"
3030 S = 236: REM ROBOT SPEED
3040 PRINT "#STEP"S:",1767,-200,0,0,0,0,0
3050 H = 7.625: REM SHOULDER HT ABOVE TABLE
3060 L = 7.0: REM SHLDER TO ELBOW&ELBOW TO WRIST
3070 LL = 3.8: REM WRIST TO FINGERTIP
3080 R1 = 1: REM ROLL WRT CARTESIAN FRAME
3090 P = - PI / 2
3100 R = 0
3110 DATA 0,-508,1162,384,384,0
3115 DATA 1125,-1125,-661.2,-244.4,-244.4,371
3120 RESTORE
3130 FOR I = 1 TO 6
3140 READ RB(1,I): REM READ INITIAL POINTS
3150 NEXT I
3160 FOR I = 1 TO 6
3170 READ RB(2,I): REM READ STEPS PER RADIAN (INCH) OF JOINTS
3180 NEXT I
3190 FOR I = 1 TO 6
3200 RB(3,I) = 0
3210 NEXT I
3220 RB(3,1) = 1767: REM CURRENT POSITION
3230 RB(3,2) = -200
3240 INPUT IR
3250 PRINT
3260 PRINT D$; "PR#0"
3270 PRINT D$; "IN#0"
3280 RETURN
E6

5 REM PROGRAM VISNPROG BY MM 12/16/88
6 REM PROGRAM TO COMPLETE THE MACHINE VISION PROCESS
10 REM PROGRAM DRIVER TO MEMORIZE OBJECT
20 DIM SX(B),SY(B),PCK(20)
30 DIM NED%(20),CCR(20,8,2),ID$(20),XC(B)
40 DIM YC(B),CD(B),AC(B),BC(B),RK%(B)
50 D$ = CHR$ (4):PI = 3.14159
100 HCT1E
110 VTAB (9): HTAB (15): INVERSE: PRINT "PLEASE WAIT": NORMAL
120 VTAB (12): HTAB (5): PRINT "The Shape Data are Being Loaded.
130 REM READ SHAPE DATA
140 Rw% = 0
150 GOSUB 3030
160 HCT1E
170 VTAB (11): HTAB (3): PRINT "DO YOU WISH TO CONTINUE THE MACHINE"
180 VTAB (12): HTAB (10): INPUT "VISION PROCESS? (Y/N) ";A$
190 IF A$ < > "Y" THEN PRINT D$ + "RUN SELNPROG,D1"
200 HCT1E
210 PRINT "BEGIN THE MACHINE VISION PROCESS"
220 CALL 27136
225 IF PEEK (36910) < > 0 THEN GOTO 965
240 NLIN% = PEEK (33792)
250 GOSUB 1040: REM LINEAR REGRESSION
260 GOSUB 1280: REM CORNER CALCULATION
270 IF ERX < > 0 THEN GOTO 960
280 GOSUB 1510: REM CALCULATE ABS. COORD.
290 GOSUB 1640: REM REORDER CORNERS
300 GOSUB 2530: REM CALCULATION OF SHAPE CENTERED COORDINATES
310 ET = .01 * PEEK (36906)
320 GOSUB 3290: REM OBJECT IDENTIFICATION
330 PRINT : PRINT : HTAB (9): PRINT "The object has ";NLIN%;" corners"
340 IF ER% = 1 THEN HTAB (7): PRINT "and has been identified as a"
350 IF ER% = 1 THEN HTAB ((40 - LEN (ID$(RECSHP%))) / 2 + 1): PRINT ID$(RECSHP%)
360 IF ER% = 0 THEN HTAB (10): PRINT "Object Not Recognized"
365 IF ER% = 0 THEN PRINT : PRINT "PRESS ANY KEY TO CONTINUE";
370 GET A$; GOTO 500
370 ER% = 0
380 IF PCK(RECSHP%) = 0 THEN PRINT : PRINT "ROBOT RETREIVAL NOT POSSIBLE": GOTO 480
390 PRINT : PRINT "ROBOT RETREIVAL OF OBJECT"
400 PRINT
410 GOSUB 3418: REM ROBOT DRIVE
420 GOTO 500
480 PRINT : PRINT : PRINT "PRESS ANY KEY TO CONTINUE";
490 GET A$
500 HOME : VTAB (6): HTAB (2): PRINT "SELECT ONE OF THE FOLLOWING OPTIONS:"
510 VTAB (9); HTAB (5); INVERSE: PRINT "1": NORMAL
520 PRINT " Run the Machine Vision Process"
530 VTAB (11); HTAB (5); INVERSE: PRINT "2": NORMAL
540 PRINT " Obtain Hardcopy Outputs of the"
550 HTAB (8): PRINT "Previous Machine Vision Run"
560 VTAB (14); HTAB (5); INVERSE: PRINT "3": NORMAL
570 PRINT " Quit the Machine Vision Process"
580 HTAB (8): PRINT "Module and Return to Main Menu"
590 VTAB (19): HTAB (2): PRINT "PLEASE ENTER THE NUMBER OF YOUR"
600 VTAB (20): HTAB (2): INPUT "SELECTION AND PRESS RETURN "; A$
610 IF A$ = "1" THEN HOME: PRINT D$: "LOAD PARAMETERS,D1": GOTO 210
620 IF VAL (A$) = 2 OR VAL (A$) = 3 THEN HOME: VTAB (9): HTAB (5): INVERSE: PRINT "PLEASE WAIT": NORMAL
625 IF VAL (A$) = 2 THEN GOSUB 4500: REM STORE CORNER PTS.
630 IF VAL (A$) = 2 THEN PRINT D$; "RUN PRROT,D1"
640 IF VAL (A$) = 3 THEN PRINT D$; "RUN SELNPROG,D1"
650 HOME: PRINT CHR$(7)
660 VTAB (10): HTAB (2): PRINT "YOU MUST ENTER A NUMBER BETWEEN 1 AND 3"
670 VTAB (13): HTAB (13): PRINT "PLEASE TRY AGAIN"
680 FOR I = 1 TO 1500: NEXT I
690 GOTO 500
950 END
962 PRINT "ERROR NO. = "; ER%
964 ER% = 0
965 PRINT ; PRINT "ERROR IN RECOGNITION"
967 POKE 36910, 0
970 INPUT "DO YOU WISH TO CONTINUE? (Y/N) "; A$
975 IF A$ < "Y" THEN PRINT D$: "RUN SELNPROG,D1"
980 PRINT D$: "LOAD parameters,D1"
985 GOTO 500
990 INPUT I; D$(NSHP-;:)
1000 REM
1010 REM SUBROUTINE TO COMPUTE LINEAR REGRESSION
1020 REM
1040 IF NLIN% < 3 THEN ER% = 1: RETURN
1050 FOR I = 1 TO NLIN%
1070 NE% = PEEK (34175 + I)
1080 SX = PEEK (34303 + I) + 256 * PEEK (34431 + I)
1090 ZX = PEEK (34559 + I) + 256 * PEEK (34687 + I) + 65536 * PEEK (34815 + I)
1100 SY = PEEK (34943 + I) + 256 * PEEK (35071 + I)
1110 XY = PEEK (35199 + I) + 256 * PEEK (35327 + I) + 65536 * PEEK (35455 + I)
1120 IF ((NE% * ZX) - (SX * SX)) = 0 THEN BC(I) = 999: AC(I) = (S X / NE%): GOTO 1150
1130 BC(I) = ((NE% * XY) - (SX * SY)) / ((NE% * ZX) - (SX * SX))
1140 AC(I) = SY / NE% - (BC(I) * (SX / NE%))
1150 NEXT I
1160 RETURN
1250 REM
1260 REM CORNER CALCULATION
1270 REM
1280 FOR I = 1 TO NLIN% - 1
1290 J = I + 1
1290 GOSUB 1400
1300 NEXT I
1310 I = NLIN%
1320 J = 1
1330 GOSUB 1400
1340 RETURN
1400 REM CORNER CALCULATION
1410 IF (BC(I) - BC(J)) = 0 THEN PRINT ; PRINT "LINES ARE PARAL
1410 REM (CORNER CALCULATI (J-4). :
1410 REM
1412 IF BC(I) = 999 THEN XC(I) = AC(I) : GOTO 1421
1414 IF BC(J) = 999 THEN XC(I) = AC(J) : GOTO 1425
1420 XC(I) = (AC(J) - AC(I)) / (BC(I) - BC(J))
1425 IF BC(J) = 999 THEN YC(I) = AC(I) + BC(I) * XC(I) : GOTO 1440
1430 YC(I) = AC(J) + BC(J) * XC(I)
1440 IF XC(I) < - 10 THEN ER% = 2: RETURN
1450 IF XC(I) > 100 THEN ER% = 3: RETURN
1460 IF YC(I) < - 10 THEN ER% = 4: RETURN
1470 IF YC(I) > 100 THEN ER% = 5: RETURN
1480 RETURN
1495 REM
1500 REM CALCULATION OF THE ABSOLUTE COORDINATES
1505 REM
1510 LWIN = PEEK (36877)
1520 TWIN = PEEK (36875)
1530 HSKP = PEEK (36879)
1540 VSKP = PEEK (36880)
1550 FOR I = 1 TO NLIN%
1560 AC(I) = 8 + (LWIN + YC(I)) * (HSKP + 1) - 120) * 0.0407126
1570 BC(I) = (120 - TWIN - XC(I)) * (VSKP + 1)) * 0.031878
1580 NEXT I
1590 RETURN
1600 REM
1610 REM REORDER CORNERS FOR SHAPE CENTERED ORIGIN
1620 REM
1630 REM CALCULATE CORNER DISTANCES
1640 FOR I = 1 TO NLIN% - 1
1650 CD(I) = (AC(I + 1) - AC(I)) * 2 + (BC(I + 1) - BC(I)) * 2
1660 NEXT I
1670 I = NLIN%
1680 CD(I) = (AC(I) - AC(I)) * 2 + (BC(I) - BC(I)) * 2
1690 FOR I = 1 TO NLIN%
1700 RK% (I) = 0
1710 NEXT I
1715 \( K = 0 \)
1720 REM ASSIGN RANK
1730 FOR I = 1 TO NLIN%
1740 IF RKX(I) = 0 THEN GOTO 1770
1750 NEXT I
1760 GOTO 1900: REM RANKING FINISHED
1770 \( K = K + 1 \)
1780 J = I
1790 FOR I = J + 1 TO NLIN%
1800 IF RKX(I) \( < \) 0 THEN GOTO 1830
1810 IF CD(I) \( < \) CD(J) THEN GOTO 1830
1820 J = I
1830 NEXT I
1840 FOR I = 1 TO NLIN%
1850 IF RKX(I) \( > \) 0 THEN GOTO 1880
1860 IF CD(I) \( < \) CD(J) * 0.98 THEN GOTO 1880
1870 RKX(I) = K
1880 NEXT I
1890 GOTO 1730: REM CHECK NEXT RANK
1900 IF K = 1 THEN RETURN: REM NO ORDERING NECESSARY
2000 REM CHECK FOR ORDERING
2010 J = 0
2020 FOR I = 1 TO NLIN%
2030 IF RKX(I) \( < \) 1 THEN GOTO 2160
2040 IF I = 1 THEN OB% = RKX(NLIN%)
2050 IF I \( > \) 1 THEN OB% = RKX(I - 1)
2060 IF I \( < \) NLIN% THEN OA% = RKX(I + 1)
2070 IF I = NLIN% THEN OA% = RKX(I)
2080 IF OA% \( < \) OB% THEN OE% = 100 * OB% - OA%
2090 IF OA% \( > \) OB% THEN OE% = OB% - 100 * OA%
2100 J = J + 1
2110 IF J = 1 THEN GOTO 2140
2120 IF ABS(OE%) \( < \) ABS(OO%) THEN GOTO 2160
2130 IF OE% \( < \) OO% THEN GOTO 2160
2140 OP% = I
2150 OD% = OE%
2160 NEXT I
2170 REM REORDER PROCEDURE
2180 K = 0
2190 OA% = 1
2200 OB% = NLIN%
2210 IF OD% \( < \) 0 THEN GOSUB 2240: REM REORDER POSITIVE
2220 IF OD% \( > \) 0 THEN GOSUB 2360: REM REORDER NEGATIVE
2226 RETURN
2230 REM POSITIVE REORDERING OF ONE SHAPE
2240 FOR I = OP% TO OB%
2250 K = K + 1
2260 \( XC(K) = AC(I) \)
2270 \( YC(K) = BC(I) \)
2280 NEXT I
2290 FOR I = OA% TO OP% - 1
K = K + 1
XC(K) = AC(I)
YC(K) = BC(I)
NEXT I
RETURN

REM NEGATIVE REORDERING OF ONE SHAPE
IF OP% < OB% THEN OP% = OP% + 1
IF OP% = OB% THEN OP% = OA%
FOR I = OP% TO OA% STEP - 1
K = K + 1
XC(K) = AC(I)
YC(K) = BC(I)
NEXT I
RETURN

NEXT I
FOR I = OB% TO OP% + 1 STEP - 1
K = K + 1
XC(K) = AC(I)
YC(K) = BE(I)
NEXT I
RETURN

REM CALCULATION OF SHAPE CENTERED COORDINATES
XO = XC(1)
YO = YC(1)
X1 = XC(2) - XO
Y1 = YC(2) - YO
XC(1) = 0
YC(1) = 0
CO(1) = SQR((XO - XC(2)) ^ 2 + (YO - YC(2)) ^ 2)
AAC = (XC(2) - XO) / CD(1)
AO = -ATN(AAC / SQR(-AAC * AAC + 1)) + PI / 2: REM ARCCOS(AAC)
IF (YC(2) - YO) < 0 THEN AO = 2 * PI - AO
CD(2) = SQR((XC(NLINX) - XO) ^ 2 + (YC(NLINX) - YO) ^ 2)
BCC = (XC(NLINX) - XO) / CD(2)
BO = -ATN(BCC / SQR(-BCC * BCC + 1)) + PI / 2: REM ARCCOS(BCC)
DO = BO - AO
IF DO < 0 THEN DO = DO + PI
IF DO > (PI / 2) THEN AO = - AO
XC(2) = CD(1)
YC(2) = 0
FOR I = 3 TO NLINX
CD(2) = SQR((XC(I) - XO) ^ 2 + (YC(I) - YO) ^ 2)
XC(I) = (X1 * (XC(I) - XO) + Y1 * (YC(I) - YO)) / CD(1)
YC(I) = SQR(1 - (XC(I) * XC(I)) / (CD(2) * CD(2))) * CD(2)
NEXT I
RETURN
REM READ(RW=0) OR WRITE(RW=1) SHAPE DATA INTO MEMORY
3020 REM
3030 PRINT D$;"OPEN SHAPES, D1"
3040 IF RW% = 1 THEN PRINT D$;"WRITE SHAPES"
3050 IF RW% < > 1 THEN PRINT D$;"READ SHAPES"
3060 REM READ/WRITE NO. OF SHAPES IN FILE
3070 IF RW% = 1 THEN PRINT NSHP%
3080 IF RW% < > 1 THEN INPUT NSHP%
3090 REM SHAPE ID, NUMBER OF CORNERS AND CORNER COORDINATES
3100 FOR I = 1 TO NSHP%
3110 IF RW% = 1 THEN PRINT ID$(I); PRINT NEDX(I)
3120 IF RW% < > 1 THEN INPUT ID$(I); INPUT NEDX(I)
3130 FOR J = 1 TO NEDX(I)
3150 IF RW% = 1 THEN PRINT CCR(I,J,1);"CCR(I,J,2)
3160 IF RW% < > 1 THEN INPUT CCR(I,J,1),CCR(I,J,2)
3180 NEXT J
3184 IF RW% = 1 THEN PRINT PCK(I)
3188 IF RW% < > 1 THEN INPUT PCK(I)
3190 NEXT I
3195 PRINT D$;"CLOSE SHAPES"
3200 RETURN
3250 REM
3260 REM SHAPE CLASSIFICATION MODULE
3270 REM
3290 FOR I = 1 TO NSHP%
3295 MCH% = 0
3300 IF NEDX(I) < > NLIN% THEN GOTO 3380
3310 FOR J = 1 TO NEDX(I)
3320 ED = ABS (CCR(I,J,1) - XC(J))
3330 IF ED > ET THEN GOTO 3360
3340 ED = ABS (CCR(I,J,2) - YC(J))
3350 IF ED < = ET THEN MCH% = MCH% + 1
3360 NEXT J
3370 IF MCH% = NLIN% THEN GOTO 3400
3380 NEXT I
3390 RETURN
3400 ER% = 1
3405 REC%HP% = 1
3410 RETURN
3415 REM INITIALIZE ROBOT VARIABLES
3418 S = 236: REM SPEED
3420 H = 7.625: REM SHLD. HGT.
3422 L = 7.0: REM SHLD TO ELBOW+ ELBOW TO WRIST
3424 LL = 3.8: REM WRIST TO FINGERTIP
3426 R1 = 1: REM ROLL RT.
3428 P = - PI / 2
3429 R = 0
3430 REM INITIALIZE ROBOT VARIABLES
3435 DATA 0,-508,1162,384,384,0
3440 DATA 1125,-1125,-661.2,-244.4,-244.4,371
3445 RESTORE
3450 FOR I = 1 TO 6
3455 READ RB(1,I): REM READ INITIAL POINTS
3460 NEXT I
3465 FOR I = 1 TO 6
3470 READ RB(2,I)
3480 NEXT I
3485 FOR I = 1 TO 6
3490 RB(3,I) = 0
3492 NEXT I
3495 RB(3,1) = 1767:RB(3,2) = -200
3500 REM CALCULATION OF THE ROBOT DRIVE COORDINATES
3510 PRINT D$;"PR#2"
3520 PRINT D$;"IN#2"
3530 X = X0 + COS (ABS (A0)) * 0.5 + COS (ABS (A0)) + SGN (A0)
10 REM PROGRAM TO HANDLE PRINTING
20 REM CAPABILITY
30 D$ = CHR$ (4)
80 HOME
90 VTAB (5); HTAB (2): PRINT "SELECT ONE OF THE FOLLOWING OPTION S:"  
100 VTAB (7); PRINT TAB (3);: INVERSE: PRINT "1";: NORMAL
110 PRINT TAB (5);"Print the Gray Scale Matrix for"  
115 VTAB (8); PRINT TAB (5);"the Entire Viewing Area"  
120 VTAB (9); PRINT TAB (3);: INVERSE: PRINT "2";: NORMAL
130 PRINT TAB (5);"Print Roberts Gradient Matrix for"  
135 VTAB (10); PRINT TAB (5);"the Entire Viewing Area"  
140 VTAB (11); PRINT TAB (3);: INVERSE: PRINT "3";: NORMAL
150 PRINT TAB (5);"Print Gray Scale Matrix for Window"  
160 VTAB (12); PRINT TAB (3);: INVERSE: PRINT "4";: NORMAL
170 PRINT TAB (5);"Print the Roberts Gradient Matrix"  
175 VTAB (13); PRINT TAB (5);"for the Window"  
180 VTAB (14); PRINT TAB (3);: INVERSE: PRINT "5";: NORMAL
190 PRINT TAB (5);"Print a Connectivity Diagram for"  
195 VTAB (15); PRINT TAB (5);"the Window"  
200 VTAB (16); PRINT TAB (3);: INVERSE: PRINT "6";: NORMAL
210 PRINT TAB (5);"Print the Edge List"  
220 VTAB (17); PRINT TAB (3);: INVERSE: PRINT "7";: NORMAL
230 PRINT TAB (5);"Print the Object's Corner Coordinates"  
240 VTAB (18); PRINT TAB (3);: INVERSE: PRINT "8";: NORMAL
250 PRINT TAB (5);"Run Machine Vision Process"  
255 VTAB (19); PRINT TAB (3);: INVERSE: PRINT "9";: NORMAL
260 PRINT TAB (5);"Return to Main Menu"  
265 VTAB (22); HTAB (2): PRINT "PLEASE ENTER THE NUMBER OF YOUR"  
270 VTAB (23); HTAB (2): INPUT "SELECTION AND PRESS RETURN"; A$  
280 IF A$ = "1" THEN GOTO 400  
290 IF A$ = "2" THEN GOTO 500  
300 IF A$ = "3" THEN GOTO 700  
310 IF A$ = "4" THEN GOTO 800  
320 IF A$ = "5" THEN GOTO 900  
330 IF A$ = "6" THEN GOTO 1000  
340 IF A$ = "7" THEN GOTO 1400  
350 IF VAL (A$) = 8 OR VAL (A$) = 9 THEN HOME: VTAB (9); HTAB (5); INVERSE: PRINT "PLEASE WAIT": NORMAL  
360 IF A$ = "8" THEN PRINT D$;"LOAD PARAMETERS,D1": PRINT D$ + "RUN VISNPROG,D1"  
370 IF A$ = "9" THEN PRINT D$ + "RUN SELNPROG,D1"  
380 HOME: PRINT CHR$ (7)  
390 VTAB (10); HTAB (2): PRINT "YOU MUST ENTER A NUMBER BETWEEN"  
395 VTAB (13); HTAB (13): PRINT "PLEASE TRY AGAIN": FOR I = 1 TO 1500: NEXT I: GOTO 80  
400 REM PRINT GRAY SCALE MATRIX  
410 GOSUB 2000  
420 IF PEEK (36911) = 255 THEN GOTO 80  
430 PRINT D$;"PR#1": REM ACTIVATE PRITNER
E14

440 PRINT TAB(15);"GRAY-SCALE MATRIX FOR THE ENTIRE VIEWING AREA"
450 GOSUB 2500: REM CALL SUBROUTINE TO PRINT MATRIX
460 PRINT D$;"PRM"; REM DEACTIVATE PRINTER
470 GOTO 80
500 POKE 36911,2: REM SET PRINT FLAG FOR ROBERTS MATRIX FOR VIEWING AREA
510 GOSUB 2000: REM CALL SUBROUTINE TO CHECK PRINTER
520 IF PEEK(36911) = 255 THEN GOTO 80: REM IF PRINT FLAG=255 CANCEL JOB
530 PRINT D$;"PRM"; REM ACTIVATE PRINTER
540 PRINT TAB(10);"MATRIX OF ROBERTS GRADIENT VALUES FOR THE ENTIRE VIEWING AREA"
550 GOSUB 2500: REM CALL SUBROUTINE TO PRINT MATRIX
560 PRINT D$;"PRM"; REM DEACTIVATE PRINTER
570 GOTO 80
600 PRINT "NOT IMPLEMENTED YET"
610 FOR I = 1 TO 5000
620 NEXT I
630 GOTO 80
700 POKE 36911,1: REM SET PRINT FLAG FOR GRAY SCALE FOR WINDOW (00000001)
710 GOSUB 2000: REM CALL SUBROUTINE TO CHECK PRINTER
720 IF PEEK(36911) = 255 THEN GOTO 80: REM IF PRINT FLAG=255 CANCEL
730 PRINT D$;"PRM"; REM ACTIVATE PRINTER
740 PRINT TAB(25);"MATRIX OF GRAY SCALE VALUES FOR THE WINDOW"
750 GOSUB 2500: REM CALL SUBROUTINE TO PRINT MATRIX
760 PRINT D$;"PRM"; REM DEACTIVATE PRINTER
770 GOTO 80
800 POKE 36911,3: REM SET PRINT FLAG FOR ROBERTS MATRIX (00000011)
810 GOSUB 2000: REM CALL SUBROUTINE TO CHECK PRINTER
820 IF PEEK(36911) = 255 THEN GOTO 80: REM IF PRINT FLAG=255 CANCEL
830 PRINT D$;"PRM"; REM ACTIVATE PRINTER
840 PRINT TAB(15);"MATRIX OF ROBERTS GRADIENT VALUES FOR THE WINDOW"
850 GOSUB 2500: REM CALL SUBROUTINE TO PRINT MATRIX
860 PRINT D$;"PRM"; REM DEACTIVATE PRINTER
870 GOTO 80
900 POKE 36911,3: REM SET PRINT FLAG TO TRANSFER ROBERTS MATRIX FOR WINDOW
902 TREDGX = PEEK(36904)
904 HOME
906 VTAB(8): PRINT TAB(3);"THE DEFAULT THRESHOLD VALUE IS ";T
908 VTAB(10): PRINT TAB(4);"PLEASE ENTER A NEW VALUE"
910 VTAB(11): PRINT TAB(4);"OR PRESS <RETURN> TO KEEP THE DEFAULT VALUE"
912 INPUT A$
914 IF LEN (A$) = 0 THEN GOTO 918
916 TREDG% = VAL (A$)
918 GOSUB 2000: REM CHECK PRINTER
920 IF PEEK (36911) = 255 THEN GOTO 80
930 PRINT CHR$ (4); "PR#1"
940 PRINT TAB(24); "CONNECTIVITY DIAGRAM (THRESHOLD=": ; TREDG%; "")
950 GOSUB 3000: REM CALL SUBROUTINE TO PRINT MATRIX
960 PRINT CHR$ (4); "PR#0"
965 D$ = CHR$ (4)
970 GOTO 80
1000 HOME
1005 CALL 37072
1010 GOSUB 2000: REM CHECK PRINTER
1020 PRINT D$; "PR#1"
1040 J = 0
1050 NLIN = PEEK (33792)
1060 FOR I = 1 TO 6
1070 PRINT : REM SCROLL DOWN 6 LINES
1080 NEXT I
1085 IF J > 0 THEN GOTO 1095
1090 PRINT TAB(35); "EDGE LIST": GOTO 1100
1095 PRINT TAB(29); "EDGE LIST (CONTINUED)"
1100 PRINT
1110 PRINT TAB(17); "X COORDINATE"; TAB(5); "Y COORDINATE"; TAB(5); "CORNER CODE"
1120 IF NLIN > 51 THEN NLIN = NLIN - 51: I = 0: LPR = 51: GOTO 1140
1130 IF NLIN < 51 THEN I = 51 - NLIN: LPR = NLIN: NLIN = 0
1140 FOR K = 1 TO LPR: REM PRINT UNTIL YOU RUN OUT OF ROOM OR DATA
1150 PRINT TAB(23);
1160 IF PEEK (32768 + J) < 10 THEN PRINT " "; PEEK (32768 + J)
1170 IF PEEK (32768 + J) ≥ 10 THEN PRINT PEEK (32768 + J);
1180 PRINT TAB(14);
1190 IF PEEK (33024 + J) < 10 THEN PRINT " "; PEEK (33024 + J)
1200 IF PEEK (33024 + J) ≥ 10 THEN PRINT PEEK (33024 + J);
1210 PRINT TAB(14); PEEK (33280 + J);
1211 PRINT TAB(14);
1212 IF PEEK (33536 + J) < 10 THEN PRINT " "; PEEK (33536 + J)
1214 IF PEEK (33536 + J) ≥ 10 THEN PRINT PEEK (33536 + J)
1220 J = J + 1
1230 NEXT K
1240 FOR K = 1 TO I + 6
1250 PRINT
1260 NEXT K
1270 IF NLIN < > 0 THEN GOTO 1060: REM PRINT NEXT PAGE
1280 PRINT D$;"PR#0": REM PRINT NEXT PAGE
1290 GOTO 80: REM RETURN TO MAIN PROGRAM
1400 REM PRINT CORNER COORD.
1410 GOSUB 2000: REM CHECK PRINTER
1420 IF PEEK (36911) = 255 THEN GOTO 80: REM CANCEL PRINT
1430 PRINT D$;"PR#1": REM ACTIVATE PRINTER
1440 FOR I = 1 TO 6
1450 PRINT
1460 NEXT I
1465 CALL 37072
1470 NLIN% = PEEK (38335)
1480 PRINT SPC(10);"X COORD.": SPC(10);"Y COORD."
1490 FOR I = 1 TO NLIN%
1500 XC(I) = PEEK (38338 + (I - 1) * 6) / 100 + PEEK (38337 + (I - 1) * 6)
1510 IF PEEK (38336 + (I - 1) * 6) = 0 THEN XC(I) = - XC(I)
1520 YC(I) = PEEK (38341 + (I - 1) * 6) / 100 + PEEK (38340 + (I - 1) * 6)
1530 IF PEEK (38339 + (I - 1) * 6) = 0 THEN YC(I) = - YC(I)
1540 IF XC(I) = 1 THEN PRINT SPC(12);XC(I);: GOTO 1580
1550 IF XC(I) = 0 THEN PRINT SPC(12);"0": XC(I);: GOTO 1580
1560 IF XC(I) = 0 THEN PRINT SPC(12);"0.00": GOTO 1580
1570 PRINT SPC(11);XC(I);
1580 IF YC(I) = 1 THEN PRINT SPC(13);YC(I);: GOTO 1620
1590 IF YC(I) = 0 THEN PRINT SPC(13);"0": YC(I);: GOTO 1620
1600 IF YC(I) = 0 THEN PRINT SPC(13);"0.00": GOTO 1620
1610 PRINT SPC(12);YC(I)
1620 NEXT I
1630 FOR I = 1 TO (59 - NLIN%)
1640 PRINT
1650 NEXT I
1660 PRINT D$;"PR#0": REM DEACTIVATE PRINTER
1670 GOTO 80
2000 HOME
2010 VTAB 12: PRINT TAB(6);"BE SURE THE PRINTER IS ON AND"
2015 VTAB 13: PRINT TAB(7);"THE PRINT HEAD IS ALIGNED"
2020 VTAB 14: PRINT TAB(7);"WITH THE TOP OF THE PAPER."
2030 VTAB 16: PRINT TAB(8);"PRESS RETURN TO CONTINUE"
2035 VTAB (17); INPUT " (PRESS ANY OTHER KEY TO CANCEL)"; CNT$;
2040 IF LEN (CNT$) = 0 THEN RETURN
2050 POKE 36911, 255
2060 RETURN
2065 CALL 36912
2100 REM MOVE MATRIX FROM BANK SWITCHED MEMORY TO LOW MEMORY
2105 REM FIND NO. OF COLUMNS EXCEPTED
2110 REM FIND NO. OF ROWS AND COLUMNS FOR GRAY SCALE VALUES
2115 NCOL% = PEEK (36872)
2120 NROW% = PEEK (36871)
2125 GOTO 2550
2528 REM THIS IS ONE LESS ROW AND COLUMN OF ROBERTS VALUES THAN GRAY SCALE VALUES
2530 NCOL% = PEEK (36872) - 1
2540 NROW% = PEEK (36871) - 1
2550 J = 0
2560 REM BEGIN DUMM LOOP FOR SETS OF COLUMNS
2570 REM CALCULATE COLUMNS TO PRINT THIS CYCLE
2575 PRINT
2580 FCOL% = 1 + J * 29
2590 LCOL% = 29 + J * 29
2600 REM SCROLL TO BOTTOM OF PAGE
2610 IF LCOL% > NCOL% THEN LCOL% = NCOL%
2620 REM BEGIN COUNTING ROWS
2630 FOR I = 1 TO NROW%
2640 REM INCREMENT COLUMNS
2650 FOR K = FCOL% TO LCOL%
2660 REM CALCULATE CELL TO BE PRINTED
2670 LOC = (I - 1) * NCOL% + (K - 1) + 32768
2680 IF PEEK (LOC) < 10 THEN PRINT PEEK (LOC); ;
2690 IF PEEK (LOC) > 9 AND PEEK (LOC) < 100 THEN PRINT PEEK (LOC); ;
2700 IF PEEK (LOC) > 99 THEN PRINT PEEK (LOC);
2710 NEXT K
2720 PRINT
2730 NEXT I
2740 REM SCROLL TO BOTTOM OF PAGE
2750 FOR I = NROW% + 2 TO 66
2760 PRINT
2770 NEXT I
2780 J = J + 1
2790 IF LCOL% < NCOL% THEN GOTO 2560
2800 RETURN
3000 CALL 36912: REM CALL PROGRAM AT $9030
3010 REM GET NO. COL. ACCEPTED, NO. ROWS ACCEPTED AND THRESHOLD
3020 NCOL% = PEEK (36872) - 1
3030 NROW% = PEEK (36871) - 1
3050 J = 0
3060 REM CALCULATE COL. TO PRINT THIS CYCLE
3070 FCOL% = 1 + J * 42
3075 PRINT
3080 LCOL% = 42 + J * 42
3090 IF LCOL% > NCOL% THEN LCOL% = NCOL%
3100 REM BEGIN PRINTING ROWS
3110 FOR I = 1 TO NROW%
3120 REM INCREMENT COLUMNS
3130 FOR K = FCOL% TO LCOL%
3140 REM CALCULATE CELL TO BE PRINTED
3150 LOC = (I - 1) * NCOL% + (K - 1) + 32768
3160 IF PEEK (LOC) < TRED% THEN PRINT * ;
3170 IF PEEK (LOC) > = TRED% THEN PRINT * 0*;
3180 NEXT K
3190 PRINT
3200 NEXT I
3210 REM SCROLL TO BOTTOM OF PAGE
3220 FOR I = NRW + 2 TO 66
3230 PRINT
3240 NEXT I
3250 J = J + 1
3260 IF LCOL < NCOL THEN GOTO 3060
3270 RETURN
10 REM PROGRAM TO DISPLAY & DELETE SHAPES
20 DIM NEDX(20), CCR(20, 8, 2), ID$(20), PCK(20)
30 D$ = CHR$ (4)
40 HOME
50 VTAB (9); HTAB (15); INVERSE: PRINT "PLEASE WAIT"; NORMAL
60 VTAB (12); HTAB (5); PRINT "The Shape Data are Being Loaded."
70 REM READ SHAPE DATA
80 RW% = 0
90 GOSUB 3000
100 HOME
110 PRINT "SELECT ONE OF THE FOLLOWING TO DISPLAY:" 
120 PRINT
130 IF I = 1 TO NSHP% 
140 IF I < 10 THEN PRINT SPC(4); I; SPC(2); 
150 IF I > 10 THEN PRINT SPC(3); I; SPC(2); 
160 PRINT ID$(I)
170 NEXT I
180 VTAB (23); PRINT "ENTER A NUMBER BETWEEN 1 AND "; NSHP%; " OR"
190 PRINT "ENTER "; INVERSE; PRINT "Q"; NORMAL; PRINT " TO RETURN TO MAIN MENU";
200 GET A$
205 HOME
210 IF A$ = "Q" THEN GOTO 1000
220 IF VAL (A$) < 1 OR VAL (A$) > NSHP% THEN HOME; PRINT CHR$ (7); PRINT "YOU MUST ENTER A NUMBER BETWEEN 1 AND 9"
230 IF VAL (A$) < 1 OR VAL (A$) > NSHP% THEN VTAB (13); HTAB (13); PRINT "PLEASE TRY AGAIN"; FOR I = 1 TO 1500: NEXT I: GOTO 80
240 A = VAL (A$)
250 PRINT ID$(A); " HAS "; NEDX(A); " CORNERS"
260 PRINT "WHOSE COORDINATES ARE:"
270 PRINT
280 VTAB (6); HTAB (5); PRINT "CORNERS"; SPC(6); "X"; SPC(8); "Y"
290 PRINT
300 FOR I = 1 TO NEDX(A)
310 HTAB (7); PRINT I;
320 IF CCR(A, I, 1) > = 1 THEN PRINT SPC(8); CCR(A, I, 1);: GOTO 360
330 IF CCR(A, I, 1) > 0 THEN PRINT SPC(8); "0"; CCR(A, I, 1);: GOTO 360
340 IF CCR(A, I, 1) = 0 THEN PRINT SPC(8); "0.000": GOTO 360
350 PRINT SPC(7); CCR(A, I, 1);
360 IF CCR(A, I, 2) >= 1 THEN PRINT SPC(4); CD(A, I, 2);: GOTO 400
370 IF CCR(A, I, 2) > 0 THEN PRINT SPC(4); "0"; CCR(A, I, 2);: GOTO 400
380 IF CCR(A, I, 2) = 0 THEN PRINT SPC(4); "0.000": GOTO 400
390 PRINT SPC(3); CCR(A, I, 2)
400 NEXT I
410 VTAB (19): INPUT "SHOULD THIS OBJECT BE DELETED? (Y/N)";A$
420 IF A$ < > "Y" THEN GOTO 100
430 INPUT "ARE YOU SURE? (Y/N)";A$
440 IF A$ < > "Y" THEN GOTO 100
450 FOR I = A + 1 TO NSHP%
460 ID$(I - 1) = ID$(I)
470 NED%(I - 1) = NED%(I)
480 FOR J = 1 TO NED%(I)
490 CCR(I - 1, J, 1) = CCR(I, J, 1)
500 CCR(I - 1, J, 2) = CCR(I, J, 2)
510 NEXT J
520 PCK(I - 1) = PCK(I)
530 NEXT I
540 NSHP% = NSHP% - 1
550 GOTO 100
1000 REM STORE NEW SHAPES FILE
1010 Rlw = 1
1020 GOSUB 3000
1030 PRINT D$ + "RUN SELNPROG,D1"
1040 END
3000 REM
3010 REM READ (RW=0) OR WRITE (RW=1) SHAPE DATA INTO MEMORY
3020 REM
3030 PRINT CHR$(4); "OPEN SHAPES,D1"
3040 IF Rlw = 1 THEN PRINT D$; "WRITE SHAPES"
3050 IF Rlw < > 1 THEN PRINT D$; "READ SHAPES"
3060 REM READ/WRITE NO. OF SHAPES IN FILE
3070 IF Rlw = 1 THEN PRINT NSHP%
3080 IF Rlw < > 1 THEN INPUT NSHP%
3090 REM SHAPE ID, NUMBER OF CORNERS AND CORNER COORDINATES
3100 FOR I = 1 TO NSHP%
3110 IF Rlw = 1 THEN PRINT ID$(I); PRINT NED%(I)
3120 IF Rlw < > 1 THEN INPUT ID$(I); INPUT NED%(I)
3130 FOR J = 1 TO NED%(I)
3150 IF Rlw = 1 THEN PRINT CCR(I, J, 1), CCR(I, J, 2)
3160 IF Rlw < > 1 THEN INPUT CCR(I, J, 1), CCR(I, J, 2)
3180 NEXT J
3184 IF Rlw = 1 THEN PRINT PCK(I)
3188 IF Rlw < > 1 THEN INPUT PCK(I)
3190 NEXT I
3195 PRINT D$; "CLOSE SHAPES"
3200 RETURN
10 REM PROGRAM SETPARAM MM 12/26/88
20 REM UPDATED PROGRAM TO CHANGE PARAMETERS
30 PRINT CHR$ (4); "LOAD PARAMETERS,D1"
40 HOME
50 VTAB (1): PRINT "SET PARAMETER MODULE Version 2"
60 VTAB (6): HTAB (2): PRINT "SELECT ONE OF THE FOLLOWING OPTIONS:
70 VTAB (9): HTAB (5): INVERSE: PRINT "1": NORMAL
80 PRINT "DISPLAY AND/OR CHANGE PARAMETERS"
90 VTAB (11): HTAB (5): INVERSE: PRINT "2": NORMAL
100 PRINT "PRINT A HARDCOPY OUTPUT OF THE PARAMETERS"
105 VTAB (12): HTAB (8): PRINT "PARAMETERS"
110 VTAB (14): HTAB (5): INVERSE: PRINT "3": NORMAL
120 PRINT "RETURN TO THE MAIN MENU"
130 VTAB (17): HTAB (2): PRINT "PLEASE ENTER THE NUMBER OF YOUR SELECTION AND PRESS RETURN"
140 VTAB (18): HTAB (2): INPUT "SELECTION AND PRESS RETURN "; A$
150 IF A$ = "1" THEN GOTO 250
160 IF A$ = "2" THEN GOSUB 4000: GOTO 40
170 IF A$ = "3" THEN HOME: VTAB (9): HTAB (15): INVERSE: PRINT "PLEASE WAIT"
175 IF A$ = "3" THEN PRINT CHR$ (4); "RUN SELECTPROG,D1"
176 IF A$ = "3" THEN END
180 PRINT CHR$ (7)
190 HOME
200 VTAB (10): HTAB (2): PRINT "YOU MUST ENTER A NUMBER BETWEEN 1 AND 3"
210 VTAB (13): HTAB (13): PRINT "PLEASE TRY AGAIN"
220 FOR I = 1 TO 1500: NEXT I
230 GOTO 40
250 HOME
270 VTAB (3): PRINT "FIRST ROW OF THE IMAGE USED BY VOS"
280 PRINT : PRINT : PRINT
290 PRINT "THIS VALUE IS THE FIRST ROW IN THE IMAGE WHICH IS USED BY THE MACHINE"
300 PRINT "VISION SYSTEM DURING THE INITIAL VIEW"
310 PRINT "OF THE ENTIRE VIEWING AREA"
320 DEVA = 8: PO = 36896
340 VA% = PEEK (PO)
350 PRINT : PRINT
360 PRINT "DEFAULT VALUE = "; DEVA
370 PRINT "CURRENT VALUE = "; VA%
380 PRINT : PRINT
390 PRINT "ENTER A NEW VALUE ("; INVERSE : PRINT "RETURN"; : NORMAL :
400 INPUT "=NO CHANGE")"; A$
410 IF LEN (A$) = 0 THEN GOTO 450
420 VA% = VAL (A$)
425 LW% = 1: UP% = 232
430 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 250
440 POKE PO, VA%
HOME
VTAB (3): PRINT "FIRST COLUMN OF THE IMAGE USED BY VOS"
PRINT : PRINT : PRINT
PRINT "This value is the first column in the"
PRINT "image which is used by the machine"
PRINT "vision system during the initial view"
PRINT "of the entire viewing area."
DEF% = 8:PO = 36897
VAX = PEEK (PO)
PRINT : PRINT
PRINT "Default Value = ";DEF%
PRINT "Current Value = ";VAX
PRINT : PRINT
PRINT "ENTER A NEW VALUE ("; INVERSE : PRINT "RETURN"; : NORMAL :
INPUT "= no change") ";A$
IF LEN (A$) = 0 THEN GOTO 650
VAX = VAL (A$)
LW% = 1:UP% = 232
IF VAX < LW% OR VAX > UP% THEN GOSUB 5000: GOTO 450
POKE PO, VAX
HOME
VTAB (3): PRINT "LAST ROW OF THE IMAGE USED BY VOS"
PRINT : PRINT : PRINT
PRINT "This value is the last row in the"
PRINT "image which is used by the machine"
PRINT "vision system during the initial view"
PRINT "of the entire viewing area."
DEF% = 232:PO = 36898
VAX = PEEK (PO)
PRINT : PRINT
PRINT "Default Value = ";DEF%
PRINT "Current Value = ";VAX
PRINT : PRINT
PRINT "ENTER A NEW VALUE ("; INVERSE : PRINT "RETURN"; : NORMAL :
INPUT "= no change") ";A$
IF LEN (A$) = 0 THEN GOTO 850
VAX = VAL (A$)
LW% = PEEK (36896) + 1:UP% = 244
IF VAX < LW% OR VAX > UP% THEN GOSUB 5000: GOTO 650
POKE PO, VAX
HOME
VTAB (3): PRINT "LAST COLUMN OF THE IMAGE USED BY VOS"
PRINT : PRINT : PRINT
PRINT "This value is the last column in the"
PRINT "image which is used by the machine"
PRINT "vision system during the initial view"
PRINT "of the entire viewing area."
DEF% = 232:PO = 36899
VAX = PEEK (PO)
PRINT: PRINT
900 PRINT "Default Value = ": print(";DEV"
910 PRINT "Current Value = ": print(";VAX"
920 PRINT: PRINT
930 PRINT "ENTER A NEW VALUE "; INVERSE : PRINT "RETURN"; : NORMAL :
1000 INPUT "=no change)"; A$
1010 IF LEN (A$) = 0 THEN GOTO 1050
1020 VAX = VAL (A$)
1025 LW% = PEEK (36897) + 1:UP% = 244
1030 IF VAX < LW% OR VAX > UP% THEN GOSUB 5000: GOTO 850
1040 POKE PO, VAX
1050 HOME
1070 VTAB (3): PRINT "HORIZONTAL SKIP MODE USED BY VOS"
1080 PRINT: PRINT
1090 PRINT "This value is the number of pixels"
1100 PRINT "that are skipped between each row of"
1110 PRINT "pixels that are used by the machine"
1120 PRINT "vision system during the initial view"
1130 PRINT "of the entire viewing area."
1140 DEV% = 3: PO = 36900
1145 VAX = PEEK (PO)
1150 PRINT: PRINT
1160 PRINT "Default Value = ": print(";DEV"
1170 PRINT "Current Value = ": print(";VAX"
1180 PRINT: PRINT
1190 PRINT "ENTER A NEW VALUE "; INVERSE : PRINT "RETURN"; : NORMAL :
1200 INPUT "=no change)"; A$
1210 IF LEN (A$) = 0 THEN GOTO 1250
1220 VAX = VAL (A$)
1225 LW% = 1: UP% = 10
1230 IF VAX < LW% OR VAX > UP% THEN GOSUB 5000: GOTO 1050
1240 POKE PO, VAX
1250 HOME
1260 VTAB (3): PRINT "VERTICAL SKIP MODE USED BY VOS"
1270 PRINT: PRINT
1280 PRINT "This value is the number of pixels"
1290 PRINT "that are skipped between each column"
1300 PRINT "of pixels that are used by the machine"
1310 PRINT "vision system during the initial view"
1320 PRINT "of the entire viewing area."
1330 DEV% = 3: PO = 36901
1340 VAX = PEEK (PO)
1350 PRINT: PRINT
1360 PRINT "Default Value = ": print(";DEV"
1370 PRINT "Current Value = ": print(";VAX"
1380 PRINT: PRINT
1390 PRINT "ENTER A NEW VALUE "; INVERSE : PRINT "RETURN"; : NORMAL :
1400 INPUT "=no change)"; A$
1410 IF LEN (A$) = 0 THEN GOTO 1450
1420 VA% = VAL (A$)
1425 LW% = INT (((PEEK (36898) - PEEK (36896)) * (PEEK (36899) - PEEK (36897))) / (PEEK (36900) + 1) / 4000) + .5)
1428 UP% = 10
1430 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 1250
1440 POKE PO, VA%
1450 HOME
1470 VTAB (3): PRINT "NUMBER OF FRAMES GATHERED BY VOS"
1480 PRINT: PRINT: PRINT
1490 PRINT "The number of frames are the total"
1500 PRINT "number of image sequences allowed"
1510 PRINT "in the data gathering process"
1520 DEV% = 0; PO = 36902
1530 VA% = PEEK (PO)
1540 PRINT: PRINT
1550 PRINT "Default Value = "; DEV%
1560 PRINT "Current Value = "; VA%
1570 PRINT : PRINT
1580 PRINT "ENTER A NEW VALUE ("; INVERSE ; PRINT "RETURN"); NORMAL ;
1590 INPUT "=no change)"; A$
1600 IF LEN (A$) = 0 THEN GOTO 1650
1610 VA% = VAL (A$)
1620 LW% = 0: UP% = 5
1630 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 1450
1640 POKE PO, VA%
1650 HOME
1670 VTAB (3): PRINT "NUMBER OF IMAGES ACCUMULATED BY VOS"
1680 PRINT: PRINT: PRINT
1690 PRINT "The value represents the number of"
1700 PRINT "frames of data that are added"
1710 PRINT "to the original frame of data"
1720 PRINT "which is gathered by VOS."
1730 DEV% = 0; PO = 36903
1740 VA% = PEEK (PO)
1750 PRINT: PRINT
1760 PRINT "Default Value = "; DEV%
1770 PRINT "Current Value = "; VA%
1780 PRINT: PRINT
1790 PRINT "ENTER A NEW VALUE ("; INVERSE ; PRINT "RETURN"); NORMAL ;
1800 INPUT "=no change)"; A$
1810 IF LEN (A$) = 0 THEN GOTO 1850
1820 VA% = VAL (A$)
1830 LW% = 0: UP% = 5
1835 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 1650
1840 POKE PO, VA%
1850 HOME
1870 VTAB (3): PRINT "EDGE THRESHOLD FOR THE ENTIRE IMAGE"
1880 PRINT: PRINT
1890 PRINT "If a calculated gradient value for a"
1900 PRINT "given pixel in the first image obtained"
1910 PRINT "by the machine vision system is above"
1920 PRINT "this threshold value than the pixel"
1930 PRINT "position will be considered a point"
1940 PRINT "along the edge of an object."
1950 DEV% = 25:PO = 36904
1960 PRINT : PRINT
1970 PRINT "Default Value = ";DEV%
1980 PRINT "Current Value = ";DEV%
1990 PRINT : PRINT
1995 PRINT "ENTER A NEW VALUE (*.; INVERSE PRINT "RETURN";: NORMAL"

2000 INPUT "=no change)";A$
2010 IF LEN (A$) = 0 THEN GOTO 2050
2020 VA% = VAL (A$)
2030 LW% = 1:UP% = 100
2035 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 1850
2040 POKE PO,VA%
2050 HOME
2060 VTAB (3): PRINT "EDGE THRESHOLD FOR THE WINDOW"
2070 PRINT : PRINT
2080 PRINT "If a calculated gradient value for a"
2090 PRINT "given pixel in the window obtained by"
2100 PRINT "the machine vision system is above this"
2110 PRINT "threshold value then the position may"
2120 PRINT "be considered a point along the edge"
2130 PRINT "of an object."
2140 DEV% = 55:PO = 36905
2145 VA% = PEEK (PO)
2150 PRINT : PRINT
2155 PRINT "Default Value = ";DEV%
2160 PRINT "Current Value = ";VA%
2165 PRINT : PRINT
2170 PRINT "ENTER A NEW VALUE (*.; INVERSE PRINT "RETURN";: NORMAL"

2200 INPUT "=no change)";A$
2210 IF LEN (A$) = 0 THEN GOTO 2250
2220 VA% = VAL (A$)
2225 LW% = 1:UP% = 256
2230 IF VA% < LW% OR VA% > UP% THEN GOSUB 5000: GOTO 2050
2240 POKE PO,VA%
2250 HOME
2260 VTAB (3): PRINT "TOLERANCE FOR OBJECT IDENTIFICATION"
2270 PRINT : PRINT
2280 PRINT "This tolerance is the allowed deviation"
2290 PRINT "of the object being viewed to the"
2300 PRINT "objects stored in the computer’s"
2310 PRINT "memory. This tolerance is in 1/100 of"
2320 PRINT "an inch (ex. 10 = .1 inch).
2330 DEV% = 10:PO = 36906
2340 VA% = PEEK (PO)
2350 PRINT : PRINT
2360 PRINT "Default Value =" ; DEV\%
2370 PRINT "Current Value =" ; VA\%
2380 PRINT : PRINT
2390 PRINT "ENTER A NEW VALUE (" ; INVERSE : PRINT "RETURN" ; ; NORMAL

2400 INPUT "=no change)" ; A$
2410 IF LEN (A$) = 0 THEN GOTO 2450
2420 VA\% = VAL (A$)
2425 LW\% = 1 ; UP\% = 100
2430 IF VA\% < LW\% OR VA\% > UP\% THEN GOSUB 5000 ; GOTO 2250
2440 POKE PO, VA\%
2450 HCtFE
2460 PRINT "NUMBER OF POINTS FOR CORNER DETECTION"
2465 PRINT : PRINT
2470 PRINT "This value is the number of points"
2480 PRINT "used by the corner detection algorithm"
2490 PRINT "to determine the position of the"
2500 PRINT "corners along the object's edge."
2510 DBfh = 4 ; PO = 36907
2520 VA\% = PEEK (PO)
2530 PRINT : PRINT
2540 PRINT "Default Value =" ; DEV\%
2550 PRINT "Current Value =" ; VA\%
2560 PRINT : PRINT
2570 PRINT "ENTER A NEW VALUE (" ; INVERSE : PRINT "RETURN" ; ; NORMAL

2580 INPUT "=no change)" ; A$
2590 IF LEN (A$) = 0 THEN GOTO 2650
2600 VA\% = VAL (A$)
2610 LW\% = 1 ; UP\% = 7
2620 IF VA\% < LW\% OR VA\% > UP\% THEN GOSUB 5000 ; GOTO 2450
2630 POKE PO, VA\%
2650 HOME
2660 PRINT "LOWER THRESHOLD FOR CORNER DETECTION"
2670 PRINT : PRINT
2680 PRINT "This value is the largest value that"
2690 PRINT "can be computed by the corner detection"
2700 PRINT "algorithm which can be considered a"
2710 PRINT "valley between two peeks that represent"
2720 PRINT "two seperate corners."
2730 DEV\% = 8 ; PO = 36908
2740 VA\% = PEEK (PO)
2750 PRINT : PRINT
2760 PRINT "Default Value =" ; DEV\%
2770 PRINT "Current Value =" ; VA\%
2780 PRINT : PRINT
2790 PRINT "ENTER A NEW VALUE (" ; INVERSE : PRINT "RETURN" ; ; NORMAL

2800 INPUT "=no change)" ; A$
2810 IF LEN (A$) = 0 THEN GOTO 2850
2820 VA% = VAL (A$)
2825 LW% = 4: UP% = 20
2830 IF VA% < (LW% OR VA%) UP% THEN GOSUB 5000: GOTO 2650
2840 POKE PO, VA%
2850 HOME
2860 PRINT "UPPER THRESHOLD FOR CORNER DETECTION"
2870 PRINT : PRINT
2880 PRINT "Only points in the edge list whose"
2890 PRINT "calculated corner strength is greater"
2900 PRINT "than or equal to this threshold may"
2910 PRINT "be considered corner points."
2920 DEV% = 13; PO = 36909
2930 VA% = PEEK (PO)
2940 PRINT: PRINT
2950 PRINT "Default Value = "; DEV%
2960 PRINT "Current Value = "; VA%
2970 PRINT : PRINT
2980 PRINT "ENTER A NEW VALUE (\":; INVERSE; PRINT "RETURN")": NORMAL
2990 INPUT ";= no change")":A$
3000 IF LEN (A$) = 0 THEN GOTO 3050
3010 VA% = VAL (A$)
3020 LW% = 4: UP% = 20
3030 IF VA% < (LW% OR VA%) UP% THEN GOSUB 5000: GOTO 2850
3040 POKE PO, VA%
3050 HOME
3060 VTAB (6): HTAB (2): PRINT "SELECT ONE OF THE FOLLOWING OPTIONS:"  
3070 VTAB (10): HTAB (5): INVERSE : PRINT "1": NORMAL
3080 PRINT "Save the New Parameters"
3090 VTAB (12): HTAB (5): INVERSE : PRINT "2": NORMAL
3100 PRINT "Print a Hardcopy Output of the"
3110 VTAB (13): HTAB (8): PRINT "Parameters"
3120 VTAB (15): HTAB (5): INVERSE : PRINT "3": NORMAL
3130 PRINT "Return to Main Menu"
3140 VTAB (19): HTAB (2): PRINT "PLEASE ENTER THE NUMBER OR YOUR"
3150 VTAB (20): HTAB (2): INPUT "SELECTION AND PRESS RETURN "; A$
3160 IF A$ = "1" THEN GOTO 3300
3170 IF A$ = "2" THEN GOSUB 4000: GOTO 3050
3180 IF A$ = "3" THEN HOME : VTAB (9): HTAB (15): INVERSE : PRINT  
"PLEASE WAIT": NORMAL
3190 IF A$ = "3" THEN PRINT CHR$ (4) ; "RUN SELNPROG,D1"
3200 IF A$ = "3" THEN END
3210 PRINT CHR$ (7)
3220 HOME
3230 VTAB (10): HTAB (2): PRINT "YOU MUST ENTER AN NUMBER BETWEEN  
1 AND 3"
3240 VTAB (13): HTAB (13): PRINT "PLEASE TRY AGAIN"
3250 FOR I = 1 TO 1500: NEXT I
3260 GOTO 3050
3290 REM STORE VALUES
3300 PRINT CHR$ (4) + "BSAVE PARAMETERS,DI,AS9020,L$000E"
3310 GOTO 40
4000 REM PRINT PARAMETERS LIST
4010 PRINT CHR$ (4); "PR#1"
4020 PRINT : PRINT : PRINT
4030 PRINT TAB(34); "PARAMETER LIST"
4040 PRINT : PRINT
4050 PRINT "FIRST ROW OF THE IMAGE USED BY VOS = "; PEEK (36896)
4060 PRINT : PRINT "LAST ROW OF THE IMAGE USED BY VOS = "; PEEK (36898)
4070 PRINT : PRINT "LAST COLUMN OF THE IMAGE USED BY VOS = "; PEEK (36899)
4080 PRINT : PRINT "HORIZONTAL SKIP MODE USED BY VOS = "; PEEK (36900)
4090 PRINT : PRINT "VERTICAL SKIP MODE USED BY VOS = "; PEEK (36901)
4100 PRINT : PRINT "NUMBER OF FRAMES GATHERED BY VOS = "; PEEK (36902)
4110 PRINT : PRINT "NUMBER OF IMAGES ACCUMULATED BY VOS = "; PEEK (36903)
4120 PRINT : PRINT "EDGE THRESHOLD FOR THE ENTIRE IMAGE = "; PEEK (36904)
4130 PRINT : PRINT "EDGE THRESHOLD FOR THE WINDOW = "; PEEK (36905)
4140 PRINT : PRINT "TOLERANCE FOR OBJECT IDENTIFICATION = "; PEEK (36906)
4150 PRINT : PRINT "NUMBER OF POINTS FOR CORNER DETECTION = "; PEEK (36907)
4160 PRINT : PRINT "LOWER THRESHOLD FOR CORNER DETECTION = "; PEEK (36908)
4170 PRINT : PRINT "UPPER THRESHOLD FOR CORNER DETECTION = "; PEEK (36909)
4175 FOR I = 1 TO 34: PRINT : NEXT I
4180 PRINT CHR$ (4); "PR#0"
4190 RETURN
5000 HOME
5010 VTAB (10): PRINT "YOU MUST ENTER A VALUE BETWEEN "; L$W%; " AND "; UP%
5020 VTAB (13): HTAB (14): PRINT "PLEASE TRY AGAIN"
5030 FOR I = 1 TO 1500: NEXT I
5040 RETURN
10 REM PROGRAM DRIVER TO MEMORIZE OBJECT
20 DIM SX(8), SY(8)
30 DIM NED%(20), CCR(20, 8, 2), ID%(20), XC(8)
40 DIM YC(8), CD(8), AC(8), BC(8), RK%(8), PCK(20)
50 D$ = CHR$(4); PI = 3.14159
100 HOME
110 VTAB(8); PRINT "In order for the machine vision system"
120 PRINT "to get an accurate representation of"
130 PRINT "the object to be memorized, a total of"
140 PRINT "5 views of the object will be analyzed"
150 PRINT "and an average representation will be"
160 PRINT "stored. Between runs you will be asked"
170 PRINT "to give the object different"
180 PRINT "orientations in order to eliminate any"
190 PRINT "rotational bias."
200 PRINT: INPUT "DO YOU WISH TO CONTINUE? (Y/N) " ; A$
205 IF LEN(A$) = 0 THEN GOTO 200
220 IF A$ < "Y" THEN PRINT D$ + "RUN SELNPROG"
225 HOME
230 VTAB(10); INPUT "HOW MANY CORNERS DOES THE OBJECT HAVE? " ; U
233 IF LEN(ULIN$) = 0 THEN GOTO 230
235 ULIN% = VAL(ULIN$)
240 IF ULIN% > 8 THEN HOME; VTAB(10); PRINT CHR$(7); HTAB(2); PRINT "I am sorry, due to memory constraints"
242 IF ULIN% > 8 THEN HTAB(2); PRINT "the computer is unable to memorize an"
243 IF ULIN% > 8 THEN HTAB(6); PRINT "object with more than 8 sides."
244 IF ULIN% > 8 THEN INPUT "WOULD YOU LIKE TO TRY ANOTHER OBJECT? " ; A$
245 IF ULIN% > 8 AND A$ < "Y" THEN PRINT D$ + "RUN SELNPROG, D1"
246 IF ULIN% > 8 THEN GOTO 225
248 IF ULIN% < 3 THEN HOME; PRINT CHR$(7); VTAB(12); PRINT "THE OBJECT MUST HAVE AT LEAST 3 CORNERS"; FOR I = 1 TO 2000; NEXT I; GOTO 225
250 HOME
260 VTAB(12); PRINT "BEGIN THE MEMORIZATION PROCESS"
263 NPIC = 1; WRC = 0
265 FOR I = 1 TO ULIN%
266 SX(I) = 0; SY(I) = 0
267 NEXT I
270 CALL 27136: REM CALL $6A00-CONTR2,OBJ0
275 IF PEEK(36910) < 0 THEN GOTO 370
280 NLIN% = PEEK(33792): REM NO. OF LINES FROM PROCESS
290 IF NLIN% = ULIN% THEN GOTO 435
295 IF NPIC < 1 AND WRC < 5 THEN PRINT D$; "BLOAD PARAMETERS, D1": WRC = WRC + 1; GOTO 270
300 HOME
310 VTAB(9); PRINT "The number of corners you input was"
PRINT "different than the number of corners"
PRINT "found by the machine vision process"
PRINT "are you sure that the object has "
INPUT "corners? (Y/N) "; A$
IF A$ < "Y" THEN HOME; PRINT D$; "LOAD PARAMETERS,D1"; GOTO 225
HOME
VTAB (9); PRINT "It appears that an error has occurred"
PRINT "during the machine vision process."
IF ER% < 0 THEN GOTO 960
GOSUB 1510: REM CALCULATE ABS. COORDINATES
GOSUB 1640: REM REORDER CORNERS
GOSUB 2530: REM CALCULATION OF SHAPE CENTERED COORDINATES
FOR I = 1 TO NLINX
SX(I) = SX(I) + XC(I)
SY(I) = SY(I) + YC(I)
NEXT I
IF NPIC = 5 THEN GOTO 575
NPIC = NPIC + 1
PRINT CHR$ (4); "LOAD PARAMETERS,D1"
HOME
VTAB (10); HTAB (15): INVERSE: PRINT "PLEASE WAIT": NORMAL
GOSUB 1040: REM LINEAR REGRESSION
GOSUB 1280: REM CORNER CALCULATION
IF ERA < 0 THEN GOTO 960
GOSUB 1510: REM CALCULATE ABS. COORDINATES
GOSUB 1640: REM REORDER CORNERS
GOSUB 2530: REM CALCULATION OF SHAPE CENTERED COORDINATES
FOR I = 1 TO NLINX
SX(I) = (INT (SX(I) / NPIC) * 1000) + .5) / 1000
SY(I) = (INT (SY(I) / NPIC) * 1000) + .5) / 1000
NEXT I
HOME
VTAB (3): PRINT "The shape centered coordinates for this"
PRINT "object are:"
710 IF SY(I) = 0 THEN PRINT SPC(4);"0.000": GOTO 720
715 PRINT SPC(3);SY(I)
720 NEXT I
730 PRINT
740 HTAB (2): PRINT "WOULD YOU LIKE TO STORE THIS?"
750 HTAB (2): INPUT "REPRESENTATION OF THE OBJECT? (Y/N) ";A$
760 IF A$ = "Y" THEN GOTO 800
770 HOME : VTAB (12): INPUT "WOULD YOU LIKE TO TRY AGAIN? (Y/N) ";A$
780 IF A$ = "Y" THEN PRINT D$ ;"BLOAD PARAMETERS,D1" : GOTO 100
790 PRINT D$ + "RUN SELNPROG,D1"
800 NSP% = 0
810 REM READ SHAPE DATA
820 RX% = 0
830 GOSUB 3030
840 GOSUB 3280
845 IF ERA = 1 THEN HOME : VTAB (12): PRINT "THE OBJECT HAS ALREADY BEEN STORED IN"
846 IF ERA = 1 THEN PRINT "THE COMPUTER'S MEMORY. DO YOU WISH TO"
847 IF ERA = 1 THEN INPUT "STORE A DIFFERENT OBJECT? (Y/N) ";A$
848 IF ERA = 1 AND A$ = "Y" THEN PRINT D$ ;"BLOAD PARAMETERS,D1" ; GOTO 100
849 IF ERA = 1 THEN PRINT D$ + "RUN SELNPROG,D1"
850 REM WRITE NEW SHAPE DATA
855 RX% = 1
860 NSHP% = NSHP% + 1
865 NED%(NSHP%,1) = SX(I)
870 FOR I = 1 TO NLIN% 
875 CCR(NSHP%,I,1) = SX(I)
880 CCR(NSHP%,I,2) = SY(I)
885 NEXT I
890 HOME : VTAB (10): PRINT "YOU MAY NOW ENTER A VERBAL DESCRIPTION"
895 PRINT "OF THE OBJECT. PLEASE KEEP THIS"
900 PRINT "DESCRIPTION AS SHORT AS POSSIBLE. FOR"
905 PRINT "EXAMPLE YOU MIGHT ENTER '3/4 INCH"
910 PRINT "SQUARE', PRESS RETURN WHEN YOU ARE DONE."
912 INPUT ID$(NSHP%)
914 HOME
916 VTAB (12): HTAB (3): INPUT "CAN THE ROBOT RETREIVE THE OBJECT? " ;A$
918 IF A$ < > "Y" THEN PCK(NSHP%) = 0
920 IF A$ = "Y" THEN PCK(NSHP%) = 1.5 * CCR(NSHP%,NLIN%,2)
922 HOME
925 VTAB (9): HTAB (15): INVERSE ; PRINT "PLEASE WAIT": NORMAL
930 VTAB (11): HTAB (8): PRINT "The object is being stored."
932 GOSUB 3030
934 HOME
935 VTAB (12): INPUT "DO YOU WISH TO STORE ANOTHER OBJECT? " ;A$
940 IF A$ = "Y" THEN PRINT D$;"BLOAD PARAMETERS,D1": GOTO 100
945 PRINT D$;"RUN SELNPROG,D1"
955 END
960 PRINT "ERROR NO.  = ";ER%
965 PRINT "ERROR IN RECOGNITION"
970 INPUT "DO YOU WISH TO TRY AGAIN? (Y/N) ": A$
975 IF A$ < » «Y» THEN PRINT D$;"BLOAD PARAMETERS,D1"
980 PRINT D$;"RUN SELNPROG,D1"
985 GOTO 100
990 INPUT ID$(NSHP)
1000 REM
1010 REM SUBROUTINE TO COMPUTE LINEAR REGRESSION
1020 REM
1040 IF NLINX < 3 THEN ER% = 1: RETURN
1050 FOR I = 1 TO NLINX
1070 NEX = PEEK (34175 + I)
1080 SX = PEEK (34303 + I) + 256 * PEEK (34431 + I)
1090 ZX = PEEK (34559 + I) + 256 * PEEK (34687 + I) + 65536 * PEEK (34815 + I)
1100 SY = PEEK (34943 + I) + 256 * PEEK (35071 + I)
1110 XY = PEEK (35199 + I) + 256 * PEEK (35327 + I) + 65536 * PEEK (35455 + I)
1120 IF (NEX * ZX) - (SX * SX) = 0 THEN BC(I) = 999; AC(I) = (S X / NEX); GOTO 1150
1130 BC(I) = ((NEX * XY) - (SX * SY)) / ((NEX * ZX) - (SX * SX))
1140 AC(I) = SY / NEX - (BC(I) * (SX / NEX))
1150 NEXT I
1160 RETURN
1250 REM
1260 REM CORNER CALCULATION
1270 REM
1280 FOR I = 1 TO NLINX - 1
1285 J = I + 1
1290 GOSUB 1400
1300 NEXT I
1310 I = NLINX
1320 J = 1
1330 GOSUB 1400
1340 RETURN
1350 GOSUB 1410: REM CALCULATE LAST CORNER IN SHAPE
1400 REM CORNER CALCULATION
1410 IF (BC(I) - BC(J)) = 0 THEN PRINT "LINES ARE PARALLEL (CORNER CALCULATION)"; ER% = 9: GOTO 400
1412 IF BC(I) = 999 THEN XC(I) = AC(I): GOTO 1425
1414 IF BC(J) = 999 THEN XC(I) = AC(J): GOTO 1425
1420 XC(I) = (AC(J) - AC(I)) / (BC(I) - BC(J))
1425 IF BC(J) = 999 THEN YC(I) = AC(I) + BC(I) * XC(I): GOTO 144
1430 YC(I) = AC(J) + BC(J) * XC(I)
1440 IF XC(I) < 0 THEN ER% = 2: RETURN
1450 IF XC(I) > 100 THEN ER% = 3: RETURN
1460 IF YC(I) < 0 THEN ERR = 4: RETURN
1470 IF YC(I) > 100 THEN ERR = 5: RETURN
1480 RETURN
1490 REM
1500 REM CALCULATION OF THE ABSOLUTE COORDINATES
1505 REM
1510 LWIN = PEEK (36896)
1520 TWIN = PEEK (36897)
1530 HSKP = PEEK (36900)
1540 VSKP = PEEK (36901)
1550 FOR I = 1 TO NLIN
1560 AC(I) = 8 + (LWIN + YC(I) * (HSKP + 1) - 120) * 0.0407126
1570 BC(I) = (120 - TWIN - XC(I) * (VSKP + 1)) * 0.031878
1580 NEXT I
1590 RETURN
1600 REM
1610 REM REORDER CORNERS FOR SHAPE CENTERED ORIGIN
1620 REM
1630 REM CALCULATE CORNER DISTANCES
1640 FOR I = 1 TO NLIN - 1
1650 CD(I) = (AC(I + 1) - AC(I)) ^ 2 + (BC(I + 1) - BC(I)) ^ 2
1660 NEXT I
1670 I = NLIN
1680 CD(I) = (AC(I) - AC(I)) ^ 2 + (BC(I) - BC(I)) ^ 2
1690 FOR I = 1 TO NLIN
1700 RK(I) = 0
1710 NEXT I
1715 K = 0
1720 REM ASSIGN RANK
1730 FOR I = 1 TO NLIN
1740 IF RK(I) = 0 THEN GOTO 1770
1750 NEXT I
1760 GOTO 1900: REM RANKING FINISHED
1770 K = K + 1
1780 J = I
1790 FOR I = J + 1 TO NLIN
1800 IF RK(I) < > 0 THEN GOTO 1830
1810 IF CD(I) < = CD(J) THEN GOTO 1830
1820 J = I
1830 NEXT I
1840 FOR I = 1 TO NLIN
1850 IF RK(I) < > 0 THEN GOTO 1880
1860 IF CD(I) < CD(J) * 0.98 THEN GOTO 1880
1870 RK(I) = K
1880 NEXT I
1890 GOTO 1930: REM CHECK NEXT RANK
1900 IF K = 1 THEN RETURN : REM NO ORDERING NECESSARY
2000 REM CHECK FOR ORDERING
2010 J = 0
2020 FOR I = 1 TO NLIN
2030 IF RK(I) < > 1 THEN GOTO 2160
2040 IF I = 1 THEN O8% = RK%(NLIN%)
2050 IF I > 1 THEN O8% = RK%(I - 1)
2060 IF I < NLIN% THEN OAX% = RK%(I + 1)
2070 IF I = NLIN% THEN OAX% = RK%(1)
2080 IF OAX% < = OB% THEN OE% = 100 * OB% - OAX%
2090 IF OAX% > OB% THEN OE% = OB% - 100 * OAX%
2100 J = J + 1
2110 IF J = 1 THEN GOTO 2140
2120 IF ABS(OE%) < ABS(OD%) THEN GOTO 2160
2130 IF OE% < = 0D% THEN GOTO 2160
2140 0P% = I
2150 0D% = OE%
2160 NEXT I
2170 REM REORDER PROCEDURE
2180 K = 0
2190 OAX% = 1
2200 OB% = NLIN%
2210 IF 0O% > = 0 THEN GOSUB 2240: REM REORDER POSITIVE
2220 IF 0O% < 0 THEN GOSUB 2360: REM REORDER NEGATIVE
2225 RETURN
2230 REM POSITIVE REORDERING OF ONE SHAPE
2240 FOR I = 0AX% TO O~h
2250 K = K + 1
2260 XC(K) = AC(I)
2270 YC(K) = BC(I)
2280 NEXT I
2290 FOR I = 0AX% TO O~h - 1
2300 K = K + 1
2310 XC(K) = AC(I)
2320 YC(K) = BC(I)
2330 NEXT I
2340 RETURN
2350 REM NEGATIVE REORDERING OF ONE SHAPE
2360 IF OP% < OB% THEN OP% = OP% + 1
2370 IF OP% = OB% THEN OP% = OAX%
2380 FOR I = OP% TO OAX% STEP - 1
2390 K = K + 1
2400 XC(K) = AC(I)
2410 YC(K) = BC(I)
2420 NEXT I
2430 FOR I = OB% TO OP% + 1 STEP - 1
2440 K = K + 1
2450 XC(K) = AC(I)
2460 YC(K) = BC(I)
2470 NEXT I
2480 RETURN
2500 REM
2510 REM CALCULATION OF SHAPE CENTERED COORDINATES
2520 REM
2530 XO = XC(1)
2540 YO = YC(1)
2550 X1 = XC(2) - XO
2560 Y1 = YC(2) - YO
2570 XC(1) = 0
2580 YC(1) = 0
2590 CD(1) = SQR ((X0 - XC(2)) * 2 + (Y0 - YC(2)) * 2)
2600 AAC = (XC(2) - XO) / CD(1)
2610 AO = - ATN (AAC / SQR (- AAC * AAC + 1)) + PI / 2: REM ARCCOS(AAC)
2620 IF (YC(2) - YO) < 0 THEN AO = 2 * PI - AO
2630 CD(2) = SQR ((XC(NLIN%) - XO) * 2 + (YC(NLIN%) - YO) * 2)
2640 BBC = (XC(NLIN%) - XO) / CD(2)
2650 B0 = - ATN (BBC / SQR (- BBC * BBC + 1)) + PI / 2: REM ARCCOS(BBC)
2660 DO = B0 - AO
2670 IF DO < 0 THEN DO = DO + PI
2680 IF DO > (PI / 2) THEN AO = - AO
2690 XC(2) = CD(1)
2700 YC(2) = 0
2710 FOR I = 3 TO NLIN%
2720 CD(2) = SQR ((XC(I) - XO) * 2 + (YC(I) - YO) * 2)
2730 XC(I) = (X1 * (XC(I) - XO) + Y1 * (YC(I) - YO)) / CD(1)
2740 YC(I) = SQR (1 - (XC(I) * XC(I)) / (CD(2) * CD(2))) * CD(2)
2750 NEXT I
2760 RETURN
3000 REM
3010 REM READ(RW=0) OR WRITE (RW=1) SHAPE DATA INTO MEMORY
3020 REM
3030 PRINT D$; "OPEN SHAPES, D1"
3040 IF RW% = 1 THEN PRINT D$; "WRITE SHAPES"
3050 IF RW% < > 1 THEN PRINT D$; "READ SHAPES"
3060 REM READ/WRITE NO. OF SHAPES IN FILE
3070 IF RW% = 1 THEN PRINT NSHP%
3080 IF RW% < > 1 THEN INPUT NSHP%
3090 REM SHAPE ID, NUMBER OF CORNERS AND CORNER COORDINATES
3100 FOR I = 1 TO NSHP%
3110 IF RW% = 1 THEN PRINT ID$(I): PRINT NED%(I)
3120 IF RW% < > 1 THEN INPUT ID$(I): INPUT NED%(I)
3130 FOR J = 1 TO NED%(I)
3140 IF RW% = 1 THEN PRINT CCR(I,J,1), CCR(I,J,2)
3150 IF RW% < > 1 THEN INPUT CCR(I,J,1), CCR(I,J,2)
3160 NEXT J
3170 IF RW% = 1 THEN PRINT PCK(I)
3188 IF R% < 1 THEN INPUT PCK(I)
3190 NEXT I
3195 PRINT D$; "CLOSE SHAPES"
3200 RETURN
3250 REM
3260 REM SHAPE CLASSIFICATION MODULE
3270 REM
3280 ET = .01 * PEEK (36906)
3290 FOR I = 1 TO NSHP%
3295 MC% = 0
3300 IF NED%(I) < NLIN% THEN GOTO 3380
3310 FOR J = 1 TO NED%(I)
3320 ED = ABS (CCR(I,J,1) - SX(J))
3330 IF ED > ET THEN GOTO 3360
3340 ED = ABS (CCR(I,J,2) - SY(J))
3350 IF ED < ET THEN MCH% = MCH% + 1
3360 NEXT J
3370 IF MCH% = NLIN% THEN GOTO 3400
3380 NEXT I
3390 RETURN
3400 ER% = 1
3410 RETURN
APPENDIX F

ASSEMBLY PROGRAM LISTINGS
2 **************************************************
3 MM 10/4/88
4
5 THIS PROGRAM DETERMINES THE DIMENSIONS AND SKIP PARAMETERS
6 OF A WINDOW WHICH WILL ENCLOSE THE OBJECT OF INTEREST
7 AND THEN CALLS NVOS TO OBTAIN A NEW WINDOW OF DATA
8
9 VARIABLES SHARED WITH NVOS
10
11 TWIN EQU $DFE5 ;LINE NUMBER OF THE TOP OF THE WINDOW
12 BWIN EQU $DFE6 ;LINE NUMBER OF THE BOTTOM OF THE WINDOW
13 LWIN EQU $DFE7 ;COLUMN NUMBER OF THE LEFT SIDE OF THE WINDOW
14 RWIN EQU $DFE8 ;COLUMN NUMBER OF THE RIGHT SIDE OF THE WINDOW
15 FPASS EQU $9006 ;1ST PASS FLAG (MUST BE SET TO 1 BEFORE CALLING)
16 NROW EQU $DFE1 ;NO. OF ROWS ACCEPTED
17 NCOL EQU $DFE2 ;NO. OF COLUMNS ACCEPTED
18 NPIXL EQU $DFE3 ;NO. OF PIXELS ACCEPTED LOW
19 NPIXH EQU $DFE4 ;NO. OF PIXELS ACCEPTED HIGH
20 CMSKP EQU $8FF1 ;SKIP PARAMETERS
21
22 VARIABLES SHARED WITH LIST OF PARAMETERS
23
24 FROW EQU $9020 ;FIRST ROW
25 FCOL EQU $9021 ;FIRST COLUMN
26 LROW EQU $9022 ;LAST ROW OF PIXELS
27 LCOL EQU $9023 ;LAST COLUMN
28 HSKP EQU $9024 ;HORIZONTAL SKIP
29 VSKP EQU $9025 ;VERTICAL SKIP
30 PFLAG EQU $902F ;PRINT FLAG
31
32
33 VARIABLES FOR PROCESS
34
35 BASE EQU $EB ;2BYTE BASE ADDRESS
36 LADX EQU $ED ;2BYTE X-ADDRESS EDLI
37 LADY EQU $FB ;2BYTE Y-ADDRESS EDLI
38
39 ERROR EQU $902E ;ERROR CODE, 0 = O.K.
40 TREDG EQU $E001 ;TRASHOLD FOR EDGE DETECTION
41 COX EQU $E002 ;X COORDINATE OF MATRIX
42 COY EQU $E003 ;Y-COORDINATE OF MATRIX
43 I EQU $E004 ;COUNTER (TEMP. X-COORDINATE)
44 J EQU $E005 ;COUNTER (TEMP. Y-COORDINATE)
45 X1 EQU $E006 ;LEFT EDGE FOR OBJECT LOCATION
46 X2 EQU $E007 ;RIGHT EDGE FOR OBJECT LOCATION
47 Y1 EQU $E008 ;BOTTOM EDGE FOR OBJECT LOCATION
48 Y2 EQU $E009 ;TOP EDGE FOR OBJECT LOCATION
49 GRAD EQU $E00A ;GRADIENT VALUE OF ROBERTS CROSS EDGE
50 VAL1 EQU $E00B ;PIXEL VALUE 1 OF MATRIX
51 FOUND EQU $E00C ;PARAMETER FOR EDGE FOUND
52 FLAG EQU $E00D ;FLAG IN PROGRAMS
53 STEP EQU $E00E ;STEP SIZE FOR LOOP
54 DIVDL EQU $E00F ; DIVIDEND LOW BYTE
55 DVDIVH EQU $E010 ; DIVIDEND HIGH BYTE
56 DIVIS EQU $E011 ; DIVISOR
57 DIRE EQU $E012 ; RESULT OF DIVISION
58 WROW EQU $E013 ; NUMBER OF ROWS IN WINDOW
59 WCOL EQU $E014 ; NUMBER OF COLUMNS IN WINDOW
60 PX EQU $E015 ; X POINTER
61 PY EQU $E016 ; Y POINTER
62 DIR EQU $E017 ; DIRECTION OF EDGE FOLLOWING
63 BXLST EQU $E018 ; BEGINNING POSITION OF X EDGE LIST
64 BYLST EQU $E019 ; BEGINNING POSITION OF Y EDGE LIST
65 DFFLAG EQU $E01A ; DIRECTION FLAG
66 NCONF EQU $E01B ; NOT CONNECTED FLAG
67 NEDPT EQU $E01C ; NO. OF EDGE POINTS
68 ;
69 NVOS EQU $7D90 ; VISION OPERATING SYSTEM ROUTINE
70 TRAXY EQU $7D70 ; TRANSFER PROGRAM
71 FINDOB EQU $7850 ; FIND OBJECT WINDOW
72 ROCED EQU $77C0 ; ROBERTS CROSS EDGE GRADIENT
73 MULTPL EQU $77A0 ; MULTIPLICATION PROGRAM
74 DIVID EQU $7770 ; DIVISION PROGRAM
75 OTMSG1 EQU $7640 ; MESSAGE PROGRAM
76 PRMAT EQU $9030 ; SUBROUTINE TO PLACE MATRIX SO IT CAN BE PRINTED
77 LNKLST EQU $6B00 ; PROGRAM TO FIND LINKED EDGE LIST
78 CORDET EQU $9150 ; CORNER DETECTION ROUTINE
79 LINREG EQU $9150 ; PROGRAM TO COMPUTE SUMS FOR LINEAR REGRESSION
80 ;
ECT FILE NAME IS CONTR2.OBJO
81 ORG $6A00
82 ;
83 *******************************************
84 BEGIN PROGRAM TO DETERMINE WINDOW
85 *******************************************
86 ;
87 LDA #0
88 STA ERROR
89 LDA $C08B ; ACTIVATE BANK SWITCHED MEMORY
90 LDA $C08B ; BY TWO CONSECUTIVE READS
91 LDA #1
92 STA FPASS ; SET FLAG TO ACTIVATE NVOS
93 JSR NVOS ; GET A FRAME OF GRAY-SCALE DATA
94 LDX #MSG2-MSG1
95 JSR OTMSG1 ; WRITE DATA COLLECTION COMPLETE
96 LDA $C08B ; ACTIVATE BANK SWITCHED MEMORY
97 LDA $C08B
98 LDA #0 ; SET UP BASE ADDRESSES FOR
99 STA LADX ; TRANSFERRING DATA FROM
100 LDA #$80 ; BASE ADDRESS $8000 TO
101 STA LADX+1 ; BASE ADDRESS $D000
102 LDA #0
103 STA LADY
104 LDA #$D0
; ACTIVATE BANK SWITCHED MEMORY
; CORNER DETECTION
; DEACTIVATE BANK SWITCHED MEMORY
; TRANSFER DATA TO SECOND BANK OF SWITCHED MEMORY
; GET EDGE LIST

; DEACTIVATE BANK SWITCHED MEMORY
; ACTIVATE SECOND BANK OF SWITCHED MEMORY
; WRITE CORNER DETECTION COMPLETE
; COMPUTE SUMS FOR LINEAR REGRESSION
; DEACTIVATE BANK MEMORY
; ERROR
; CMP #0
; BNE L99
; JSR CORDET ; DO CORNER DETECTION
; LDA $#082 ; DEACTIVATE BANK SWITCHED
; LDX #0
; JSR OTMSG1 ; WRITE CORNER DETECTION COMPLETE
; JSR LINREG ; COMPUTE SUMS FOR LINEAR REGRESSION
; LDA $#082 ; DEACTIVATE BANK MEMORY
; LDA ERROR
; CMP #0
; BNE L99
; JSR OTMSG1 ; WRITE SUMS FOR LINEAR REGRESSION COMPLETE
; JSR LINREG
; LDA $#082
; RTS ; BACK TO BASIC

ECT FILE NAME IS CONTR2.OBJ1

; SUBROUTINE TRAXY
; THIS SUBROUTINE TRANSFERS DATA FROM
; BASE ADDRESS LADX TO LADY
156 ;
157 ;INPUT LADX,LADX+1, LADY AND LADY+1
158 ;
159 LDX NPIXH ;LOAD NO. OF PAGES TO BE TRANSFERRED
160 LDY #0
161 NEXT LDA (LADX),Y ;LOAD WORD TO BE TRANSFERRED
162 STA (LADY),Y ;STORE VALUE AT LADY+1 + LADY + Y
163 DEY
164 BNE NEXT ;IF PAGE HAS NOT BEEN TRANSFERED MOVE NEXT WOR
165 NEXBLK INC LADX+1
166 INC LADY+1 ;INCREMENT TO NEXT PAGE
167 DEX ;DECREMENT NO. OF PAGES REMAINING
168 BNE DONE
169 BNE NEXT
170 LDY NPIXL ;TRANSFER REMAINING PIXELS
171 DONE RTS
173 **************************************************
174 ; SUBROUTINE FINDOB
175 **************************************************ECT FILE NAME IS CONTR2.OBJ2
176 ORG $7850
177 LDX #MSG5-MSG1
178 JSR OTMSG1 ;FIND OBJECT WINDOW
179 LDA $C088 ;ACTIVATE BANK SWITCHED
180 LDA $C088 ;MEMORY
181 LDA $9028 ;INITIALIZE THRESHOLD
182 STA TREDG ;FOR TOTAL IMAGE
183 STA $D9EB ;STORE THRESHOLD IN BANK SWITCHED AREA
184 LDA #2 ;INITIALIZING
185 STA X1
186 STA Y1
187 LDA NROW
188 STA X2
189 DEC X2
190 DEC X2
191 DEC X2 ;X2=NROW-3
192 LDA NCOL
193 STA Y2
194 DEC Y2
195 DEC Y2
196 DEC Y2 ;Y2=NCOL-3
197 ;FIND X1 OF THE OBJECT WINDOW
198 LDA #0
199 STA FOUND
200 LDA X1
201 STA COX
202 DEC COX
203 FI1 INC COX ;LOOP FROM X1 TO X2
204 LDA Y1
205 STA COY
206 DEC COY
; LOOP FROM Y1 TO Y2
207 FI2 INC COY
208 JSR FINDME
209 LDA FOUND ; CHECK FOR TOP EDGES
210 CMP #2
211 BEQ FI3 ; START OF OBJECT FOUND
212 LDA COY
213 CMP Y2
214 BCC FI2 ; NEXT Y
215 LDA COX
216 CMP X2
217 BCC FI1 ; NEXT X
218 LDA $C08A ; DEACTIVATE BANK SWITCHED MEMORY
219 LDX #MSG6-MSG1
220 STX ERROR
221 JSR OTMSG1 ; TOP OF WINDOW NOT FOUND
222 RTS ; EXIT
223 FI3 LDA COX
224 STA X1 ; STORE NEW X1
225 ; FIND Y1 OF THE OBJECT WINDOW
226 LDA #0
227 STA FOUND
228 LDA Y1
229 STA COY
230 DEC COY
231 FI4 INC COY ; LOOP FROM Y1 TO Y2
232 LDA X1
233 STA COX
234 DEC COX
235 FI5 INC COX ; LOOP FROM X1 TO X2
236 JSR FINDME
237 LDA FOUND ; CHECK FOR TWO EDGES
238 CMP #2
239 BEQ FI6 ; START OF OBJECT FOUND
240 LDA COX
241 CMP X2
242 BCC FI5 ; NEXT X
243 LDA COY
244 CMP Y2
245 BCC FI4 ; NEXT Y
246 LDA $C08A ; DEACTIVATE BANK SWITCHED MEMORY
247 LDX #MSG7-MSG1
248 STX ERROR
249 JSR OTMSG1 ; LEFT SIDE OF WINDOW NOT FOUND
250 RTS ; EXIT
251 FI6 LDA COY
252 STA Y1 ; STORE NEW Y1
253 ; FIND X2 OF THE OBJECT WINDOW
254 LDA #0
255 STA FOUND
256 LDA X2
257 STA COX
258 INC COX
259 FI7 DEC COX ; LOOP FROM X2 TO X1
260 LDA Y1
261 STA COY
262 DEC COY
263 FI8 INC COY ; LOOP FROM Y1 TO Y2
264 JSR FINDME
265 LDA FOUND ; CHECK FOR TWO EDGES
266 CMP #2
267 BEQ FI9 ; END OF OBJECT FOUND
268 LDA COY
269 CMP Y2
270 BCC FI8 ; NEXT Y
271 LDA X1
272 CMP COX
273 BCC FI7 ; NEXT X
274 LDA $C08A ; DEACTIVATE BANK SWITCHED MEMORY
275 LDX #MSG8-MSG1
276 STX ERROR
277 JSR OTMSG1 ; BOTTOM OF WINDOW NOT FOUND
278 RTS ; EXIT
279 FI9 LDA COX
280 STA X2 ; STORE NEW X2
281 ; FIND Y2 OF THE OBJECT WINDOW
282 LDA #0
283 STA FOUND
284 LDA Y2
285 STA COY
286 INC COY
287 FI10 DEC COY ; LOOP FROM Y2 TO Y1
288 LDA X1
289 STA COX
290 DEC COX
291 FI11 INC COX ; LOOP FROM X1 TO X2
292 JSR FINDME
293 LDA FOUND ; CHECK FOR TWO EDGES
294 CMP #2
295 BEQ FI12 ; END OF OBJECT FOUND
296 LDA COX
297 CMP X2
298 BCC FI11 ; NEXT X
299 LDA Y1
300 CMP COY
301 BCC FI10 ; NEXT Y
302 LDA $C08A ; DEACTIVATE BANK SWITCHED MEMORY
303 LDX #MSG9-MSG1
304 STX ERROR
305 JSR OTMSG1 ; RIGHT SIDE OF WINDOW NOT FOUND
306 RTS ; EXIT
307 FI12 LDA COY
308 STA Y2 ; STORE NEW Y2
309 ; CALCULATE ACTUAL COORDINATES
310 ;
311  LDA X1
312  ASL A
313  ASL A ;4*X1 BECAUSE OF SKIP
314  CLC
315  ADC TWIN ;ADD ABSOLUTE ADR
316  SEC
317  SBC #$8 ;MINUS 8 FOR WINDOW FRAME
318  STA FROW ;=X1+TWIN-8
319  LDA Y1
320  ASL A
321  ASL A ;4*Y1 BECAUSE OF SKIP
322  CLC
323  ADC LWIN ;ADD ABSOLUTE ADR
324  SEC
325  SBC #$8 ;MINUS 8 FOR WINDOW FRAME
326  STA FCOL ;=Y1+LWIN-8
327  LDA X2
328  ASL A
329  ASL A ;4*X2 BECAUSE OF SKIP
330  CLC
331  ADC TWIN ;ADD ABSOLUTE ADR
332  ADC #$8 ;ADD 8 FOR WINDOW FRAME
333  STA LROW ;X2+TWIN+8
334  LDA Y2
335  ASL A
336  ASL A ;4*Y2 BECAUSE OF SKIP
337  CLC
338  ADC LWIN ;ADD ABSOLUTE ADR
339  ADC #$8 ;ADD 8 FOR WINDOW FRAME
340  STA LCOL ;Y2+LWIN+8
341 ;CALCULATE NO. OF ROWS AND COLUMNS
342  LDA LROW
343  SEC
344  SBC FROW
345  STA WROW ;WROW=X2-X1
346  INC WROW ;WROW=WROW+1
347  LDA LCOL
348  SEC
349  SBC FCOL
350  STA WCOL ;WCOL=Y2-Y1
351  INC WCOL ;WCOL=WCOL+1
352 ;FIND APPROPRIATE SKIP MODES
353  ;
354  LDA WROW
355  STA I
356  LDA WCOL
357  STA J
358  JSR MULTPL ;BASE=WROW*NCOL
359  JSR CKSUM ;OVER 4000 ?
360  LDA FLAG
361  CMP #1
362  BNE FI20 ;TO MANY PIXELS
363  LDA  #0 ;NO SKIP NECESSARY
364  STA  HSKP
365  STA  VSKP
366  STA  CMSKP
367  RTS ;EXIT
368  ;
369  F120  LSR  BASE+1
370  ROR  BASE ;DIVIDE BASE BY 2
371  JSR  CKSUM ;OVER 4000 ?
372  LDA  FLAG
373  CMP  #1
374  BNE  F122 ;TO MANY PIXELS
375  LDA  WROW ;SKIP = 2
376  CMP  WCOL
377  BCC  F121 ;NROW<NCOL
378  LDA  #0 ;NROW>NCOL
379  STA  HSKP
380  LDA  #1
381  STA  VSKP
382  STA  CMSKP
383  DEC  WROW
384  LSR  WROW ;HALF OF WROW
385  INC  WROW
386  RTS ;EXIT
387  ;
388  F121  LDA  #1 ;NROW<NCOL
389  STA  HSKP
390  LDA  #0
391  STA  VSKP
392  LDA  #$10
393  STA  CMSKP
394  DEC  WCOL
395  LSR  WCOL ;HALF OF WCOL
396  INC  WCOL
397  RTS ;EXIT
398  F122  LSR  BASE+1
399  ROR  BASE ;DIVIDE BASE BY 2 (4)
400  JSR  CKSUM ;OVER 4000 ?
401  LDA  FLAG
402  CMP  #1
403  BNE  F125A ;TO MANY PIXELS
404  LDA  #0 ;SKIP = 3 OR 4
405  STA  DIVDH
406  LDA  WROW
407  STA  DIVDL
408  DEC  DIVDL
409  LDA  #$3
410  STA  DIVIS
411  JSR  DIVID ;DIVIDE NROW/3
412  LDA  DIRE
413  STA  I
414  INC  I
415 JSR MULTPL ; \text{BASE} = (\text{NROW}/3) \times \text{NCOL}
416 JSR CKSUM ; \text{OVER} 4000 \ ?
417 LDA FLAG
418 CMP #1
419 BNE F124 ; \text{SKIP} = 4
420 LDA WROW ; \text{SKIP} = 3
421 CMP WCOL
422 BCC F123 ; \text{NROW} < \text{NCOL}
423 LDA #0 ; \text{NROW} > \text{NCOL}
424 STA HSKP
425 LDA #2
426 STA VSKP
427 STA CMSKP
428 LDA 1
429 STA WROW ; \text{WROW} = \text{WROW}/3
430 RTS ; \text{EXIT}
431 ;
432 F123 LDA #2 ; \text{NROW} < \text{NCOL}
433 STA HSKP
434 LDA #0
435 STA VSKP
436 LDA #20
437 STA CMSKP
438 LDA #0
439 STA DIVDH
440 LDA WCOL
441 STA DIVDL
442 DEC DIVDL
443 LDA #3
444 STA DIVIS
445 JSR DIVID ; \text{NCOL}/3
446 LDA DIRE
447 STA WCOL
448 INC WCOL
449 RTS ; \text{EXIT}
450 ;
451 F125A JMP F125 ; \text{JUMP OVER PAGE LENGTH}
452 F124 LDA #1 ; \text{SKIP} = 4
453 STA HSKP
454 STA VSKP
455 LDA #11
456 STA CMSKP
457 DEC WROW
458 LSR WROW ; \text{HALF OF NROW}
459 INC WROW
460 DEC WCOL
461 LSR WCOL ; \text{HALF OF WCOL}
462 INC WCOL
463 RTS ; \text{EXIT}
464 ;
465 F125 LSR BASE+1
466 ROR BASE ; \text{DIVIDE BASE BY 2 (8)}
JSR CKSUM ;OVER 4000?
LDA FLAG
CMP #1
BNE F129A ;TO MANY PIXELS
LDA WROW ;SKIP = 6 OR 8
CMP WCOL
BCC F127 ;NROW<NCOL
LDA #0 ;NROW>NCOL
STA DIVDH
LDA WROW
STA DIVDL
DEC DIVDL
LDA #3
STA DIVIS
JSR DIVID ;DIVIDE NROW/3
LDA DIRE
STA I
INC I
DEC J
LSR J ;DIVIDE NCOL/2
INC J
JSR MULTPL ;BASE=(NROW/3)*(NCOL/2)
JSR CKSUM ;OVER 4000?
LDA FLAG
CMP #1
BNE F126 ;SKIP = 8
LDA #1 ;SKIP = 6
STA HSKP
LDA #2
STA VSKP
LDA #$12
STA CMSKP
LDA I
STA WROW ;WROW=WROW/3
LDA J
STA WCOL ;WCOL=WCOL/2
RTS ;EXIT
FI26 LDA #1 ;SKIP = 8
STA HSKP
LDA #3
STA VSKP
LDA #$13
STA CMSKP
DEC WROW
LSR WROW
LSR WROW ;WROW=WROW/4
INC WROW
DEC WCOL
LSR WCOL ;WCOL=WCOL/2
INC WCOL
RTS ;EXIT

519  F129A   JMP  F129 ; JUMP OVER PAGE LENGTH
520  ;
521  F127   LDA  #0 ; NROW < NCOL
522  STA  DIVDH
523  LDA  WCOL
524  STA  DIVDL
525  DEC  DIVDL
526  LDA  #3
527  STA  DIVIS
528  JSR  DIVID ; DIVIDE NCOL/3
529  LDA  DIRE
530  STA  J
531  INC  J
532  DEC  I
533  LSR  I ; DIVIDE NROW/2
534  INC  I
535  JSR  MULTPL ; BASE = (NROW/2) * (NCOL/3)
536  JSR  CKSUM ; OVER 4000 ?
537  LDA  FLAG
538  CMP  #1
539  BNE  F128 ; SKIP = 8
540  LDA  #2 ; SKIP = 6
541  STA  HSKP
542  LDA  #1
543  STA  VSKP
544  LDA  #$21
545  STA  CMSKP
546  LDA  I
547  STA  WROW ; WROW = WROW/2
548  LDA  J
549  STA  WCOL ; WCOL = WCOL/3
550  RTS ; EXIT
551  F128   LDA  #3 ; SKIP = 8
552  STA  HSKP
553  LDA  #1
554  STA  VSKP
555  LDA  #$31
556  STA  CMSKP
557  DEC  WROW
558  LSR  WROW ; WROW = WROW/2
559  INC  WROW
560  DEC  WCOL
561  LSR  WCOL
562  LSR  WCOL ; WCOL = WCOL/4
563  INC  WCOL
564  RTS ; EXIT
565  ;
566  F129   LDA  #0
567  STA  DIVDH
568  LDA  WROW
569  STA  DIVDL
570  DEC  DIVDL
571 LDA #3
572 STA DIVIS ;DIVIDE NROW/3
573 JSR DIVID
574 LDA DIRE
575 STA I
576 INC I
577 LDA #0
578 STA DIVDH
579 LDA WCOL
580 STA DIVDL
581 DEC DIVDL
582 LDA #3
583 STA DIVIS ;DIVIDE NCOL/3
584 JSR DIVID
585 LDA DIRE
586 STA J
587 INC J
588 JSR MULTPL ;BASE=(NROW/3)*(NCOL/3)
589 JSR CKSUM ;OVER 4000 ?
590 LDA FLAG
591 CMP #1
592 BNE FI30 ;TO MANY PIXELS
593 LDA #2 ;SKIP = 9
594 STA HSKP
595 STA VSKP
596 LDA #22
597 STA CMSKP
598 LDA I
599 STA WROW ;WROW=WROW/3
600 LDA J
601 STA WCOL ;WCOL=WCOL/3
602 RTS ;EXIT
603 FI30 DEC I
604 LSR I
605 LSR I ;DIVIDE NROW/4
606 INC I
607 LDA #0
608 STA DIVDH
609 LDA WCOL
610 STA DIVDL
611 DEC DIVDL
612 LDA #3
613 STA DIVIS ;DIVIDE NCOL/3
614 JSR DIVID
615 LDA DIRE
616 STA J
617 INC J
618 JSR MULTPL ;BASE=(NROW/4)*(NCOL/3)
619 JSR CKSUM ;OVER 4000 ?
620 LDA FLAG
621 CMP #1
622 BNE FI32 ;SKIP = 16
623  LDA WROW ;SKIP = 12
624  CMP WCOL
625  BCC FI31 ;NROW<NCOL
626  LDA #2 ;NROW>NCOL
627  STA HSKP
628  LDA #3
629  STA VSKP
630  LDA #$23
631  STA CMSKP
632  LDA I
633  STA WROW ;WROW=WROW/4
634  LDA J
635  STA WCOL ;WCOL=WCOL/3
636  RTS ;EXIT
637  
638  FI31  LDA #3 ;NROW<NCOL
639  STA HSKP
640  LDA #2
641  STA VSKP
642  LDA #$32
643  STA CMSKP
644  LDA #0
645  STA DIVDH
646  LDA WROW
647  STA DIVDL
648  DEC DIVDL
649  LDA #3
650  STA DIVIS
651  JSR DIVID ;DIVIDE NROW/3
652  LDA DIRE
653  STA WROW
654  INC WROW
655  DEC WCOL
656  LSR WCOL
657  LSR WCOL ;DIVIDE WCOL/4
658  INC WCOL
659  RTS ;EXIT
660  FI32  LDA #3 ;SKIP = 16
661  STA HSKP
662  STA VSKP
663  LDA #$33
664  STA CMSKP
665  DEC WROW
666  LSR WROW
667  LSR WROW ;WROW=WROW/4
668  INC WROW
669  DEC WCOL
670  LSR WCOL
671  LSR WCOL ;WCOL=WCOL/4
672  INC WCOL
673  RTS ;EXIT
674  **************************************************
675 ;FIND THE MAXIMAL EDGE POINTS
676 ;PARAMETER COX,COY,TREDG
677 ;RESULT FOUND
678 ;
679 FINDME JSR ROCED ;GET GRADIENT
680 LDA GRAD
681 CMP TREDG
682 BCS FM1 ;GRADIENT>THRESHOLD
683 LDA #0
684 STA FOUND
685 RTS
686 FM1 INC FOUND
687 RTS
688 **************************************************
689 ;SUBROUTINE TO CHECK WHETHER SUM OF PIXELS IS OVER 4000 ($0FA0)
690 ;IF SMALLER SET FLAG=1
691 ;
692 ;
693 CKSUM LDA #0
694 STA FLAG
695 LDA BASE+1
696 CMP #$0F
697 BEQ CS2 ;HIGH BYTE EQUAL
698 BCC CS1 ;SUM SMALLER
699 RTS ;SUM BIGGER, EXIT
700 CS1 LDA #1
701 STA FLAG
702 RTS ;EXIT
703 CS2 LDA BASE
704 CMP #$A0
705 BCC CS1 ;SUM SMALLER
706 RTS ;SUM BIGGER, EXIT
707 **************************************************
708 ;SUBROUTINE ROCED
709 **************************************************
ECT FILE NAME IS CONTR2.OBJ3
710 ORG $77C0
711 ;ROBERTS CROSS EDGE CALCULATION
712 ;VARIABLES COX,COY
713 ;RESULT GRAD
714 ;
715 LDA COX
716 STA I
717 LDA COY
718 STA J
719 JSR MATVAL ;GET LEFT UPPER VALUE
720 LDA VAL1
721PHA ;SAVE VAL1
722 INC I
723 INC J
724 JSR MATVAL ;GET RIGHT LOWER VALUE
725 PLA ;GET VAL1 BACK
F15

ASSEMBLY-CONTROL-PROGRAM 09-05-1989 003 PAGE 15

; DIFFERENCE OF FIRST DIAGONAL
726  SEC
727  SBC  VAL1
728  BPL  R01
729  STA  VAL1
730  LDA  #0
731  SEC
732  SBC  VAL1
733  R01  STA  GRAD
734  DEC  I
735  JSR  MATVAL
736  LDA  VAL1
737  PHA
738  INC  I
739  DEC  J
740  JSR  MATVAL
741  PLA
742  SEC
743  SBC  VAL1
744  BPL  R02
745  STA  VAL1
746  LDA  #0
747  SEC
748  SBC  VAL1
749  R02  CLC
750  ADC  GRAD
751  BCC  R03
752  LDA  #$FF
753  R03  STA  GRAD
754  RTS
755 ******************************************************
756 ; GET THE POSITION OF THE VALUE WITH THE COORDINATES I,J
757 ; AND STORE THE VALUE IN VAL1
758 ;
759  MATVAL  LDA  J
760  PHA
761  LDA  NCOL
762  STA  J
763  JSR  MULTPL
764  PLA
765  STA  J
766  CLC
767  ADC  BASE
768  STA  BASE
769  LDA  #$00
770  ADC  BASE+1
771  STA  BASE+1
772  LDY  #0
773  LDA  <BASE>,Y
774  STA  VAL1
775  RTS
776 ******************************************************
777 ; SUBROUTINE MULTPL
MULTIPLICATION OF I AND J,

RESULT BASE

LDA I ;SAVE I
PHA

LDA #0 ;INIT RESULT (HIGH)
STA BASE ;INIT RESULT (LOW)
LDX #8 ;X IS SHIFT COUNTER

MU1 LSR I ;SHIFT MPR (LOOP)

BCC MU2

CLC ;CARRY WAS ONE. CLEAR IT
ADC J ;A=A + MPD

MU2 ROR A ;SHIFT RESULT

ROR BASE ;CATCH BIT INTO B
DEX ;DECREMENT COUNTER

BNE MU1 ;LAST SHIFT?

STA BASE+1 ;SAVE HIGH BYTE

PLA ;GET I BACK

STA I

RTS

DIVISION ROUTINE DIRE=DIVD(H&L)/DIVIS

LDA #0 ;INITIALIZE RESULT
STA DIRE

LDA DIVDH ;DIVIDEND HIGH
LDY #8
SEC

SBC DIVIS

PHP ;LOOP

ROL DIRE

ASL DIVDL

ROL A

PLP

BCC DI2

SBC DIVIS

JMP DI3 ;NEXT

DI2 ADC DIVIS

DI3 DEY

BNE DI1

BCS DI4

ADC DIVIS

CLC

DI4 ROL DIRE

RTS

*********
F17

ASSEMBLY-CONTROL-PROGRAM 09-05-1989 003 PAGE 17

828 ******************************************************
829 ; SUBROUTINE PRMAT
829 ******************************************************
ECT FILE NAME IS CONTR2.OBJ6

831 ORG $9030
832 CLC
833 LSR PFLAG ;ROTATE LSB OF PRINT FLAG TO INDICATE VIEWING
834 BCC BNK1 ;IF VIEWING AREA OPEN FIRST BANK OF MEMORY
835 BNK2 LDA $C083 ;OTHERWISE ACTIVATE BANK 2 OF BANK SWITCHED MEM
836 LDA $C083 ;(AREA WHERE WINDOW DATA IS STORED)
837 JMP BNK3
838 BNK1 LDA $C08B ;ACTIVATE SECOND BANK OF BANK SWITCHED MEMORY
839 LDA $C08B ;(AREA WHERE VIEWING AREA DATA IS STORED)
840 BNK3 LDA $DFEB ;MOVE THRESHOLD FROM UPPER MEMORY TO LOWER MEM
841 STA TREDG
842 LDA NROW ;MOVE THE NO. OF ROWS AND COLUMNS
843 STA $9007 ;FROM BANK SWITCHED MEMORY TO LOWER MEMORY
844 LDA NCOL
845 STA $9008
846 CLC
847 LSR PFLAG ;IF SECOND BIT OF PRINT FLAG IS SET THEN CREATI
848 BCS BNKR ;ROBERTS VALUES, ELSE MOVE GRAY SCALES
849 LDA #0 ;SET UP BASE ADDRESSES FOR TRANSFERRING
850 STA LADX ;DATA FROM BASE ADDRESS
851 LDA #$D0 ;$0000 TO $8000
852 STA LADX+1
853 LDA #0
854 STA LADY
855 LDA #$80
856 STA LADY+1
857 JSR TRAXY ;TRANSFER DATA
858 LDA $C082 ;DEACTIVATE EITHER BANK OF BANK SWITCHED MEM
859 RTS ;RETURN TO BASIC

860 
861 ;ESTABLISH ROBERTS MATRIX IN LOWER MEMORY
862 
863 BNKR LDA #0
864 STA COX
865 STA COY
866 STA LADX
867 LDA #$80
868 STA LADX+1
869 LDA NCOL
870 STA Y2 ;Y2=NCOL-1
871 DEC Y2
872 LDA NROW ;X2=NROW-1
873 STA X2
874 DEC X2
875 BNKR1 LDA #0 ;LOWER MEMORY
876 STA COY ;RESET CURRENT COLUMN NUMBER TO 0
877 BNKR2 JSR ROCED ;CALCULATE GRADIENT
878 LDA GRAD ;LOAD ROBERTS GRADIENT VALUE
LDY #0
STA (LADX),Y ;STORE VALUE IN LOWER MEMORY
INC LADX ;MOVE TO NEXT MEMORY LOCATION
BNE BNKR3 ;IF Y=256 INCREMENT BASE ADDRESS
INC LADX+1
BNKR3 INC COY
LDA COY
CMP Y2 ;HAS THE VALUE BEEN CALCULATED FOR THE LAST COI
BCC BNKR2 ;IF NOT NEXT X
INC COX
LDA COX
CMP X2 ;HAS THE VALUE BEEN CALCULATED FOR LAST ROW?
BCC BNKR1 ;IF NOT NEXT Y
LDA #$C082 ;DEACTIVATE BANK SWITCHED MEMORY
LDA #2
STA PFLAG ;SET PFLAG TO INDICATE ROBERTS VALUES
RTS ;RETURN TO BASIC
*******************************************************************************
SUBROUTINE OTMSG1
*******************************************************************************
ECT FILE NAME IS CONTR2.OBJ7
ORG $7640
ROUTINE TO OUTPUT A MESSAGE TO THE SCREEN
ROUTINE TO OUTPUT THE MESSAGE INDEXED BY X TO THE SCREEN

OUTPUT EQU $FDFO
CR EQU $80

NOW FOR THE MESSAGES

MSG1 DFB CR
ASC 'CORNER DETECTION COMPLETE.'

MSG2 DFB CR
ASC 'DATA COLLECTION COMPLETE.'
920 MSG3  DFB  CR
921 ASC  'LINEAR  REGRESSION SUMS COMPLETE.'

922 MSG4  DFB  CR
923 ASC  'DATA  HAS BEEN TRANSFERRED.'

924 MSG5  DFB  CR
925 ASC  'FIND  OBJECT WINDOW.'

926 MSG6  DFB  CR
927 ASC  'NO  OBJECT DETECTED.'

928 MSG7  DFB  CR
929 ASC  'LEFT  SIDE OF WINDOW NOT FOUND.'
930 MSG8  DFB  CR
931    ASC  'BOTTOM OF WINDOW NOT FOUND.'

932 MSG9  DFB  CR
933    ASC  'RIGHT SIDE OF WINDOW NOT FOUND.'

ASSEMBLY: NO ERRORS
VISION-OPERATING-SYSTEM 09-05-1989 002 PAGE 1

;******************************************************************************
;NVOS IS A SHORTENED VERSION OF VOS
;CREATED ON OCTOBER 4, 1988. IT
;PERFORMS THE SAME BASIC TASK EXCEPT
;MANY SUBROUTINES USED IN THE INTERACTIVE
;MODULES HAVE BEEN ELIMINATED
;
;GOFLAG EQU $9000 ;FLAG- TELLS MAIN TO PROC DATA
;TCRSR EQU $9001 ;TOP OF CURSOR (1ST SCAN LINE)
;BCRSR EQU $9002 ;BOTTOM OF CURSOR
;LCRSR EQU $9003 ;LEFT EDGE OF CURSOR
;RCRSR EQU $9004 ;RIGHT EDGE OF CURSOR
;WIDTH EQU $9005 ;CALCULATED WIDTH OF CURSOR
;
;FPASS EQU $9006 ;1ST PASS FLAG (1),RESET BY VOS
;NROW EQU $9007 ;NUMBER OF ROWS ACCEPTED
;NCOL EQU $9008 ;NUMBER OF COL’S ACCEPTED
;NPIXL EQU $9009 ;NUMBER OF PIXELS ACCEPTED
;NPIXH EQU $900A ;NUMBER OF PIXELS ACCEPTED (HIGH)
;
;TWIN EQU $9008 ;LINE NO. OF TOP OF WINDOW
;BUIN EQU $900C ;LINE NO. OF BOTTOM OF WINDOW
;LUIN EQU $900D ;COL NO. OF LEFT SIDE OF WINDOW
;RUIN EQU $900E ;COL NO. OF RIGHT SIDE OF WINDOW
;HSKP EQU $900F ;NUMBER OF HORIZ. POINTS TO SKIP
;VSKP EQU $9010 ;NUMBER OF VERT. POINTS TO SKIP
;NFRM EQU $9011 ;NUMBER OF FRAMES TO SKIP
;NIIG EQU $9012 ;NO. OF FRAMES TO INJECT INHIBIT
;NDIF EQU $9013 ;DIFF. BETWEEN NFRM AND NIIG
;
;TMFLG EQU $9014 ;TERMINATE FLAG
;CRDSP EQU $9015 ;CURSOR BEING DISPLAYED FLAG
;SCNFG EQU $9016 ;TV DISPLAY MODE FLAG
;IIGON EQU $9017 ;FLAG TO INDICATE IIG TURN ON
;MEMFL EQU $9018 ;MEM. FILLED FLAG (NEW DATA)
;ERRCD EQU $9019 ;ERROR CODE (NORMALLY 0)
;
;TEMPA EQU $901A ;TEMP. STORAGE ‘ACC’
;TEMPX EQU $901B ;TEMP. STORAGE ‘X’
;TEMPLY EQU $901C ;TEMP. STORAGE ‘Y’
;
;CAMIO EQU $8FF0 ;MAIN CONTROL PORT
;CMSKP EQU $8FF1 ;SKIP PARAMETERS
;VCNTL EQU $8FF8 ;VERTICAL SYNC COUNTER LOW
;VCNTH EQU $8FF9 ;VERTICAL SYNC COUNTER HI
;CMACR EQU $8FFB ;AUX CONTROL REG
;CMPCR EQU $8FFC ;PERIPHERAL CONTROL REG
;CMIFR EQU $8FFD ;INTERRUPT FLAG REG

F21
; INITIALIZE THE VISION SYSTEM
53 CMIER EQU $8FE8 ; INTERRUPT ENABLE REG
54 ; ROW, COL PARAMETERS (SAME AS WINDOW PARAMETERS)
55 FROW EQU $8FE7 ; FIRST ROW TO ACCEPT
56 LROW EQU $8FE6 ; LAST ROW TO ACCEPT
57 FCOL EQU $8FE5 ; FIRST COL TO ACCEPT
58 LCOL EQU $8FE4 ; LAST COL TO ACCEPT
59 ;
60 ; DIGITAL TO ANALOG CONVERTERS
61 DACA EQU $8FE3 ; D/A 'A'
62 DACB EQU $8FE2 ; D/A 'B'

; FILE NAME IS NVOS.OBJ0
63 ORG $7D90
64 ; THIS PROGRAM SYNCHRONIZES DATA ANALYSIS WITH DATA
65 ; COLLECTION BY THE IRQ ROUTINE.
66 ; AFTER INITIALIZATION THE PROGRAM WAITS FOR NEW DATA
67 ; (MEMFL=1, SET BY THE IRQ ROUTINE) THEN DISPLAYS
68 ; THE MESSAGE 'FRAME OF DATA GATHERED' AND THEN RETURNS
69 ;
70 VOS JSR INTLZ ; INITIALIZE THE VISION SYSTEM
71 LDA $C0BD ; DEACTIVATE BANK SWITCHED MEMORY
72 START JSR STTCYC ; START THE INTERRUPT CYCLE
73 MNLP CLI ; ENABLE INTERRUPTS
74 MN01 LDA MEMFL ; WAIT FOR NEW DATA TO FILL MEMORY
75 BEQ MN01
76 ;
77 JSR STPCYC ; STOP IRQ CYCLE, PREPARE TO RETURN
78 RTS
79 ******************************************
80 ; SUBROUTINE 'INTLZ'
81 ; THIS ROUTINE INITIALIZES THE VISION SYSTEM.
82 ;
83 INTLZ SEI ; INIT THE 6502
84 CLD
85 ; INITIALIZE I/O PORTS OF VCI
86 LDA #$FF ; DDR'S
87 STA $8FF3 ; DDRA
88 LDA #$3F
89 STA $8FF2 ; DDRB
90 LDA #$3F ; PRESET SWITCHES
91 STA CAMIO
92 LDA FPASS ; ONLY PROMPT USER FOR PARAM
93 BEQ CONT ; ON FIRST FRAME GATHERED
94 JSR GTPRM ; GET CAM. CONTROL PRMTRS
95 CONT RTS
96 ******************************************
97 ; SUBROUTINE 'STTCYC' - STARTS THE INTERRUPT CYCLE OF THE VCI.
98 ;
99 STTCYC LDA NIIG ; SHOULD IIGON BE ON?
100 STA IIGON ; STORE IF YES
101 LDA #$00
102 STA MEMFL ; INIT. ALL FLAGS
VISION-OPERATING-SYSTEM 09-05-1989 002 PAGE 3

103  STA  ERRCD
104  STA  SCNFG
105  STA  GOFLAG
106  STA  TMFLG
107  LDA  #$00
108  STA  FPASS  ;CLEAR FIRST PASS FLAG
109 ORA CAMIO  ;SET FOR COMPUTER ACCESS
110  STA  CAMIO
111  ;
112  LDA  #$INTRT  ;INIT. IRQ VECTOR
113  STA  #$3FE
114  LDA  #$<INTRT
115  STA  #$3FF
116  LDA  #$B0  ;INIT. IER
117  STA  CMIER
118  ;
119  LDA  #$00  ;PCR
120  STA  CMPCR  ;EOF IRQ(CB1)
121  LDA  #$20  ;ACR
122  STA  CMACR
123  ;
124  LDA  #$00  ;START COUNTER
125  STA  VCNTL
126  STA  VCNTH
127  RTS
128  ****************************************
129  iSUBROUTINE 'STPCyc'- STOPS THE INTERRUPT CYCLE OF THE VCI.
130  ;
131  STPCyc  SEI  ;DISABLE IROs
132  ;
133  LDA  #$7F  ;DISABLE VIA IROs
134  STA  CMIER
135  ;
136  LDA  #$65  ;RESET IRQ VECTORS TO DEFAULT VALUES
137  STA  #$3FE
138  LDA  #$FF
139  STA  #$3FF
140  ;
141  LDA  #$01  ;SET FOR MEM. ACCESS
142  ORA  CAMIO
143  STA  CAMIO
144  RTS
145  ****************************************
146  iTHIS SUBROUTINE GETS THE INITIAL PRMTRS NEEDED TO RUN THE
147  iINTERFACE, CALCULATES SOME VALUES AND LOADS THE INTERFACE
148  iWITH THE PROPER INFORMATION.
149  ;
150  GTPRM  LDA  #$020
151  STA  FROWS
152  STA  TWIN  ;FIRST ROW
153  STA  #$DFE5  ;STORE TWIN IN BANK SWITCHED
154 LDA $9021
155 STA FCOL
156 STA LWIN ;FIRST COL
157 STA $DFE7 ;STORE LWIN IN BANK SWITCHED
158
159 LDA $9022
160 STA LROW
161 STA BWIN ;LAST ROW
162 STA $DFE6 ;STORE BWIN IN BANK SWITCHED
163 LDA $9023
164 STA LCOL
165 STA RWIN ;LAST COL
166 STA $DFE8 ;STORE RWIN IN BANK SWITCHED
167
168 LDA $9024 ;SKIP PARAMETERS
169 STA HSKP
170 STA $DFE9 ;STORE HSKP IN BANK SWITCHED
171 LDA $9025
172 STA VSKP
173 STA $DFEA ;STORE VSKP IN BANK SWITCHED
174 LDA HSKP
175 ASL A
176 ASL A
177 ASL A
178 ASL A
179ORA VSKP
180 STA CMSKP
181
182 LDX TWIN ;CALCULATE NROW
183 LDA BWIN
184 LDY VSKP
185 JSR CALC
186 STX NROW
187 STX $DFE1 ;STORE NROW IN BANK SWITCHED
188
189 LDX LWIN ;CALCULATE NCOL
190 LDA RWIN
191 LDY HSKP
192 JSR CALC
193 STX NCOL
194 STX $DFE2 ;STORE NCOL IN BANK SWITCHED
195
196 JSR CALCPX ; CALCULATE NPIX
197 LDA $9026
198 STA NFRM ;NUM FRAMES TO SKIP
199 LDA $9027
200 STA NIIG ;NUM IIG'S
201 LDA NFRM
202 CLC
203 ADC #1
204 SEC
SBC NIIG
STA NDIF

; NDIF = NUM FRAMES TIL NEXT VSYNCH CTR INTERRUPT
RTS ; BACK (NO QUESTIONS)

; SUBROUTINE 'CALC'
; THIS SUBRTN TAKES 1ST COUNT IN X, LAST COUNT IN ACCUM.,
; AND THE SKIP VALUE IN Y & CALC. THE TOTAL NUMBER OF
; ACCEPTED ELEMENTS (ROWS, OR COLS) BY THE INTRFC.
; RESULT IS LEFT IN X

CALC STY TEMPY ; STORE VALUES
STX TEMPX

CLD
SEC

SBC TEMPX ; SUBTR 1ST VAL FROM LAST VAL
BCC CA04 ; FIRST > LAST?
ADC #$00 ; NO-MAKE DIFFERENCE INCLUSIVE

INC TEMPY ; INC SKIP VAL TO INC VAL
LDX #$00 ; ZERO CNTR
CA01 SBC TEMPY ; SUBTRACT INC VAL
INX ; INC CNTR

BCS CA01 ; BORROW REQUIRED YET?

RTS ; RETURN WITH VALUE IN X

CA04 LDX #$00 ; ERROR-NO ELEMENTS ACCEPTED

RTS

******************************************************************

; THIS SUBRTN MULTIPLIES THE TWO VALUES NROW AND NCOL TO GET
; THE TWO WORD RESULT NPIX.

CALCPX LDX NROW
BEQ CL02 ; NROW=0?
LDA #$00 ; LEAST SIG WORD OF RESULT
LDY #$00 ; MOST SIG WORD OF RESULT

CLC

CL00 ADC NCOL
BCC CL01 ; LEAST SIG WORD OVERFLOW?
INY 7ED8:18

CL01 DEX ; DONE NROW'S YET?
BNE CL00 ; NO

STA NPIXL ; YES-STORE L.S.WORD
STA $DFE3 ; STORE M.S.WORD
STY NPIXH ; STORE NPIXL IN BANK SWITCHED MEMORY

RTS
256 ;
257  CL02  STX  NPIXL  ;ZERO PIXELS ACCEPTED
258  STX  NPIXH
259  RTS
260  ******************************************************
261 ;
262  ;THIS ROUTINE SERVICES 3 TYPES OF INTERRUPTS: VERTICAL SYNCH
263  ;COUNT (VCNTR-PB6), VSYNCH (C82), AND END-OF-FRAME (EOF-CB1).
264  ;WHENEVER THE VCNTR INTERRUPTS, THE VSYNCH ALSO WILL BE SET,
265  ;THUS BOTH MUST BE RESET IN THE ROUTINE. THE VCNTR INTERRUPT
266  ;CAUSES EITHER THE I1G TO BE SET OR THE INTERFACE SET TO
267  ;ACCEPT FRAME DATA. THE I1G FLAG CONTROLS THIS.
268  ;THE VSYNCH INTERRUPT CHOOSES WHAT SORT OF INFORMATION
269  ;SHOULD BE DISPLAYED ON THE TV MONITOR (SCAN LINE OR
270  ;CURSOR). THE SCNFG CONTROLS THIS. THE END OF FRAME
271  ;INTERRUPT SETS THE MEMFL FLAG IF THAT FRAME HAD BEEN
272  ;ACCEPTED AND SETS THE INTERFACE FOR COMPUTER ACCESS
273  ;(COM/CAM). IT ALSO CHECKS FOR KYBOARD INPUT.
274 ;
275  INTRT  PHA  ;SAVE ACC.
276  LDA  #$10
277  BIT  CMIFR  ;WHICH INTRPT?
278  BMI  INT00  ;BRNCH IF THIS DEV.
279  JMP  INT40  ;JMP TO OTHER IRQ DEV.
280 ;
281  INT00  BNE  INT10  ;END OF FRAME IRQ
282 ;
283  LDA  #$20
284  BIT  CMIFR
285  BNE  INT20  ;VSYNC COUNTER IRQ
286 ;
287  LDA  #$08
288  BIT  CMIFR
289  BNE  INT30  ;VSYNCH IRQ
290 ;
291  LDA  #$02  ;NO IRQ RECOGNIZED
292  STA  ERRCD  ;SIGNAL ERROR
293  LDA  #$00  ;CLEAR IRQ
294  STA  CMIFR
295  PLA
296  RTI
297 ;
298  END OF FRAME INTERRUPT.
299  INT10  LDA  CAM10  ;CLEAR IRQ
300  AND  #$01  ;CHECK COM/CAM SWITCH
301  BNE  INT11  ;SWITCH NOT SET-RETURN
302 ;
303  LDA  #$01
304  STA  MEMFL  ;SET MEM. FILLED FLAG
305  ORA  CAM10  ;RESET SWITCH
306  STA  CAM10  ;RESET SWITCH
307 INT11 TXA ;SAVE X
308 PHA
309 TYA ;SAVE Y
310 PHA
311 PLA
312 TAY ;REPLACE Y
313 PLA
314 TAX ;REPLACE X
315 PLA ;REPLACE ACC
316 RTI
317 ;
318 ;VCNTR IRQ. EITHER TURN ON IIG OR SET INTERFACE TO INPUT
319 ;THIS FRAME.
320 ;
321 INT20 LDA IIGON ;TURN ON IIG?
322 BNE INT23 ;YES-TURN IT ON
323 ;
324 LDA #$FE ;NO-ACCEPT THIS FRAME
325 AND CAMIO ;RESET CAM/CAM SWITCH
326 STA CAMIO ;(RESET IRQ)
327 LDA NIIG ;SET IIG FLAG
328 STA IIGON ;IIGON>0 IF NIIG>0
329 LDA #$02 ;TURN OFF IIG
330 ORA CAMIO
331 STA CAMIO
332 ;
333 LDA NDIF ;IRQ IN NDIF FRAMES
334 SEC ;SUBTRACT 1
335 SBC #$01
336 STA VCNTH ;NDIF FRAMES
337 LDA #$00 ;(AND RESET IRQ)
338 STA VCNTH
339 ;
340 JSR LDSCN ;SET PARAMETERS
341 PLA
342 RTI
343 ;
344 INT23 LDA #$FD ;TURN ON IIG
345 AND CAMIO
346 STA CAMIO
347 ;
348 LDA #$00 ;RESET IIG FLAG
349 STA IIGON
350 ;
351 LDA NIIG ;IRQ IN NIIG FRAMES
352 SEC ;SUBTRACT 1
353 SBC #$01
354 STA VCNTH
355 LDA #$00 ;(AND CLEAR IRQ)
356 STA VCNTH
357 ;VSYNC INTRPT-CHOOSE WHAT INFO TO DISPLAY ON TV
358 INT30  LDA  CAM10  ;RESET IRQ
359    LDA  SCNFG  ;GET FLAG
360    BEQ  INT34  ;ALTERNATE DISPLAY
361    BMI  INT35  ;DISPLAY CURSOR
362    PLA    ;MUST BE POS.-DISPLAY
363    RTI    ;SCAN LINES(ALREADY SET)
364  |
365 INT34  LDA  CRDSP  ;IS CURSOR BEING DISPLAYED
366    BEQ  INT36  ;NO-DISPLAY CURSOR
367  |
368    JSR  LDSCN  ;DISPLAY SCAN LINES
369    PLA
370    RTI
371  |
372 INT35  LDA  CRDSP  ;IS CURSOR DISPLAY SET?
373    BEQ  INT36  ;NO-SET IT
374    PLA
375    RTI
376  |
377 INT36  JSR  LDWID  ;TO DISPLAY CURSOR
378    PLA
379    RTI
380  |
381 ;THIS SECTION FOR OTHER IRQ’S
382 INT40  LDA  #01  ;‘NO OTHER IRQ DEV’
383    STA  ERRCD
384    PLA
385    PLP    ;SET IRQ DISABLE
386    SEI
387    PHP
388    RTI
389  *********************************************************
390  |
391 ;THIS SUBRTN SETS THE INTERFACE TO DISPLAY THE SCAN LINES
392 ;(FOR DISPLAY OR FOR DATA ACQUISITION).
393  |
394 LDSCN  LDA  TWIN  ;LINES
395    STA  FROW
396    LDA  BWIN
397    STA  LRW
398    LDA  LWIN
399    STA  FCOL
400    LDA  RWIN
401    STA  LCOL
402  |
403    LDA  HSKP
404    ASL  A
405    ASL  A
406    ASL  A
407    ASL  A
408    ORA  VSKP
409 STA CMSKP
410 ;
411 LDA #$00
412 STA CRDSP ;RESET CURSOR DISPLAY FLAG
413 ;
414 RTS
415 ;
416 ****************************************
417 ;
418 ;THIS SUBRTN SETS THE INTERFACE TO DISPLAY THE CURSOR.
419 ;ON THE MONITOR.
420 LDWID LDA #$00 ;DISPLAY CURSOR
421 STA CMSKP ;SET SKIP TO ZERO
422 LDA LCRSR ;SET LEFT EDGE
423 STA FCOL
424 LDA RCRSR ;SET RIGHT EDGE
425 STA LCOL
426 LDA TCRSR ;SET TOP EDGE
427 STA FROW
428 LDA BCRSR ;SET BOTTOM EDGE
429 STA LROW
430 ;
431 LDA #$01 ;SET CURSOR DISPLAY
432 STA CRDSP ;FLAG
433 RTS

ASSEMBLY: NO ERRORS
EDGE-LINKING-PROCEDURE  09-06-1989  005 PAGE 1

2 *********************************************************************************************
3 ;
4 ; EDGE LINKING PROGRAM   MM  10/28/88
5 ;
6 *********************************************************************************************
7 ; VARIABLES FOR EDGE LINKING PROGRAM
8 BASE   EQU $EB   ;2 BYTE BASE ADDRESS
9 LADX   EQU $ED   ;2 BYTE BASE ADDRESS FOR X COORDINATE OF EDGE
10 LADY   EQU $FB   ;2 BYTE BASE ADDRESS FOR Y COORDINATE OF EDGE
11 ;
12 ; VARIABLES SHARED WITH CONTR2
13 ;
14 NROW EQU $DFE1 ;NO. OF ROWS ACCEPTED
15 NCOL EQU $DFE2 ;NO. OF COLUMNS ACCEPTED
16 NPIXL EQU $DFE3 ;NO. OF PIXELS ACCEPTED LOW
17 NPIXH EQU $DFE4 ;NO. OF PIXELS ACCEPTED HIGH
18 ERROR EQU $902E ;ERROR FLAG 0=O.K.
19 TREDG EQU $9029 ;THRESHOLD FOR EDGE DETECTION
20 COX EQU $E002 ;X COORDINATE OF MATRIX
21 COY EQU $E003 ;Y COORDINATE OF MATRIX
22 I EQU $E004 ;COUNTER (TEMP. X COORDINATE)
23 J EQU $E005 ;COUNTER (TEMP. Y COORDINATE)
24 X2 EQU $E007 ;
25 Y2 EQU $E009 ;
26 GRAD EQU $E00A ;GRADIENT VALUE OF ROBERTS GRADIENT
27 DIVDL EQU $E00F ;DIVIDEND LOW BYTE
28 DIVDH EQU $E010 ;DIVIDEND HIGH BYTE
29 DIVIS EQU $E011 ;DIVISOR
30 DIRE EQU $E012 ;RESULT OF DIVISION
31 PX EQU $E015 ;X POINTER
32 PY EQU $E016 ;Y POINTER
33 DIR EQU $E017 ;DIRECTION OF EDGE FOLLOWING
34 BXLST EQU $FA00 ;BEGINNING POSITION OF X EDGE LIST
35 BYLST EQU $FB00 ;BEGINNING POSITION OF Y EDGE LIST
36 DRFLAG EQU $E01A ;DIRECTION FLAG
37 LDIR EQU $E01B ;DIRECTION USED TO FIND LAST EDGE PT.
38 NEDPT EQU $E01C ;NO. OF EDGE POINTS
39 TEMPY EQU $E01D
40 ;
41 ;PROGRAM LOCATIONS
42 ;
43 ROCED EQU $77C0 ;PROGRAM FOR ROBERTS CROSS EDGE GRADIENT
44 MULTPL EQU $77A0 ;MULTIPLICATION PROGRAM
45 DIVID EQU $7770 ;DIVISION PROGRAM
46 ;DEFINE START POSITION OF THIS PROGRAM
47 ;
ECT FILE NAME IS LNKLST.OBJ0
48 ORG $6800
49 ;
50 *********************************************************************************************
51 ;BEGIN PROGRAM TO FIND LINKED LIST
52 *********************************************************************************************
53 LDA $C082
EDGE-LINKING-PROCEDURE 09-06-1989 005 PAGE 2

54  LDX #MSG3-MSG1
55  JSR OTMSG1
56  ;OPEN SECOND BANK OF BANK SWITCHED MEMORY
57  BEGN  LDA $C083
58  LDA $C083
59  ;INITIALIZE VARIABLES
60  LDA #0
61  STA PX
62  STA PY
63  STA DRFLAG ;SET DIRECTION FLAG TO ZERT
64  LDA NCOL
65  STA Y2 ;Y2=NCOL-1
66  DEC Y2
67  LDA NROW
68  STA X2
69  DEC X2 ;X2=NROW-1
70  ;BEGIN LOOKING FOR BEGINNING OF EDGE
71  LNK1  LDA #0
72  STA PY
73  LNK2  LDA PX
74  STA COX
75  LDA PY
76  STA COY
77  JSR ROCED ;GET GRADIENT VALUE
78  LDA GRAD
79  CMP TREDG
80  BCC LNK3 ;IF GRADIENT < THRESHOLD CHECK NEXT POINT
81  ;OTHERWISE SEE IF THIS PT. HAS A NEIGHBOR W/ GRAD)>=TREDG
82  LDA #3
83  STA DIR
84  LNK5  JSR PIXPS3 ;GET NEXT PIXEL FOR SEARCH FOR
85  JSR CHKPOS ;IS THIS PIXEL POSITION VALID?
86  LDA ERROR
87  CMP #1
88  BEQ LNK6 ;IF THIS IS NOT A VALID PIXEL, TRY NEXT SURROUND
89  JSR ROCED ;GET ROBERTS GRADIENT VALUE FOR THIS POSITION
90  LDA GRAD
91  CMP TREDG
92  BCS EDGFOL ;IF ROBERTS GRAD >= TO TREDG BEGIN EDGE FOLLOW
93  LNK6  INC DIR
94  LDA DIR
95  CMP #7
96  BNE LNK5 ;IF 3<=POSITION <7, CHECK NEXT SURROUND POINT
97  LNK3  INC PY
98  LDA PY
99  CMP Y2
100  BCC LNK2 ;CHECK POINT IN NEXT COLUMN
101  INC PX
102  LDA PX
103  CMP X2
104  BCC LNK1 ;IF YOU HAVE NOT CHECKED LAST ROW, CHECK NEXT R
105  LDA #0
106  STA NEDPT ;SET NUMBER OF EDGE PTS. TO ZERO
EDGE-LINKING-PROCEDURE

09-06-1989

005 PAGE 3

107 LDA #$C082 ;IF ALL ROWS HAVE BEEN CHECKED
108 LDX #0 ;THEN SOMETHING IS WRONG
109 JSR OTMSG1 ;PRINT OBJECT NOT FOUND
110 LDA #1
111 STA ERROR ;LOAD ERROR FLAG WITH 1
112 RTS ;RETURN TO MAIN PROGRAM DRIVER
113
114 SECTION OF PROGRAM TO PERFORM EDGE FOLLOWING
115
116 EDGFOL LDA #$C082
117 LDX #$MSG4-MSG1 ;PRINT FOLLOW EDGE
118 JSR OTMSG1
119 LDA #$C083
120 LDA #$C083
121 LDY #0
122 LDA #$FA ;SET UP BASE ADDRESSES TO STORE
123 STA LADX+1 ;X EDGE LIST FROM $FA00 TO $FAFF
124 LDA #$FB ;AND Y EDGE-LIST FROM $FB00 TO $FBFF
125 STA LADY+1
126 LDA #0
127 STA LADX
128 STA LADY
129 LDA PX
130 STA (LADX),Y ;STORE PX AS FIRST PT. IN EDGE LIST
131 LDA PY
132 STA (LADY),Y ;STORE PY AS FIRST Y COORDINATE IN EDGE LIST
133 EDLN1 INY
134 TYA
135 CMP #$FF
136 BNE EDLN9 ;MOVE DATA TO SAFE LOCATION
137 JSR MOVER ;DEACTIVATE BANK SWITCHED MEMORY
138 STA NEDPT
139 LDA #$C082
140 LDX #$MSG2-MSG1 ;PRINT MORE THAN 255 EDGE POINTS
141 JSR OTMSG1 ;REACTIVATE BANK SWITCHED MEMORY
142 LDA #$C083
143 LDA #$C083
144 LDA #1
145 STA ERROR ;RETURN TO PROGRAM DRIVER
146 RTS
147 EDLN9 LDA CX
148 STA PX ;MOVE TO NEXT POINT IN EDGE LIST
149 STA (LADX),Y ;STORE NEW VALUE IN EDGE LIST (X COORD.)
150 LDA COY
151 STA PY ;MOVE TO NEXT PT. IN EDGE LIST (Y COORD.)
152 STA (LADY),Y ;STORE Y COORD. OF NEW VALUE IN EDGE LIST
153 LDA DIR
154 STA LDIR ;SAVE THE DIRECTION USED TO FIND THE LAST EDGE
155 TYA
156 CMP #5 ;IF THERE ARE LESS THAN 5 PTS. IN THE EDGE LIST
157 BCS BFIN1 ;THE EDGE LIST IS NOT COMPLETE
158 JMP FIN
159 ;
EDGE-LINKING-PROCEDURE          09-06-1989          005 PAGE 4

160 ;CHECK FOR END OF SHAPE
161 ;THE END OF THE SHAPE IS DETECTED IF BOTH THE X + Y COORDINATES
162 ;OF THE FIRST POINT STORED ARE WITHIN 1 OF THE LAST POINT STORED
163 ;THIS SECTION CHECKS IF IT IS WITHIN A DISTANCE OF 1
164 ;
165 BF1N1  LDA  BXLST  ;LOAD FIRST X COORDINATE
166  CMP  COX  ;COMPARE TO THE LAST
167  BEQ  FINY1  ;IF EQUAL CHECK Y
168  BCC  FINX  ;IF FIRST X COORD. LAST THEN SUBT. LAST FROM F:
169  CLD
170  SEC
171  SBC  COX
172  CMP  #2
173  BEQ  FING1
174  CMP  #1 ;IF THIS DIFFERENCE =1 THEN
175  BEQ  FINY1 ;CHECK Y COORDINATES
176  JMP  FIN ;IF NOT THEN THIS POINT IS NOT THE END OF THE
177  FINX  CLD
178  SEC ;IF LAST X COORD. > FIRST X
179  LDA  COX ;COORD. SUBT. THE FIRST FROM LAST
180  SBC  BXLST
181  CMP  #2
182  BEQ  FING2 ;SEE IF IT IS WITHIN 2 OF JOINING
183  CMP  #1
184  BEQ  FINY1 ;IF >1 THEN NOT END OF SHAPE
185  JMP  FIN
186  FINY1  LDA  BYLST ;CHECK Y CANDIDATES THE SAME AS THE X CANDIDATE
187  CMP  COY
188  BEQ  FINY3 ;IF THE Y DIFFERENCE=0 THEN END OF SHAPE
189  BCC  FINY2
190  CLD
191  SEC
192  SBC  COY
193  CMP  #2
194  BEQ  FING3
195  CMP  #1
196  BEQ  FINY3 ;END OF SHAPE IF DIFFERENCE=1
197  JMP  FIN ;OTHERWISE NOT END OF SHAPE
198  FINY2  CLD
199  SEC
200  LDA  COY
201  SBC  BYLST
202  CMP  #2
203  BEQ  FING4
204  CMP  #1
205  BNE  FIN
206  FINY3  LDA  #0
207  STA  ERROR ;RESET ERROR FLAG INDICATE SUCCESSFUL LINKING F
208  INY
209  STY  NEDPT ;STORE THE NO. OF EDGE PTS IN EDGE LIST
210  RTS ;RETURN TO PROGRAM DRIVER
211  FING1  LDA  BYLST ;IF THE X COORDINATE DIFFERS
212  CMP  COY ;BY 2 AND THE DIFFERENCE BETWEEN
EDGE-LINKING-PROCEDURE

213  BNE  FIN  ;THE FIRST AND LAST POINT IS 0 THEN
214  INY  ;DECLARE THE POINT THE LAST POINT
215  LDX  PX  ;IN THE EDGE LIST
216  DEX
217  TXA
218  STA  (LADX),Y
219  LDA  PY
220  STA  (LADY),Y
221  JMP  FINY3
222  FING2  LDA  BYLST
223  CMP  COY
224  BNE  FIN
225  INY
226  LDX  PX
227  INX
228  TXA
229  STA  (LADX),Y
230  LDA  PY
231  STA  (LADY),Y
232  JMP  FINY3
233  FING3  LDA  BXLST  ;IF THE Y COORDINATES
234  CMP  COX  ;DIFFER BY 2 AND THE X COORDINATES
235  BNE  FIN  ;ARE EQUAL THEN YOU ARE BACK TO
236  INY  ;THE BEGINNING STORE THE GAP POINT AND END
237  LDX  PY
238  DEX
239  TXA
240  STA  (LADY),Y
241  LDA  PX
242  STA  (LADX),Y
243  JMP  FINY3
244  FING4  LDA  BXLST
245  CMP  COX
246  BNE  FIN
247  INY
248  LDX  PY
249  INX
250  TXA
251  STA  (LADY),Y
252  LDA  PX
253  STA  (LADX),Y
254  JMP  FINY3
255  FIN  JSR  PIXPS3  ;GET NEXT PIXEL POSITION
256  JSR  CHKPOS  ;IS THE NEWEST PIXEL POSITION VALID?
257  LDA  ERROR
258  CMP  #1
259  BEQ  RN1  ;IF IT IS NOT VALID THEN GET NEXT COORDINATES
260  TYA
261  PHA  ;STORE STATUS OF Y REGISTER
262  JSR  ROCED  ;GET ROBERTS CROSS EDGE GRADIENT
263  PLA
264  TAY  ;RETRIEVE STATUS OF Y REGISTER
265  LDA  GRAD
EDGE-LINKING-PROCEDURE
09-06-1989 005 PAGE 6

266   CMP  TREDG  ;IF TREDG < ROCED
267   BCC  RN1
268   JMP  EDLN1
269   ;
270   RN1  LDA  #1
271   CMP  DRFLAG  ;DID EDGE FOLLOWING BLOCK LOOK LEFT (DRFLAG=0)  
272   BNE  RN1A  ;(DRFLAG=1) LAST TIME?
273   JMP  LN1
274   RN1A  STA  DRFLAG  ;IF LEFT THEN LOOK RIGHT THIS TIME AND VICE VERSA
275   LDA  DIR
276   CLC
277   ADC  #7  ;DETERMINE THE NEXT DIRECTION
278   AND  #7
279   STA  DIR
280   JSR  PIXPS3  ;GET NEXT PIXEL POSITION
281   JSR  CHKPOS  ;IS THE NEWEST PIXEL POSITION VALID?
282   LDA  ERROR
283   CMP  #1
284   BEQ  RN2
285   TYA
286   PHA  ;STORE STATUS OF Y REGISTER
287   JSR  ROCED  ;GET ROBERTS CROSS EDGE GRADIENT
288   PLA
289   TAY  ;RETRIEVE STATUS OF Y REGISTER
290   LDA  GRAD
291   CMP  TREDG  ;IS TREDG < ROCED
292   BCC  RN2
293   JMP  EDLN1
294   ;
295   RN2  LDA  DIR  ;GET LAST DIRECTION
296   CLC
297   ADC  #2  ;UPDATE DIRECTION
298   AND  #7
299   STA  DIR
300   JSR  PIXPS3  ;GET NEXT PIXEL POSITION
301   JSR  CHKPOS  ;IS THE NEWEST PIXEL POSITION VALID?
302   LDA  ERROR
303   CMP  #1
304   BEQ  RN3
305   ;IF IT IS NOT VALID THEN GET NEXT POSITION
306   TYA
307   PHA  ;STORE STATUS OF Y REGISTER
308   JSR  ROCED  ;GET ROBERTS GRADIENT
309   PLA
310   TAY  ;RETRIEVE STATUS OF Y REGISTER
311   LDA  GRAD
312   CMP  TREDG
313   BCC  RN3  ;IF ROCED<TREDG TRY NEXT DIRECTION
314   JMP  EDLN1
315   ;
316   RN3  LDA  DIR  ;GET DIRECTION
317   CLC
318   ADC  #5  ;UPDATE DIRECTION
319  AND #7
320  STA DIR
321  JSR PIXPS3 ;GET NEXT PIXEL POSITION
322  JSR CHKPOS ;IS THE NEW POSITION VALID?
323  LDA ERROR
324  CMP #1
325  BEQ RN4 ;IF IT IS NOT THEN GET NEXT PIXEL POSITION
326  TYA
327  PHA ;STORE STATUS OF Y REGISTER
328  JSR R0CED ;GET ROBERTS GRADIENT
329  PLA
330  TAY ;RETRIEVE STATUS OF Y REGISTER
331  LDA GRAD
332  CMP TREDG ;IF TREDG < R0CED
333  BCC RN4
334  JMP EDLN1 ;IF R0CED>TREDG STORE VALUE
335  ;
336  RN4  LDA DIR
337  CLC
338  ADC #4
339  AND #7
340  STA DIR
341  JSR PIXPS3
342  JSR CHKPOS
343  LDA ERROR
344  CMP #1
345  BEQ RN5
346  TYA
347  PHA
348  JSR R0CED
349  PLA
350  TAY
351  LDA GRAD
352  CMP TREDG ;IF TREDG < R0CED
353  BCC RN5
354  JMP EDLN1
355  ;
356  RN5  LDA DIR
357  CLC
358  ADC #3
359  AND #7
360  STA DIR
361  JSR PIXPS3
362  JSR CHKPOS
363  LDA ERROR
364  CMP #1
365  BEQ RN6
366  TYA
367  PHA
368  JSR R0CED
369  PLA
370  TAY
371  LDA GRAD
DURING EDGE FOLLOWING

{EDGE-LINKING-PROCEDURE 09-06-1989 005 PAGE 8

372 CMP TREDG
373 BCC RN6
374 JMP EDLN1
375 ;
376 RN6 LDA DIR
377 CLC
378 ADC #6
379 AND #7
380 STA DIR
381 JSR PIXPS3
382 JSR CHKPOS
383 LDA ERROR
384 CMP #1
385 BEQ RN7
386 TYA
387 PHA
388 JSR ROCED
389 PLA
390 TAY
391 LDA GRAD
392 CMP TREDG
393 BCC RN7
394 JMP EDLN1
395 ;
396 RN7 JMP EDLN4 ;SEARCH USING 5X5
397 ;
398 LN1 LDA #0
399 STA DRFLAG
400 ;BEGIN ROUTINE FOR LOOKING LEFT DURING EDGE FOLLOWING
401 LDA DIR
402 CLC
403 ADC #1 ;FIND NEXT DIRECTION
404 AND #7
405 STA DIR
406 JSR PIXPS3 ;GET NEXT PIXEL POSITION
407 JSR CHKPOS ;IS THE NEWEST PIXEL POSITION VALID?
408 LDA ERROR
409 CMP #1
410 BEQ LN2
411 TYA
412 PHA ;STORE STATUS OF Y REGISTER
413 JSR ROCED ;GET ROBERTS GRADIENT
414 PLA
415 TAY ;RETRIEVE STATUS OF Y REGISTER
416 LDA GRAD
417 CMP TREDG ;IS TREDG < ROCED
418 BCC LN2
419 JMP EDLN1
420 ;
421 LN2 LDA DIR
422 CLC
423 ADC #6
424 AND #7}
425 STA DIR
426 JSR PIXPS3 ; GET PIXEL POSITION
427 JSR CHKPOS
428 LDA ERROR
429 CMP #1
430 BEQ LN3
431 TYA
432 PHA
433 JSR ROCED
434 PLA
435 TAY
436 LDA GRAD
437 CMP TREDG
438 BCC LN3
439 JMP EDLN1
440 ;
441 LN3 LDA DIR
442 CLC
443 ADC #3
444 AND #7
445 STA DIR
446 JSR PIXPS3
447 JSR CHKPOS
448 LDA ERROR
449 CMP #1
450 BEQ LN4
451 TYA
452 PHA
453 JSR ROCED
454 PLA
455 TAY
456 LDA GRAD
457 CMP TREDG
458 BCC LN4
459 JMP EDLN1
460 ;
461 LN4 LDA DIR
462 CLC
463 ADC #4
464 AND #7
465 STA DIR
466 JSR PIXPS3
467 JSR CHKPOS
468 LDA ERROR
469 CMP #1
470 BEQ LN5
471 TYA
472 PHA
473 JSR ROCED
474 PLA
475 TAY
476 LDA GRAD
477 CMP TREDG
EDGE-LINKING-PROCEDURE

478 BCC LN5
479 JMP EDLN1
480 ;
481 LN5 LDA DIR
482 CLC
483 ADC #5
484 AND #7
485 STA DIR
486 JSR PIXPS3
487 JSR CHKPOS
488 LDA ERROR
489 CMP #1
490 BEQ LN6
491 TYA
492 PHA
493 JSR ROCED
494 PLA
495 TAY
496 LDA GRAD
497 CMP TREDG
498 BCC LN6
499 JMP EDLN1
500 ;
501 LN6 LDA DIR
502 CLC
503 ADC #2
504 AND #7
505 STA DIR
506 JSR PIXPS3
507 JSR CHKPOS
508 LDA ERROR
509 CMP #1
510 BEQ EDLN4 ;EDGE LIST IS NOT EIGHT CONNECTED
511 TYA
512 PHA
513 JSR ROCED
514 PLA
515 TAY
516 LDA GRAD
517 CMP TREDG
518 BCC EDLN4 ;THE EDGE LIST IS NOT EIGHT CONNECTED
519 JMP EDLN1
520 ;
521 ; SEARCH USING 5X5 MATRIX
522 ;
523 EDLN4 LDA LDIR
524 STA DIR ;DIR=2*LDIR
525 ASL DIR
526 JSR PIXPS5
527 JSR CHKPOS ;IS THE PIXEL POSITION VALID
528 LDA ERROR
529 CMP #1
530 BEQ R51 ;IF NOT GET NEXT VALUE
EDGE-LINKING-PROCEDURE

531  TYA
532  PHA ;STORE STATUS OF Y REGISTER
533  JSR ROCED ;GET ROBERTS GRADIENT VALUE
534  PLA
535  TAY ;RETRIEVE STATUS OF Y REGISTER
536  LDA GRAD
537  CMP TREDG ;IF TREDG<ROCED
538  BCC R51
539  JMP FILPT
540  
541  R51  LDA DIR
542  CLC
543  ADC #15 ;FIND NEXT DIRECTION
544  AND #15
545  STA DIR
546  JSR PIXPS5
547  JSR CHKPOS ;IS THE NEWEST POSITION VALID?
548  LDA ERROR
549  CMP #1
550  BEQ R52
551  TYA
552  PHA
553  JSR ROCED ;GET ROBERTS GRADIENT
554  PLA
555  TAY
556  LDA GRAD
557  CMP TREDG
558  BCC R52
559  JMP FILPT
560  
561  R52  LDA DIR
562  CLC
563  ADC #2 ;FIND NEXT DIRECTION
564  AND #15
565  STA DIR
566  JSR PIXPS5
567  JSR CHKPOS
568  LDA ERROR
569  CMP #1
570  BEQ R53
571  TYA
572  PHA
573  JSR ROCED
574  PLA
575  TAY
576  LDA GRAD
577  CMP TREDG
578  BCC R53
579  JMP FILPT
580  
581  R53  LDA DIR
582  CLC
583  ADC #13
AND #15
STA DIR
JSR PIXPS5
JSR CHKPOS
LDA ERROR
CMP #1
BEQ R54
TYA
PHA
JSR ROCED
PLA
TAY
LDA GRAD
CMP TREDG
BCC R54
JMP FILPT

LDA DIR
CLC
ADC #4
AND #15
STA DIR
JSR PIXPS5
JSR CHKPOS
LDA ERROR
CMP #1
BEQ R55
TYA
PHA
JSR ROCED
PLA
TAY
LDA GRAD
CMP TREDG
BCC R55
JMP FILPT

LDA DIR
CLC
ADC #11
AND #15
STA DIR
JSR PIXPS5
JSR CHKPOS
LDA ERROR
CMP #1
BEQ R56
TYA
PHA
JSR ROCED
PLA
TAY
LDA GRAD
EDGE-LINKING-PROCEDURE

637   CMP   TREDG
638   BCC   R56
639   JMP   FILPT
640   ;
641   R56   LDA   DIR
642   CLC
643   ADC   #6
644   AND   #15
645   STA   DIR
646   JSR   PIXPS5
647   JSR   CHKPOS
648   LDA   ERROR
649   CMP   #1
650   BEQ   R57
651   TYA
652   PHA
653   JSR   ROCED
654   PLA
655   TAY
656   LDA   GRAD
657   CMP   TREDG
658   BCC   R57
659   JMP   FILPT
660   ;
661   R57   LDA   DIR
662   CLC
663   ADC   #9
664   AND   #15
665   STA   DIR
666   JSR   PIXPS5
667   JSR   CHKPOS
668   LDA   ERROR
669   CMP   #1
670   BEQ   R58
671   TYA
672   PHA
673   JSR   ROCED
674   PLA
675   TAY
676   LDA   GRAD
677   CMP   TREDG
678   BCC   R58
679   JMP   FILPT
680   ;
681   R58   LDA   DIR
682   CLC
683   ADC   #8
684   AND   #15
685   STA   DIR
686   JSR   PIXPS5
687   JSR   CHKPOS
688   LDA   ERROR
689   CMP   #1
EDGE-LINKING-PROCEDURE

690  BEQ  R59
691  TYA
692  PHA
693  JSR  ROCED
694  PLA
695  TAY
696  LDA  GRAD
697  CMP  TREDG
698  BCC  R59
699  JMP  FILPT

; R59
700  LDA  DIR
701  CLC
702  ADC  #7
703  AND  #15
704  STA  DIR
705  JSR  PIXPS5
706  JSR  CHKPOS
707  LDA  ERROR
708  CMP  #1
709  BEQ  R510
710  TYA
711  PHA
712  JSR  ROCED
713  PLA
714  TAY
715  LDA  GRAD
716  CMP  TREDG
717  BCC  R510
718  JMP  FILPT ; FILL IN MISSING POINT

; R510
719  LDA  DIR
720  CLC
721  ADC  #10
722  AND  #15
723  STA  DIR
724  JSR  PIXPS5
725  JSR  CHKPOS
726  LDA  ERROR
727  CMP  #1
728  BEQ  R511
729  TYA
730  PHA
731  JSR  ROCED
732  PLA
733  TAY
734  LDA  GRAD
735  CMP  TREDG
736  BCC  R511
737  JMP  FILPT

; R511
738  LDA  DIR
739  CLC
ADC  #5
AND  #15
STA  DIR
JSR  PIXPS5
JSR  CHKPOS
LDA  ERROR
CMP  #1
BEQ  R512
TYA
PHA
JSR  ROCED
PLA
TAY
LDA  GRAD
CMP  TREDG
BCC  R512
JMP  FILPT

; R512
LDA  DIR
CLC
ADC  #12
STA  DIR
JSR  PIXPS5
JSR  CHKPOS
LDA  ERROR
CMP  #1
BEQ  R513
TYA
PHA
JSR  ROCED
PLA
TAY
LDA  GRAD
CMP  TREDG
BCC  R512
JMP  FILPT

; R513
STY  TEMPY  ;STORE THE VALUE OF THE Y REGISTER IN TEMPY
TYA
CMP  #5
BCC  AV4
CLD
SEC
SBC  #4  ;DECREMENT THE Y REGISTER BY 5
TAY
LDA  #0
STA  DIVDL
STA  DIVDH  ;ZERO OUT DIVIDEND
AV1
LDA  <LADX>,Y
STA  COX
LDA  <LADY>,Y
STA  COY
TYA
EDGE-LINKING-PROCEDURE

796       PHA
797       JSR ROCED ;GET GRADIENT
798       PLA
799       TAY
800       LDA DIVDL
801       CLC
802       ADC GRAD
803       BCC AV2
804       INC DIVDH
805       AV2 STA DIVDL
806       INY
807       TYA
808       CMP TEMPY ;SEE IF ALL OF THE LAST 5 GRADIENT VALUES ARE 0
809       BCC AV1
810       BEQ AV1
811       LDY TEMPY
812       LDX #6
813       AV3 LSR DIVDL ;DIVIDE VALUE STORED IN DIVDH & DIVDL BY 64
814       CLC ; (6 LSR S = 5/64 OF AVERAGE THRESHOLD VALUE AT
815       LSR DIVDH
816       BCC AV6
817       LDA DIVDL ;IF THE BIT SHIFTED OUT OF DIVDH IS 1
818       CLC ; THEN ADD 10000000 BINARY TO DIVDL
819       ADC #128
820       STA DIVDL
821       AV6 DEX
822       BNE AV3
823       JMP AV5
824 R8BMOB4BY5 LDA #5 ;IF FEWER THAN 5 PTS. HAVE BEEN STORED IN EDGE
825       STA DIVDL
826       AV5 LDA TREDG
827       CLD
828       SEC
829       SBC DIVDL
830 R83THRESHOLD CMP #15 ;IF THE THRESHOLD -5 IS > THEN 15 THEN PERFORM
831       BEQ REN ;OTHERWISE AN ERROR HAS OCCURRED SAVE VALUES AN
832       BCC REN
833       STA TREDG
834       LDY #0
835       LDA #0
836       RINT STA (LADX),Y ;REINITIALIZE EDGE LIST
837       STA (LADY),Y
838       INY
839       CPY TEMPY
840       BNE RINT
841       LDA #$C082 ;DEACTIVATE BANK SWT MEM
842       LDX #MSG5-MSG1 ;PRINT DECREASE THRESHOLD BY 5
843       JSR OTMSG1
844       JMP BEGN ;BEGIN LINKING AGAIN
845       REN JSR MVORE ;MOVE EDGE LINK DATA TO SAFE LOCATION
846       LDA #1
847       STA ERROR ;SET ERROR FLAG
848       RTS ;RETURN TO MAIN PROGRAM
EDGE-LINKING-PROCEDURE

849 ;
850 **********************************************
851 ;SUBROUTINE TO FILL MISSING POINTS IN EDGE LIST
852 ;
853 FILPT LDA DIR
854 CMP #2 ;IF DIRECTION IS > 1 BRANCH
855 BCS FIL2 ;IF DIRECTION IS 0 OR 1
856 INY
857 LDX PX ;THEN FILL IN PT AT PX-1
858 DEX ;PY-1
859 TXA
860 STA (LADX),Y
861 LDX PY
862 DEX
863 TXA
864 STA (LADY),Y
865 LSR DIR ;DIVIDE DIRECTION BY 2, DROPPING CARRY SCALE D(
866 JMP EDLN1
867 ;
868 FIL2 CMP #3
869 BCS FIL3
870 INY
871 LDX PX ;IF DIRECTION = 2
872 DEX
873 TXA
874 STA (LADX),Y ;THEN SAVE PT. (PX-1,PY)
875 LDX PY
876 TXA
877 STA (LADY),Y
878 LSR DIR
879 JMP EDLN1
880 ;
881 FIL3 CMP #6
882 BCS FIL6
883 INY
884 LDX PX ;IF DIRECTION =3, 4, OR 5
885 DEX ;THEN SAVE PT. AT (PX-1,PY+1) IN
886 TXA ;EDGE LIST
887 STA (LADX),Y
888 LDX PY
889 INX
890 TXA
891 STA (LADY),Y
892 LSR DIR
893 JMP EDLN1
894 ;
895 FIL6 CMP #7
896 BCS FIL7
897 INY
898 LDA PX ;IF DIR=6 THEN
899 STA (LADX),Y
900 LDX PY
901 INX
F47

EDGE-LINKING-PROCEDURE 09-06-1989 005 PAGE 18

902    TXA
903    STA (LADY),Y
904    LSR DIR
905    JMP EDLN1
906 ;
907    FIL7 CMP #10
908    BCS FIL10
909    INY
910    LDX PX
911    INX ;IF DIR=7,8 OR 9 THEN FILL
912    TXA ;IN PT AT PX+1, PY+1
913    STA (LADX),Y
914    LDX PY
915    INX
916    TXA
917    STA (LADY),Y
918    LSR DIR
919    JMP EDLN1
920 ;
921    FIL10 CMP #11
922    BCS FIL11
923    INY
924    LDX PX ;DIR=15 FILL IN PT AT
925    INX ;PX+1, PY
926    TXA
927    STA (LADX),Y
928    LDX PY
929    TXA
930    STA (LADY),Y
931    LSR DIR
932    JMP EDLN1
933 ;
934    FIL11 CMP #14
935    BCS FIL14
936    INY
937    LDX PX
938    INX
939    TXA
940    STA (LADX),Y ;SAVE (PX+1, PY-1)
941    LDX PY
942    DEX
943    TXA
944    STA (LADY),Y
945    LSR DIR
946    JMP EDLN1
947 ;
948    FIL14 CMP #15
949    BCS FIL15
950    INY
951    LDA PX
952    STA (LADX),Y ;SAVE (PX, PY-1)
953    LDX PY
954    DEX
EDGE-LINKING-PROCEDURE

955  TXA
956  STA (LADY),Y
957  LSR DIR
958  JMP EDLN1
959 ;
960  FIL15  INY
961  LDX PX
962  DEX
963  TXA
964  STA (LADY),Y ;SAVE (PX-1,PY-1)
965  LDX PY
966  DEX
967  TXA
968  STA (LADY),Y
969  LSR DIR
970  JMP EDLN1
971 ****************************
972 ;PROGRAM TO FIND THE PIXEL SEARCH POSITION
973 ;USING A 3X3 MATRIX, LOOKING RIGHT FIRST
974 ;PARAMETER PX,PY,DIR
975 ;RESULT COX,COY
976 ;
977  PIXPS3 LDX DIR
978  BEQ PIO
979  DEX
980  BEQ PI1 ;DIRECTION 1
981  DEX
982  BEQ PI2 ;DIRECTION 2
983  DEX
984  BEQ PI3 ;DIRECTION 3
985  DEX
986  BEQ PI4 ;DIRECTION 4
987  DEX
988  BEQ PI5 ;DIRECTION 5
989  DEX
990  BEQ PI6 ;DIRECTION 6
991  DEX
992  BEQ PI7 ;DIRECTION 7
993 ;
994  ******************************************
995 ;
996  PIO LDX PX ;IF DIRECTION WAS 0
997  DEX
998  STX COX ;COX=PX-1
999  LDX PY
1000  DEX
1001  STX COY ;COY=PY-1
1002  RTS
1003 ;
1004  PI1 LDX PX ;IF DIRECTION IS 1
1005  DEX
1006  STX COX ;COX=PX-1
1007  LDX PY
EDGE-LINKING-PROCEDURE 09-06-1989 005 PAGE 20

1008  STX  COY  ;COY=PY
1009   RTS
1010  ;
1011 PI2  LDX  PX  ;IF DIRECTION IS 2
1012   DEX
1013 STX  COX  ;COX=PX-1
1014 LDX  PY
1015 INX
1016 STX  COY  ;COY=PY+1
1017 RTS
1018  ;
1019 PI3  LDX  PX  ;COX=PX
1020 STX  COX
1021 LDX  PY
1022 INX
1023 STX  COY  ;COY=PY+1
1024 RTS
1025  ;
1026 PI4  LDX  PX  ;COY=PY+1
1027 INX
1028 STX  COX  ;COX=PX+1
1029 LDX  PY
1030 INX
1031 STX  COY
1032 RTS
1033  ;
1034 PI5  LDX  PX  ;COX=PX+1
1035 INX
1036 STX  COX
1037 LDX  PY
1038 STX  COY  ;COY=PY
1039 RTS
1040  ;
1041 PI6  LDX  PX  ;COX=PX+1
1042 INX
1043 STX  COX
1044 LDX  PY
1045 DEX
1046 STX  COY  ;COY=PY-1
1047 RTS
1048  ;
1049 PI7  LDX  PX  ;COX=PX
1050 STX  COX
1051 LDX  PY
1052 DEX
1053 STX  COY  ;COY=PY-1
1054 RTS
1055 ************************************************************
1056 ;SUBROUTINE TO FIND THE PIXEL SEARCH POSITION
1057 ;USING THE OUTSIDE OF A 5X5 MATRIX
1058 ;PARAMETER PX,PY,DIR
1059 ;RESULT COX, COY
1060 ;
EDGE-LINKING-PROCEDURE

1061 PIX55 LDX DIR
1062 BEQ P50 ;DIRECTION 0
1063 DEX
1064 BNE DM51 ;DIRECTION 1
1065 JMP P51
1066 DM51 DEX
1067 BNE DM52 ;DIRECTION 2
1068 JMP P52
1069 DM52 DEX
1070 BNE DM53 ;DIRECTION 3
1071 JMP P53
1072 DM53 DEX
1073 BNE DM54 ;DIRECTION 4
1074 JMP P54
1075 DM54 DEX
1076 BNE DM55 ;DIRECTION 5
1077 JMP P55
1078 DM55 DEX
1079 BNE DM56 ;DIRECTION 6
1080 JMP P56
1081 DM56 DEX
1082 BNE DM57 ;DIRECTION 7
1083 JMP P57
1084 DM57 DEX
1085 BNE DM58 ;DIRECTION 8
1086 JMP P58
1087 DM58 DEX
1088 BNE DM59 ;DIRECTION 9
1089 JMP P59
1090 DM59 DEX ;DIRECTION 10
1091 BNE DM510
1092 JMP P510
1093 DM510 DEX ;DIRECTION 11
1094 BNE DM511
1095 JMP P511
1096 DM511 DEX ;DIRECTION 12
1097 BNE DM512
1098 JMP P512
1099 DM512 DEX ;DIRECTION 13
1100 BNE DM513
1101 JMP P513
1102 DM513 DEX ;DIRECTION 14
1103 BNE DM514
1104 JMP P514
1105 DM514 JMP P515 ;DIRECTION 15
1106 ;
1107 P50 LDX PX ;DIRECTION=0
1108 DEX
1109 DEX
1110 STX COX ;COX=PX-2
1111 LDX PY
1112 DEX
1113 DEX
EDGE-LINKING-PROCEDURE

1114  STX  COY  ;COY=PY-2
1115  RTS
1117  LDX  PX  ;DIRECTION=1
1118  DEX
1119  DEX
1120  STX  COX  ;COX=PX-2
1121  LDX  PY
1122  DEX
1123  STX  COY  ;COY=PY-1
1124  RTS
1126  LDX  PX  ;DIRECTION=2
1127  DEX
1128  DEX
1129  STX  COX  ;COX=PX-2
1130  LDX  PY
1131  STX  COY  ;COY=PY
1132  RTS
1134  LDX  PX  ;DIRECTION=3
1135  DEX
1136  DEX
1137  STX  COX  ;COX=PX-2
1138  LDX  PY
1139  INX
1140  STX  COY  ;COY=PY+1
1141  RTS
1143  LDX  PX  ;DIRECTION=4
1144  DEX
1145  DEX
1146  STX  COX  ;COX=PX-2
1147  LDX  PY
1148  INX
1149  INX
1150  STX  COY  ;COY=PY+2
1151  RTS
1153  LDX  PX  ;DIRECTION=5
1154  DEX
1155  STX  COX  ;COX=PX-1
1156  LDX  PY
1157  INX
1158  INX
1159  STX  COY  ;COY=PY+2
1160  RTS
1162  LDX  PX  ;DIRECTION=6
1163  STX  COX  ;COX=PX
1164  LDX  PY
1165  INX
1166  INX
EDGE-LINKING-PROCEDURE 09-06-1989 005 PAGE 23

1147    STX  COY  ;COY=PY+2
1148    RTS
1149
1150    P57   LDX  PX  ;DIRECTION=7
1151    INX
1152    STX  COX  ;COX=PX+1
1153    LDX  PY
1154    INX
1155    INX
1156    STX  COY  ;COY=PY+2
1157    RTS
1158
1159    P58   LDX  PX  ;DIRECTION=8
1160    INX
1161    INX
1162    STX  COX  ;COX=PX+2
1163    LDX  PY
1164    INX
1165    INX
1166    STX  COY  ;COY=PY+2
1167    RTS
1168
1169    P59   LDX  PX  ;DIRECTION=9
1170    INX
1171    INX
1172    STX  COX  ;COX=PX+2
1173    LDX  PY
1174    INX
1175    INX
1176    STX  COY  ;COY=PY+1
1177    RTS
1178
1179    P510  LDX  PX  ;DIRECTION=10
1180    INX
1181    INX
1182    STX  COX  ;COX=PX+2
1183    LDX  PY
1184    INX
1185    INX
1186    STX  COY  ;COY=PY
1187    RTS
1188
1189    P511  LDX  PX  ;DIRECTION=11
1190    INX
1191    INX
1192    STX  COX  ;COX=PX+2
1193    LDX  PY
1194    DEX
1195    STX  COY  ;COY=PY-1
1196    RTS
1197
1198    P512  LDX  PX  ;DIRECTION=12
1199    INX
1200    INX
1201    STX  COX  ;COX=PX+2
1202    LDX  PY
1203    DEX
1204    STX  COY  ;COY=PY+2
1205    LDX  PY
1206    DEX
1207
1208    STX  COX  ;COX=PY+2
1209    DEX
1210    DEX
1211
1212
EDGE-LINKING-PROCEDURE

1220  STX COY ;COY = PY-2
1221  RTS
1222 P513 LDX PX ;DIRECTION 13
1223  INX
1224  STX COX ;COX=PX+1
1225  LDX PY
1226  DEX
1227  DEX
1228  STX COY ;COY=PY-2
1229  RTS
1230 P514 LDX PX ;DIRECTION 14
1231  STX COX ;COX=PX
1232  LDX PY
1233  DEX
1234  DEX
1235  STX COY ;COY=PY-2
1236  RTS
1237 P515 LDX PX ;DIRECTION = 15
1238  DEX
1239  STX COX ;COX=PX-1
1240  LDX PY
1241  DEX
1242  DEX
1243  STX COY ;COY=PY-2
1244  RTS
1245  
1246  ******************************************
1247  ;PROGRAM TO CHECK IF EDGE POINT IS VALID OR IF IT IS OUTSIDE THE IMAGE
1248  CHKPOS LDA #0
1249  STA ERROR ;CLEAR ERROR FLAG
1250  LDA X2
1251  CMP COX ;COMPARE CURRENT X POSITION TO NROWS
1252  BCC CHK4 ;IF CURRENT POSITION LARGER THAN NROWS ERROR
1253  BEQ CHK4
1254  LDA Y2
1255  CMP COY
1256  BCC CHK4 ;IF CURRENT Y POSITION LARGER THAN NCOL ERROR
1257  BEQ CHK4
1258  ;HAS THE PROSPECTIVE EDGE PT BEEN STORED AS ONE OF THE
1259  ;LAST 6 EDGE POINTS OR IS IT ONE OF THE 5 EDGE POINTS NEIGHBORS?
1260  ;IF IT IS THEN IT IS AN INELIGIBLE EDGE PT DOWN FIELD-5 YRD PENALTY
1261  STY TEMPY ;STORE STATUS OF Y REGISTER TEMPORARILY
1262  LDY #0
1263  LDA TEMPY
1264  CMP #6 ;IF FEWER THAN 6 PTS. HAVE BEEN STORED THEN
1265  ;ONLY CHECK THE POINTS THAT HAVE BEEN STORED
1266  BCC CHK1
1267  SEC ;SUBTRACT 6 FROM THE NO. OF PTS
1268  SBC #6 ;THAT WERE STORED IN THE Y REGISTER
1269  TAY ;AND TRANSFER THEM TO THE Y REGISTER
1270  CHK1 LDA (LADX),Y
1271  CMP COX
1272  BEQ CHK2 ;COMPARE THE X COORDINATES OF THE CURRENT
EDGE-LINKING-PROCEDURE

;POINT TO THE X COORDINATES OF THE X-6 TO X-1
1273 TAX
1274 INX
1275 TXA
1276 CMP COX
1277 BEQ CHK2
1278 TAX
1279 DEX
1280 DEX
1281 TXA
1282 CMP COX
1283 BEQ CHK2
1284 JMP CHK3
1285 CHK2 LDA (LADY),Y
1286 CMP COY
1287 BEQ CHK4
1288 TAX
1289 INX
1290 TXA
1291 CMP COY
1292 BEQ CHK4
1293 TAX
1294 DEX
1295 DEX
1296 TXA
1297 CMP COY
1298 BEQ CHK4
1299 CHK3 INY
1300 TYA
1301 CMP TEMPY
1302 BNE CHK1
1303 LDY TEMPY
1304 RTS
1305 CHK4 LDA #1
1306 STA ERROR
1307 LDY TEMPY
1308 RTS
1309 **************************************************
1310 ; PROGRAM TO MOVE EDGE LIST DATA TO A SAFE LOCATION IF AN ERROR OCCURS
1311 ;
1312 MVOER LDA #0
1313 STA LADX
1314 STA LADY
1315 LDA #$FA
1316 STA LADX+1
1317 STA LADY+1
1318 LDA #$F7
1319 STA LADY+1
1320 LD X #2
1321 LDY #0
1322 MV1 LDA (LADX),Y
1323 STA (LADY),Y
1324 INY
1325 BNE MV1
1326  INC LADX  ;MOVE TO NEXT PAGE OF DATA
1327  INC LADY
1328  DEX
1329  BNE MVI  ;IF BOTH PAGES HAVE NOT BEEN TRANSFERRED MOVE
1330  RTS  ;RETURN TO PROGRAM
1331  ;
1332  **************************************************
1333  **************************************************
1334  ;ROUTINE TO OUTPUT A MESSAGE TO THE SCREEN
1335  ;
1336  OUTPUT EQU $FDF0
1337  CR EQU $8D
1338  ;
1339  OTMSG1 LDA MSG1,X
1340  JSR OUTPUT  ;PRINT THAT CHARACTER TO THE SCREEN
1341  INX
1342  CMP #$AE  ;IS IT A PERIOD?
1343  BEQ RETURN  ;IF SO DONE
1344  JMP OTMSG1
1345  RETURN RTS  ;DONE, RETURN
1346  MSG1 DFB CR
1347  ASC 'NO OBJECT DETECTED.'

1348  MSG2 DFB CR
1349  ASC 'MORE THAN 255 EDGE POINTS.'

1350  MSG3 DFB CR
1351  ASC 'ENTER EDGE LINKING PROGRAM.'

1352  MSG4 DFB CR
1353  ASC 'FOLLOW EDGE.'
1354 MSG5  DFB CR
1355 ASC 'THRESHOLD IS DECREASED.'

ASSEMBLY: NO ERRORS
CORNER-DETECTION 09-05-1989 005 PAGE 1

2 **************************************************
3 ; PROGRAM TO DETECT CORNERS FROM A CHAIN CODE
4 ; AND A TRANSFER PROGRAM TO MOVE CODE TO LOWER MEMORY
5 ;
6 ; **************************************************
7 ; VARIABLES
8 ;
9 BASE EQU $EB ;2 BYTE ADDRESS
10 LADX EQU $ED ;2 BYTE ADDRESS
11 LADY EQU $FB ;2 BYTE ADDRESS FOR INDEXED INDIRECT ADDRESSING
12 TOPX EQU $F9 ;TWO BYTE ADDRESS
13 ;
14 ; VARIABLES SHARED WITH PARAMETER LIST
15 ;
16 SEll PIXR EQU $9028 ;PIXEL RANGE (3 MEANS 3 PTS. BEFORE AND AFTER CANI
17 LCTHR EQU $902C ;LOWER CORNER DETECTION THRESHOLD
18 CTHR EQU $9020 ;UPPER CORNER THRESHOLD
19 ;
20 ; VARIABLES SHARED WITH OTHER PROGRAMS
21 ;
22 I EQU $E004 ;USED WITH MULTIPLICATION PROGRAM
23 J EQU $E005 ;DITTO
24 X1 EQU $E006
25 X2 EQU $E007
26 Y1 EQU $E008
27 Y2 EQU $E009
28 DIVDL EQU $E00F ;DIVIDEND LOW BYTE
29 DIVDH EQU $E010 ;DIVIDEND HIGH BYTE
30 DIVIS EQU $E011 ;DIVISOR
31 DIRE EQU $E012 ;RESULT OF DIVISION
32 NEDPT EQU $E01C ;NUMBER OF EDGE POINTS IN LINKED EDGE LIST
33 TEMPY EQU $E01D ;LOCATION OF COORDINATES OF CANDIDATE CORNER PT.
34 TEMPS EQU $E01E ;TEMPORARY POSITION-WILDCARD
35 CORVAL EQU $E01F ;CALCULATED CORNER VALUE
36 ATBB EQU $E020 ;POINT IN ORIGINAL EDGE LIST WHICH CORRESPONDS TO
37 YTWO EQU $E021 ;VALUE OF Y POINTER FOR UPDATED EDGE LIST
38 LCORVL EQU $E022 ;CORNER STRENGTH FOR THE LAST CORNER POINT
39 APPCOR EQU $E023
40 TCORN EQU $E024 ;POSITION IN EDGE-LIST OF POSSIBLE CORNER
41 NOLL EQU $E025 ;NUMBER OF EDGE POINTS IN LIST (NO. OF CORNERS)
42 ;
43 ; PROGRAMS USED
44 ;
45 MULTPL EQU $77A0
46 TRANED EQU $9000
47 **************************************************
48 ; SUBROUTINE TO CALCULATE THE STRENGTH OF A CORNER POINT
49 ; BASED UPON THE DISTANCE TO THE CORNERS
50 ; PARAMETERS CORVAL, TEMPY, NEDPT, PIXR
CORNER-DETECTION
09-05-1989
005 PAGE 2

CORNER DETECTION ROUTINE

FILE NAME IS CORDET.OBJ

ORG $9300
LDA #$083
LDA #$083
LDA NEDPT
STA FPNT
LDA #0
STA NOLL
STA TOPX
STA YTWO
STA LCORVL
LDY PIXR
DEY
STY TEMPY
LDA #$00
STA LADX
STA LADY
LDA #$FA
STA LADX+1
LDA #$FB
TO READ FROM $FA00- AND $FB00- (X&Y COORD. RESPECTIVELY)
STA LADY+1
INC TEMPY
JSR CALCOR
LOA CORVAL
CMP CTHR
INC TEMPY
LOA TEMPY
CMP FPNT
BNE COR3
STA #2
LDY YTWO
RTS
INC TEMPY
;PREPARE TO CHECK NEXT POSITION
LDA TEMPY
CMP FPNT
BNE COR2
BNE COR1
LDY #0
INC TEMPY
;SET A PARITY CHECK FOR THE LINEAR REGRESSION PROGRAM
LDA #$0
CMP NEDPT
BCS COR1
CMP CTHR
BNE COR3
BNE COR4
CMP CTHR
BNE COR2
STA TOPX
LDA CORVAL
JA COR4
STA FPNT
LDA #$F6
STA TOPX+1
LDY YTWO
LDA CORVAL
;STORE THE CALCULATED STRENGTH OF CORNER FOR FIRST LOCATION
105 STA (TOPX), Y
106 INC TOPX+1
107 LDY TEMPY ; STORE VALUE OF X COORD. IN BANK SWITCHED
108 LDA (LADX), Y ; MEMORY AT $F800
109 LDY YTWO
110 STA (TOPX), Y
111 INC TOPX+1 ; STORE VALUE OF Y COORD.
112 LDY TEMPY ; IN BANK SWITCHED MEMORY
113 LDA (LADY), Y ; AT $F900
114 LDY YTWO
115 STA (TOPX), Y
116 INC TOPX+1
117 LDA #0
118 LDY YTWO
119 STA (TOPX), Y ; FIRST POINT HAS BEEN STORED
120 LDA #1 ; SET APPROACHING CORNER FLAG
121 STA APPCOR
122 INC YTWO ; INCREMENT COUNTER FOR UPDATED LINKED LIST
123 INC TEMPY ; INCREMENT COUNTER FOR OLD LINKED LIST
124 LDA TEMPY
125 CMP FPNT
126 BNE COR7
127 LDA #2
128 LDY NEDPT ; STORE 2 AT THE END OF THE CORNER
129 DEY ; CODE AS A PARITY CHECK
130 STA (TOPX), Y
131 RTS ; BACK TO BEGINNING OF LIST, RETURN TO MAIN PROG
132 COR6 INC YTWO ; INCREMENT COUNTER FOR UPDATED LINKED LIST
133 INC TEMPY ; INCREMENT COUNTER FOR OLD LINKED LIST
134 LDA TEMPY
135 CMP FPNT ; COMPARE CURRENT POSITION IN EDGE LIST (TEMпы)
136 BCC COR8 ; TO NO. EDGE Pts., IF THE END HAS BEEN
137 LDA #0
138 STY TEMPY
139 COR8 LDA CORVAL
140 STA LCORVL
141 JSR CALCOR ; GET THE CORNER VALUE FOR THE CANDIDATE CORNER PT
142 LDA #$F6 ; STORE THIS VALUE FOR LATER REFERENCE
143 STA TOPX+1 ; IN $F600-$F6FF OF BANK SWITCHED MEM.
144 LDY YTWO
145 LDA CORVAL
146 STA (TOPX), Y
147 INC TOPX+1 ; STORE X COORDINATE OF EDGE PT. IN
148 INC TOPX+1 ; THE UPDATED LIST (IN BANK SWITCHED MEMORY)
149 INC TOPX+1 ; STORE THE VALUE OF THE Y COORDINATE
150 LDY TEMPY
151 LDA (LADY), Y
152 LDY YTWO
153 STA (TOPX), Y
154 INC TOPX+1 ; STORE A CORNER CODE OF ZERO
155 LDA #0
156 STA (TOPX), Y
157  LDA   APPCOR  ;CHECK APPROACHING CORNER FLAG
158  CMP   #0      ;IF 0 THEN LOOK FOR AN INCREASE IN CORVAL
159  BNE   COR9    ;IF <>0 THEN LOOK FOR CORNER POINT
160  LDA   CORVAL  ;NOT YET NEARING CORNER
161  CMP   CTHR    ;IS CORVAL>8 IF YES THEN SET
162  BCC   COR6    ;FLAG TO INDICATE WE ARE NEARING
163  LDA   #1      ;A CORNER
164  STA   APPCOR
165  JMP   COR6
166  COR9  CMP   #1      ;WE ARE NEAR A CORNER IF SET TO 1
167  BNE   COR11   ;IF NOT SET TO 1 WE HAVE JUST PASSED A CORNER
168  LDA   CORVAL
169  CMP   LCORVL  ;IS THE NEW CORNER VALUE LESS THAN THE LAST ONE?
170  BCS   COR6
171  LOY   TCORN
172  STA   APPCOR
173  JMP   COR6
174  COR10 CMP   CTHR   ;IF NEW PT NOT LESS THAN 8 GOTO COR 12
175  LDA   #2      ;IF IT IS THEN STORE 2 IN APPROACHING CORNER FLAG
176  STA   APPCOR
177  JMP   COR6
178  COR11 LDA   #0      ;IF A CORNER HAS BEEN FOUND AND THE
179  STA   APPCOR   ;CALCULATED CORNER VAL IS BELOW 8
180  LDA   #1
181  LDY   TCORN
182  DEY
183  STA   <TOPX>,Y ;SET APPROACH CORNER FLAG TO 0-NOT NEAR CORNER
184  JMP   COR6
185  COR12 LDA   APPCOR
186  CMP   #2
187  BNE   COR15
188  LDA   CORVAL
189  CMP   LCORVL
190  BCS   COR13
191  BEQ   COR13
192  BCS   COR12
193  COR13 CMP   CTHR   ;IF APPCOR WAS SET TO 2 AND THE CORVAL
194  BCC   COR14   ;CONTINUES TO DECREASE THEN COMPARE CORVAL TO 8
195  JMP   COR6
196  COR14 LDA   #0      ;IF IT IS BELOW 8 SET APPCOR TO 0 (NOT NEAR CORNE:
197  STA   APPCOR   ;IF IT IS NOT BELOW 8 THEN TRY NEXT PT. APPCOR REI
198  LDA   #1
199  LDY   TCORN  ;STORE THE CORNER POINT AT
200  DEY
201  STA   <TOPX>,Y
202  JMP   COR6
203  COR12 LDA   #3      ;IF APPCOR=2 AND THE CORNER VALUE IS
204  STA   APPCOR   ;INCREASING AGAIN THEN WE ARE REALLY NEAR A CORNE.
205  JMP   COR6   ;SET APPCOR TO 1
206  COR15 LDA   CORVAL
207  CMP   LCORVL  ;HAS THE CORNER VALUE BEGUN TO DECREASE YET?
208  BCC   COR16
JMP COR6 ; IF NOT THEN CHECK NEXT POINT
210 COR16 LDA YTWO ; IF IT HAS THEN THE CORNER
211 SEC ; IS LOCATED HALF WAY BETWEEN CURRENT
212 CLD ; EDGE POSITION AND THE POSITION STORED
213 SBC TCORN ; FOR THE TEMPORARY CORNER POSITION
214 STA TCORN

M2THE CURRENT LBRNEBORN ; SO DETERMINE DIFFERENCE OF TWO CORNERS/2 AND THE
216 SEC
217 LDA YTWO
218 SBC TCORN
219 TAY
220 LDA #1 ; STORE NEWLY CALCULATED CORNER
221 STA (TOPX),Y
222 LDA #0
223 STA TCORN
224 LDA CORVAL
225 CMP LCTHR ; IF NEWEST CORNER VALUE IS BELOW 8 THEN CHANGE ST
226 BCC COR17 ; CHANGE STATUS TO 2
227 LDA #2
228 STA APPCOR
229 JMP COR6
230 COR17 LDA #0
231 STA APPCOR
232 JMP COR6

CALCOR LDY TEMPY ; GET X COORD. OF THE CORNER CANDIDATE
234 LDA (LADX),Y
235 STA X1 ; STORE VALUE AS X1
236 ASL X1 ; MULTIPLY THE VALUE BY 2
237 LDA (LADY),Y ; GET Y COORD. OF THE CORNER CANDIDATE
238 STA Y1
239 ASL Y1 ; MULTIPLY VALUE BY 2
240 LDA TEMPY ; CALCULATE THE POSITION OF THE PRIOR PT.
241 CMP PXIR ; IN THE LINKED EDGE LIST
242 BCC CCR1 ; IF VALUE IN Y REGISTER>PXIR THEN SUBTRACT
243 CLD ; PXIR FROM THE Y-REGISTER
244 SEC
245 SBC PXIR ; STORE THIS AS THE POSITION IN LINKED EDGE-LIST
246 TAY ; FOR X1〈PT. BEFORE CAND. EDGE-PT〉
247 JMP CCR2
248 CCR1 LDA PXIR
249 CLD ; O.W. THE POSITION OF THE PT. BEFORE THE
250 SEC ; THE CANDIDATE EDGE PT = NEDPTS-(PIXR-TEMPY)
251 SBC TEMPY
252 STA TEMPS
253 LDA NEDPT
254 CLD
255 SEC
256 SBC TEMPS
257 TAY
258 CCR2 LDA (LADX),Y
259 STA X2 ; X2=THE X COORD. OF THE PT. BEFORE THE EDGE CANDI
LDA (LADY),Y  ;Y2=THE Y COORD. OF THE PT. BEFORE THE EDGE CANDI
STA Y2
LDA TEMPY  ;CALCULATE POSITION OF COORDINATES OF THE PT.
CLD  ;IN FRONT OF THE EDGE CANDIDATE EDGE PT.
CLC
ADC PIXR  ;CALCULATE TEMPY+PIXR
STA TEMPS
TAY
CMP NEDPT  ;COMPARE THIS VALUE TO THE NO. OF EDGE PTS. IN EC
IF CALCULATED POSITION=NEDPT THEN
BCC CCR3  ;CORRECT VALUE, IF NOT GOTO CCR3 & CONTINUE
CLD
SEC
SBC NEDPT  ;TRUE POSITION=CALC. POS.-NEDPT-1
TAY
STA TEMPS
CCR3
LDA (LADX),Y
CLD  ;ADD THE X COORDINATE FOR THE POINT
CLC  ;AFTER THE EDGE CANDIDATE TO THE
ADC X2  ;VALUE PRECEDING THE EDGE CANDIDATE
STA X2
CMP X1
BCC CCR4  ;IF (X1+X2) FROM THESIS >2XM THEN
CLD  ;FIND (X1+X2)-2XM & STORE IT IN X1
SEC
SBC X1
STA X1
JMP CCR5
CCR4
LDA X1
CLD  ;O.W. CALCULATE & STORE 2XM-(X1+X2) FROM
SEC  ;THESIS IN MEMORY LOCATION X1
SBC X2
STA X1
CCR5
CMP #0  ;IF THIS VALUE = 0 THEN DONE WITH X
BEQ CCR6  ;IF NOT THEN CALCULATE THE SQUARE VALUE
STA I
STA J
JSR MULTPL  ;CALCULATE (2XM-(X1+X2))^2
LDA BASE
STA X1  ;STORE RESULT IN X1
LDA BASE+1
BEQ CCR6  ;IF VALUE IN BASE+1 (HIGH BYTE)<<0 THEN
LDA #FF  ;STORE 255 IN X1
STA X1
CCR6
LDY TEMPS
LDA (LADY),Y  ;ADD THE Y COORDINATE FOR THE POINT
CLD  ;AFTER THE EDGE CAND. TO THE
CLC
ADC Y2  ;VALUE PRECEDING THE EDGE CANDIDATE
STA Y2
CMP Y1
BCC CCR7  ;IF (Y1+Y2) FROM THESIS>2YM THEN
;FIND \( (Y_1+Y_2)-2YM \) AND STORE IT IN \( X_1 \)

;ADD \( X_1 \) AND \( Y_1 \)

;O.W. CALCULATE & STORE \( 2XM-(X_1+X_2) \) FROM THESIS IN MEMORY LOCATION \( X_1 \)

;IF THIS VAL=0 THEN DONE WITH \( Y \)

;IF NOT THEN CALCULATE SQUARE VALUE

;DEACTIVATE BANK SWITCHED MEMORY

;CALCULATE \( (2YM-(Y_1+Y_2))^2 \)

;ZERO OUT \( Y \) REGISTER

;IF VALUE EXCEEDS 255 THEN STORE 255 IN \( Y_1 \)

;IF VALUES ADDED OK THEN STORE VALUE IN CORVAL

;RETURN FROM SUBROUTINE

;IF NO. EXCEEDED 255 THEN STORE 255 IN CORVAL

;SET HIGHBYTE OF LADY TO MOVE DATA TO \$8000-$80FF

;ZERO OUT LOW BYTE OF ADDRESS REGISTERS

;SET HIGHBYTE LADX TO MOVE DATA FROM \$F700-$F7FF

;SET HIGHBYTE OF LADY TO MOVE DATA TO \$8000-$80FF

;ZERO OUT \( Y \) REGISTER

;SET HIGHBYTE \( (LADX),Y \)

ECT FILE NAME IS CORDET.OBJI

ORG \$9000

;ACTIVATE BANK SWITCHED MEMORY

;BEGIN PROGRAM

LDA \#0

STA LADX

STA LADY

LDA \#F7

STA LADX+1

LDA \#80

STA LADY+1

LDY \#0

;ZERO OUT LAD

TR1 LDA (LADX),Y
364 STA (LADY),Y ;MOVE DATA FROM HIGH TO LOW MEMORY
365 INY
366 TYA
367 CMP NEDPT ;HAVE ALL OF THE X COORD. OF THE EDGE POINTS BEEN
368 BCC TR1 ;IF NOT MOVE THE NEXT POINT
369 BEQ TR1
370 ;MOVE Y COORDINATES
371 LDA #$F8
372 STA LADX+1 ;SET LADX TO READ FROM BANK SWITCHED
373 LDA #$81
374 STA LADY+1
375 LDY #0
376 TR2 LDA (LADX),Y ;READ ONE BIT FROM BANK SWITCHED MEMORY
377 STA (LADY),Y ;STORE POINT IN LOWER MEMORY
378 INY
379 TYA
380 CMP NEDPT
381 BCC TR2
382 BEQ TR2
383 ;MOVE CORNER CODE
384 LDA #$F9
385 STA LADX+1 ;SET LADX TO READ FROM BANK SWITCHED
386 LDA #$82
387 STA LADY+1 ;SET LADY TO WRITE TO LOWER MEMORY
388 LDY #0
389 TR3 LDA (LADX),Y
390 STA (LADY),Y
391 INY
392 TYA
393 CMP NEDPT
394 BCC TR3
395 BEQ TR3
396 LDA #$F6
397 STA LADX+1 ;SET LADX TO READ FROM BANK SWITCHED MEMORY
398 LDA #$83
399 STA LADY+1 ;SET LADY TO WRITE TO LOWER MEMORY
400 LDY #0
401 TR4 LDA (LADX),Y
402 STA (LADY),Y
403 INY
404 TYA
405 CMP NEDPT
406 BCC TR4
407 BEQ TR4
408 LDA NEDPT ;STORE NO. OF EDGE POINTS IN LOWER MEMORY
409 STA $8400
410 LDA NOLL
411 STA $8401
412 ;DEACTIVATE BANK SWITCHED MEMORY
413 LDA #$C082
414 RTS
**SUMS-FOR LINEAR-REGRESSION**  09-05-1989  006 PAGE 1

```assembly

*****************************************************************

;LINEAR REGRESSION PROGRAM MIKE MILLER (11/21/88)

*****************************************************************

; VARIABLES FOR PROCESS

BASE EQU $EB ; 2BYTE BASE ADDR

; VARIABLES SHARED WITH OTHER PROGRAMS

ERROR EQU $902E ; ERROR FLAG (0=O.K.)
I EQU $E004 ; COUNTER (TEMP. X-COORDINATE)
J EQU $E005 ; COUNTER (TEMP. Y-COORDINATE)
NEDPT EQU $E01C ; NUMBER OF EDGE POINTS IN LINKED LIST
PEDG EQU $E026 ; POINTER FOR EDGE LIST
PLIN EQU $E027 ; POINTER FOR LINE LIST

; LIST OF PROGRAMS

MULTPL EQU $77A0 ; MULTIPLICATION PROGRAM

; PROCEDURE NAME IS LINREG.OBJO

ORG $9150

*****************************************************************

; SUMMATION OF THE COORDINATES FOR EACH LINE

; FOR LINEAR REGRESSION ANALYSIS

; PARAMETER NOLL,NSHP,PLIN,PEDG

; INTERN  I,J,BASE

LDA $C083 ; ACTIVATE BANK SWITCHED MEMORY
LDA $C083
LDX #0
LDY #0
STY PEDG ; EDGE POINTER
STX PLIN ; LINE POINTER

LR1 LDA $F900,Y
CMP #1
BEQ LR2 ; LOOK FOR FIRST PT ALONG EDGE
INY
JMP LR1
LR2 STY PEDG
LI1 LDA #1 ; INITIALIZATION PROC LINE
STA $B580,X ; NE
LDA #0
STA $B680,X ; INIT. HIGH STORE
STA $B800,X
STA $B900,X
STA $BA0,X
```

---

**Notes:**

- The text contains an assembly language program for linear regression analysis.
- The program includes variables for process and shared with other programs.
- It uses specific instructions and addresses for memory manipulation.
- The purpose of the program is to calculate the sums for linear regression analysis.
- The program is written in assembly language and utilizes specific addresses and flags.