

Robotic Weed Control System for Tomatoes

BY

WON SUK LEE

B.S. (Seoul National University) 1986

M.S. (Seoul National University) 1988

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Biological and Agricultural Engineering

in the

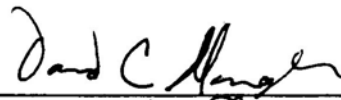
OFFICE OF GRADUATE STUDIES

of the

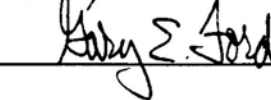
UNIVERSITY OF CALIFORNIA

Davis

Approved:







Committee in Charge

1998

To my wife,

Inok

ABSTRACT

A real-time intelligent robotic weed control system was developed for selective herbicide application to in-row weeds using machine vision and precision chemical application. The image processing algorithm took 0.34s to process one image containing 10 plant objects, representing a 11.43 cm by 10.16 cm region of seedline, and allowing the prototype robotic weed control system to travel at a continuous rate of 1.20 km/h. In actual field trials conducted in a commercial processing tomato field, the robotic weed control system correctly identified and didn't spray 75.8% of the tomato plants and correctly sprayed 47.6% of the weeds.

The color segmentation look-up-tables made in hue, saturation, intensity color space were generally better than those made in normalized red, green, blue and un-normalized red, green, blue color spaces. Overall, look-up-tables built only with hue gave the best performance, correctly classifying 77.8% of color pixels.

In validation tests with 290 field images from 13 different commercial processing tomato fields, the image processing algorithm correctly identified 58.5% - 80.7% of tomato cotyledons, 9.0% - 21.2% of tomato true leaves, 8.9% - 12.9% of tomato leaves that were curled, occluded, bug-eaten or partially hidden by the edge of the image, and 93.0% - 95.5% of weeds using plant area, length to perimeter ratio, and occupational ratio. For separation of occluded plant leaves, 5 modifications to the watershed algorithm were investigated. The performance of *opening*, feature criteria, and combined *opening* and feature criteria modifications were better than others. The recognition of occluded tomato cotyledons and true leaves improved 33.3% and 41.1%, respectively, after the modified watershed algorithm was applied to occluded objects. The angle of tomato

cotyledon orientation for some varieties changed at different times of day. The critical orientation angle for tomato cotyledon recognition was estimated as 27.5° with the vertical axis of the plant. Spray trials with 2.54 cm and 1.27 cm diameter targets showed that the robotic spraying system correctly sprayed these simulated “weeds”, targeting their centers within an average spatial error of 0.51 - 1.36 cm and a standard deviation of 0.21 - 0.71 cm on three different ground surfaces.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisor, Dr. David C. Slaughter for his invaluable guidance, financial support, and encouragement. Without his dedication, this work would have not been done. I also would like to thank the rest of my dissertation committee, Dr. D. Ken Giles for his extensive support and advice for implementing precision chemical application system, and Dr. Gary E. Ford for his revision and comments on my dissertation. Thanks are also given to professor Shrinii Upadhyaya for his helpful discussion and assistance.

I also acknowledge Dr. Daehie Hong for helpful advice in serial communication, Dr. Serge Beucher for valuable suggestion in watershed method, Garry Pearson and Jim Jackson for their support for field work, and Ken Zeff from Ag-Seeds Unlimited, Woodland, CA for supplying tomato seeds. Special thanks are given to processing tomato farmers (Tony Turkovich, Mark Cooley, Emmett and Jim Heidrick, and Joe Heidrick) for supplying their own commercial processing tomato fields for our field tests. I would like to thank all professors, staff members, and fellow students in the Biological and Agricultural Engineering department, who have helped me in my study and research.

Special appreciation are given to my parents for their care, love and support all these years. Finally, sincere thanks are given to my lovely wife, In Ok Kim, for her patience, understanding and support throughout this work. Jae Young and Jae Hee, my lovely son and daughter, are also a part of my dissertation.

Won Suk Lee

TABLE OF CONTENTS

TITLE PAGE	i
ABSTRACT	iii
ACKNOWLEDGMENTS	v
TABLE OF CONTENTS	vi
GLOSSARY OF ABBREVIATIONS AND NOTATIONS	ix
1. INTRODUCTION	1
2. BACKGROUND	7
2.1 Plant identification	10
2.2 Real-time machine vision application	15
2.3 Separation of partially occluded objects	16
2.4 Precision farming and selective chemical application	18
2.5 Color image segmentation and Bayes' classifier	22
2.6 Definitions of image processing commands	25
3. MATERIALS AND METHODS	28
3.1 Overview of the research and the robotic weed control system	28
3.2 Robotic weed control system	30
3.2.1 Image Processing Hardware	30
3.2.2 Outdoor image acquisition and uniform illumination device	34
3.2.3 Precision spraying system	35
3.3 Image analysis	37
3.3.1 Image processing algorithm - Programs for field tests	37
3.3.3.1 Image acquisition preparation	37
3.3.3.2 Image acquisition	40
3.3.3.3 Color image segmentation with Bayes' decision rule	42
3.3.3.4 Look-Up Table construction for image binarization	52

3.3.3.5	Look-up table and image quality	57
3.3.3.6	Binary image pre-processing	60
3.3.3.7	True leaf recognition: Curvature calculation	61
3.3.3.8	Partially occluded leaves: Watershed algorithm	67
3.4	Bayesian classifier with features	80
-Method I		91
-Method II		92
-Method III		94
3.5	Displacement sensing and calibration of encoder	96
3.6	Precision chemical application system	98
3.6.1	Nozzle design	99
3.6.2	Valve / nozzle design	101
3.6.3	Valve driver circuit	104
3.6.4	Valve and encoder software	105
3.7	Test procedure of precision chemical application system	114
3.7.1	Performance of encoder	114
3.7.2	Spray targeting accuracy without imaging	114
3.7.3	Spray targeting with imaging on different ground surfaces	116
3.8	Procedure for field testing of the prototype system	118
3.8.1	Speed of the image processing commands and algorithm	118
3.8.2	Field testing of the prototype system	119
3.9	Cotyledon opening experiment in a field	122
3.10	Transgenic purple tomato plants	128
4.	RESULTS AND DISCUSSIONS	132
4.1	LUT performance	132
- Method I		135
- Method II		147
4.2	Plant recognition performance	149
4.2.1	Cotyledon opening experiment results	149

4.2.2	Bayesian classifier with features	154
	- Method I	154
	- Method II	177
	- Method III	188
	- Final selection of the best feature subsets	192
4.2.3	Separation of touching leaves: Watershed method	198
4.3	Performance of precision chemical application system	216
4.3.1	Displacement sensor performance	216
4.3.2	Spray targeting accuracy without imaging	218
4.3.3	Spray targeting with imaging on different ground surfaces	227
	- Indoor tests of the prototype system with rectangular (“tomato cotyledon”) and circular (“weeds”) targets	230
	- Comparison of spraying-only test with imaging & spraying test	232
4.4	Speed of the prototype system and field testing	234
4.5	Recognition of transgenic purple tomato plant	239
5.	SUMMARY AND CONCLUSIONS	244
5.1	Real-time prototype system	244
5.2	Performance of precision chemical application system	245
5.3	LUT performance	246
5.4	Plant recognition performance	247
5.5	Separation of touching leaves: Watershed method	248
5.6	Diurnal changes in plant appearance	250
5.7	Recognition of transgenic purple tomato plant	250
6.	RECOMMENDATIONS FOR FUTURE WORK	252
	REFERENCES	254
	APPENDIX	266

GLOSSARY OF ABBREVIATIONS AND NOTATIONS

<i>A</i>	area of a nozzle
<i>ABSAVGC</i>	average of absolute value of curvature
<i>ABSUMINV</i>	sum of absolute value of radius of curvature
<i>AREA</i>	area of an object
<i>ATC</i>	ratio of area to average of the absolute values of curvature
<i>ATL</i>	ratio of area to length
<i>ATP</i>	ratio of area to projection area
<i>AVGC</i>	average curvature
<i>B</i>	blue
<i>CMP</i>	compactness
<i>CNTRD</i>	centroid of an object
<i>CTC</i>	ratio of compactness to average of the absolute values of curvature
<i>ECCN</i>	eccentricity
<i>ELG</i>	elongation
<i>ETC</i>	ratio of elongation to average of the absolute values of curvature
<i>G</i>	green
<i>H</i>	hue
<i>HET</i>	height
<i>I</i>	intensity
<i>IPC</i>	image processing computer
<i>LHW</i>	logarithm of ratio of height to width
<i>LTP</i>	ratio of length to perimeter
<i>LUT</i>	look-up table
<i>M</i>	number of pattern classes
<i>M₀₂</i>	second moment of an object along the y-axis
<i>M₁₁</i>	multiplied moment of inertia of an object around the centroid
<i>M₂₀</i>	second moment of an object along the x-axis
<i>MAXC</i>	maximum curvature
<i>MIC</i>	microcontroller

<i>MINC</i>	minimum curvature
<i>MJX</i>	major axis
<i>MNX</i>	minor axis
<i>MTM</i>	ratio of MJX to MNX
<i>MTMC</i>	ratio of difference of MAXC & MINC to sum of MAXC & MINC
<i>NEG</i>	occurrence of negative curvature
<i>OCCR</i>	occupational ratio
$P(\omega_i)$	<i>a priori</i> probability of class ω_i
$P(\omega_i \mathbf{x})$	<i>a posteriori</i> probability of \mathbf{x} , given ω_i
<i>PERIM</i>	perimeter
<i>PRINAXIS</i>	orientation of principal axis of inertia of an object
<i>PTB</i>	ratio of perimeter to broadness
<i>PTC</i>	ratio of perimeter to average of the absolute values of curvature
<i>PTP</i>	ratio of Pythagorean length to perimeter
\dot{Q}	flow rate
<i>R</i>	red
<i>S</i>	saturation
<i>SUMINV</i>	sum of radius of curvature
<i>STDEVC</i>	standard deviation of curvature
T_{AREA}	threshold of AREA between non-occluded and occluded objects
T_{CCAWE}	threshold of concavity between non-occluded and occluded objects
<i>V</i>	exit velocity of a spray drop from a nozzle
<i>W0</i>	original watershed algorithm
<i>W1</i>	watershed algorithm modified with <i>opening</i> algorithm
<i>W2</i>	watershed algorithm modified with pre-flooding
<i>W3</i>	watershed algorithm modified with feature criteria
<i>W4</i>	watershed algorithm modified with concavity criteria
<i>W5</i>	watershed algorithm modified with combined <i>opening</i> and feature criteria

<i>WID</i>	width
<i>YCNRD</i>	y-coordinate of a centroid
<i>acount</i>	number of character ‘a’ sent to IPC from MIC
<i>aflag</i>	set to 1 after a character ‘a’ is sent to IPC from MIC
<i>b</i>	normalized blue
<i>ccount</i>	character count read from serial buffer in MIC
<i>ccount0</i>	a variable used to store previous value of <i>ccount</i> .
<i>char0</i>	<i>position byte</i> indicating column number (1 - 18) sent from IPC to MIC
<i>char1</i>	<i>valve byte</i> sent from IPC to MIC
<i>cpc</i>	number of encoder counts per spray cell
<i>display_timer</i>	a variable keeps serial buffer being checked for 1000 ms
<i>f(i, j)</i>	binary intensity level of a pixel at the (i, j) location
<i>g</i>	normalized green
<i>h_{min}</i>	minimum height
<i>k</i>	number of columns that the <i>store_ptr0</i> must be advanced
<i>k0</i>	<i>k0 = 0</i> indicates that IPC and MIC are in synchronization
<i>m1</i>	number of occluded leaves over cut
<i>m2</i>	number of occluded leaves uncut
<i>m3</i>	number of occluded leaves properly cut
<i>m4</i>	number of total correctly cut leaves
$\mathbf{m}_i = E\{\mathbf{x} \mid \omega_i\}$	mean vector of class ω_i
<i>mycount</i>	indicates that IPC finishes processing an image
<i>n</i>	dimension of feature space
<i>n₁</i>	number of plant pixels in plant-only image
<i>n₂</i>	number of non-black background pixels in background-only image
<i>n₃</i>	number of non-black pixels in segmented plant-only binary image
<i>n₄</i>	number of non-black pixels in segmented background-only binary image
<i>noz_ofs</i>	number of spray cells between two lines of valve arrays

$p(\mathbf{x})$	probability density of \mathbf{x}
$p(\mathbf{x} \omega_i)$	conditional probability density of \mathbf{x} , given ω_i
<i>position_counter</i>	number of columns sprayed by valve system in the current image
<i>r</i>	normalized red
<i>s</i>	arc length
<i>spray_array</i>	array variables which has <i>valve byte</i> . Cleared after spraying
<i>spray_delay</i>	spray delay from the camera to first line of valves
<i>spray_ptr</i>	<i>spray_array</i> index indicating which <i>valve byte</i> is sent to nozzles
<i>stored</i>	array variables which stores <i>valve bytes</i> to compare with those sent from IPC
<i>store_ptr0</i>	<i>spray_array</i> index used to store <i>valve byte</i> into <i>spray_array</i>
\mathbf{x}	feature vector, e.g. {r, g, b} or {CMP, ELG, LTP}
Λ	likelihood ratio
Σ_i	Covariance matrix of class ω_i
κ	curvature
α	significance level
θ	polar angle of the unit tangent vector
γ	openness of a cotyledon (angle of a cotyledon)
ω_i	<i>i</i> th pattern class
\forall	For all

Copyright by
WON SUK LEE
1998

1. INTRODUCTION

A weed can be thought of any plant growing in the wrong place at the wrong time and doing more harm than good. Weeds compete with the crop for water, light, nutrients and space, and therefore reduce crop yields and also affect the efficient use of machinery (Parish, 1990). Many methods are used for weed control. Among them, mechanical cultivation is commonly practiced in many vegetable crops to remove weeds, aerate soil, and improve irrigation efficiency, but this technique cannot selectively remove weeds located in the seedline between crop plants. The most widely used method for weed control is to use agricultural chemicals (herbicides and fertilizer products). In fact, the success of U.S. agriculture is attributable to the effective use of chemicals. For example, a total of 5.9 million kg of agricultural chemicals (herbicides, insecticides, fungicides, and other chemicals) were used to produce processing tomatoes in California alone in 1994 (USDA, NASS and ERS, 1995). This heavy reliance on chemicals raises many environmental and economic concerns, causing many farmers to seek alternatives for weed control in order to reduce chemical use in farming. For some crop/weed situations there are no selective herbicides. Selective herbicides selectively kill only weeds and not crop plants, thus they are important for weed control.

Since hand labor is costly, an automated weed control system may be economically feasible. A real-time precision robotic weed control system could also reduce or eliminate the need for chemicals. Although there have been many efforts to control in-row weeds, no system is currently available for real-time field use. In this research, an intelligent real-time robotic weed control system has been developed to identify and locate outdoor plants for selective spraying of in-row weeds using an

environmentally sound and friendly chemical application system based upon machine vision technology, pattern recognition techniques, knowledge-based decision theory, and robotics. Figure 1.1 shows testing of the prototype robotic weed control system in a commercial processing tomato field.

The juvenile processing tomato field plants were selected as the target crop plant since tomatoes are one of the more profitable leading crop plants in California which have serious weed control problems. Tomato plants also provide a challenge for machine vision recognition with their highly variable shapes and sizes. However this research could also easily be applied to many other crops. Figure 1.2 shows juvenile tomato plants and Figure 1.3 shows some of the commonly found weeds in commercial processing tomato fields in northern California.



Figure 1.1 Field testing of the prototype robotic weed control system.

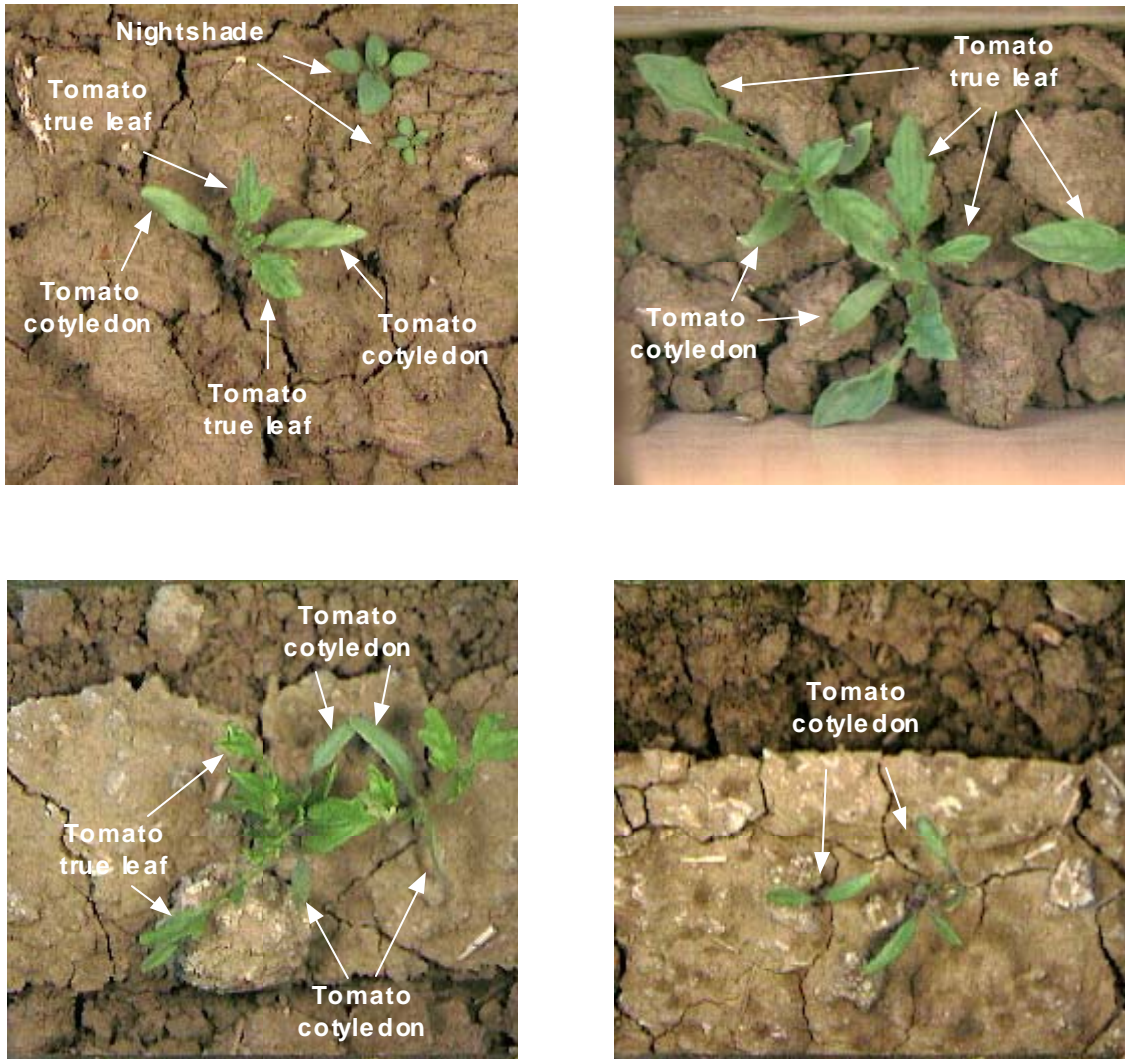


Figure 1.2 Tomato plants in commercial fields.



(a) Black nightshade.



(b) Groundcherry.



(c) Hairy nightshade.



(d) Lambsquarters.



(e) Mustard.



(f) Nettleleaf goosefoot.



(g) Shepherdspurse.



(h) Cheeseweed.



(i) Redroot pigweed.



(j) Groundsel.



(k) Velvetleaf.



(l) Field bindweed.



(m) Yellow nutsedge.



(n) Large crabgrass.

Figure 1.3 Some of the most common and troublesome weeds in California tomato fields. (University of California Statewide Integrated Pest Management Project, 1985. Photographer: Jack Kelly Clark. Used with permission.)

Tomatoes are one of the leading vegetable crops produced in California. In 1996, over 9 billion kg of processing tomatoes were produced in California, accounting for 93% of all processing tomatoes produced in the U.S. (USDA and NASS, 1997). However, the current in-row weed control method is highly dependent on labor-intensive and costly hand hoeing. A significant amount of manual work is still required for weed control in crop rows, which hopefully can be automated with today's rapidly growing state of the art computer technologies.

For processing tomatoes in California, the cost for weed control was about \$50 per 0.4 ha for herbicides and \$80 per 0.4 ha for hand weeding in northern California in 1996. According to the economic analysis for the prototype machine by D. C. Slaughter (1996), if a prototype robotic system could travel at 0.8 km/hr, the savings would justify a purchase price of over \$110,000 per machine considering the current cost. This assumes a three-row machine for rows spaced 1.52 m apart, an operating period of 45 days per season, 60 % of overhead and operating costs, no interest, and a five year machine life.

Agriculture should take advantage of today's state of the art technology. More and more computers are being used in farming to minimize cost and maximize yields. Even satellites are used for smart and precision farming using the Global Positioning System (GPS) and the Geographic Information System (GIS). Complete farm automation would be an ultimate goal. This research explored the feasibility of using a robot as a means of cultivating and thinning and as one of the stepping stones for automated farming. Costly, tedious and labor-intensive manual operations need to be automated.

OBJECTIVES

The goal of this project was to build a real-time machine vision based robotic weed control system that can detect crop and weed locations, kill weeds and thin crop plants. The system needed to recognize tomato plants and weeds outdoors in commercial tomato fields using image processing techniques while moving forward at a constant speed. Specific objectives were to:

- (1) develop and implement a color machine vision algorithm in real-time for identifying crop plants and weeds in the seedline,
- (2) develop a precision chemical application system for the control of in-row weeds for optimal use of pesticides and high efficiency,
- (3) develop a uniform illumination device for field use with a real-time computer vision system to facilitate high image quality,
- (4) develop an algorithm which can distinguish tomato plants older and larger than the cotyledon stage,
- (5) increase the accuracy and speed of the algorithms implemented in real-time in order to distinguish tomato seedlings in the field, and
- (6) evaluate the performance of a prototype system in commercial processing tomato fields.

2. BACKGROUND

Ever since humans started farming, weeds have been one of the major obstacles to maximizing production. Until recent technologies evolved, farming has been dependent on human power. Although there have been a lot of efforts to control in-row weeds effectively, no system is currently available to replace tedious and time-consuming hand weeding. Figure 2.1 shows the current in-row weed control technique (hand hoeing) in a commercial processing tomato field in northern California.



Figure 2.1 Current in-row weed control technique (hand hoeing) in a commercial processing tomato field in northern California.

Hand hoeing is costly, time consuming and labor intensive. For example, the cost for hand weeding was about \$80 per 0.4 ha (1 acre) for processing tomato production in northern California in 1996. In California, there were over 12 million ha of farm land in 82,000 farms in 1996. Among them, 140,183 ha were used for tomato production, Table 2.1.

Table 2.1 Area harvested, production and value per ton for tomato production in California in 1996.

	Fresh	Processing
Area (ha)	13,516.5	126,666.6
Production (ton)	484,300.1	10,660,780.0
Value per ton (\$)	504.0	62.30

One of the major issues concerning agricultural production in California was the current heavy reliance on agricultural chemicals to optimize yields and minimize costs. A total of 53.0 million kg of agricultural chemicals (fertilizers, herbicides, insecticides, fungicides and other chemicals) were used to produce tomatoes in California alone in 1994 (USDA, NASS and ERS (1995)), Table 2.2.

Table 2.2 Chemical usage in tomato production in California in 1994.

Chemicals	Fresh (x 1000 kg)	Processing (x 1000 kg)	Total (x 1000 kg)
Fertilizers	5,797.8	40,034.1	45,831.9
Herbicides	10.5	200.4	210.9
Insecticides	78.6	99.2	177.8
Fungicides	511.9	4,453.1	4,965.0
Others	418.6	1,423.8	1,842.4
Total	6,817.4	46,210.6	53,028.0

However, this heavy reliance on chemicals raised many concerns about health and environmental aspects. The current concern over the use of methyl bromide is an example of some of the issues associated with the use of agricultural chemicals. Ferguson and Padula (1994) investigated the economic effects of banning methyl bromide (MB)

for soil fumigation which had been widely used to control soil pests and protect stored commodities, since MB contributes to the depletion of the stratospheric ozone layer. They reported that the Environmental Protection Agency (EPA) might ban the use of MB by initiating action under the Clean Air Act that required a phaseout of MB uses by the year 2001. They predicted that losses for tomato growers would be about \$86 million per year if the available alternatives for tomatoes were used, while the net revenue loss would be about \$100 million annually if no alternatives were available.

There has been a lot of effort to control weeds non-chemically in order to reduce chemical costs in response to environmental pressure. These methods can be largely divided into cultural weed control methods, mechanical control methods, and biological control methods. In this research, mechanical control methods were the main focus. These include hand pulling or hoeing, tillage, cultivation, burning, flame cultivation, and electrical devices (Cooperative Extension Service (1995)). There were some researchers who investigated non-chemical weed control methods (Parish (1990) and Bond (1992)), but as Bond pointed out, few attempts have been made to selectively control weeds in the seedline. However, with advances in image processing and machine vision technologies, many researchers have applied these techniques to agriculture to identify individual crop plants.

Image processing and machine vision technologies have been applied successfully to many agricultural settings recently. Properly applied machine vision techniques improve manufactured product quality and provide valuable process control information (Novini, 1992). Their primary agricultural applications are automatic inspection and sorting of agricultural products (Shearer and Payne (1990), Miller and

Delwiche (1991), Al-Janobi and Kranzler (1994), Lan et al. (1996) and Crowe and Delwiche (1996a, b)), and identifying and locating individual crop plants (Jia et al. (1990) and Tian and Slaughter (1993)). Machine vision has been also used for guiding a robotic system in the harvest of fruits (Slaughter and Harrell (1989)), detecting fruit defects, evaluating chemical applications (Jiang and Derksen (1993)) and investigating plant architectural measurements (Tarbell and Reid (1989)).

2.1 Plant identification

Machine vision technologies have been applied to agriculture to identify and locate individual plants. Many researchers have tried various image processing methods, working in different environments; however, most of the work has been done indoors with controlled illumination and an adequate setup for the acquisition of high quality images. In an early study typical of those to follow, Guyer et al. (1986) studied the feasibility of using machine vision to identify the species of potted, greenhouse-grown weeds. In this study, Guyer et al. noted that plants grown in a greenhouse had a different appearance from those grown under the natural outdoor environment of a commercial farm.

The variety of visual characteristics that have been used in indoor plant identification can be divided into three categories: spectral reflectance, morphology, or texture. Many studies (e.g. Slaughter and Harrell (1989), Franz et al. (1991b), Woebbecke et al. (1995a), Zhang and Chaisattapagon (1995), Brivot and Marchant (1996), and Shiraishi and Sumiya (1996)) have used color or near infrared reflectance to distinguish the fruits or plants from the background. In a few situations, researchers have

found that spectral characteristics alone can be used to distinguish between selected plant species, but this technique is usually insufficient to distinguish crop plants from weeds on a typical California farm. Morphological characteristics of plant leaves such as central moment, complexity, principal axis of moment of inertia, first invariant moment, aspect ratio, radius permutation, ratio of perimeter to longest axis, compactness, and elongation have been used to classify plant species with some success (e.g. Guyer et al. (1986), Franz et al. (1991a), Woebbecke et al. (1995b), and Shiraishi and Sumiya (1996)). In a few cases textural feature analysis has also been used to identify plant species (e.g. Shearer and Holmes (1990), Burks and Shearer (1995), Zhang and Chaisattapagon (1995) and Ahmad et al. (1996)).

Among morphological characteristics of plant leaves, curvature has also been used as one of the major tools to identify plant leaves. Many researchers studied how to estimate curvature accurately and precisely from a discrete grid (Asada and Brady (1986), O’Gorman (1988), Worring and Smeulders (1992 and 1993)).

Sarkar and Wolfe (1985) used the minimum curvature of the chain coded tomato boundary as the feature for detecting shape for classification of fresh market tomatoes, however they made some typical assumptions such as non-occlusion, pre-orientated objects, diffused light, and darker background. Franz et al. (1991a) used curvature to describe boundaries of both completely visible and partially occluded leaves. They reported that partially occluded leaves were identified by aligning the resampled curvatures with species models. However, they found that curvature was an inadequate shape descriptor for the leaves with variable leaf serration while a curvature matching method was applied. Tao et al. (1995) used boundary information to identify shapes of

potatoes. Even though they didn't use curvature directly, they used normalized radius boundary and its Fourier transform and achieved 89% agreement with human judgment for shape grading of potatoes. Shiraishi and Sumiya (1996) also used the curvature of leaf boundaries along with other features for plant identification using quasi-sensor fusion and total occurrence range and reported that they obtained satisfactory discrimination results. They reported difficulties in recognizing overlapped leaves.

As a preliminary effort to develop new techniques and new weed control machinery for the California agricultural industry (which has been developed at the University of California, Davis), Tian and Slaughter (1993) developed and tested a computer vision algorithm in a laboratory environment to detect and locate individual tomato plants with images taken in commercial tomato fields. They used hue and excessive green ($= 2\text{Green}-\text{Red}-\text{Blue}$) to get binary images by thresholding and extracting features such as compactness (CMP), elongation (ELG), and y-coordinate of the centroid of leaves from binary images. With 28 field images, the algorithm was able to identify almost all the isolated tomato cotyledons, and determined the inward position of occluded cotyledons with an accuracy of 61.2%.

Further, Tian (1995) studied the feasibility of using a machine vision system to identify individual plants with images taken in the natural outdoor environment. He reported problems associated with non-uniform illumination. The four features, elongation (ELG), compactness (CMP), the logarithm of the ratio of height to width (LHW), and the ratio of length to perimeter (LTP) were used as the optimum subset among all features for tomato cotyledon recognition. He identified between 61 to 82 percent of all the individual plants in about 270 frames of field images in a laboratory

environment. However, the research ended before high speed algorithms were developed for implementation in a real-time computer vision system for use in a commercial field. The following feature definitions were used by Tian (1995) and these definitions were also used to identify tomato plants and weeds reported in this research along with other features defined later in chapter 3.

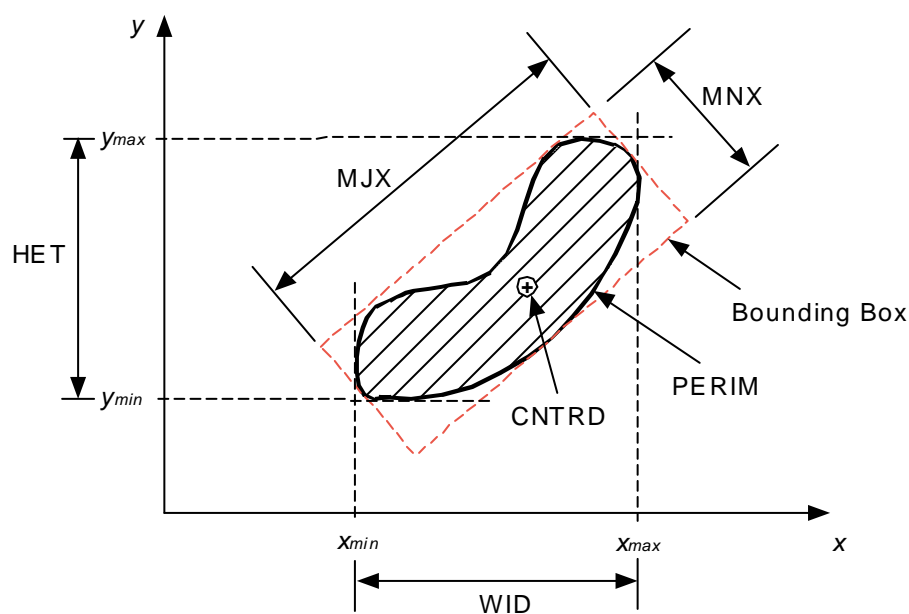


Figure 2.2 Definitions of major axis, minor axis, perimeter and centroid.

Area (AREA) was defined as the number of pixels in an object. Height (HET) was the difference between the largest (y_{max}) and the smallest vertical coordinate (y_{min}) plus one. Width (WID) was defined as the difference between the largest (x_{max}) and the smallest (x_{min}) horizontal coordinate plus one. Major axis (MJX) was defined as the longest axis of a bounding box by which an object is enclosed (Figure 2.2). Minor axis (MNX) was defined as the shortest axis of a bounding box perpendicular to the major

axis. Perimeter (PERIM) was defined as the number of boundary pixels. Centroid (CNTRD) of an object, (\bar{x}, \bar{y}) , was defined as the center of area as in Eq. (2.1) and Eq. (2.2).

$$\bar{x} = \frac{\iint_I xb(x, y)dx dy}{\iint_I b(x, y)dx dy} \quad (2.1)$$

$$\bar{y} = \frac{\iint_I yb(x, y)dx dy}{\iint_I b(x, y)dx dy} \quad (2.2)$$

where I is an image, and

$b(x, y)$ is a pixel value of a binary image.

$$\text{HET} = y_{\max} - y_{\min} + 1 \quad (2.3)$$

$$\text{WID} = x_{\max} - x_{\min} + 1 \quad (2.4)$$

$$\text{ATL} = \frac{\text{AREA}}{\text{MJX}} \quad (2.5)$$

$$\text{CMP} = \frac{16 \text{ AREA}}{\text{PERIM}^2} \quad (2.6)$$

$$\text{ELG} = \frac{\text{MJX} - \text{MNX}}{\text{MJX} + \text{MNX}} \quad (2.7)$$

$$\text{LHW} = \log_{10} \left(\frac{\text{HET}}{\text{WID}} \right) \quad (2.8)$$

$$\text{LTP} = \frac{\text{MJX}}{\text{PERIM}} \quad (2.9)$$

$$\text{PTB} = \frac{\text{PERIM}}{2(\text{HET} + \text{WID})} \quad (2.10)$$

The area to length ratio (ATL) was defined as Eq. (2.5). The compactness (CMP) was used to describe complexity of planar objects and was defined in Eq. (2.6) so that the CMP was one for a perfect square. Elongation (ELG) was the measurement of how long and narrow an object was. The logarithm of the ratio of height to width (LHW) gave a symmetric measure of the aspect ratio of an object. The ratio of length to perimeter (LTP) was a measure of the 2-D distribution pattern of the boundary of an object. The ratio of perimeter to broadness (PTB) was a measurement of a convex region.

2.2 Real-time machine vision application

After image processing technologies have been developed, the natural transition is its “real-time” field application. “Real-time” application means that the system could keep up with the system input rate. There are a few machine vision systems which have achieved real-time application. Slaughter et al. (1992 and 1997) developed a real-time guidance system for precision cultivation (later named the “UC Davis Robotic Cultivator”) that could identify the center of the row under normal field conditions. The vision guidance system identified the location of the seedline, then the offset between the current position and the desired position was adjusted by moving the toolbar laterally. The system was tested in tomato fields and the test results indicated that the prototype could operate at speeds exceeding 8.0 km/h while precisely positioning the cultivator with an overall RMS error ranging from 4.2 mm when there were no weeds to 11.9 mm when the area ratio of weed to tomato was 3:1. This precision UC Davis Robotic Cultivator was used as a guidance system for the robotic weed control system reported in this research.

Liao et al. (1994) studied the feasibility of real-time detection of color and surface defects of maize kernels. Using a Matrox Image-1280 real-time image processing board, they reported that the processing time required from acquiring live images to the end of primitive (basic) feature extraction was from 0.87 s for 1 object to 2.08 s for 12 objects.

Haney et al. (1994) applied machine vision to sort wood based on its color. They reported that the system could operate at conveyor speeds up to 110 m/min. Alchanatis and Searcy (1995) built and tested a high speed inspection system for fresh-market carrots. They reported that the system could handle 2 carrots/s with a classification accuracy of more than 90%. Crowe and Delwiche (1996a and 1996b) developed an image processing algorithm for real-time defect detection in produce. Apples and peaches were tested at a rate of 5 fruits/s and the sorting error rate was 25%.

A group of researchers in Spain have worked to distinguish crop plants from weeds (Molto et al. (1996) and Molto et al. (1997)). They developed a machine vision system for robotic weeding of artichokes and lettuce using color differences between crop plants, weeds and background. The average processing time was about 500 ms per image. A mobile robot for non-chemical weed control is planned.

2.3 Separation of partially occluded objects

One of the difficult tasks for machine vision applications is to recognize partially occluded objects. Human eyes can distinguish overlapped objects very easily since they are working in 3-dimensions. However, it has been a difficult task for machine vision systems to recognize overlapped objects using only a single top view. Occluded objects need to be cut apart so that they can be measured separately.

In an early study, Lester et al. (1978) applied two boundary finding algorithms (heuristic searching and the least maximum cost techniques) to white blood cell images through a graph searching method. They reported that the heuristic searching method was an effective and efficient procedure. Whittaker et al. (1987) used a modified circular Hough transform to locate partially hidden tomatoes based on shape. However, this algorithm was computationally intensive and could not be performed in real-time. Shatadal et al. (1995) developed an algorithm to disconnect touching kernels using a mathematical morphology-based approach, however they reported limitations of the algorithm in that it failed when the connected kernels formed a relatively long isthmus or bridge between them.

One of the widely used methods for separation of occluded objects is the watershed algorithm. Since it was first introduced as a morphological tool by H. Digabel and C. Lantu—joul (1978), many variations have been produced for this technique. The watershed algorithm can be applied to either gray scale images or binary images.

Russ (1990) described conditions for successful application of the watershed algorithm as “The method makes the implicit assumption that the features are both actually convex, so that they should be segmented, and also assumes that the degree of touching or overlap is sufficiently small that there is a valley between the peaks at the center of each feature in the brightness-coded distance map.”

Vincent and Soille (1991) developed an efficient and fast algorithm for computing watersheds in digital gray scale images based on immersion simulation. The algorithm was composed mainly of two steps: a sorting step and a flooding step. In the sorting step, all pixels were sorted in the increasing order of their gray values to induce

direct assignment of each pixel to a unique cell in the stored array. In the flooding step, a hole was pierced in every local minima and its catchment basin (a separate region corresponding to each local minima) was eventually filled with water from the hole. When the two catchment basins would merge, a dam was built to keep water from flooding to other basins. This dam was the watershed (separation line) of overlapped objects. A first-in-first-out data structure was introduced to speed up the computations in the flooding step. This algorithm was used to separate occluded plant leaves in this research (see Chapter 3.5.5).

In order to make thin watershed lines and straightening the watershed lines to produce identical watershed lines regardless of the orientation of the object, Orbert et al. (1993) also proposed a modified watershed algorithm by fusing the catchment basins. They provided two example applications and reported that the new algorithm worked well.

Casasent et al. (1996) applied the binary watershed algorithm to separate pistachio nuts in X-ray images and reported that 250 of the 253 clusters were segmented correctly. However, it was not implemented in real-time. They also reported over segmentation (excessive cutting) problems if the boundary was irregular or complex.

2.4 Precision farming and selective chemical application

As technologies have evolved, site-specific application of agricultural chemicals and nutrients has been one of the major concerns in precision farming. Farmers want to concentrate their efforts only where needed to make their farming efficient. Precision farming refers to carefully customizing crop and soil management to fit the different

conditions found in each field. Precision farming is sometimes called “site specific farming” or “variable rate technology”. It has been realized with the use of remote sensing systems, Global Positioning Systems (GPS) and Geographic Information Systems (GIS). More recently farmers have gained access to site-specific technology through GPS. GPS utilizes a series of military satellites that identify the location of farm equipment within a meter of its actual site in the field. In conjunction with GPS technology, Geographic Information Systems (GIS) can be used to accurately produce a map of the data, layering various nutrient levels, soil types, fertility, yield potential and other pertinent information. Therefore, spatially variable application of herbicides would be possible and eventually lead environmental benefit as well as cost reduction.

In the same context as saving agricultural chemical costs and being a more environmentally sound application, there has been some work done in selective application of herbicides. However, in the majority of the work, ‘selective’ referred to the selectivity of plants vs. soil, not crop vs. weeds. Most of this work utilized a difference in reflectance levels between plants and soil background based upon the chlorophyll in the foliage absorbing the red radiation which is reflected by the soil. In earlier studies, the ratio of visible to near-infrared radiation (Hooper, Harries and Ambler (1976)) and the ratio of red to near-infrared (Haggar, Stent and Isaac (1983), Felton et al. (1991), Felton and McCloy (1992), and Merritt et al. (1994)) were used to distinguish green vegetation from the soil background. Some of these led to commercial plant detector-sprayers (Weed Seeker PhD 1620, Patchen California, Inc., Los Gatos, CA and Detectspray-S45, Concord Inc., Fargo, ND).

Tauzer et al. (1994) and Tauzer (1995) developed a prototype machine vision controlled, boomless, offset herbicide sprayer for roadside vegetation control. Target detection was 100% successful for 5 cm x 5 cm targets at travel speed of 12.9 km/h, 89% for 2.5 cm x 2.5 cm targets and 26% for 1.25 cm x 1.25 cm targets. This system also identified only green plants, not crop vs. weeds.

Visser and Timmermans (1996) developed an automatic selective herbicide spraying system for weed control. They used the chlorophyll fluorescence effect and optically filtered LEDs in a sensor to detect weeds, and solenoid valves to control weed spray. However, the system also detected and sprayed all green plants as “weeds”. None of the systems described above could spray weeds selectively as opposed to crop plants.

For future implementation of a crop protection robot for autonomous spraying, Brivot and Marchant (1996) explored the feasibility of using infrared images to segment plants from soil background in the images of transplanted cauliflower using thresholding and replacing pixel values by its minimum neighborhood. They reported possibilities of up to 92% correct classification of crop plants, weeds and background with good images and up to 73% with bad images. In their trials, the weeds were smaller than the crop plants, thus it might have been possible to classify crop plants, weeds and background using only gray scale images. However, this technique may not be feasible when the crop plants are the same size as weeds, such as non-transplanted crop plants.

For selective chemical application to weeds vs. crop plants, Tian et al. (1997) developed and tested a precision sprayer guided by a machine vision system. They used the wavelet transformation to identify weeds in groups from a field of view of 3.7 m x 0.43 m by simultaneously analyzing images in spatial and frequency domains. The

overall accuracy of the sprayer was 75% for weeds and 48% for crops. However, this precision sprayer system did not identify individual plants, but groups of plants.

While most of the previous research actually implemented selective chemical application for weed control, utilizing GPS and GIS for weed control is not at the implementation stage yet, but only at the stage of mapping weed locations in the field for later spraying or for tracking weed infestations. Lass and Callihan (1993) tried to assess the accuracy of two types of GPS receivers by mapping weed locations and found that GPS data agreed closely with U.S. Geological survey data. Wilson et al. (1993) described and developed a procedure for identifying potential pesticide contamination problems for groundwater on weed-infested areas. They reported that their algorithms were used to generate daily precipitation, evapotranspiration, soil carbon, and final hazard maps. Prather and Callihan (1993) developed a geographical information system to evaluate its utility in an eradication program for common weeds. They constructed a database to maintain treatment and efficacy data for tracking the process of eradication for each infestation. Stafford et al. (1996) developed and evaluated a portable weed mapping system using a palm-top PC data logger and a GPS receiver. They found that the system had good potential for aiding a farmer to log weed patches with a good user interface to ease learning and operating the system. Webster and Cardina (1997) conducted experiments to test the accuracy of a global positioning system for measuring the area of simulated weed patches of varying size and to determine the accuracy in navigating back to particular points in a field. They reported that the relationship between GPS error and patch size had an excellent fit and errors decreased as patch size increased.

2.5 Color image segmentation and Bayes' classifier

One important step in implementing a machine vision system is image segmentation. Due to the fact that color images provide more information than gray scale images, color information has been widely used in image segmentation since the late 1980's where color was an important characteristic to identify fruits (Slaughter and Harrell, (1987)) and individual or groups of plants (Tian (1995) and Tauzer (1995)), to implement a precision guidance system (Slaughter et al. (1997)) or to sort agricultural products (Luo et al. (1997)). Some researchers studied building a color classifier (Precetti and Krutz (1993)), analyzing plant images in the HLS color space (Anderson and Wendorf (1993)), or image segmentation in color space (Gao et al. (1995)).

Most of these applications (Tian (1995), Tauzer (1995) and Slaughter et al. (1997)) utilized a color look-up table for real-time implementation using the Bayes' rule. The Bayes' classifier is known to be an optimum classifier in the sense that it minimizes the classification error. Suppose the class-conditional probability density function, $p(\mathbf{x} | \omega_i)$ is Gaussian, then

$$p(\mathbf{x} | \omega_i) = \frac{1}{(2\pi)^{n/2} |\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i)\right\}, \quad i = 1, 2, \dots, M \quad (2.11)$$

where \mathbf{m}_i = mean vector of class ω_i , Σ_i = covariance matrix of class ω_i ,

and M = number of pattern classes.

The *a posteriori* probability, $P(\omega_i | \mathbf{x})$ of class ω_i membership is calculated using Bayes' rule, given an observation \mathbf{x} :

$$\mathbf{P}(\omega_i | \mathbf{x}) = \frac{p(\mathbf{x} | \omega_i) \mathbf{P}(\omega_i)}{p(\mathbf{x})} \quad (2.12)$$

$$\text{where } p(\mathbf{x}) = \sum_{k=1}^M p(\mathbf{x} | \omega_k) \mathbf{P}(\omega_k)$$

Expected loss or conditional risk in assigning \mathbf{x} to ω_i is defined as:

$$R_i(\mathbf{x}) = \sum_{j=1}^M l_{ij} \mathbf{P}(\omega_j | \mathbf{x}), \quad i = 1, 2, \dots, M \quad (2.13)$$

where l_{ij} = loss sustained if \mathbf{x} is assigned to ω_i when it is actually ω_j .

The decision rule that minimizes the probability of error results from the use of the zero-one loss function:

$$l_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad (2.14)$$

Then, conditional risk becomes:

$$R_i(\mathbf{x}) = \sum_{\substack{j=1 \\ j \neq i}}^M \mathbf{P}(\omega_j | \mathbf{x}), \quad i = 1, 2, \dots, M \quad (2.15)$$

However, note that

$$\sum_{j=1}^M \mathbf{P}(\omega_j | \mathbf{x}) = \left\{ \sum_{\substack{j=1 \\ j \neq i}}^M \mathbf{P}(\omega_j | \mathbf{x}) \right\} + \mathbf{P}(\omega_i | \mathbf{x}) = 1$$

Thus,

$$R_i(\mathbf{x}) = 1 - \mathbf{P}(\omega_i | \mathbf{x}) \quad (2.16)$$

Therefore, the minimum risk or minimum probability of error decision rule becomes:

$$\text{Assign } \mathbf{x} \text{ to } \omega_i \text{ if } P(\omega_i | \mathbf{x}) > P(\omega_k | \mathbf{x}), \quad \forall k \neq i \quad (2.17)$$

Restating the decision rule in terms of discriminant function $d_k(\mathbf{x})$,

$$\text{Assign } \mathbf{x} \text{ to } \omega_i \text{ if } d_i(\mathbf{x}) > d_k(\mathbf{x}), \quad \forall k \neq i \quad (2.18)$$

where the discriminant function for Bayes' minimum error rate classification is:

$$d_k(\mathbf{x}) = P(\omega_k | \mathbf{x}) = \frac{p(\mathbf{x} | \omega_k) P(\omega_k)}{p(\mathbf{x})} \quad (2.19)$$

Since $p(\mathbf{x})$ does not depend on k and is always non-negative, it does not affect the decision and can be eliminated. The decision function becomes

$$d_k(\mathbf{x}) = p(\mathbf{x} | \omega_k) P(\omega_k) \quad (2.20)$$

Assuming the class-conditional probability density function, $p(\mathbf{x} | \omega_k)$, is a Gaussian distribution, a log decision function is more useful. Therefore, using Eq. (2.1) and Eq. (2.20), the discriminant function becomes:

$$\begin{aligned} d_k(\mathbf{x}) &= \ln p(\mathbf{x} | \omega_k) + \ln P(\omega_k) \\ &= -\frac{1}{2} \{ (\mathbf{x} - \mathbf{m}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{m}_k) + \ln |\Sigma_k| \} + \ln P(\omega_k) \end{aligned} \quad (2.21)$$

To implement the above Bayes' discriminant function on a given feature vector, \mathbf{x} , the class-conditional probability density function $p(\mathbf{x} | \omega_k)$ is calculated for each class k from Eq. (2.1). Then, the class associated with the largest probability is assigned to the feature

vector \mathbf{x} . A mean vector of class ω_i , \mathbf{m}_i and a covariance matrix of class ω_i , Σ_i , need to be estimated from a training data set prior to implementation.

In previous real-time applications (Slaughter and Harrell (1987), Tauzer (1995) and Slaughter et al. (1997)), the Bayes' classifier was implemented in software, by allocating a large array in RAM and then developing a mechanism to allow the CPU to access the specific decision in the array in real-time. However, in this research, a recently developed new hardware based system (Sharp AUXLUT card) was used to implement the Bayes' classifier in real-time, which did not require the CPU. This was a significant improvement from what was done in the previous applications. This new system allowed a real-time color LUT conversion (about 3 ms with an image size of 256 x 240) in a faster and different way by one pass pixel by pixel multiplication of two images (details are explained in chapter 3).

2.6 Definitions of image processing commands

Once a binary image is obtained from color image segmentation, the resulting binary image must be processed in order to remove any noise and to get an accurate shape of the plant leaves. This section explains definitions of such processing steps as translation, reflection, dilation, erosion, shrinking, swelling and deletion of isolated points, which were used to enhance the binary image throughout this research.

Let A and B be sets in the 2-D integer space Z^2 , with components $a = (a_1, a_2)$ and $b = (b_1, b_2)$, respectively. The *translation* of A by $x = (x_1, x_2)$, denoted $(A)_x$, is defined as

$$(A)_x = \{c \mid c = a + x, \text{ for } a \in A\} \quad (2.22)$$

The *reflection* of B , denoted \hat{B} , is defined as

$$\hat{B} = \{x \mid x = -b, \text{ for } b \in B\} \quad (2.23)$$

For sets A and B in the 2-D integer space Z^2 and ϕ denoting the empty set, the *dilation* of A by B is denoted $A \oplus B$ and is defined as

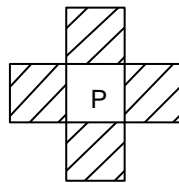
$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \phi\} \quad (2.24)$$

The dilation of A by B is the set of all x displacements such that \hat{B} and A overlap by at least one nonzero element. Set B is commonly referred to as *structuring element*.

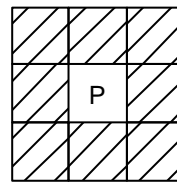
The *erosion* of A by B is denoted $A \ominus B$ and is defined as

$$A \ominus B = \{x \mid (B)_x \subseteq A\} \quad (2.25)$$

The erosion of A by B is the set of all points x such that B , translated by x , is contained in A .



(a) 4-connected neighbors.



(b) 8-connected neighbors.

Figure 2.3 Neighborhood connectedness.

Opening of set A by structuring element B , denoted $A \circ B$, is defined as

$$A \circ B = (A \ominus B) \oplus B \quad (2.26)$$

Shrinking is accomplished by removing all border pixels according to a specified neighborhood connectedness. The neighborhood connectedness for a center pixel P is defined in Figure 2.3. Likewise, *swelling* is accomplished by expanding all border pixels by one pixel to the background according to neighborhood connectedness. *Deletion of isolated points* is to remove any single pixels in an image.

The application of these processing steps to plant images was important in order to obtain more realistic leaf shapes in the binary images since these binary images were eventually used for distinguishing tomato plants and weeds. In this research, plant identification was mainly based on their shape analysis. More realistic shape was important in order to produce correct results.

3. MATERIALS AND METHODS

3.1 Overview of the research and the robotic weed control system

The objective of this research was to develop a real-time robotic weed control system for seedline weeds and to replace costly hand weeding with robotic weed control using machine vision, pattern recognition, and robotics. All research was conducted with juvenile processing tomato plants in commercial tomato fields in northern California during 1996 and 1997. Tomato images were acquired in 5 different fields in 1996 and in 8 different fields in 1997.

The UC Davis Robotic Cultivator (Slaughter et al., 1997) was utilized as a guidance system to center the prototype system over the row, Figure 3.1. A seedline image was acquired for the recognition of plant leaves. Plant leaves were identified using their shape characteristics and the Bayesian discriminant function. After distinguishing tomato plants from weeds, the main computer sent the weed locations to a precision chemical application system which opened a corresponding spray valve in the valve/nozzle array when it was above a weed plant. The precision chemical application system consisted of 8 valves/nozzles, which due to space constraints were aligned in two rows (four in each row). For precise herbicide application, the image was subdivided into a spray grid of 8 rows by 18 columns. Each cell in the spray grid corresponded to a 1.27 cm by 0.64 cm region on the seedbed. The precision spray system was capable of applying chemical herbicides to individual spray cells providing a level of precision unparalleled in existing agricultural spray systems.

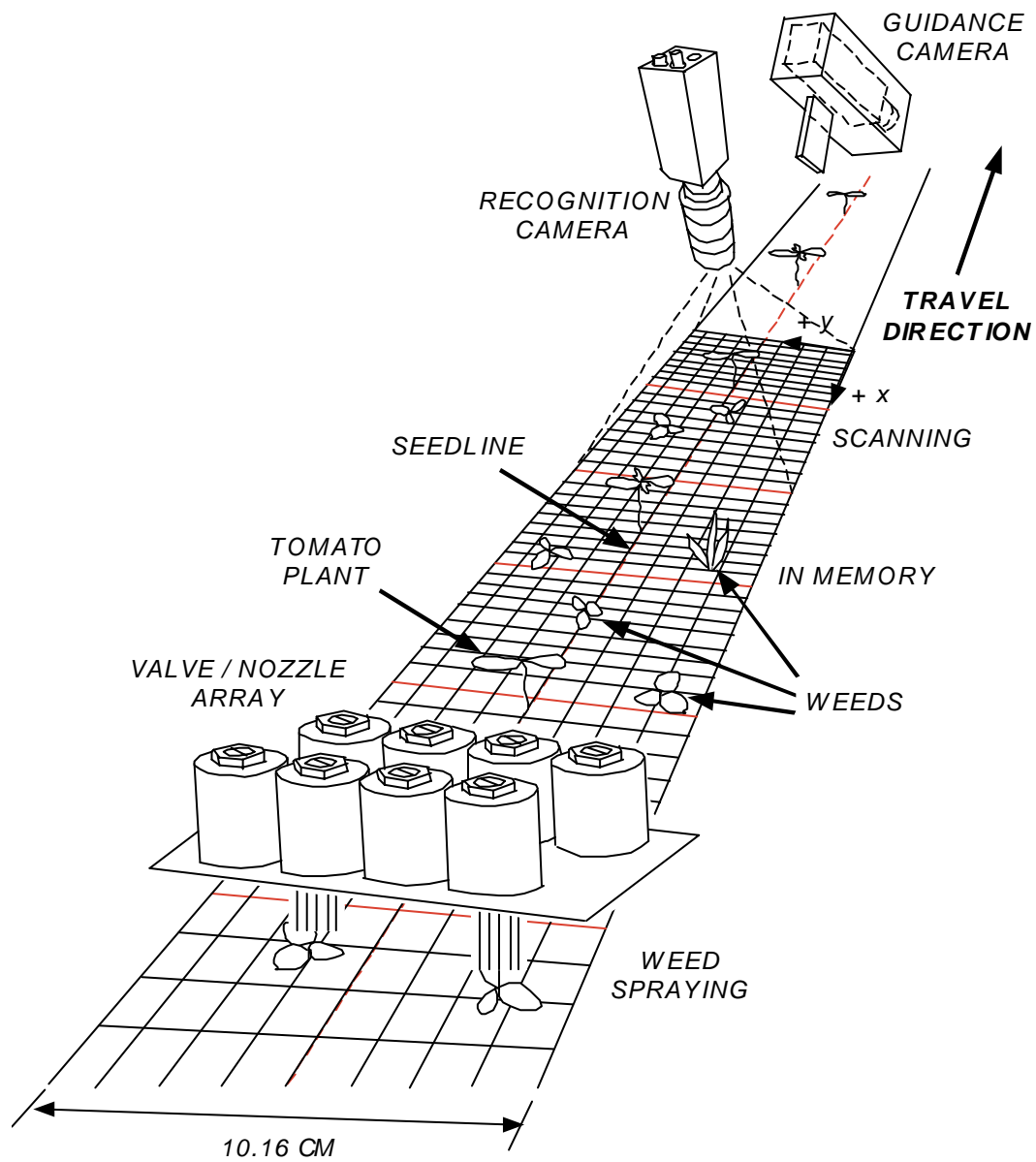


Figure 3.1 Concept drawing of the robotic weed control system.

3.2 Robotic weed control system

3.2.1 Image Processing Hardware

A real-time computer vision system was developed for plant species recognition. The following equipment was used to implement the real-time computer vision system, Figure 3.2. The system consisted of an IBM compatible computer, a set of three real-time image processing boards, a camera, and a camera controller board.

A SHARP GPB-2 board was used as the basis for the hardware portion of the image processing system. The GPB-2 board performed image processing subroutines using twelve 512 row by 512 column by 8 bit image memory banks.

A SHARP INCARD was used to acquire the three simultaneous NTSC (National Television System Committee) color video inputs (red, green, and blue video signals) for color image processing. The INCARD was directly interfaced to the GPB-2, which allowed real-time color image transfer to occur without sending the image across an external computer bus.

A SHARP AUXLUT card was used for real-time image segmentation. Like the INCARD, the AUXLUT card was also directly interfaced to the GPB-2 for real-time operation. This card provided look-up table (LUT) conversion for data coming from one, two, or three GPB memory buffers. The LUT output was placed in another GPB memory buffer for further processing. The AUXLUT card had two 16 bit inputs to 8 bit LUTs, each made up of two 65K bytes by 4 static RAMs.

A R, G, B color video camera (Model 2222-1340/0000, Cohu, Inc.) was used for high resolution NTSC color video image acquisition. The camera was equipped with a zoom lens (C mount manual lens, 12.5 - 75 mm focal length, F1.8) and was mounted with

the lens 32 cm above the seedbed to provide a field of view of the seedline 11.43 cm wide by 10.16 cm long. A camera shutter speed of 1/500 second was used to prevent blurring due to motion of the tractor and wind.

A multifunction I/O board (Model CIO-DAS 1600, ComputerBoards, Inc.) was used to generate an asynchronous reset signal for the camera. The Microsoft C Compiler Version 7.0 was used to develop all computer vision software and a Dell Dimension XPS Pro200n with 200 MHz Pentium Pro CPU was used for the computer.

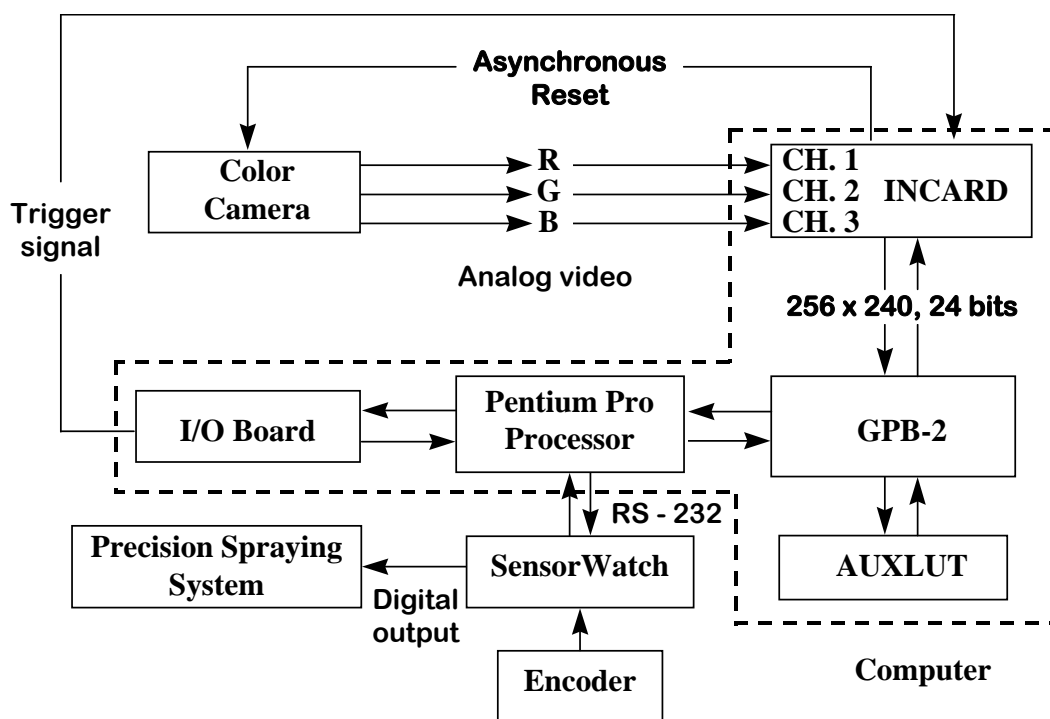


Figure 3.2 Schematic of the real-time machine vision system.

The prototype robotic weed control system is shown in Figure 3.3. The UC Davis Robotic Cultivator (Slaughter et al., 1997) was utilized as a guidance system (camera #1 and Alloway cultivation tool in Figure 3.3) to center the prototype system over the row.

Each step, from field image acquisition to the weed control actuation, was synchronized using a heavy duty optical incremental encoder (Model HR6251000000A, Danaher Controls) attached to a gage wheel on the toolbar of the cultivation sled. The cultivation sled was mounted on the three-point hitch of a tractor (Model 7800, John Deere Co.). The encoder generated a pulse whenever the tractor moved 0.13 mm forward. The encoder output was 1000 pulses/revolution and in order to obtain higher resolution from the encoder, an intermediate pulley was used between the encoder and the axle of the gage wheel, Figure 3.4. A new image was taken every 879 pulses (11.43 cm) by triggering an asynchronous reset signal to the color camera.

A microcontroller (SensorWatch™, TERN Inc., Davis, CA) was used to count the number of encoder pulses. The location of the prototype was obtained by monitoring this counter. The SensorWatch™ is a C/C++ programmable 16-bit controller designed for data acquisition and control applications. The SensorWatch™ communicated via a RS-232 serial port to the computer containing the image processing boards.

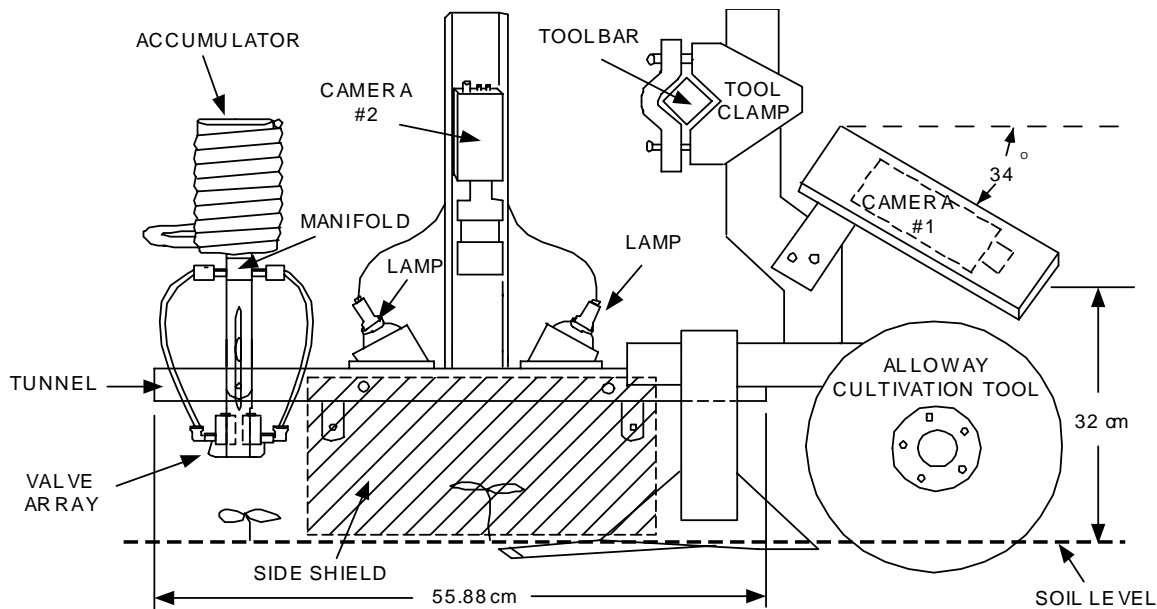
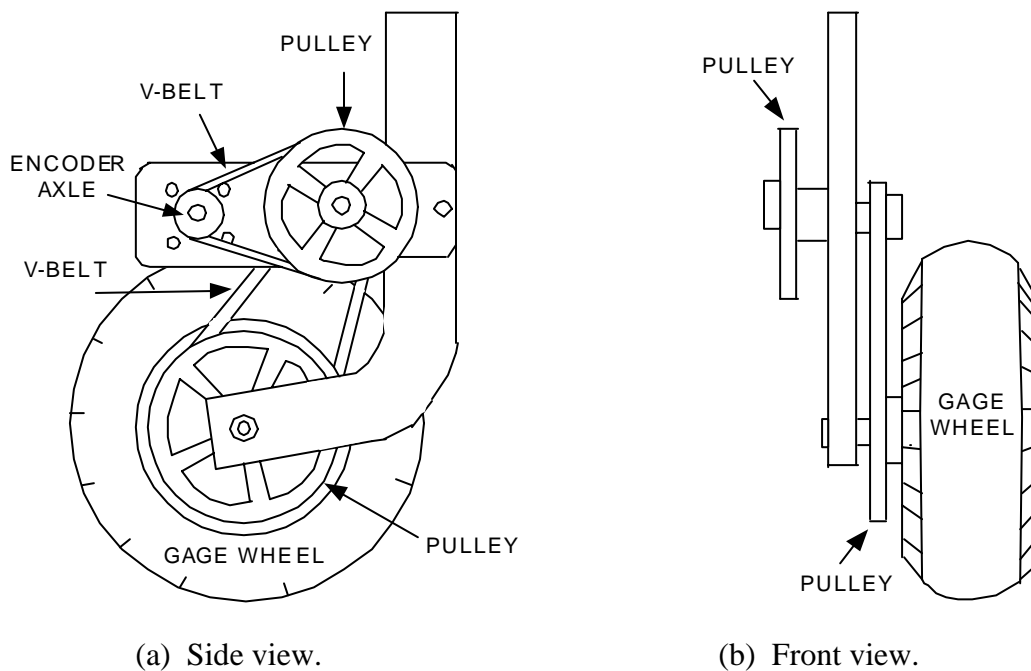


Figure 3.3 The prototype robotic weed control system.



(a) Side view.

(b) Front view.

Figure 3.4 Gage wheel with encoder and intermediate pulley.

3.2.2 Outdoor image acquisition and uniform illumination device

The first step in a machine vision application is image acquisition. Frequently the better the images are, the better the results obtained. Most machine vision systems have been used in controlled environments with consistently positioned objects, good illumination, and objects easily distinguished from the background. Outdoor image acquisition is quite different, especially for systems working with biological materials. Natural objects are not consistently positioned or aligned. Their size and shapes vary widely. Sunlight changes from time to time according to environmental conditions. The color of the background (soil) also changes greatly from time to time and from field to field. Plants are easily moved by wind.

Uniform scene illumination is critical for successful color-based pattern recognition of plant species (Tian, 1995). In this research, in order to eliminate many of the problems described above, a uniform illumination device was developed for the prototype using a specially designed cultivation tunnel which was attached to the frame of the 'Alloway' cultivation tool.

The illumination tunnel was composed of a C channel beam (10.16 cm wide, 60.96 cm long, and 0.48 cm thick), two dichroic halogen lamps (Iwasaki Electric Co. Model MR16CG, 12Vdc, and 50W), two flashed opal optical diffusers (Oriol Model No. 48030, 5.08 cm diameter and 0.22 cm thick), two metal side shields and front and rear rubber flaps. The side shields and rubber flaps were designed to block the sunlight and to minimize the amount of soil falling on top of the tomato plants during cultivation. The two lamps were positioned at 30° relative to the optical axis of the camera, to provide uniform illumination in the camera's field of view.

Figure 3.3 shows the uniform illumination device attached to the end of the UC Davis Robotic Cultivator. This device provided more uniform lighting and helped in acquiring better images. The cultivation/image acquisition tunnel was used to reduce wind movement of plants.

3.2.3 Precision spraying system

After distinguishing tomatoes from weeds, an image was divided into a spray grid of 8 rows by 18 columns (Figure 3.5) in order to determine which spray valves/nozzles were to be activated to spray weeds. The 8 rows corresponded to the eight solenoid valves, and each image was divided into 18 columns to maximize the precision of the spray application. This grid size gives each spray cell a size of 1.27 cm by 0.64 cm. Once the center of the weed leaves were located by the real-time computer vision system, the information was sent to the spray valve microcontroller.

A *valve byte*, a *sync byte*, and a *position byte* were used to control the serial communication between the image processing computer and the microcontroller. The *valve byte* was one byte of information indicating which valves the microcontroller opened or closed while the valve array was traveling over the corresponding weed location in an image. The *sync byte* was used to tell the microcontroller the time at which a new picture was taken. The *position byte* was information which specified the order of the *valve bytes*, indicated by column numbers (i.e., 1 - 18), and was used as a check against communication errors.

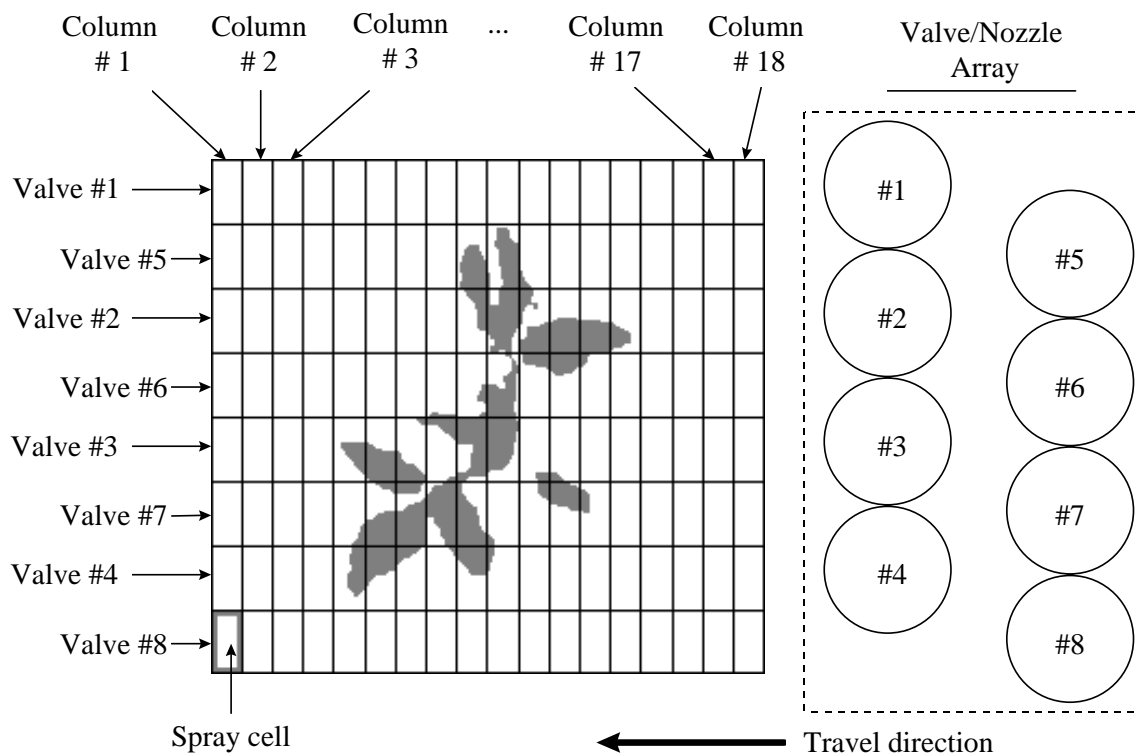


Figure 3.5 Spray grid of an image (Top view).

Figure 3.5 illustrates column numbers and corresponding valve numbers for the spray cells in an image. The eight valves and nozzles were aligned in two rows (four in each row) due to physical constraints in order to allow the entire 10.16 cm wide seed line to be sprayed when they were opened at the same time.

3.3 Image analysis

3.3.1 Image processing algorithm - Programs for field tests

Figure 3.6 shows a flowchart of the software for the entire robotic weed control system. The software is composed of image acquisition, binarization of a raw color image, binary image pre-processing, plant recognition, and detection of weed location procedures.

3.3.3.1 Image acquisition preparation

The serial communication link between the image processing computer and the microcontroller was used to synchronize image acquisition and weed mapping, Figure 3.7. At the beginning of the processing loop, the microcontroller sent an 'a' character to the image processing computer to initiate the acquisition of a new picture. The microcontroller subsequently sent an 'a' character whenever the tractor traveled 11.43 cm forward.

Since the number of objects in an image varied, processing time for every image was not always constant. Two conditions could cause the image processing computer to fail to finish processing the current image in time to acquire the next image: excessive tractor speed, or an unusually high number of weeds in the image.

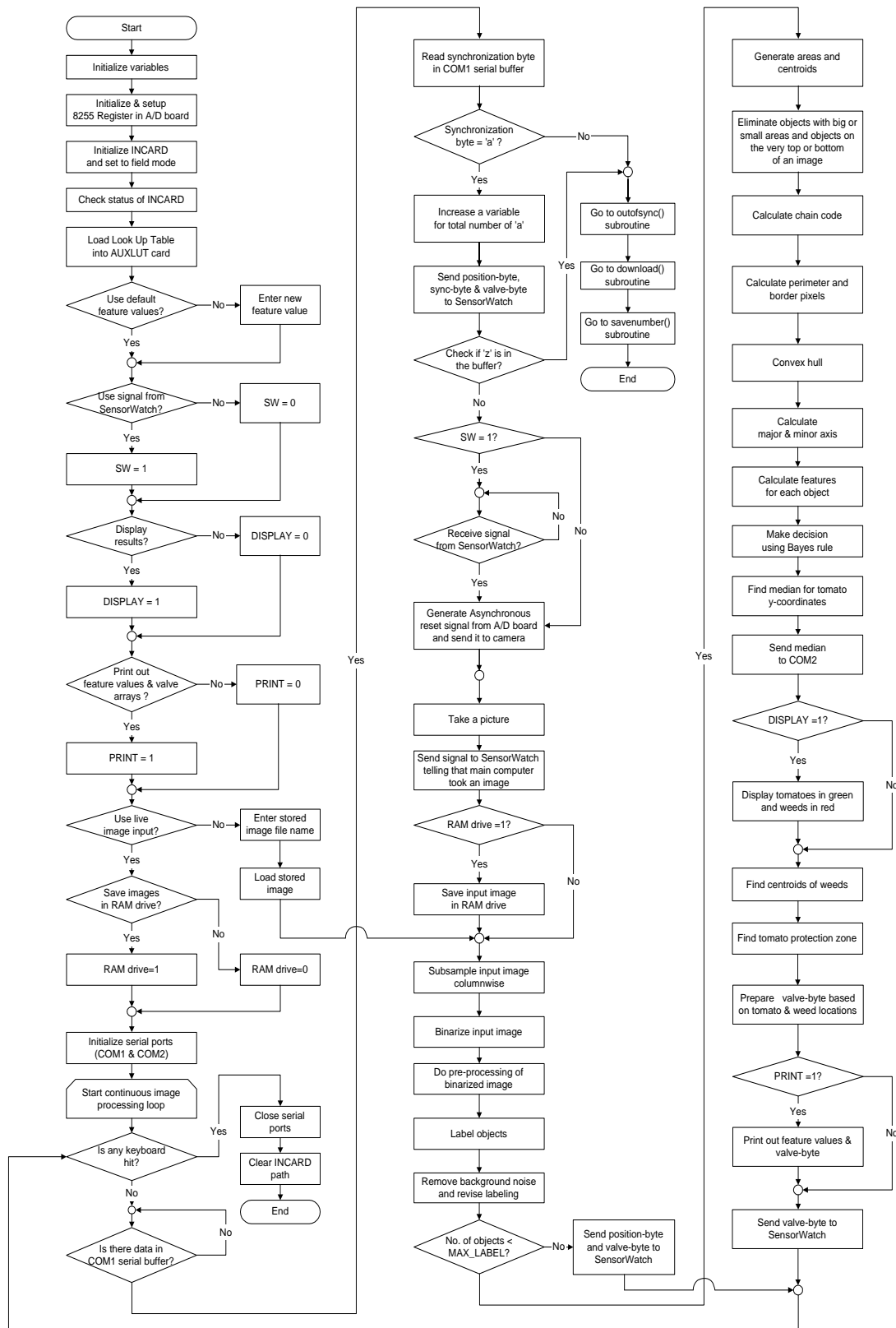


Figure 3.6 Flow chart for robotic weed control system software.

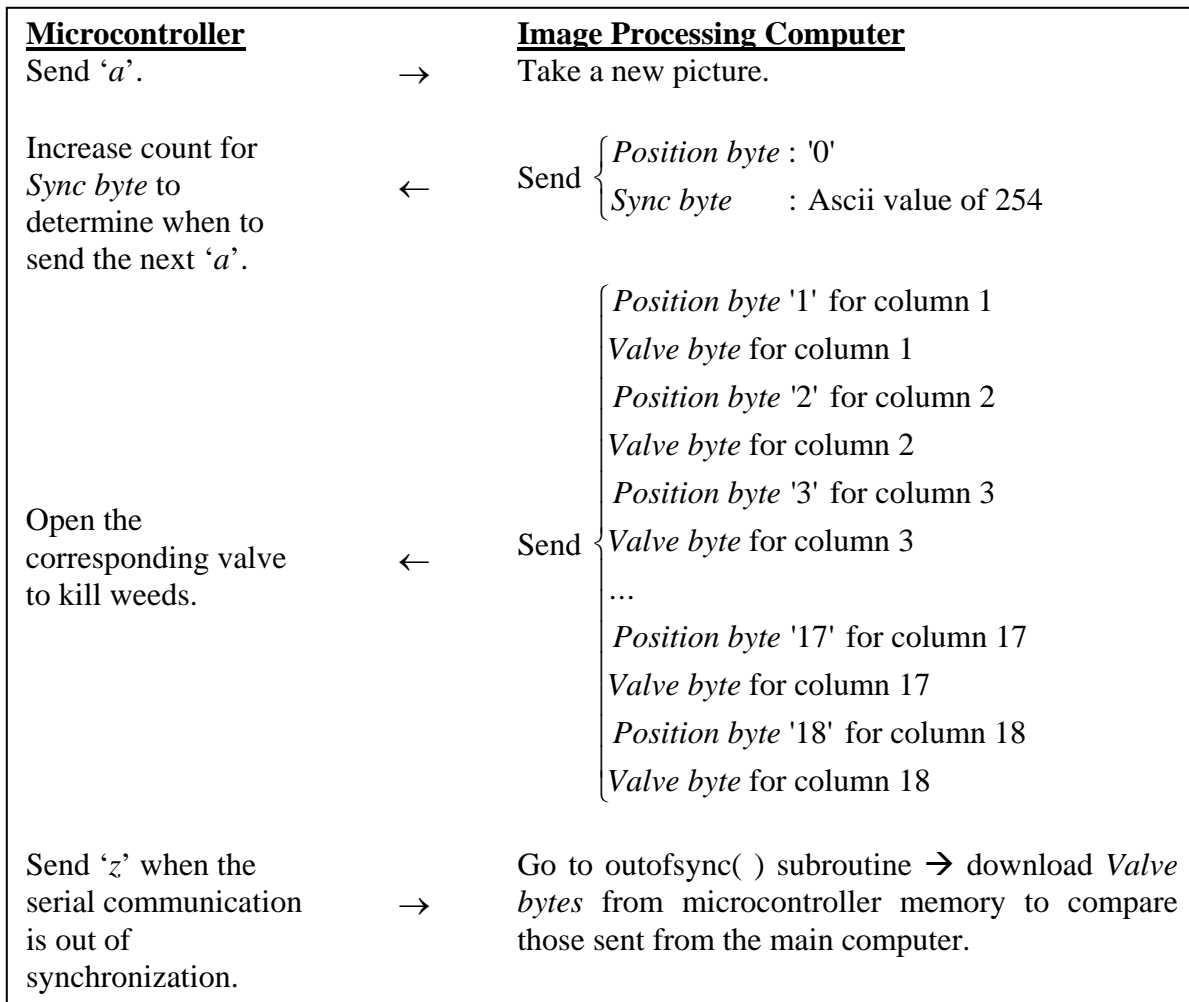


Figure 3.7 Serial communication between the image processing computer and the microcontroller.

To prevent the microcontroller from getting out of synchronization and spraying incorrect weed locations, an additional serial communication step was used before acquiring an image. Figure 3.7 illustrates the serial communication between the image processing computer and the microcontroller. The image processing computer checked its serial buffer and counted the number of 'a' characters present. If more than one 'a' character were present in the buffer, the last 'a' character was used to take a picture and

the remaining 'a' characters in the buffer were used to send eighteen zeros (indicating no known weeds) to the microcontroller for each character.

The main computer sends a '0' *position byte* and an ASCII value of 254 as a *sync byte* to the microcontroller after reading an 'a' character from the serial buffer. The *sync byte* was used by the spray valve microcontroller to determine the exact seedbed location where the image was acquired. If the prototype has advanced part way into a new image field of view when the *sync byte* is received, then the columns in the spray grid passed by are automatically filled with zeros (indicating no known weeds). This process allowed the two computer systems to maintain synchronization during field operation.

The *sync byte* and *position byte* were used by the microcontroller to detect communication errors. If a communication error occurred, a 'z' character was sent to the image processing computer, and the image processing computer stopped processing and downloaded the valve array data, which was stored in the microcontroller memory to allow the operator to diagnose the error.

3.3.3.2 Image acquisition

An asynchronous reset signal was used to acquire an image at a specified instant in time from a moving vehicle in order to obtain an image at the desired region of the seedline. When the INCARD received an active low TTL signal from the I/O board which was at least 70 μ s long (Figure 3.8 (a)), it reset the camera by sending an asynchronous reset signal (Figure 3.8 (b)) and counted three NTSC video field times. The first video field was not acquired due to unwanted accumulation on the CCD (Charge Coupled Device) between prolonged triggers. The second and third fields were

captured from the camera and the INCARD retrieved the second field. Thus, there was a one field delay in image acquisition from the time the asynchronous reset signal was sent.

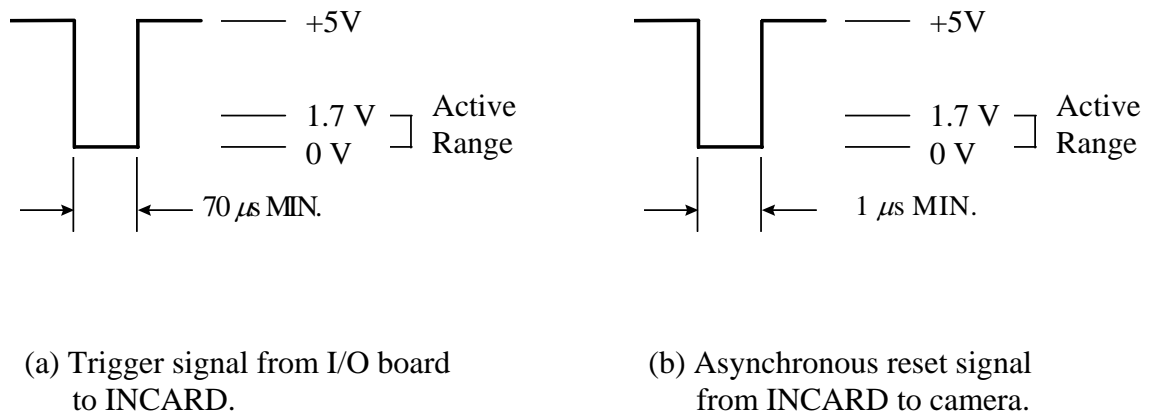


Figure 3.8 Asynchronous reset signal.

Upon receiving an asynchronous reset signal from the I/O board, the image processing computer sent a *position byte* and a *sync byte* to the microcontroller telling it that it had received the asynchronous reset signal to synchronize them. To acquire an image of the seed line, the red, green, and blue RS-170 interlaced video signals were input to the INCARD. The INCARD digitized a single field of the interlaced video picture and subsampled the input image columnwise to eliminate blurring due to camera motion. This process took 33.0 ms (16.7 ms for acquiring one *field* and 16.7 ms for subsampling) and allowed saving of 16.7 ms for acquiring an image. If a whole video frame (512 pixels by 480 pixels) image were taken and every row and column subsampled, the image acquisition would take 49.6 ms (33.3 ms for acquiring a frame and 16.3 ms for subsampling). The actual field of view of the camera was about 11.43 cm x 10.16 cm and was digitized into 256 x 240 pixels.

3.3.3.3 Color image segmentation with Bayes' decision rule

After a color image was digitized and stored as a 24 bit color image in computer memory, the image was segmented into plant and non-plant regions using color information such as hue, saturation and intensity. Color images provide more information than gray scale images. Even though the gray level of two objects are similar, their color can be different. Since all image processing steps in this study were conducted on binary images, image segmentation was an essential preliminary step.

In order to make binary images, many methods could be used such as thresholding, supervised learning, statistical classification, and unsupervised learning. In this research, a statistical classification method (Bayes' minimum-risk classifier) was applied to build a color look-up-table (LUT) which was loaded into the AUXLUT card where it was used for real-time conversion of the color image to a binary image (white for plant leaves and black for background). This process took less than 3 ms using the AUXLUT card in a Dell Dimension XPS Pro200n computer with a 200 MHz Pentium Pro processor (please refer to chapter 4.4).

Utilizing a computer look-up-table is a common way to implement high-speed binary conversion from a color image. There are different ways to make LUTs such as using red (R), green (G), and blue (B) color components, normalized red (r), green (g), and blue (b) color components, which are defined in Eq. (3.1) - (3.3), or hue (H), saturation (S), and intensity (I) components. Hue is defined as an attribute associated with the dominant wavelength in a distribution of light frequencies. Hue represents dominant color as perceived by an observer. Saturation refers to relative purity or the amount of white light mixed with a hue. Intensity represents the perceived luminance. H ,

S , and I are transformed from R , G , and B using Eq. (3.4) - (3.6) (Gonzalez and Woods, 1993: p. 234).

$$r = \frac{R}{R + G + B} \quad (3.1)$$

$$g = \frac{G}{R + G + B} \quad (3.2)$$

$$b = \frac{B}{R + G + B} \quad (3.3)$$

$$H = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \right\} \quad (3.4)$$

$$S = 1 - \frac{3}{(R + G + B)}[\min(R, G, B)] \quad (3.5)$$

$$I = \frac{1}{3}(R + G + B) \quad (3.6)$$

A preliminary study was conducted to determine which color space, components, or number of classes produce a LUT with the best segmentation performance. In order to investigate LUT performance, 10 training images and 15 validation images were randomly chosen among the images acquired in 5 different commercial processing tomato fields in northern California. From the 10 training images, sample pixel values of all components in RGB , rgb , and HSI color spaces were obtained for three different classes, tomato plants, weeds, and background. Table 3.1 shows the number of pixels for each class from these training images.

Table 3.1 Number of pixels used to evaluate LUT performance.

Class	Number of pixels
Tomato	3242
Weed	6155
Background	8583

Histograms of these pixels are shown in Figures 3.9 - 3.11 for the three different color spaces. In these histograms, we can see the relative frequency of pixels of each class for the different components of each color space. This information can be used to help identify which components might be useful in distinguishing one class from another.

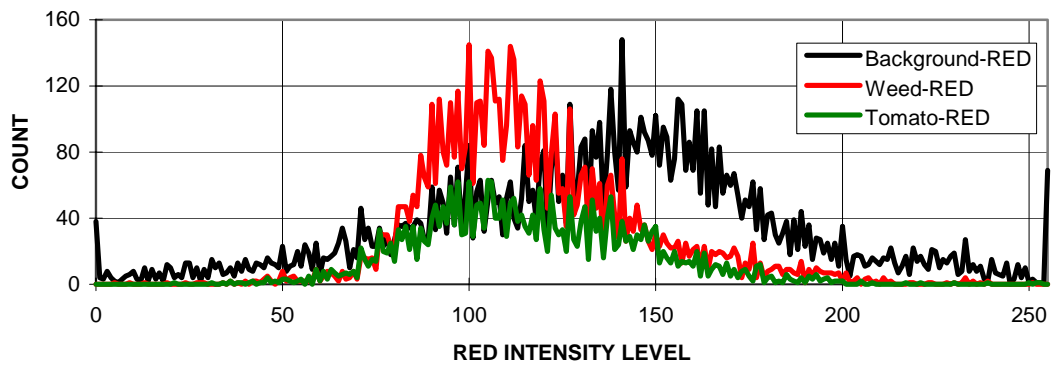
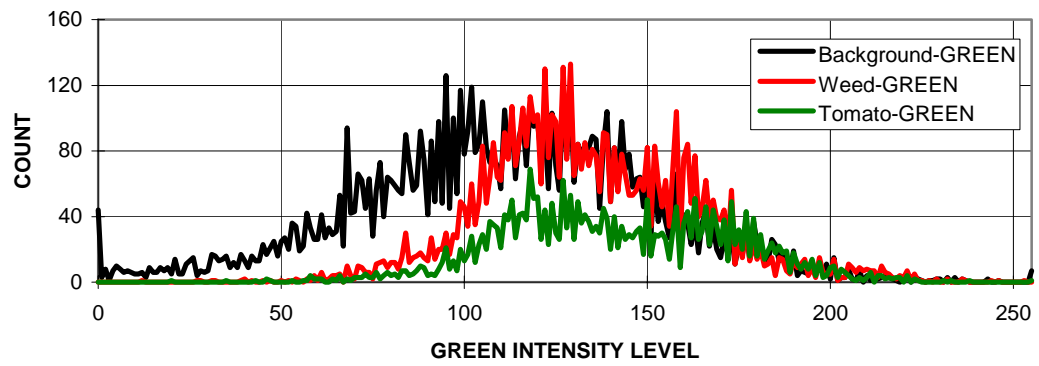
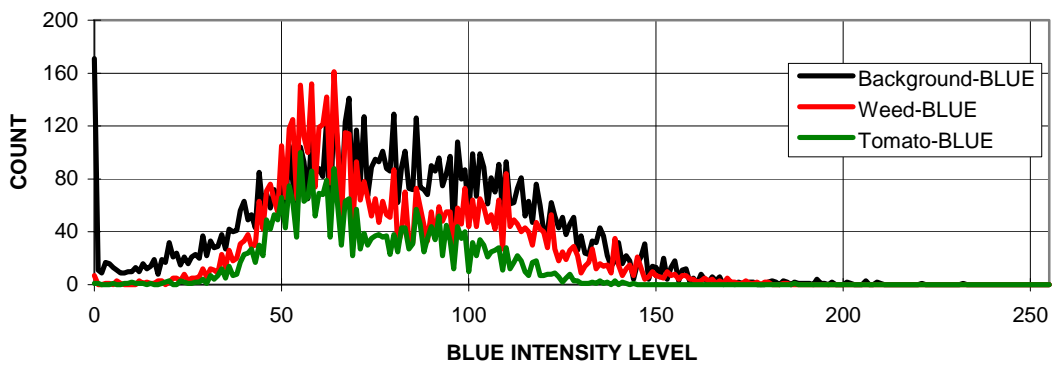
From the histogram in *RGB* color space (Figure 3.9), there was considerable overlap between plant (tomato and weed) and background classes, and there was complete overlap between tomato plants and weeds. This implied that a simple color classifier could not be used to distinguish tomato plants from weeds. The least overlap between plant and soil occurred in the *R* component, then in the *G* component, and was greatest in the *B* component. We can see that the blue channel had a lower intensity than red and green channels.

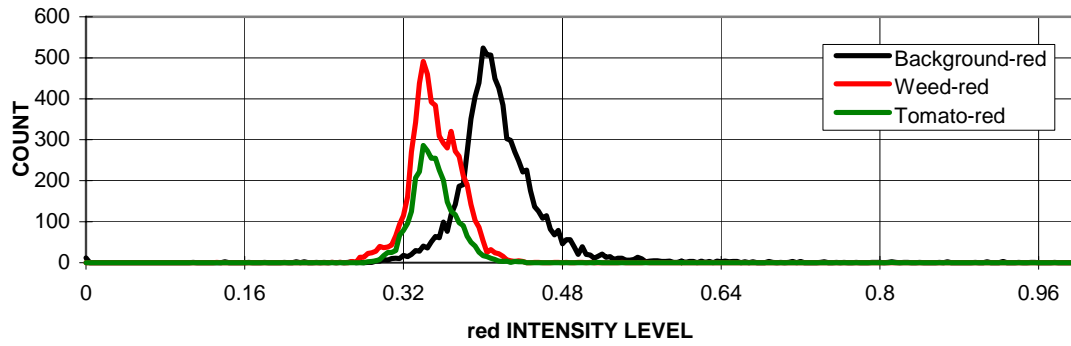
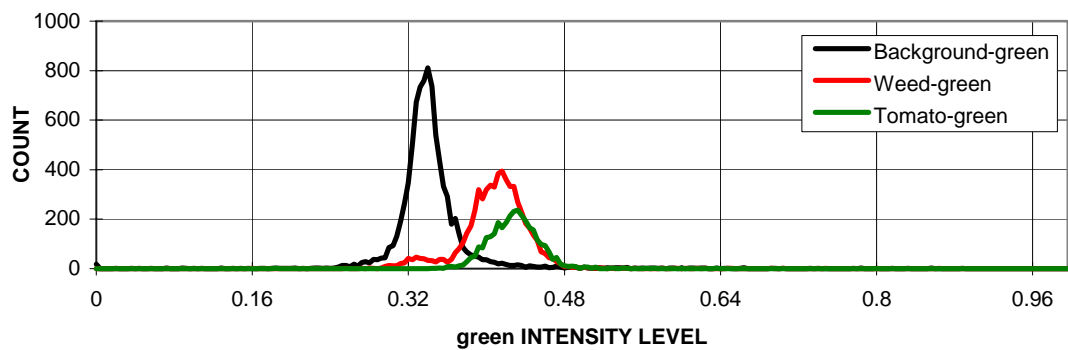
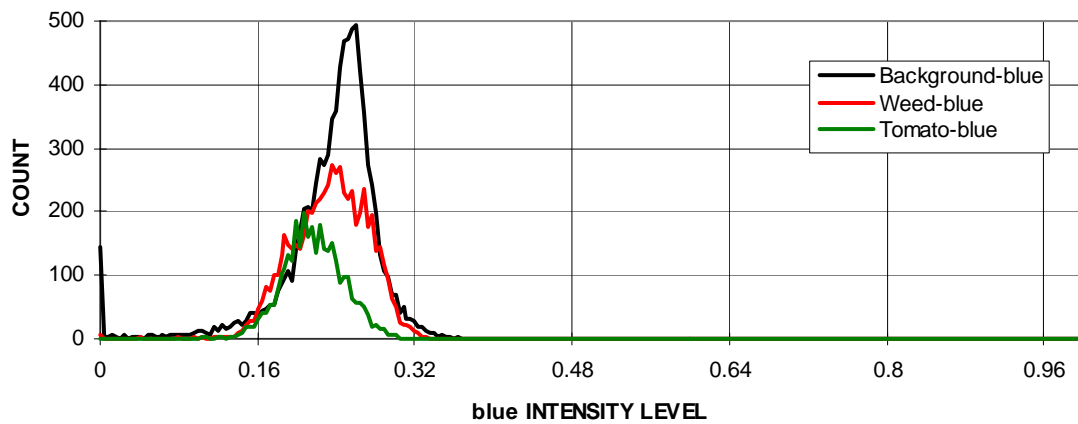
In *rgb* color space (Figure 3.10), there was also considerable overlap in the *b* component between plants (tomato plants and weeds) and background, less overlap in the *r* component, and separation in the *g* component between plants and background. In this color space, tomato plants and weeds were also completely overlapped.

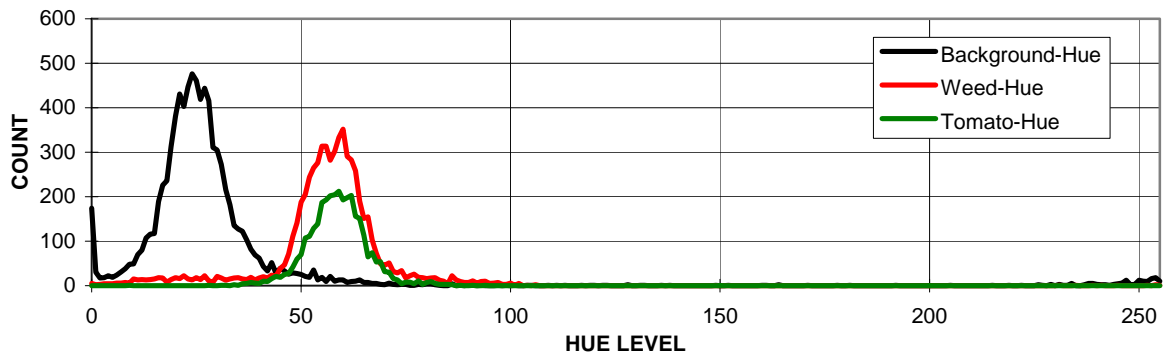
In *HSI* color space (Figure 3.11), the hue component was a good characteristic to use in distinguishing plants from background. However, in this color space, tomato plants and weeds were also completely overlapped.

Figures 3.12 - 3.14 show 2-D and 3-D scatter plots of a randomly chosen 100 pixel sample for each class (tomato plants, weed, and background) in different color spaces. In *RGB* color space (Figure 3.12), there was a separation between plants and background in *R-G* plane. However, in the *R-B* plane and *B-G* plane, all three classes were overlapped. In *rgb* color space (Figure 3.13), separation lines could be drawn in all three planes. This was because there existed a color difference between plants and backgrounds, not an intensity difference. In *rgb* color space, it only takes two components to know all 3 components, because $r + g + b = 1$. Since there was a separation in *R-G* plane, any two components of *r*, *g*, and *b* also shown this separation since the intensity was not important.

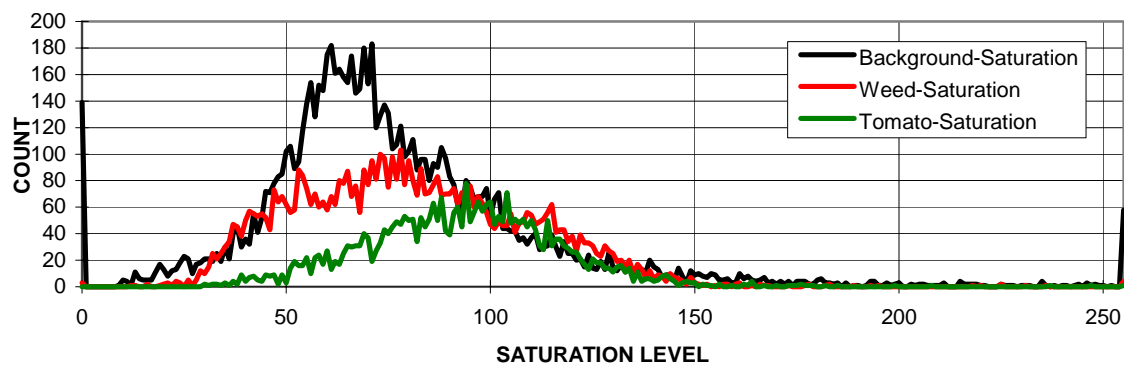
In *HSI* color space, there was no distinction between plants and background in the *I-S* plane (Figure 3.14 (d)). This shows that the predominant characteristic in distinguishing plants and background is hue.

(a) *R*.(b) *G*.(c) *B*.Figure 3.9 Histogram of *R*, *G*, and *B* component of training images.

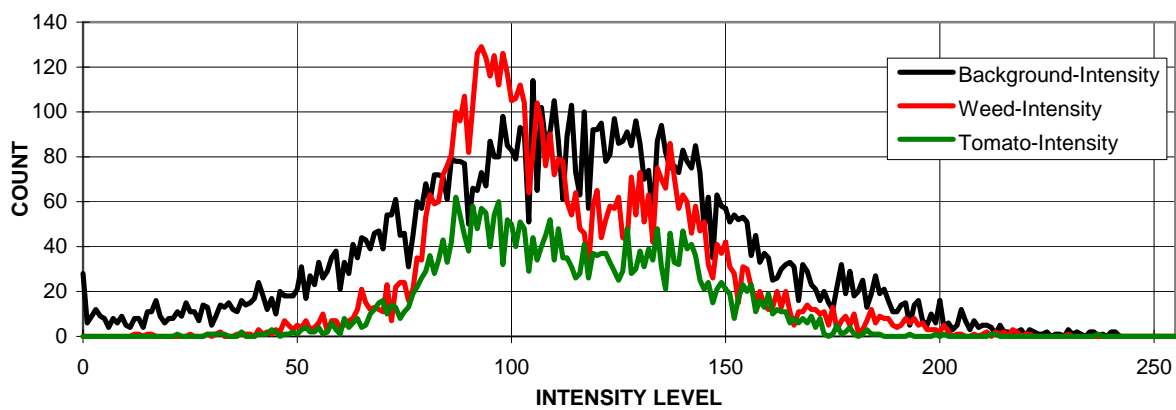
(a) *r*.(b) *g*.(c) *b*.Figure 3.10 Histogram of *r*, *g*, and *b* component of training images.



(a) Hue.



(b) Saturation.



(c) Intensity.

Figure 3.11 Histogram of hue, saturation, and intensity component of training images.

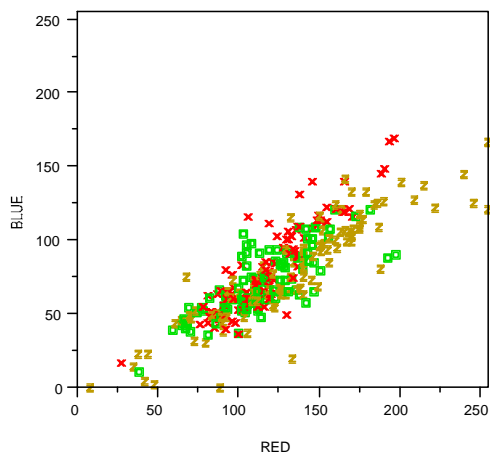
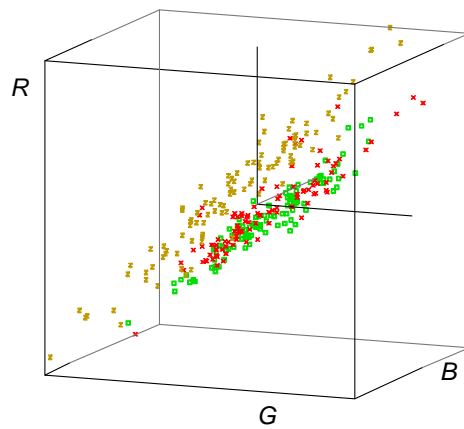
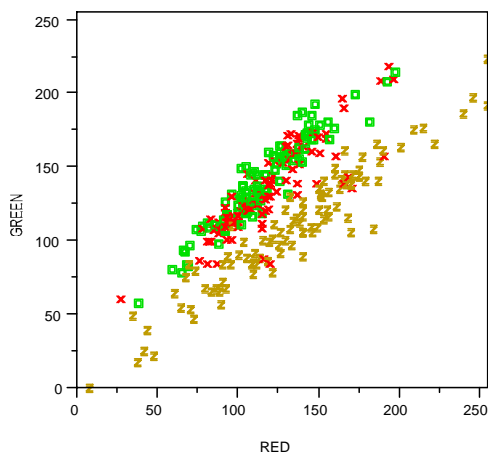
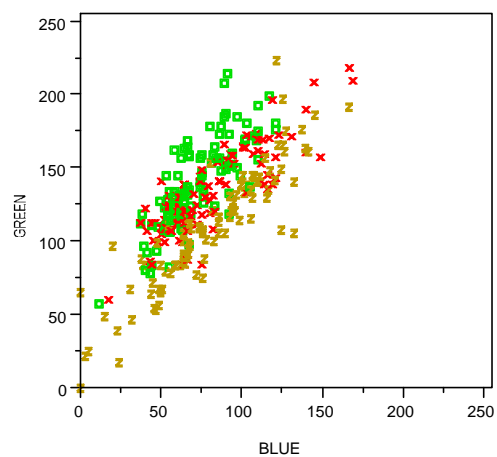
(a) R vs. B .(b) R vs. G vs. B .(c) R vs. G .(d) B vs. G .

Figure 3.12 Scatter plots of tomato plants (\square), weed (\times), and background (\mathbb{Z}) pixel samples in $R - G - B$ color space.

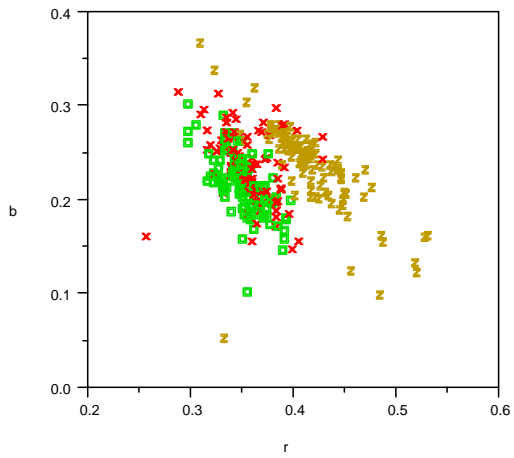
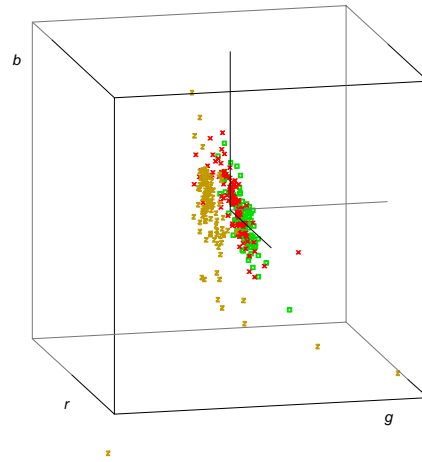
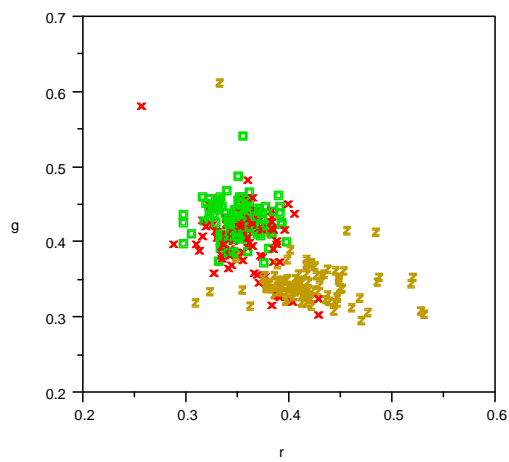
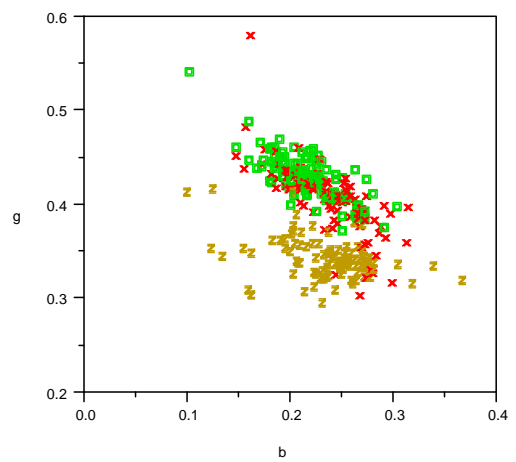
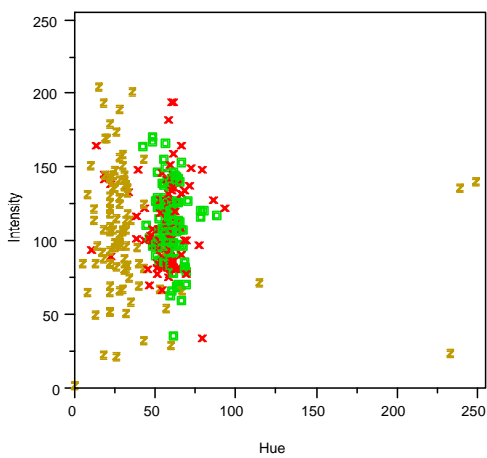
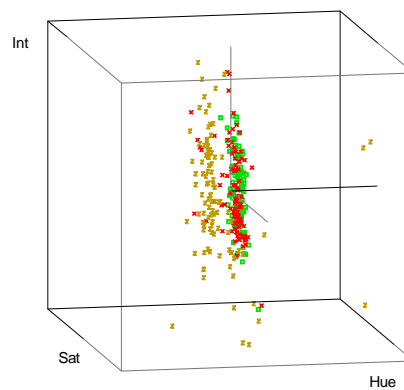
(a) r vs. b .(b) r vs. g vs. b .(c) r vs. g .(d) b vs. g .

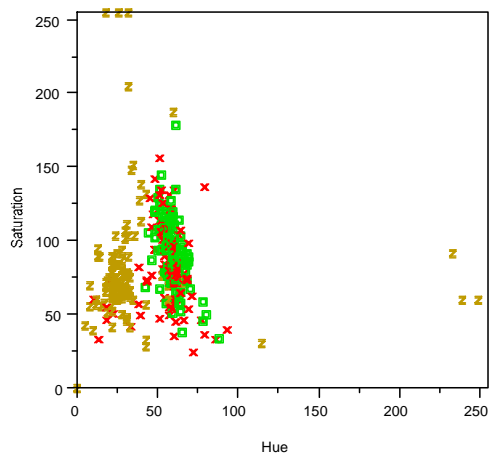
Figure 3.13 Scatter plots of tomato plants (\square), weed (\times), and background (\mathbf{z}) pixel samples in $r - g - b$ color space.



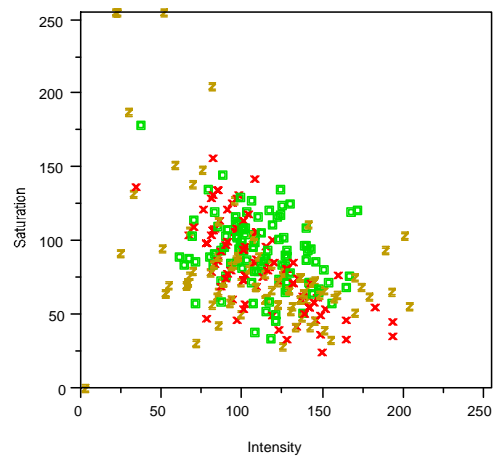
(a) Hue vs. Intensity



(b) Hue vs. Saturation vs. Intensity



(c) Hue vs. Saturation.



(d) Intensity vs. Saturation.

Figure 3.14 Scatter plots of tomato plants (\square), weed (\times), and background (\mathbf{z}) pixel samples in Hue - Saturation - Intensity color space.

3.3.3.4 Look-Up Table construction for image binarization

A look-up table (LUT) is a list of output values for the corresponding different inputs. By using LUT, many computational steps can be saved at a cost of additional memory usage. The AUXLUT card has the ability of taking 16 bits from three different input images (Figure 3.15) and producing an 8 bit output image in less than 3 ms making it very suitable for real-time use.

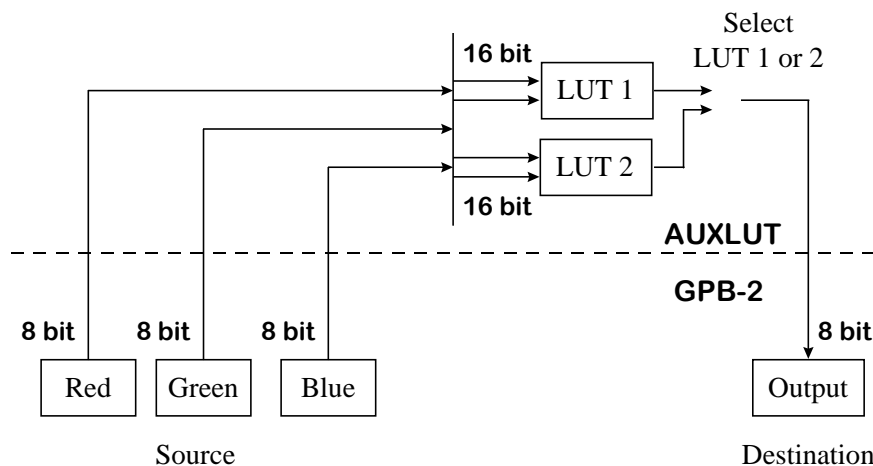


Figure 3.15 Structure of AUXLUT card.

Since the AUXLUT card takes only a 16 bit input, the 24 bit R , G , and B inputs needed to be reduced to 16 bits. An input of 5 bits of R , 6 bits of G , and 5 bits of B was chosen due to the fact that the green component was more important in dealing with plant objects and more sensitive to human eyes.

The following pseudo code generates R , G , and B from a single LUT *address*, which increments from 0 to 65535 (SHARP, 1994).

$$\begin{aligned}
 R = & \text{(Double)} ((((\text{address} \ll 17) \gg 31) \ll 7) & (3.7) \\
 & | (((\text{address} \ll 20) \gg 31) \ll 6) \\
 & | (((\text{address} \ll 23) \gg 31) \ll 5) \\
 & | (((\text{address} \ll 26) \gg 31) \ll 4) \\
 & | (((\text{address} \ll 29) \gg 31) \ll 3));
 \end{aligned}$$

$$\begin{aligned}
 G = & \text{(Double)} ((((address \ll 16) \gg 31) \ll 7) & (3.8) \\
 & | (((address \ll 19) \gg 31) \ll 6) \\
 & | (((address \ll 22) \gg 31) \ll 5) \\
 & | (((address \ll 25) \gg 31) \ll 4) \\
 & | (((address \ll 28) \gg 31) \ll 3) \\
 & | (((address \ll 31) \gg 31) \ll 2));
 \end{aligned}$$

$$\begin{aligned}
 B = & \text{(Double)} ((((address \ll 18) \gg 31) \ll 7) & (3.9) \\
 & | (((address \ll 21) \gg 31) \ll 6) \\
 & | (((address \ll 24) \gg 31) \ll 5) \\
 & | (((address \ll 27) \gg 31) \ll 4) \\
 & | (((address \ll 30) \gg 31) \ll 3));
 \end{aligned}$$

When R and B were entered into the LUT as 5 bits, 3 bits were lost. Thus, new R and B values were obtained by dividing by 8 or by shifting R value three bits to the right ($= R \gg 3$). Likewise, a new G value was obtained by dividing by 4 ($= G \gg 2$). If we took integer values of R , G , and B , then new R' , G' , and B' values were:

$$R' = \text{Int}(R / 8) \times 8 \quad (3.10)$$

$$G' = \text{Int}(G / 4) \times 4 \quad (3.11)$$

$$B' = \text{Int}(B / 8) \times 8 \quad (3.12)$$

The maximum possible integer value of R' and B' was 248 ($= \text{Int}(255 / 8) \times 8$) since the maximum value of R and B was 255. The maximum possible integer value of G' was 252 ($= \text{Int}(255 / 4) \times 4$).

All possible combinations of red, green and blue components of a pixel were then converted into hue, saturation and intensity components using Eq. (3.13) - (3.15).

$$\text{Intensity} = \left(\frac{R' + G' + B'}{3} \right) \times 255.0 / 249.0 + 0.5 \quad (3.13)$$

$$\text{Saturation} = \begin{cases} 0.0 & \text{if Intensity} < 16.0 \\ \left(255.0 \times \left(1.0 - \frac{\text{Min}(R', G', B')}{\text{Intensity}} \right) \right) + 0.5 & \text{Otherwise} \end{cases} \quad (3.14)$$

$$\text{Hue} = \frac{\text{HueTemp}}{360.0} \times 255.0 \quad (3.15)$$

Where HueTemp =

$$\begin{cases} 0.0 & \text{if Saturation} < 16.0 \text{ or} \\ & \text{Intensity} < 16.0 \\ & \text{Otherwise} \\ 90.0 - \tan^{-1} \left(\frac{2R' - G' - B'}{(G' - B' + 0.01) / \sqrt{3}} \right) \times \frac{180.0}{\pi} + \text{Factor} + 1.5 & \text{Factor} = \begin{cases} 180.0 & \text{if } B' > G' \\ 0.0 & \text{Otherwise} \end{cases} \end{cases}$$

For calculating Intensity, the value of 249 was used to scale up to 255, since the maximum value of $(R' + G' + B') / 3$ would be 249 ($= (248 + 252 + 248) / 3$).

For binarization, hue, saturation, and intensity components of pixel values were used as the input feature vector \mathbf{x} in the Bayesian discriminant function (Eq. (3.17)) for the LUT since they showed great possibility for binarization. (This choice also turned out to be the best one according to the analysis of LUT performance in Chapter 4.1.) The estimates of the mean, \mathbf{m}_k and the covariance, Σ_k of hue, saturation, and intensity component were also calculated to be used as inputs to the discriminant function.

$$\mathbf{x} = \begin{bmatrix} \text{Hue} \\ \text{Saturation} \\ \text{Intensity} \end{bmatrix} \quad (3.16)$$

$$d_k(\mathbf{x}) = -\frac{1}{2} \{ (\mathbf{x} - \mathbf{m}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{m}_k) + \ln |\Sigma_k| \} + \ln P(\omega_k) \quad (3.17)$$

After calculating $d_k(\mathbf{x})$ for all input feature vectors, a class was assigned to each feature vector by the following decision rule. For binarization of a raw color image, there were only two classes, i.e., plant leaf or background.

$$\text{Assign } \mathbf{x} \text{ to } \omega_i \text{ if } d_i(\mathbf{x}) > d_k(\mathbf{x}), \quad \forall k \neq i \quad (3.18)$$

The resulting LUT was written as a SHARP 8 bit image format (512 x 128) in raster scan order, from the top left to the bottom right. The contents of the LUT at each address was set to 0 if \mathbf{x} was assigned to background and to 255 if \mathbf{x} was assigned to plant. Thus, the LUT took R, G, B as inputs and produced a binary image. For a given pixel with R, G, B components, a corresponding *address* in the LUT was obtained using Eq. (3.7) - (3.9), and the value at the address was read from the LUT as an output to the binary image. An example is shown in Figure 3.16. If a pixel with $R = 8, G = 4,$ and $B = 0$ goes through the LUT, an *address* obtained is 5 (Eq. (3.7) - (3.9)). Then, the output 255 in this example, is read from the corresponding pixel location (i.e., row 0 and column 5).

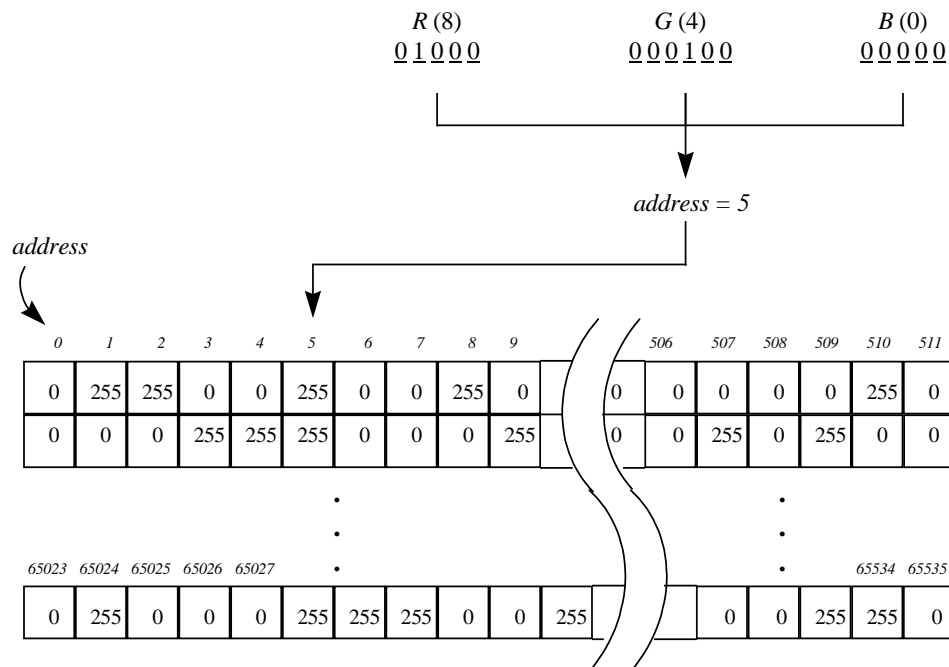


Figure 3.16 An example of LUT reading.

Typical field images acquired under different illuminations, stages of growth, or times of day, were used to make a LUT for binarization. The R , G , B values of each object class (e.g., plant leaves or background) were obtained from these images to calculate representative mean and covariance matrices for both classes.

3.3.3.5 Look-up table and image quality

Building a good LUT is a fundamental and important step in pattern recognition since binary images are used as the basis for all subsequent image processing steps. The better a LUT is, the less image processing steps are needed. Better binary images produce better shape recognition results. In this chapter, LUT performance was evaluated using binary images created using different LUTs based upon different color spaces and using 2 or 3 input classes (i.e., 2 classes: plants and background, or 3 classes: tomato plants, weeds, and background).



(a) Color image

(b) Plant-only image

(c) Background-only image

Figure 3.17 Separation of plant-only and background-only images from a color image.

The first step was to make a plant-only image and a background-only image from a color image taken in a commercial processing tomato field, Figure 3.17. In this step, Adobe Photoshop (Adobe systems, Inc., version 4.0) was used to manually separate plants from background to create the images shown in Figure 3.17 (b) and (c). It is a tedious, time consuming and difficult process to manually classify every pixel in an image containing 61440 pixels (plant and background together). This process could be a

source of error since it was difficult to separate all plant pixels from the background and the author tried to make as few errors as possible in this step.

The number of plant pixels in the plant-only image were counted and denoted as n_1 . The number of non-black background pixels in the background-only image were also counted and denoted as n_2 . These plant-only and background-only images were passed through a set of LUTs and binary images were created. The plant-only binary image was used to evaluate the number of plant pixels correctly passed through LUT. The number of non-black pixels in the segmented plant-only binary image were counted and denoted as n_3 . The background-only binary image was used to evaluate the number of non-black background pixels incorrectly passed through the LUT. The number of non-black pixels in the segmented background-only image were counted and denoted as n_4 .

A total of 12 LUTs (Table 3.2) were made from 10 training images using different numbers of input classes in different color spaces. The names of LUTs were assigned according to the color components and number of input classes used. For example, LutRGBc2 was a LUT made with R , G , and B components in RGB color space and two input classes: plants (tomato plants and weeds together) and background, whereas LutRGBc3 was made with three input classes: tomato plants, weeds, and background. The LUT, Lutrgc3, was a LUT made with r and g components and three input classes (tomato plants, weeds, and background).

Note that when making a LUT in rgb color space, if all r , g , and b components were used as inputs to LUT, then the covariance matrix became singular ($r + g + b = 1$) so that there was no inverse for the covariance matrix, and thus Bayes' rule (Eq. (2.11))

could not be applied to build a LUT. In this case only the r and g components were used to build a LUT instead of r , g , and b components.

Table 3.2 List of LUT names used to determine performance.

Component used	2 Class (Plant, Background)	3 Class (Tomato plant, Weed, Background)
R, G, B	LutRGBC2	LutRGBC3
R, G	LutRGC2	LutRGC3
r, g	LutrgC2	LutrgC3
H, S, I	LutHSIC2	LutHSIC3
H, S	LutHSC2	LutHSC3
H	LutHC2	LutHC2

The values of $n_1 - n_4$ for each LUT were used to evaluate its performance according to its total error rate (Eq. (3.21)). The LUT error has two components, the % of plant pixels not correctly classified (Eq. (3.19)) and the % of background pixels not correctly classified (Eq. (3.20)). Ideally the total error rate would be zero.

$$\text{Plant error rate} = \frac{(n_1 - n_3)}{n_1} \times 100 (\%) \quad (3.19)$$

$$\text{Background error rate} = \frac{n_4}{n_2} \times 100 (\%) \quad (3.20)$$

$$\text{Total error rate} = \text{Plant error rate} + \text{Background error rate} \quad (3.21)$$

$$= \left[\frac{(n_1 - n_3)}{n_1} + \frac{n_4}{n_2} \right] \times 100 (\%)$$

The LUT performance was evaluated by two methods. The first method (Method I) was to use the binary images from LUTs with no operation done to them. The second method (Method II) was to evaluate LUTs with single point noise (one pixel noise in the background) removed from the background-only binary images. This method was used since a single point noise could be easily removed by the Sharp board.

3.3.3.6 Binary image pre-processing

After color segmentation, the binary image was enhanced through a series of image processing steps including erosion, dilation, shrinking and swelling to remove any digitization noise and to smooth the object edges to provide a more accurate shape for leaf recognition. More specifically, the following series of operations were done in the order listed in Table 3.3. This combination of operations was chosen among other combinations by trial and error to get the most accurate shape. Then, all objects in the image were labeled from 1 to the number of objects for further processing.

Table 3.3 Pre-processing steps for a binary image.

Operation	Connectedness
shrink	4
delete isolated points	-
binary erode	8
binary dilate	8
swell	8
shrink	4
swell	8
shrink	4
swell	8
shrink	8

Figure 3.28 (b) page 122, shows the segmented and enhanced field image of a tomato seedling and weeds.

3.3.3.7 True leaf recognition: Curvature calculation

In an effort to recognize tomato seedling true leaves, the curvature of the leaf boundaries was studied. True leaves of tomato plants usually have some notches (concave regions) in their boundaries (Figure 3.18), while most of the weed leaves are round and convex. The curvature at a point P is defined as the value of the derivative of the polar angle of the unit tangent vector with respect to arc length (Bers, 1969). As shown in Fig. 3.19, the curvature κ , at a point P is calculated using Eq. (3.22).



Figure 3.18 True leaves of tomato plant.

$$\kappa = \frac{d\theta}{ds} \quad (3.22)$$

where θ = polar angle of the unit tangent vector in radians, and s = arc length.

$$\theta_1 = \tan^{-1} \left(\frac{y_2 - y_1}{x_2 - x_1} \right) \text{ (rad)} \quad (3.23)$$

$$\theta_2 = \tan^{-1} \left(\frac{y_3 - y_2}{x_3 - x_2} \right) \text{ (rad)} \quad (3.24)$$

$$\kappa = \frac{\theta_2 - \theta_1}{\Delta S} \times 1000 \quad (3.25)$$

where (x_i, y_i) = average of coordinates at segment i ,

θ_i = tangent angle at segment i (rad), and

ΔS = arc length (pixel)

The curvature was calculated using a discrete version of Eq. (3.22) from the boundary pixels of each object using a segment and gap as shown in Eq. (3.23) - (3.25). A pixel on the bottom right of the object was used as the starting pixel and the curvature was calculated in a counterclockwise direction along the boundary. The segment (SEG) was defined as a group of contiguous pixels used to estimate the average value of the x and y coordinates of a boundary point using a boxcar smoothing operator. The gap (GAP) was defined as the number of boundary pixels between two consecutive segments and was used to tune the discrete derivative to accentuate concave regions of a desired size. In an object, the point P was repeatedly moved by an offset (C_OFFSET) along the leaf boundary to calculate the curvature along the entire boundary until all boundary pixels were used. The default values used for SEG, GAP, and C_OFFSET were 3, 5, and 5 respectively. These default values were found by trial and error in order to accurately find the leaf boundary curvature and to minimize the effect of digitization noise at the boundary. The 1000 multiplicative factor in Eq. (3.25) was used to enlarge the curvature value.

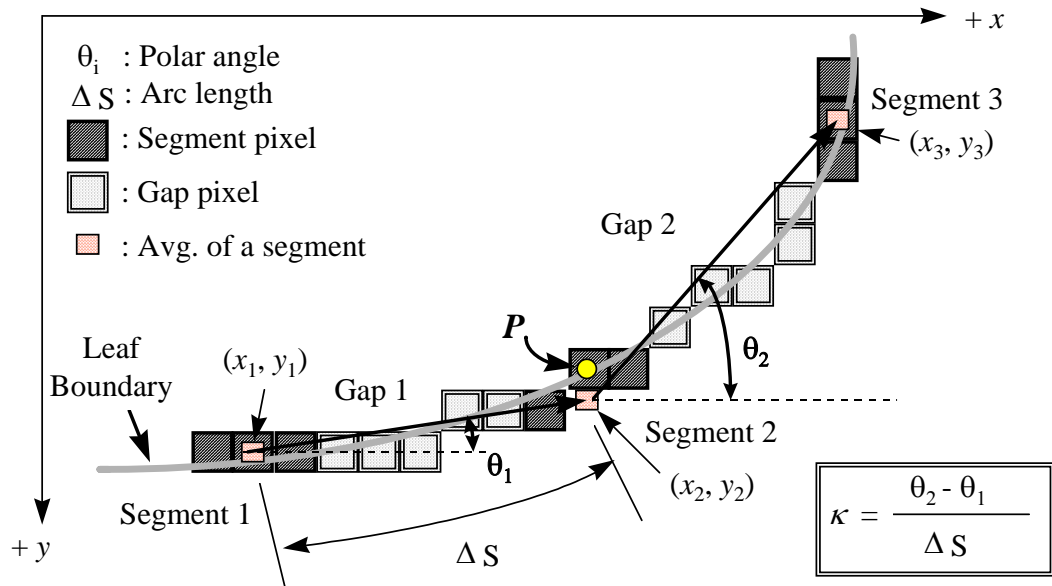


Figure 3. 19 Curvature Calculation.

Using curvature, the following features were calculated for each leaf in order to build a Bayesian classifier to distinguish tomato plants and weeds: arc length (S), polar angle of the tangent vector (θ), occurrence of negative curvature (NEG), maximum curvature (MAXC), minimum curvature (MINC), average curvature (AVGC), average of absolute value of curvature (ABSAVGC), standard deviation of curvature (STDEVC), sum of radius of curvature (SUMINV), sum of absolute value of radius of curvature (ABSUMINV), ratio of area to average of the absolute values of curvature (ATC), ratio of compactness to average of the absolute values of curvature (CTC), ratio of elongation to average of the absolute values of curvature (ETC), ratio of difference of MAXC & MINC to sum of MAXC & MINC (MTMC), and ratio of perimeter to average of the absolute values of curvature (PTC). A factor of 100 was multiplied in calculating CTC and ETC to increase their magnitude.

$$ATC = \frac{AREA}{ABSAVGC} \quad (3.26)$$

$$CTC = \frac{100 \text{ CMP}}{ABSAVGC} \quad (3.27)$$

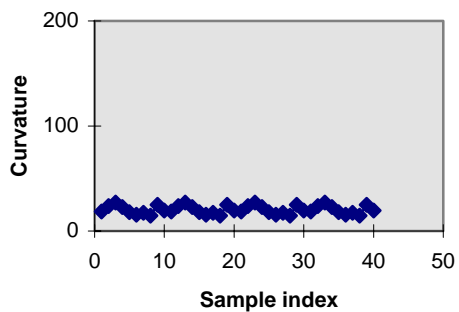
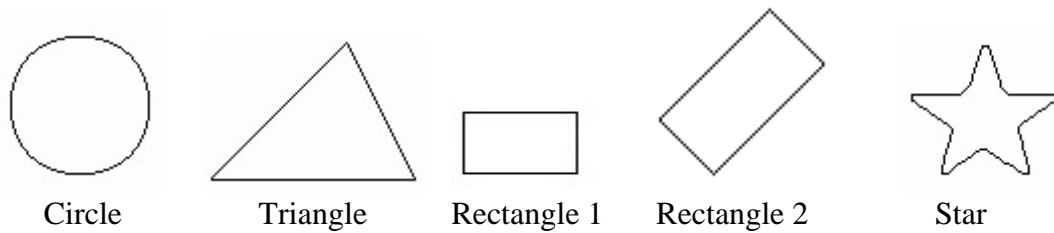
$$ETC = \frac{100 \text{ ELG}}{ABSAVGC} \quad (3.28)$$

$$MTMC = \frac{MAXC - MINC}{MAXC + MINC} \quad (3.29)$$

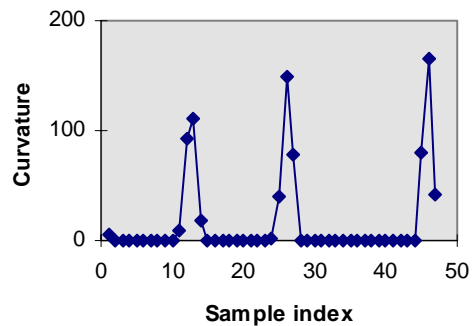
$$PTC = \frac{PERIM}{ABSAVGC} \quad (3.30)$$

$$SUMINV = \sum \frac{1}{\kappa} \quad (3.31)$$

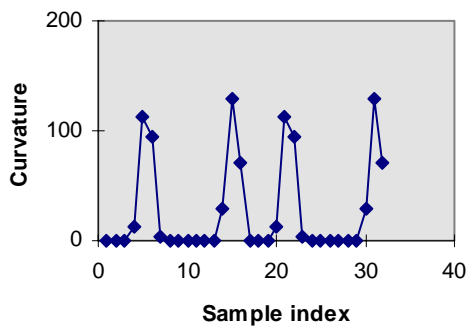
Figures 3.20 and 3.21 show sample curvature feature values for standard shapes and tomato cotyledon, tomato true leaf and cotyledon of nightshade weed, respectively.



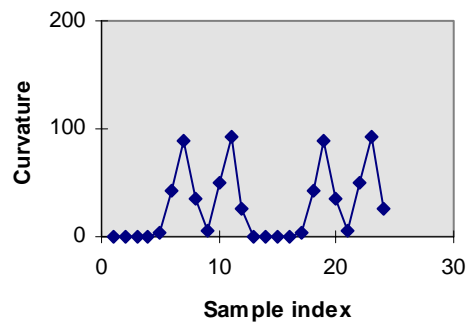
(a) Circle.



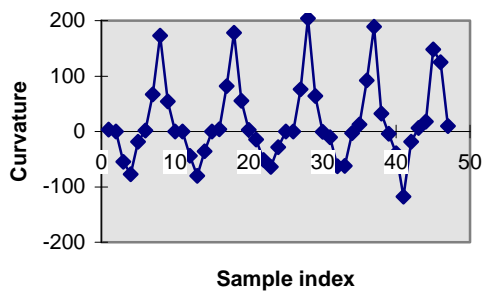
(b) Triangle.



(c) Rectangle 1.

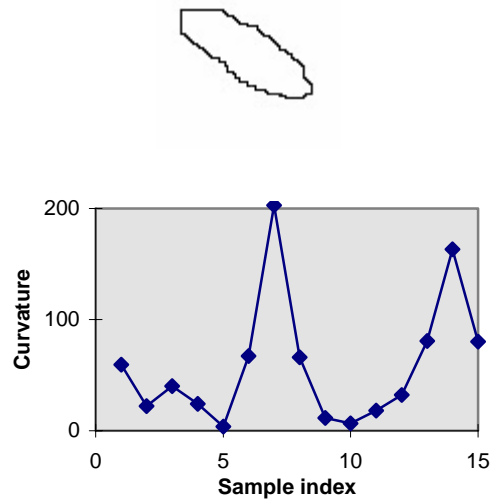


(d) Rectangle 2.

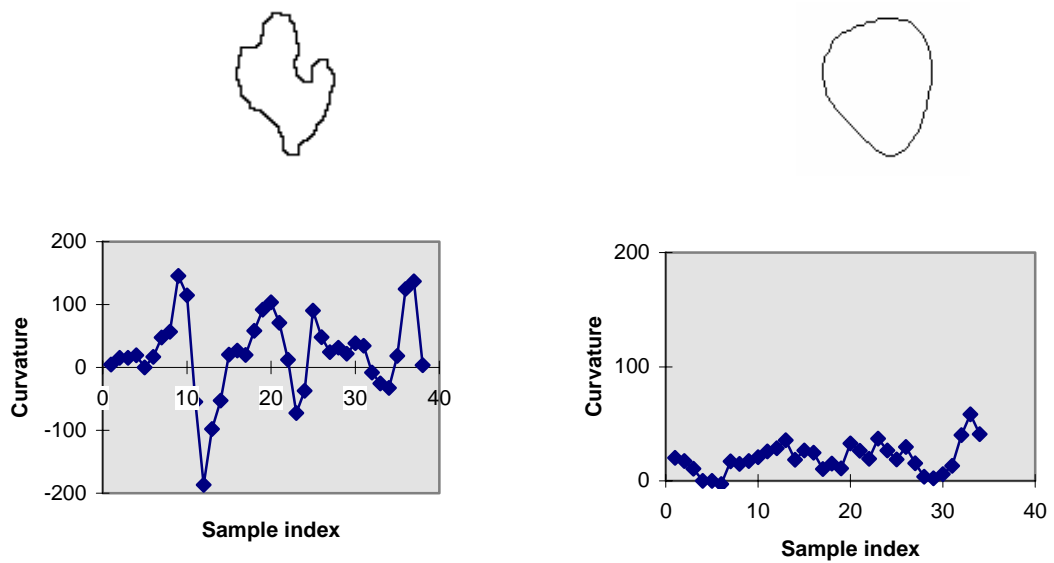


(e) Star.

Figure 3.20 Sample curvatures of different shapes.



(a) Cotyledon of tomato plant.



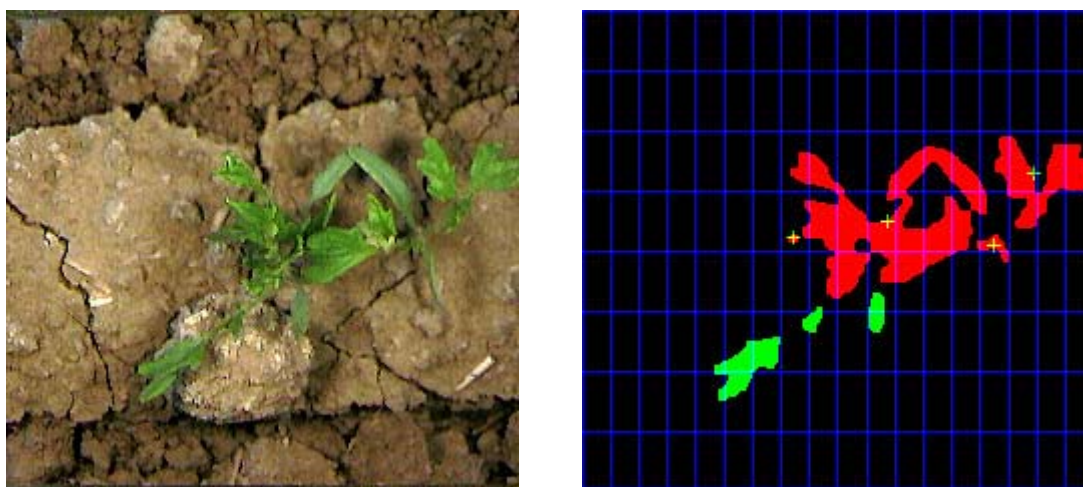
(b) True leaf of tomato plant.

(c) Cotyledon of nightshade weed.

Figure 3.21 Sample curvatures of tomato cotyledon, tomato true leaf and nightshade cotyledon.

3.3.3.8 Partially occluded leaves: Watershed algorithm

Occlusion has been one of the most difficult obstacles in machine vision since occluded objects are difficult to identify. Multiple occluded objects appear as one object in the segmented binary image, producing an unusual set of feature values. Figure 3.22 shows an example of unrecognized tomato leaves due to occlusion. Occluded objects need to be separated before extracting their features. One of the faster and more widely used methods to segment occluded objects is the watershed algorithm.



(a) Occluded tomato cotyledons. (b) Unrecognized cotyledons shown in red.

Figure 3.22 Incorrectly classified tomato cotyledons due to occlusion.

The watershed method was developed in 1977 and many variations of the method have been tried. The watershed method can be used on either binary or gray scale images. In this research, a binary image was used to separate partially occluded plant leaves as illustrated in Figure 3.23. The algorithm used in this research was based on the algorithm by Vincent and Soille (1991). The basic idea for this algorithm is that a digital image is considered as a topographical surface and the value of each pixel represents the

elevation at that point. An imaginary hole is pierced at each regional minimum (Figure 3.23) in the topographical surface and then the entire surface is “flooded” with water through the imaginary hole. Water fills the surface beginning the lowest regional minima and a “dam” is built at the point where two different regional minima would merge. After the entire image is “filled with water”, the “dams” provide separation lines for each regional minima. These “dams” are called the watersheds of the image. Each regional minimum is called a “catchment basin”.

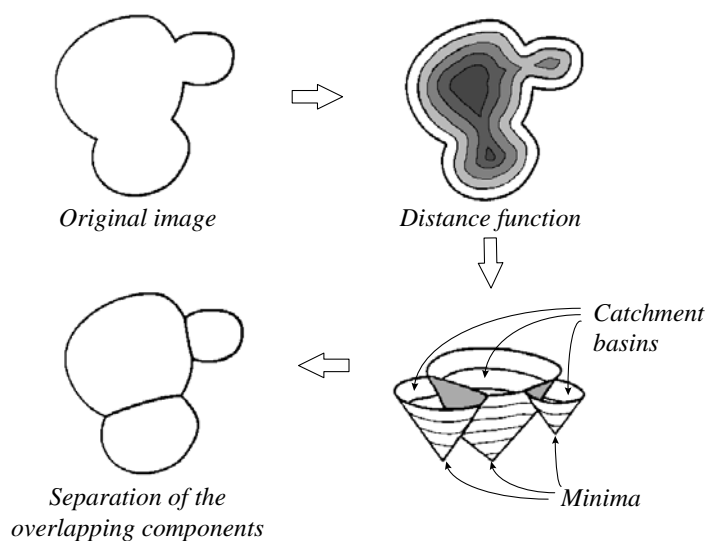
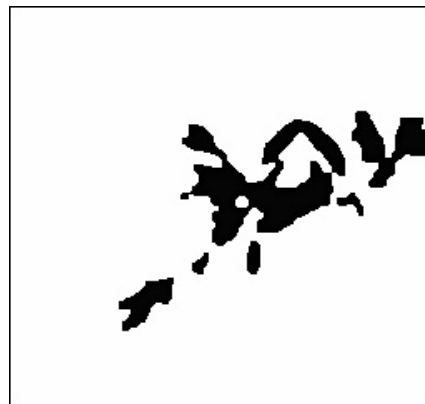


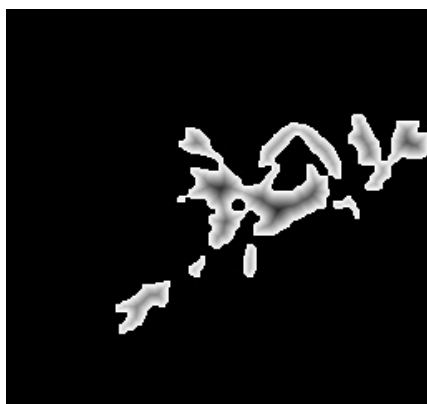
Figure 3.23 Binary separation by watersheds of the opposite of the distance function.
(Source: Vincent and Soille (1991))



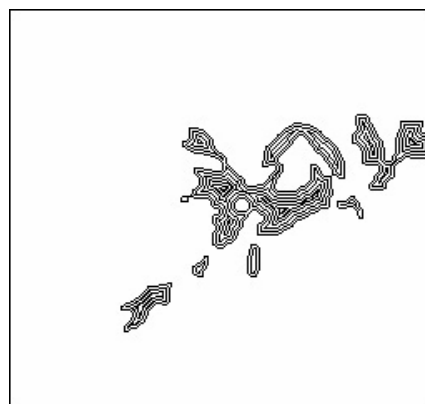
(a) Occluded tomato leaves.



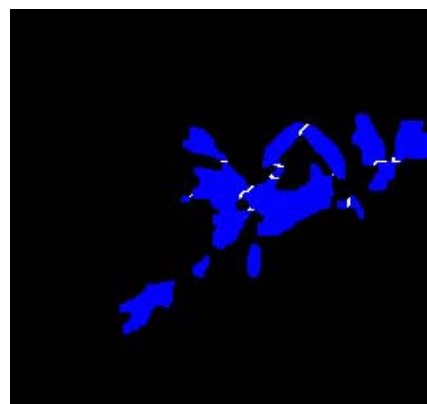
(b) Binary image.



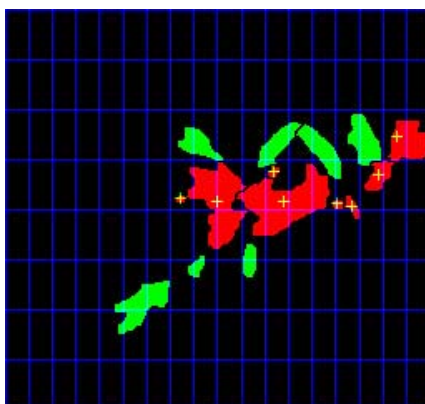
(c) Distance function.



(d) Level lines of distance function.



(e) Watershed lines generated by the algorithm.



(f) Correctly recognized tomato leaves shown in green.

Figure 3.24 Example of segmentation of overlapped leaves by the watershed method.

Figure 3.24 shows an example of the watershed algorithm applied to occluded leaves. The application of the watershed method starts with making a distance function. The distance function (Figure 3.24 (c)) is defined as a function which associates every pixel value with a value that is inversely proportional to its distance to the background. In this research, the distance from the background was calculated for every pixel in an object and the result of subtraction of the distance from 255 was stored in that pixel as the distance function for that pixel. Thus, the farther a pixel is from the background, the darker (“lower”) the pixel is. Figure 3.24 (d) shows the level lines of the distance function.

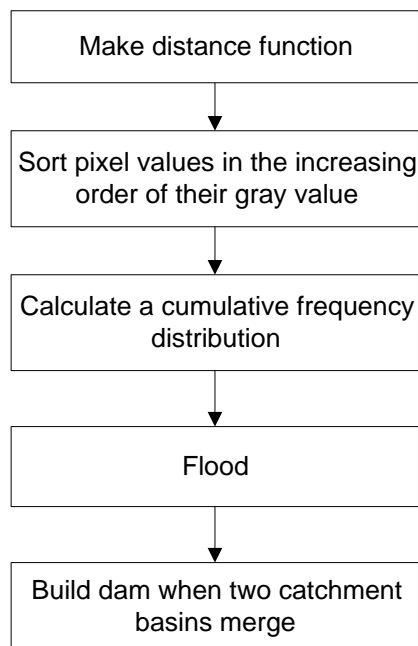


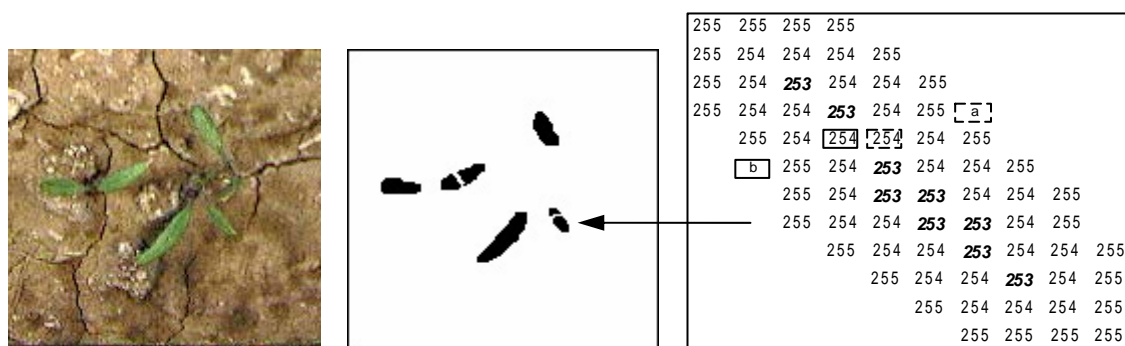
Figure 3.25 Processing steps in the watershed algorithm.

The next step is to sort all pixels in an image in the increasing order of their gray values and calculate a cumulative frequency distribution. This step assigns each pixel to

a unique cell in the stored array and allows direct access to the pixels at a given gray level h . The final step is the flooding step. In this step, holes are punched in the regional minima and the catchment basins are flooded from below by letting water rise from the holes at a uniform rate across the entire image. When the rising water in distinct catchment basins would merge, a dam is built to prevent merging. The dam corresponds to a watershed line and a particular value (pixel value of 0 for this research) is assigned to the pixels where the dam is built. Figure 3.24 (e) shows watershed lines generated by the watershed algorithm. Figure 3.24 (f) shows correctly recognized tomato cotyledons shown in green after separation by the algorithm. In this figure, some of the true leaves are still incorrectly recognized as weeds. However, 4 leaves unrecognized in Figure 3.22 (b) are now recognized.

The watershed method has two drawbacks: over-separation of leaves and being too time-consuming. Over-separation is excessive cutting, where a single object is cut into pieces. Beucher and Meyer (1993) observed that “the over segmentation produced by direct construction of the watershed line is due to the fact that every regional minimum becomes the center of a catchment basin.” More specifically, regional minima are sometimes made of two or three connected components (for example, in Figure 3.26 (c), there are two separate regional minima (a pixel value of 253) as connected components in one leaf), which produces over separation. Figure 3.26 shows an example of over-cutting due to disconnected local minima. When a distance is calculated between a pixel from the local minima and the background, its distance depends on the structure of the object. The fifth row in Figure 3.26 (c) has 6 pixels and there are no pixels with a value of 253, disconnecting the minima. If there was a pixel in dotted square- a ($\square_{\bar{a}}$)

position, the pixel 254 in dotted square ($\begin{smallmatrix} \square & 254 \end{smallmatrix}$) would become 253, or if there was a pixel in square- b position ($\begin{smallmatrix} \square & b \end{smallmatrix}$), the pixel 254 in square ($\begin{smallmatrix} \square & 254 \end{smallmatrix}$) would become 253. In either of these two cases, the object would have one connected local minima and would not be over cut. This boundary noise can have a significant impact on over cutting.



(a) Color image.

(b) Over separated binary image.

(c) Distance function of the indicated leaf in (b).

Figure 3.26 Example of over segmentation due to disconnected local minima.

A classical solution to avoid over cutting consists of slightly smoothing the distance image by performing a morphological *opening* operation after the distance image is made but before the flooding step. *Opening* is defined as erosion followed by dilation and is basically a smoothing operation. Applying the *opening* operation to the local minima increases the chance that over cut objects will contain only a single minima eliminating the over cutting problem. This application of *opening* was the first modified algorithm attempted to remove over cutting.

In order to obtain a larger connected local minima, the *opening* operation was applied to those pixels in the distance image with values up to a certain “height” (e.g.,

$h_{min} + k$, where h_{min} is a minimum “height” or value in the distance image for an object and k is found by trial and error). The area resulting from the *opening* operation was used as a new local minima. After performing *opening*, the pixel locations of the opened area were given the pixel value of $h_{min} + k$. Then, the distance function was re-sorted by increasing order of gray level, the cumulative frequency distribution of the distance function was calculated again and the original watershed algorithm was applied.

This modification is illustrated in Figure 3.27 using the tomato cotyledon shown in part a. Figure 3.27 (b) shows the distance image of the cotyledon in part a. The symbols of 1, 2, 3, 4, and 5 were used for the heights of 251, 252, 253, 254, and 255 respectively. Figure 3.27 (c) shows two disconnected local minima ($h_{min} = 251$), which would cause over cutting in the original watershed algorithm. If these minima were opened once (Figure 3.27 (d)), the local minima were removed, failing to create a single connected local minima. Thus, the *opening* was performed after defining the local minima as those pixels in the distance image with values of 251 and 252. Figure 3.27 (e) shows the local minima as defined by pixels with values of 251 and 252 and Figure 3.27 (f) shows the area resulting from application of the *opening* operation to the local minima shown in Figure 3.27 (e). The resulting area shown in Figure 3.27 (f) still contained two disconnected (4 connectedness) minima. Thus, the *opening* was performed once again after defining the local minima as those pixels in the distance image with values of 251, 252, and 253. Figure 3.27 (g) shows the local minima as defined by pixels with values of 251, 252 and 253. Figure 3.27 (h) shows the area resulting from application of the *opening* operation to the local minima shown in Figure 3.27 (g), which was now a smoothed single larger connected local minima. For this object the area resulting from

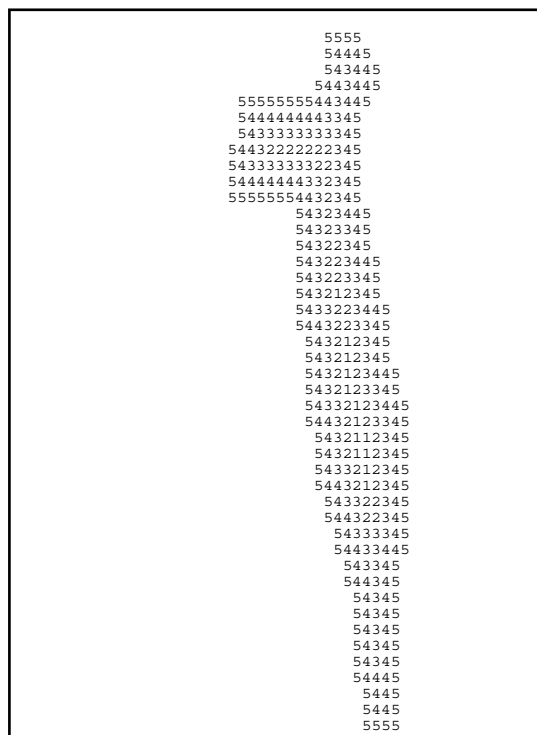
the application of the *opening* operation to the new local minima as defined by pixels with values up to 253 ($h_{min} + 2$) prevented the object from being over cut. In this example the MJX of the cotyledon was 35.2 and the minimum height was 251.

Since long objects (e.g., tomato cotyledons) tended to be over cut, the “height” used to define the local minima before *opening* was determined based on the length of the major axis (MJX) of the object with the following criteria, which was determined by trial and error from sample images. If the MJX of an object was less than 20 pixels, no *opening* was conducted. If MJX was between 20 and 35, a new object was made using pixels with values less than or equal to the local minimum (h_{min}) plus 1 and *opening* was applied to the new object. The result of *opening* this new object was used as a new local minima.

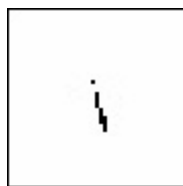
<u>Condition</u>	<u>Operation</u>
$0 < \text{MJX} \leq 20$: no opening
$20 < \text{MJX} \leq 35$: opening using $h_{min} + 1$
$35 < \text{MJX} \leq 80$: opening using $h_{min} + 2$
$80 < \text{MJX}$: opening using $h_{min} + 3$



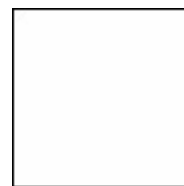
(a) Original object to be opened.



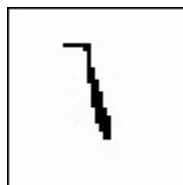
(b) Distance image of binary object.
 (where 1 = 251, 2 = 252, 3 = 253, 4 = 254, and 5 = 255)



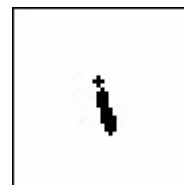
(c) Before opening: $h = 251$.



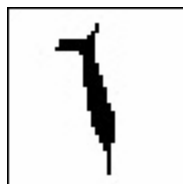
(d) After opening: $h = 251$.



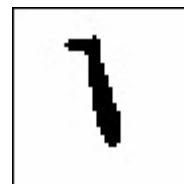
(e) Before opening: $h \leq 252$.



(f) After opening: $h \leq 252$.



(g) Before opening: $h \leq 253$.



(h) After opening: $h \leq 253$.

Figure 3.27 Example of obtaining connected local minima by opening.

The second modified algorithm was to use pre-flooding, which combined the local minima of an object before the watershed method was applied. Pre-flooding is defined as raising the local minima pixel values in an object to a pre-determined height (pixel value), if the pixel values were less than the pre-determined height. If an object had pixels with values less than the pre-determined height, the objects were pre-flooded, then the watershed algorithm was applied.

Since the appropriate level of pre-flooding varied with object shape and the shape varied a lot, it would be difficult to come up with some general technique to obtain pre-flooding level for each object without extensive study. Thus, instead of determining the pre-flooding level for each object, it would be more useful to determine when the watershed algorithm needs to be applied. Thus, the features of AREA, ELG and CMP were used to explore the feasibility of determining when the watershed algorithm needed to be applied. The AREA could be used to determine whether the watershed algorithm should be applied to an object, since objects with a small area should indicate that they were a single object and probably not occluded. ELG could be used to identify long and thin objects which tended to be over cut and CMP could also be used to distinguish compact objects which had a tendency of not being separated properly due to their lower local minima. Hence, in order to determine the proper range of feature values for occluded and non-occluded leaves, training images were chosen randomly from the set of field images and plant leaves in the training images were divided into two groups: non-occluded and occluded. Table 3.4 shows the mean and standard deviation values for AREA, ELG and CMP of these two groups in 36 sample images.

Table 3.4 Mean and standard deviation of AREA, ELG & CMP for occluded and non-occluded plant objects.

	Non-occluded group			Occluded group		
	AREA	ELG	CMP	AREA	ELG	CMP
No. of objects	192	192	192	55	55	55
Mean	181.1	0.273	1.038	945.1	0.246	0.408
Std. Dev.	192.1	0.156	0.374	512.6	0.150	0.122
Mean \pm 1 Std. Dev.	0 ~ 373.2	0.117 ~ 0.429	0.664 ~ 1.413	432.6 ~ 1457.7	0.095 ~ 0.396	0.286 ~ 0.530
Mean \pm 2 Std. Dev.	0 ~ 565.2	0 ~ 0.586	0.290 ~ 1.787	0 ~ 1970.3	0 ~ 0.546	0.164 ~ 0.652

In order to explore the feasibility of using the feature characteristics to determine when the watershed algorithm should be applied, thresholds were calculated with AREA and CMP between non-occluded and occluded groups using Bayes' rule. Suppose T_{AREA} is a threshold of AREA between non-occluded group and occluded group. Then, a threshold T_{AREA} would be solved from

$$\frac{T_{AREA} - 181.1}{192.1} = \frac{945.1 - T_{AREA}}{512.6} \Rightarrow T_{AREA} = 389.4$$

A threshold for CMP, 0.563, was calculated as the same way. ELG was not used since the ELG distributions of the two groups were greatly overlapped. Therefore, the following criteria was used to determine when the watershed algorithm should be applied.

If $AREA \geq 389.4$ and $CMP \leq 0.563$, apply watershed algorithm.

Otherwise, do not apply watershed algorithm.

The last modification of the watershed algorithm was to use the number of concave regions since there were certain number of concave regions of a certain size if

the objects were partially occluded. The curvature of each object was calculated first and the number of concave regions was obtained so that the watershed algorithm was applied based on the concavity of an object. If objects were partially occluded, then there were concavities of a certain size and the watershed algorithm would be applied in this stage. However, a non-overlapping single object could have concave regions on its boundary. Thus, a threshold value for the number of concave region was used.

Table 3.5 shows the mean and standard deviation of the number of concave regions as defined by gap size of 5 pixels and offset of 5 pixels of non-occluded and occluded groups from the same 36 sample images previously used in the feature criteria.

Table 3.5 Mean and standard deviation of concave regions for two groups.

	Non-occluded group	Occluded group
No. of object	192	55
Mean no. of concavities	1.20	9.84
Std. Dev.	1.52	3.86
Mean \pm 1 Std. Dev.	0 - 2.71	5.97 - 13.70
Mean \pm 2 Std. Dev.	0 - 4.23	2.11 - 17.56
Min	0	3
Max	7	18
Median	1	9

Based on the curvature distributions of the two groups, a threshold of 4 would be an appropriate cutoff to determine if the watershed algorithm should be applied. The threshold of 4 was obtained similarly using Bayes' rule.

$$\frac{T_{CCAVE} - 1.20}{1.52} = \frac{9.84 - T_{CCAVE}}{3.86} \Rightarrow T_{CCAVE} = 3.64$$

Therefore, the rule becomes

If the number of concavities is greater than or equal to 4, then apply the watershed algorithm.

Otherwise, do not apply the watershed algorithm.

Finally, the modification with *opening* and the feature criteria were applied together to optimize cutting. For simplification, the following names (W0 - W5) were used to describe the original and each of 5 different modifications to the watershed algorithm.

Table 3.6 Names of the watershed modifications and their description.

Name	Modification description
W0	Original algorithm
W1	Modification with <i>opening</i> operation
W2-251	Modification with pre-flooding up to 251
W2-252	Modification with pre-flooding up to 252
W2-253	Modification with pre-flooding up to 253
W3	Modification with feature criteria
W4	Modification with the number of concavities
W5	Modification with <i>opening</i> operation and feature criteria combined

3.4 Bayesian classifier with features

This section describes the feature selection procedure for identifying tomato plants and weeds. The following features were also used to identify plant leaves in addition to the features in Eq. (2.1) - (2.10) and Eq. (3.26) - (3.31).

$$ATP = \frac{AREA}{HET \times WID} \quad (3.32)$$

$$MTM = \frac{MJX}{MNX} \quad (3.33)$$

$$OCCR = \frac{AREA}{MJX \times MNX} \quad (3.34)$$

$$PTP = \frac{\sqrt{WID^2 + HET^2}}{PERIM} \quad (3.35)$$

$$M_{20} = \sum_i \sum_j (j - x)^2 f(i, j) \quad (3.36)$$

$$M_{02} = \sum_i \sum_j (i - y)^2 f(i, j) \quad (3.37)$$

$$M_{11} = \sum_i \sum_j (i - y)(j - x) f(i, j) \quad (3.38)$$

$$PRINAXIS = \frac{1}{2} \left(\tan^{-1} \left(\frac{2M_{11}}{M_{20} - M_{02}} \right) \right) \times \frac{180}{\pi} \quad (3.39)$$

$$ECCN = \frac{(M_{20} - M_{02})^2 + 4M_{11}}{AREA} \quad (3.40)$$

ATP is defined as the ratio of area to projected area (width x height). MTM is the ratio of major axis to minor axis. OCCR is the occupation ratio of an object with respect to the major diameter (MJX) and the minor diameter (MNX). PTP is the ratio of pythagorean maximal length to the perimeter. M_{20} is the second moment of an object along the x -axis and M_{02} is the second moment along the y -axis, where $f(i, j)$ is the binary intensity level of a pixel at the (i, j) location. M_{11} is the multiplied moment of inertia of an object around the centroid. PRINAXIS is the orientation of the principal axes of inertia of an object measured counterclockwise from the horizontal line. The units of PRINAXIS are in degrees. ECCN is a measure of eccentricity calculated from the second moments and the multiplied moment of inertia of an object.

Table 3.7 Performance of the prototype machine vision system using validation data sets with ELG and CMP.

Group	Good	Bad	Total
No. of images in training set	10	16	26
No. of images in validation set	41	46	87
Total no. of tomato leaves	192	128	320
Total no. of weed leaves	26	102	128
Avg. no. of tomato leaves per image	4.7	2.8	3.7
Avg. no. of weeds per image	0.6	2.2	1.5
Cotyledons found	80.0 %	62.5 %	75.0 %
Trueleaves found	38.0 %	14.7 %	30.5 %
Third group objects found	52.5 %	32.9 %	42.0 %
Weeds found	53.9 %	72.6 %	68.8 %
Avg. tomato leaves found per image	85.1 %	53.6 %	73.1 %
Avg. no. of weeds found per image	53.9 %	72.6 %	68.8 %

For the first preliminary classifier, two features (ELG and CMP) were selected based on their performance (Lee et al., 1997). In this test, a Bayesian classifier was created with 10 and 16 training images from the good and bad quality groups

respectively. The performance of the prototype system was tested using 41 and 46 validation data images from the good and bad quality groups respectively.

Table 3.7 shows the results of the preliminary classifier. With this classifier, 73.1% of tomatoes and 68.8% of weeds were correctly identified from 87 validation images acquired from the commercial processing tomato fields in northern California.

Feature subset selection procedure

For real-time identification of tomato plants and weeds, a minimum number of features needs to be selected among the features described in the previous chapters. In order to select the best feature subset, field images were used which were taken from 13 commercial processing tomato fields in Northern California starting in late May - late June 1996 and late March until mid-May 1997. The tomato plants were in various stages of maturity from just emerging to the second true leaf stage.

All images were divided into two groups of good and bad image quality based on the focus, camera aperture, wind, cotyledon opening, state of maturity, and occlusion. It was an especially windy spring in Northern California in 1997 and most of the tomato plants in the commercial fields were lying down along the direction of wind travel.

Tomato plants in the good image quality group were easier to recognize with an image processing algorithm since they retained their original shape. Tomato plants in bad images were harder to recognize since many of them lost their original shape due to occlusion and from being blown down by wind. From each group, a training set and a validation set were created in order to estimate the plant recognition performance by the image processing algorithm. For the good image group, a total of 117 images were used

in the training set and 157 images were used in the validation set. For the bad group, a total of 129 images and 133 images were used respectively (Table 3.8). The images in the training sets were selected carefully to represent the entire group while those in the validation sets were selected randomly from each group. There was no overlap between training and validation sets.

Table 3.8 Number of images used for feature selection in each group.

	Good group	Bad group	Total
Training set	117	129	242
Validation set	157	133	290
Total	274	262	536

Objects were divided into 4 classes; tomato cotyledon, tomato true leaf, third group, and weed classes (Table 3.9). The objects in the third group consisted of cotyledons and tomato true leaves which were curled, occluded, eaten by bugs, and partially hidden by the edge of the image. For this chapter, the class numbers (1, 2, 3, and 4) will be used instead of class description (tomato cotyledon, tomato true leaf, third group, and weed).

Table 3.9 Class assignment for plant leaves in an image.

Class	Description
1	Tomato cotyledon
2	Tomato true leaf
3	Tomato Third group
4	Weed

In order to find out whether objects from the third group could be separated from the weed class, both training and validation sets in the good group (Table 3.10) were used together to test the feasibility of separating the third group objects from the weed

class using canonical discriminant analysis. This test would indicate which classes could be separated. In this test, all of the following 35 features were used. YCNTRD is the y-coordinate of a leaf centroid.

AREA, YCNTRD, PERIM, MJX, MNX, ELG, CMP, MAXC, MINC, AVGC,
 AVGABSC, STDEVC, NEG, ATL, PTB, LHW, LTP, SUMINV, ABSUMINV, WID,
 HET, M₂₀, M₀₂, M₁₁, PRINAXIS, ATP, MTM, OCCR, PTP, MTMC,
 ATC, PTC, ETC, CTC, ECCN

Table 3.10 Number of plant leaves in good training and validation sets together used for exploring separation feasibility of third group by canonical discriminant analysis.

Class	No. of objects	Proportion (%)
1	228	0.141
2	306	0.189
3	687	0.424
4	400	0.247

Table 3.11 and Figure 3.28 show the result of canonical discriminant analysis with good training and validation sets together. Canonical discriminant analysis is a dimension-reduction technique related to principal component analysis and canonical correlation. Given a classification variable and several quantitative variables, this analysis derives canonical variables (linear combinations of the quantitative variables) that summarize between-class variation in much the same way that principal components summarize total variation. For canonical analysis, the SAS procedure CANDISC was used.

Table 3.11 Result of canonical discriminant analysis with good training and validation sets together.

Canonical Discriminant Analysis Pairwise Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \text{COV}^{-1} (\bar{X}_i - \bar{X}_j)$$

Squared Distance to CLASS

From CLASS	1	2	3	4
1	0	6.31045	6.19911	6.57798
2	6.31045	0	3.41924	2.30020
3	6.19911	3.41924	0	1.18015
4	6.57798	2.30020	1.18015	0

F Statistics, NDF=35, DDF=1583 for Squared Distance to CLASS

From CLASS	1	2	3	4
1	0	23.06099	29.68265	26.71963
2	23.06099	0	20.24701	11.15438
3	29.68265	20.24701	0	8.34502
4	26.71963	11.15438	8.34502	0

Prob > Mahalanobis Distance for Squared Distance to CLASS

From CLASS	1	2	3	4
1	1.0000	0.0001	0.0001	0.0001
2	0.0001	1.0000	0.0001	0.0001
3	0.0001	0.0001	1.0000	0.0001
4	0.0001	0.0001	0.0001	1.0000

Canonical Discriminant Analysis

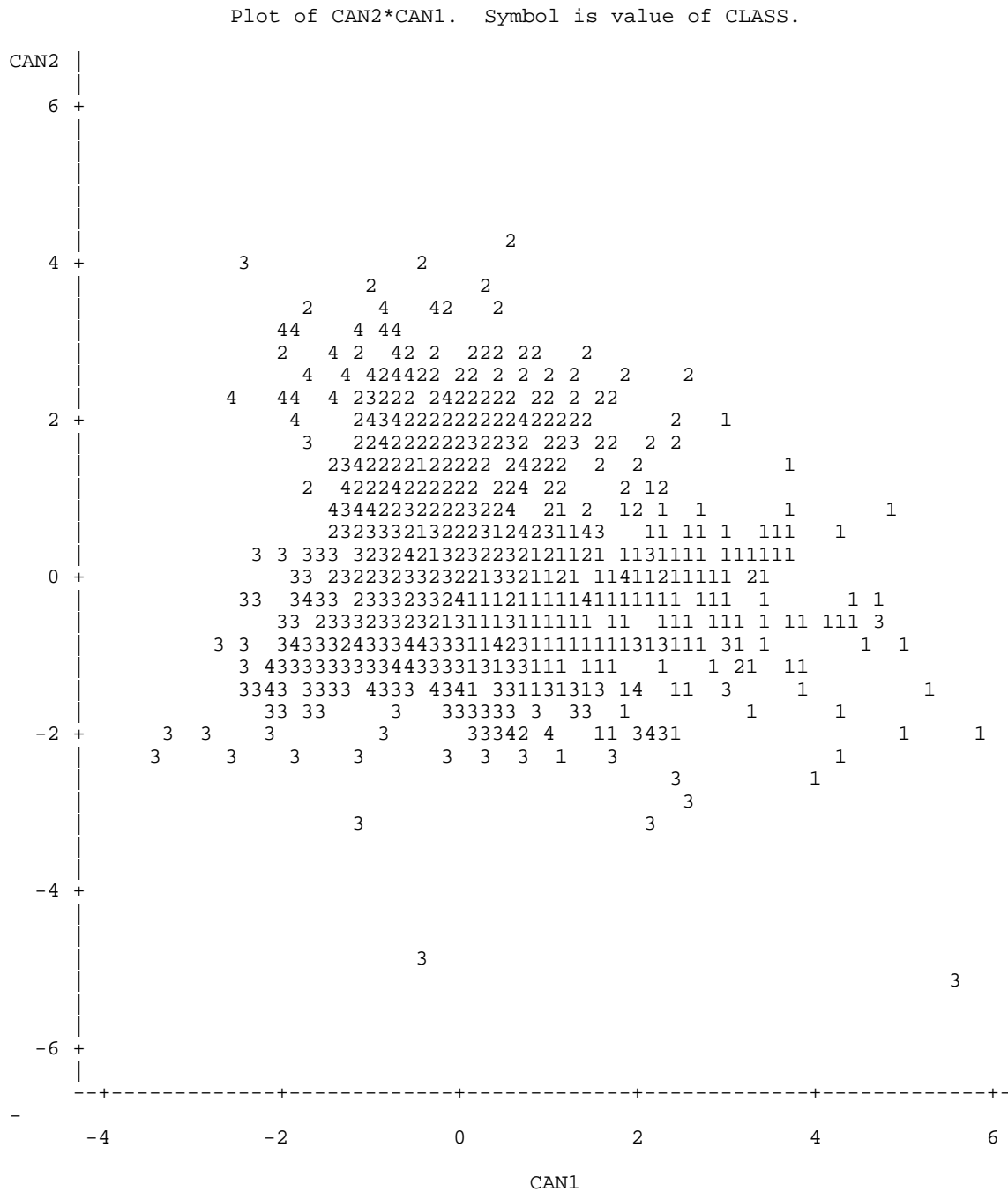
	Canonical Correlation	Adjusted Canonical Correlation	Approx Standard Error	Squared Canonical Correlation
1	0.644056	0.633081	0.014539	0.414808
2	0.548430	0.536029	0.017372	0.300775
3	0.343368	0.319173	0.021916	0.117902

Eigenvalues of $\text{INV}(E)*H$
= $\text{CanRsqr}/(1-\text{CanRsqr})$

	Eigenvalue	Difference	Proportion	Cumulative
1	0.7088	0.2787	0.5570	0.5570
2	0.4302	0.2965	0.3380	0.8950
3	0.1337	.	0.1050	1.0000

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approx F	Num DF	Den DF	Pr > F
1	0.36093778	18.3045	105	4741.31	0.0001
2	0.61678518	12.7329	68	3168	0.0001
3	0.88209827	6.4198	33	1585	0.0001



NOTE: 1108 obs hidden.

Figure 3.28 Plot of canonical variable 1 and variable 2 with good training and validation sets.

From Table 3.11, the squared distance from class 1 to classes 2, 3, and 4 is over 6 whereas the distance from class 3 to class 4 is 1.18, indicating that the classes 3 and 4 could not be easily separated. The plot of canonical variable 1 and variable 2 (Figure 3.28) also confirmed the same result. These results indicate that the third group could not be separated from weed class using only a single 2-D top view even though only the good image group was used. In addition, if the third group could not be separated from the weed class with good group altogether, the third group would not be separated for a smaller set from the good group or the bad group. Therefore, the objects in class 3 were temporarily set aside during the feature selection process and only the classes 1, 2, and 4 were used for the feature selection process.

Next, the *a priori* probability for each class needs to be determined for discriminant analysis. In theory, the *a priori* probability should be set to the proportion that, if a plant leaf were chosen, a decision should be made as to what class that leaf would fall in without being observed. However, changing the *a priori* probability is a very useful way to adjust the decision process and should be used. It is very useful to try and weight one class more heavily than another. The training and validation sets used in this chapter came from different fields at different stages of development and the *a priori* probability of the different classes changed from field to field. Often the images were collected disproportionately to the actual occurrence in the field since some unusual appearances needed to be included in the training set. In addition, the classifier needed to work correctly on ideal images too.

In this research there is more emphasis on recognizing weeds than recognizing tomato plants since if the weed recognition rate is low, more work still needs to be done

after the robotic weed control system eliminates weeds. Usually tomatoes are over planted since their germination rate is variable and often tomato growers perform a thinning operation to maintain their desired stand.

In order to determine the proper level of the *a priori* probabilities for the three classes, the good training and validation sets together were used as sample data set and discriminant analyses were conducted with different *a priori* probabilities with all of the 35 features. Table 3.12 shows the result of discriminant analysis with different *a priori* probabilities for good training set using all of the 35 features. Table 3.13 shows the discriminant analysis result with proportional *a priori* probability and *a priori* probabilities of 0.1, 0.1 and 0.8 for classes 1, 2, and 4 respectively.

Table 3.12 Discriminant results with different *a priori* probabilities with good training and validation sets using all 35 features.

<i>A priori</i> probability			Recognition rate		
class 1	class 2	class 4	class1	class2	class4
0.0395	0.3276	0.4283	96.1	61.4	65.8
0.1	0.1	0.8	95.6	59.5	68.8
0.1	0.2	0.7	95.6	59.8	67.3
0.1	0.3	0.6	95.2	60.8	66.5
0.1	0.4	0.5	95.6	61.4	66.3
0.1	0.5	0.4	95.6	62.4	65.8
0.1	0.6	0.3	95.2	63.1	65.3
0.1	0.7	0.2	95.2	65.0	64.0
0.1	0.8	0.1	95.2	68.0	62.3

In this test, the *a priori* probability for the class 1 was fixed to 0.1 and the *a priori* probabilities for classes 2 and 4 were changed with 0.1 increment for the class 2 and 0.1 decrement for the class 4. For the class 4, the recognition rates decreased as the *a priori* probability decreased. On the other hand, the classification rates for the class 2 increased

as the *a priori* probability increased. Thus, since the emphasis was on recognizing more weeds than more tomato plants, the *a priori* probabilities of 0.1, 0.1, and 0.8 were used for classes 1, 2, and 4 respectively for further discriminant analysis throughout the feature selection procedures.

Table 3.13 Discriminant analysis results with different *a priori* probabilities for good training and validation data set with all of the 35 features.

A priori probability: proportional

From CLASS	1	2	4	Total
1	219 96.05	3 1.32	6 2.63	228 100.00
2	50 16.34	188 61.44	68 22.22	306 100.00
4	76 19.00	61 15.25	263 65.75	400 100.00
Total	345	252	337	934
Percent	36.94	26.98	36.08	100.00
Priors	0.2441	0.3276	0.4283	

Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0395	0.3856	0.3425	0.2259

A priori probability: class 1 = 0.1, class 2 = 0.1, class 4 = 0.8

From CLASS	1	2	4	Total
1	218 95.61	2 0.88	8 3.51	228 100.00
2	48 15.69	182 59.48	76 24.84	306 100.00
4	73 18.25	52 13.00	275 68.75	400 100.00
Total	339	236	359	934
Percent	36.30	25.27	38.44	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0439	0.4052	0.3125	0.2238

This choice of the *a priori* probability (tomato plants = 0.2 and weeds = 0.8) was probably acceptable since the *a priori* probabilities were used here as a tool to recognize more weeds than to recognize more tomato plants.

Table 3.14 *A priori* probability assignment for discriminant analysis.

Class	<i>A priori</i> probability
1	0.1
2	0.1
4	0.8

The first step in plant recognition was to select features which could discriminate plant leaves into their original classes. To find the best subset of features, training sets in both good and bad groups were used with all 35 input variables unless stated otherwise. Table 3.15 shows the number of plant leaves used for the feature selection procedure in each class in each group.

Table 3.15 Number of plant leaves used for feature selection for each class in each group. (where, 1 = Tomato cotyledon, 2 = Tomato true leaf, and 4 = Weed)

Good group			Bad group		
Class	Training	Validation	Class	Training	Validation
1	78	150	1	90	123
2	127	179	2	114	111
4	198	202	4	138	157
Total	403	531	Total	342	391

The following three procedures were used to find the best features to identify tomato plants and weeds.

Method I

The first method was to use canonical discriminant analysis and principal component analysis to choose the best features by removing any useless features and to take advantage of the large feature set but to eliminate the problems with multi-collinearity. Features extracted from the same object tend to be correlated or have multi-collinearity by nature. If there exists multi-collinearity among features, classification with those highly correlated features does not reflect any inherent effect of the particular feature on the classification result but only a marginal or partial effect. Thus, a multi-collinearity problem should be solved.

This method produced the best solution regardless of cost. Principal component analysis is generally used to maximize the variance of a linear combination of the variables and is used when highly correlated independent variables may produce unstable estimates. In these two tests (canonical discriminant analysis and principal component analysis), the following 35 features were used as input features.

AREA, YCNTRD, PERIM, MJX, MNX, ELG, CMP, MAXC, MINC, AVGC,
 AVGABSC, STDEVC, NEG, ATL, PTB, LHW, LTP, SUMINV, ABSUMINV, WID,
 HET, M₂₀, M₀₂, M₁₁, PRINAXIS, ATP, MTM, OCCR, PTP, MTMC,
 ATC, PTC, ETC, CTC, ECCN

In this method, the multivariate test for differences between the classes was conducted at the 0.0001 level using Wilk's Λ test and other tests. Wilk's Λ test is a likelihood ratio test for population mean difference ($H_0: \mu_1 = \mu_2 = \dots = \mu_k$) and is given by

$$\Lambda = \frac{|\mathbf{E}|}{|\mathbf{E} + \mathbf{H}|} \quad (3.41)$$

where, $\mathbf{H} = n \sum_{i=1}^k (\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_{..})(\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_{..})'$ (between sample sums of squares) and

$$\mathbf{E} = \sum_{i=1}^k \sum_{j=1}^n (\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)(\mathbf{y}_{ij} - \bar{\mathbf{y}}_i)'$$
 (within sample sums of squares).

The null hypothesis, H_0 , is rejected if Λ is smaller than or equal to the table value.

Method II

The second method was to try to find a feature subset for real-time field use. Since the goal of this research was to build a real-time robotic weed control system, features to be used for recognition of tomato plants and weeds should not take too much time to be calculated while providing enough discriminatory power to identify tomato plants and weeds. In this method, the number of features were limited to 4 for real-time implementation. In addition, multi-collinearity among features should also be solved since highly correlated independent variables (features) provide only marginal or partial effect to the dependent variable (resulting class for an object).

This method (II) was to try to reduce the number of features to the most important subset by correlation analysis and to find a good combination of features using the linear regression model selection procedure based on the R^2 criteria. With correlation analysis, if two or more features were highly correlated, only one feature could be chosen from the set and other features were removed.

Then, the linear regression model selection procedure by R^2 criteria was used with the 35 features as independent variables and class number as a dependent variable. This method was the only method that could try all of the possible combinations of the maximum number of features in a reasonable amount of time. This process produced linear regression models and associated R^2 values for all of the possible combinations of input features. Feature combinations with higher R^2 values were important feature subsets.

For correlation analysis, the SAS procedure CORR was used and the procedure REG was used for linear regression model selection with the R^2 criteria. From correlation analysis, the following subsets in square bracket had correlations of over 0.9 with the feature on the left in Table 3.16.

Table 3.16 Highly correlated feature subsets in both good and bad training sets.

Good training set	Bad training set
AREA \propto [PERIM, MJX, MNX, ATL, ATC, PTC]	AREA \propto [PERIM, ATC]
PERIM \propto [AREA, MJX, NEG, PTC]	PERIM \propto [AREA, MJX, NEG]
MJX \propto [AREA, PERIM, PTC]	MJX \propto [PERIM, PTC]
MNX \propto [AREA, ATL]	MNX \propto [ATL]
ELG \propto [MTM]	ELG \propto [MTM]
PTB \propto [PTP]	PTB \propto [PTP]
ATC \propto [AREA, PTC]	ATC \propto [AREA, PTC]

Therefore, from the result of correlation analysis, the following 6 features were removed to reduce multi-collinearity since they were common in both groups:

ATC, ATL, MJX, MTM, PERIM, and PTP.

The basic idea here was first to find minimum good feature subsets that gives good classification of tomato cotyledons vs. weeds, then to find good features for tomato true leaves vs. weeds so that the overall classification rate could be improved. Feature

models could not be developed using this technique for all three classes simultaneously, but were restricted to two class models due to difficulties in assigning appropriate class values for the dependent variable when more than two classes are present in the analysis. For example, if the values of tomato cotyledon = 1, tomato true leaf = 2, and weed = 4 were assigned to the dependent variable and the regression analysis was conducted, then the regression analysis considered that the true leaf had twice as much value as tomato cotyledons and weeds had 4 times as much value as cotyledons and twice as much value as tomato true leaves which might not be appropriate. Therefore, only two classes were used at one time for the R^2 model selection procedure.

Prior to finding the best features for real-time field use, execution time for each feature was measured using the computer clock. In this test, a 200 MHz Pentium processor was used to process the image. The sample image (256 x 240 pixels) had 10 tomato cotyledons with each cotyledon containing about 400 pixels. Each execution time was for only one object and was the average of 1000 iterations.

Method III

This method was used to find the best feature subset for real-time use by trial and error method. In this method the maximum number of features was limited to 4 for real-time field use.

Prior to using this method, stepwise discriminant analysis was performed with good and bad training sets by backward elimination and stepwise selection options after removing 6 features by correlation analysis. The forward selection option was not used since this option did not remove any redundant features from the set. Although stepwise

discriminant analysis does not select the best subset of features due to multi-collinearity among features, the result from this analysis would be used as supplementary information. Table 3.17 shows the selected features from stepwise discriminant analysis listed by their partial R^2 values. For this process, the SAS STEPDISC procedure was used with both good and bad training sets.

Thus, feature subsets for the Method III were selected based on the following three analysis: principal component analysis, canonical discriminant analysis, and stepwise discriminant analysis.

Table 3.17 Selected features from stepwise discriminant analysis for good and bad training sets.

Good training set		Bad training set	
Backward elimination	Stepwise selection	Backward elimination	Stepwise selection
MNX	ELG	AREA	MNX
M_{02}	PTC	LTP	LTP
ETC	OCCR	OCCR	OCCR
PTC	ETC	PTB	AVGABSC
LTP	AREA	M_{20}	PTB
PTB	M_{02}		
AREA	HET		
OCCR	M_{20}		
NEG	MNX		
M_{20}			
WID			
AVGABSC			

3.5 Displacement sensing and calibration of encoder

Displacement sensing was very important in this research, since timing for image acquisition and valve control were based on the distance traveled. In order to sense travel distance correctly, many devices were tried. First, two radar sensor units (John Deere and Raven-Model No. 063-0159-835, Raven Industries, Sioux Falls, SD), which are normally used in tractors were tested. However, these radar units did not work at speeds less than 1.6 km/h. Since the prototype robotic system had a maximum traveling speed of about 0.8 km/h, these units were not appropriate.

A bicycle wheel with an encoder mounted on its axle was evaluated to sense travel distance. However, the tire was narrow and light, and did not provide enough stability, frequently jumping around a lot when it was used in tomato fields where many dirt clods or bumps were present. Hence, the bicycle wheel was not suitable for correct distance measurement.

A cultivator gage wheel was finally used which was attached to the cultivator toolbar with an encoder (Model HR6251000000A, Danaher Controls) mounted on its axle. The gage wheel provided enough stability and worked at a lower speed range (less than 1.6 km/h). The encoder generated a pulse whenever the tractor moved 0.13 mm forward on the soil. The encoder output was 1000 pulses/revolution and in order to obtain higher resolution from the encoder, an intermediate pulley was used between the encoder and the axle of the gage wheel.

The encoder was calibrated each time prior to field tests of the robotic weed control system. The number of encoder pulses were counted after traveling a pre-determined distance. The resolution of the encoder was then calculated by dividing

the traveled distance by the number of pulses. The average resolution was 0.13 mm per pulse on soil and 0.14 mm per pulse on a smooth concrete floor.

3.6 Precision chemical application system

A precision chemical application system was used as the actuator of the robotic weed control system to kill weeds after weed locations were identified. The system, Figure 3.29, consisted of a valve driver circuit, eight 12 Vdc solenoid valves (Capstan Ag Systems, Inc., Topeka, Kansas), a metal valve alignment plate (13.97 cm x 6.35 cm x 0.48 cm), a stainless steel manifold (3.18 cm x 3.18 cm x 13.97 cm), a specially designed accumulator, a CO₂ tank, a pressure regulator, and a spray mix tank. The robotic spraying system was mounted at the end of the tunnel about two image frames behind the camera.

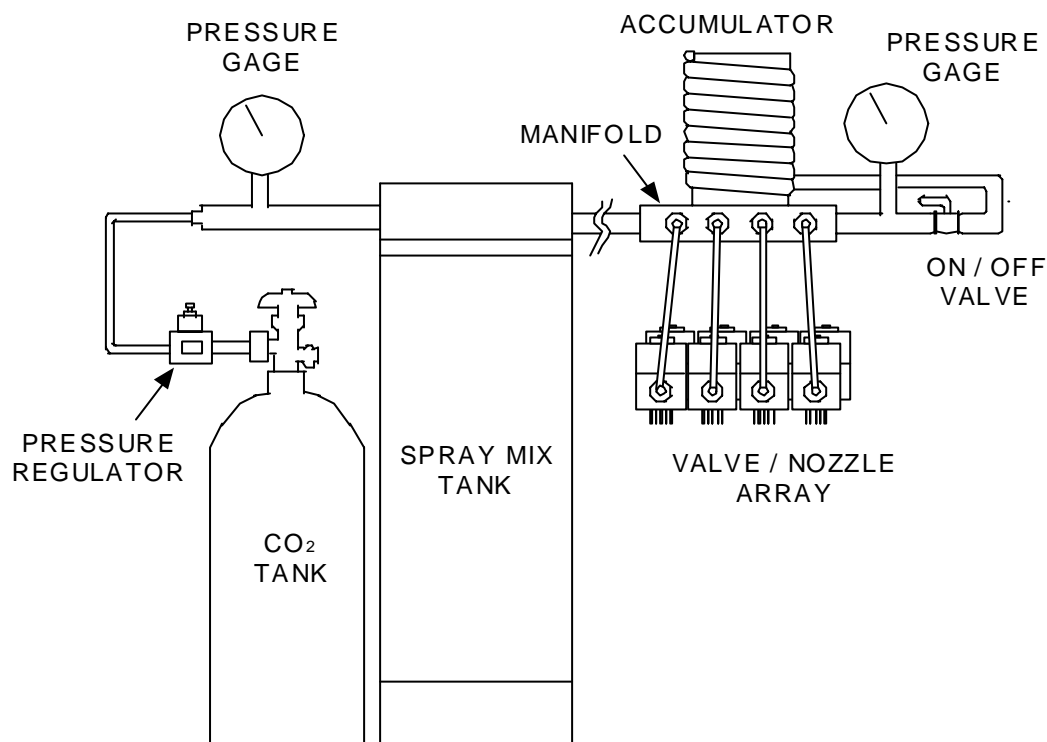


Figure 3.29 Spray mix supply system

3.6.1 Nozzle design

The precision spray nozzles were designed to spray a 0.64 cm x 1.27 cm region on the soil and to keep the flow rate as low as possible. Several nozzle prototypes were built and tested to determine which could deposit an appropriate amount of spray on a spray region. For example, commercially available nozzles (TP1501, TP1509, TP2508, TP4001E, TP150033, TP400067, and TP400067E, Spraying Systems Inc., Wheaton, Illinois), a nylon unslotted hex head screw (size 8-32, 1.27 cm long) with a variety of center holes (diameter = 0.34 mm - 0.71 mm), and a plexiglass orifice plate (5 holes on a center line, each with a diameter = 0.25 mm, 2.54 mm apart from each other) were evaluated. However, none of these were acceptable. The commercially available nozzles provided much larger spray drops than desired. The spray drop from the nylon hex head screw was a single solid stream and it did not disperse and did not provide enough spray coverage. The plexiglass orifice could not be used since the spray droplets were not aligned at the center and they landed in a random pattern on the ground, since the holes in the plate were very small (diameter = 0.25 mm) and it was very difficult to drill the holes straight.

After all these trials, hypodermic tubes of very small diameter were found to be plausible as spray nozzles. The hypodermic tube nozzles provided straight spray streams and a properly sized elliptical deposit on the ground.

As shown in Figure 3.30, five hypodermic tubes (Heavy Wall Stainless Steel Type 304-W, Part no. E-HTX-22HW, 22 gauge, O.D. = 0.71 mm, I.D. = 0.28 mm, 12.7 mm long, Small Parts, Inc., Miami Lakes, Florida) were attached in a line along the center of a stainless steel plate (2.54 cm x 2.54 cm, 0.48 cm thick) using Epoxy glue

(Cole Parmer Instrument Co., Chicago, IL60648). The tubes were 2.54 mm apart from each other. This “micro-spray” nozzle was attached to the solenoid valve using four bolts screwed into the valve body and a thin round rubber sheet (2.54 cm diameter, 0.41 mm thick) was used as a gasket.

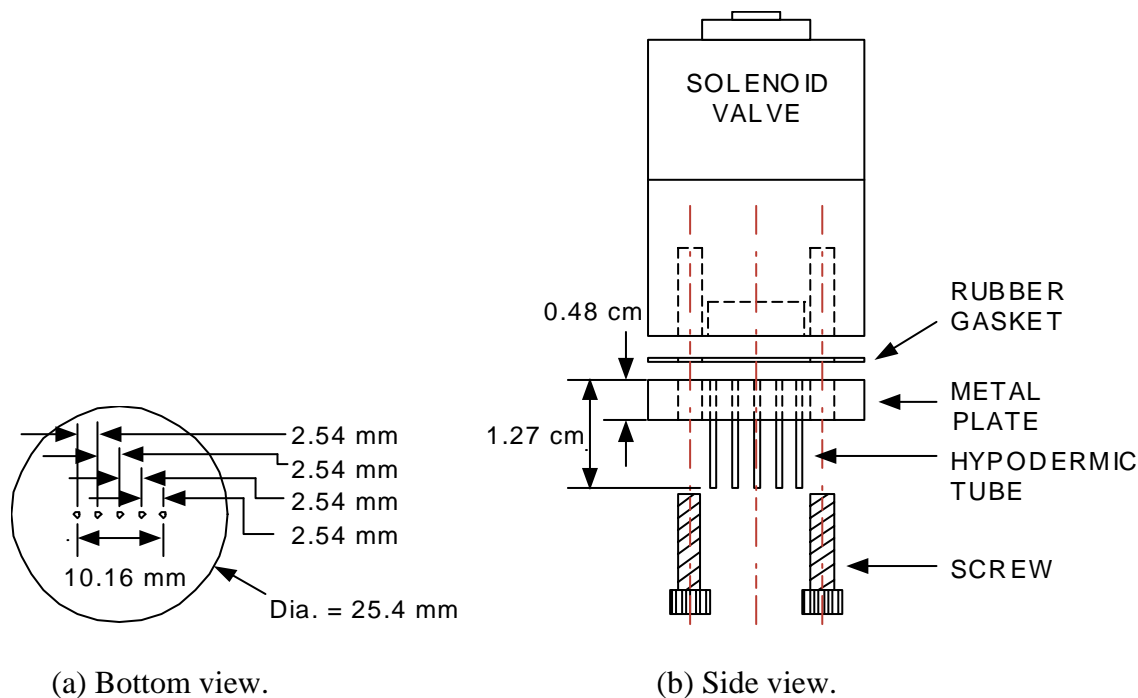


Figure 3.30 Valve / nozzle assembly structure.

When moving at 1.20 km/h, each micro-spray nozzle provided an elliptical deposit, 0.90 cm along the axis of travel and 1.27 cm normal to axis of travel, using a 10 ms valve opening time at 103 kPa and a nozzle height of 10.16 cm above the ground. A spraying time of 10 ms gave a flow rate of 0.098 L/min for each valve and an exit velocity from the nozzle of 6.4 m/s. A CO₂ tank was used to pressurize the spray system. The eight solenoid valves (2.54 cm diameter) were aligned in two rows (four valves in each row,

Figure 3.5) in order to allow the entire 10.16 cm wide seedline to be sprayed when they were all opened simultaneously. An accumulator was attached to the manifold in order to maintain a constant flow rate independent of the number of valves opened simultaneously.

3.6.2 Valve / nozzle design

The optimal valve operating conditions such as nozzle height, valve opening time, flow rate, exit velocity, manifold pressure, and accumulator design, were determined by trial and error. If a nozzle was too far away or too close to the ground, or pressure inside the manifold was too high or too low, the spray pattern size was different than desired. There was also a significant pressure drop inside the manifold when all 8 valves were opened simultaneously. An accumulator was used to stabilize the pressure drop inside the manifold, regardless of the number of valves opened at the same time.

The flow rate of a valve was measured with one valve chosen randomly out of 8 valves. The valve was opened for pre-determined time (Table 3.18) and the mass of distilled water was measured for each trial. The flow durations were 5, 10, 15, 20, 25, and 30 seconds and three repetitions were performed for each opening duration. From the results of Table 3.18, an average flow rate, \dot{Q} , was 1.632 ml/sec.

$$\dot{Q} = 513.99 \text{ g} / 315 \text{ s} = 1.632 \text{ ml} / \text{sec} \quad (3.42)$$

Table 3.18 Valve flow for different periods of operation.

Flow duration (sec)	Mass of water flowing through valve (g)
5	8.41
5	8.28
5	8.23
10	16.69
10	16.45
10	16.48
15	24.69
15	24.41
15	24.58
20	32.52
20	32.64
20	32.58
25	40.81
25	40.64
25	40.62
30	48.62
30	48.72
30	48.62
Total: 315	Total: 513.99

Two methods were used to measure the exit velocity of the spray drops. One was to calculate the velocity using flow rate and the area of flow. A hex head screw was used as a nozzle with a hole (dia. = 0.05715 cm) at the center for calculating the exit velocity of spray droplets from the nozzle. The area of a nozzle, A was 0.00256 cm^2 ($= \pi / 4 \times (0.05175 \text{ cm})^2$). Thus, the exit velocity, V was calculated as follows.

$$V = \frac{\dot{Q}}{A} = \frac{1.632 \text{ ml / sec}}{0.00256 \text{ cm}^2} = 6.361 \text{ m/sec} \quad (3.43)$$

The other method to calculate the exit velocity used a pressure transducer and an accelerometer to get the flight time of a spray droplet for a pre-measured distance, Figure 3.31.

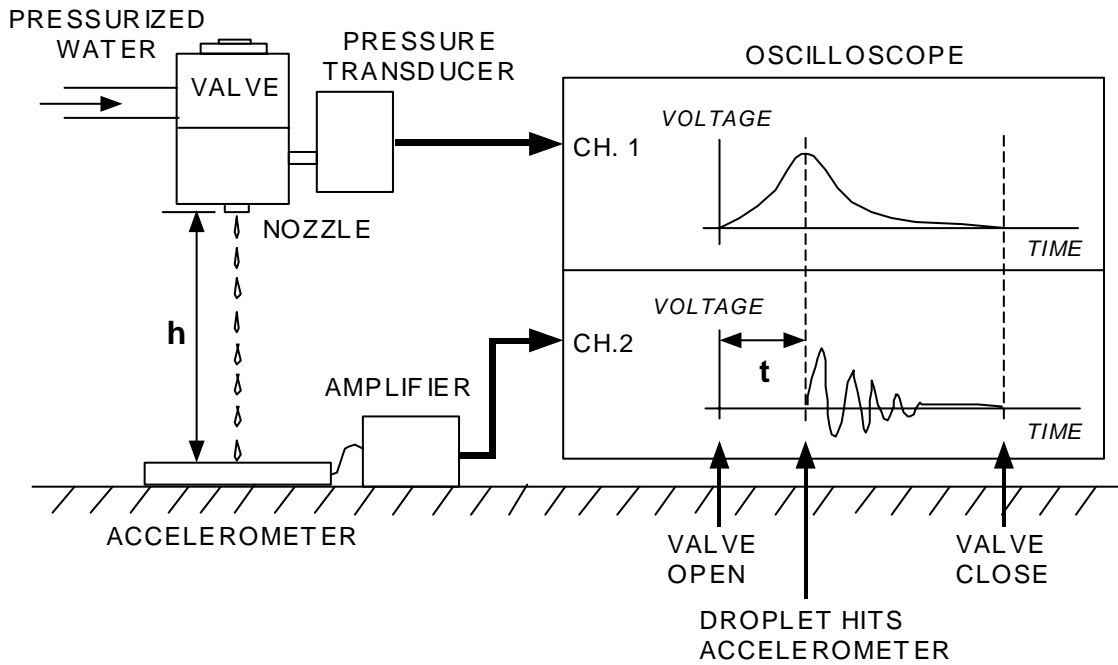


Figure 3.31 A schematic for measuring an exit velocity of spray droplet.

A valve was positioned above an accelerometer (model ACH-01, Pennwalt Corp., Kynar Piezo Film Dept., Valley Forge, PA) at a height h . When a spray droplet hit the accelerometer, the output signal was amplified (model IB-ACH01-01, Pennwalt Corp., Valley Forge, PA) and displayed on one channel of an oscilloscope (model TDS520, Tektronics Inc., Wilsonville, Oregon). At the same time, the pressure change inside the valve was monitored via a pressure transducer (model PX176, OMEGA Engineering Inc., Stamford, CT). Then, the flight time, t , from valve opening to the time the droplet hit the accelerometer was read from the oscilloscope. The flight time t and the height h are shown in Table 3.19.

Table 3.19 Nozzle height and flight time of spray droplet.

Height (cm)	7.62	10.16	12.70
Flight time (ms)	12.9	16.0	19.9
	13.0	16.6	20.0
	12.8	15.6	20.1
	12.8	14.4	19.7
	12.5	16.4	19.8
	12.5	16.8	18.7
	12.4	16.5	18.7
Average (ms)	12.70	16.04	19.56

Thus, when the valve was 10.16 cm high from the ground, the exit velocity was calculated (ignoring gravitational effects) as 6.35 m/s ($= 10.16 \text{ cm} / 16.04 \text{ ms}$). This result matched the one from the previous method.

3.6.3 Valve driver circuit

A microcontroller (SensorWatch™, TERN Inc., Davis, CA) and a valve control circuit (Figure 3.32) were used to control the micro-spray valve array. One of the tasks of the microcontroller was to control the precision valve system, i.e., to open the corresponding valve for 10 ms when the valve was directly above a weed location.

The microcontroller was programmed prior to use via an RS-232 serial link using the Borland C compiler. The microcontroller had eight digital output ports, which were connected to the eight TTL inputs of the valve driver circuit. The operation of these valves was manipulated by writing a 0 or 1 to the microcontroller digital output ports.

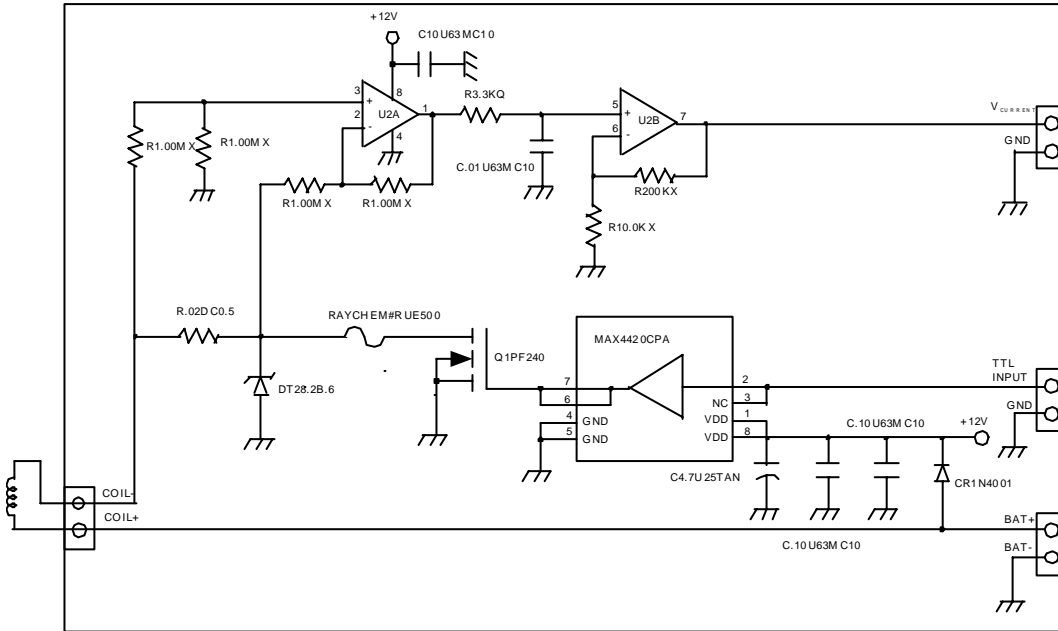


Figure 3.32 Spray valve driver circuit (Courtesy of Capstan Ag Systems, Topeka, KS).

3.6.4 Valve and encoder software

The objective of the valve software was to open and close spray valves when they were needed. The valve software was composed of nine subroutines, which were *main*, *ready*, *running*, *download*, *set_param*, *kb_scan*, *kb_decode*, *tb_isr* and *ct1_isr*. The flow charts of the functions *main*, *running*, *tb_isr* and *ct1_isr* are shown in Figures 3.33 - 3.35.

The function *main* (Figure 3.33) was used to initialize the microcontroller, liquid crystal display, time base interrupt, serial port, and counter interrupt and to execute the function *ready*. The function *ready* was used to provide the current status of the microcontroller to the function *main*.

The function *running* (Figure 3.34) was used to actually control the opening of the valves with a time base interrupt service routine, *tb_isr*, and a counter interrupt service routine, *ct1_isr* (Figure 3.35). This function was used during the operation of the prototype weed control machine, reading encoder counts, sending them to the main computer, and controlling the valve operation. This function loaded an initial *cpc* (count per cell) count to the counter, prevented simultaneous serial communication between the image processing computer and the microcontroller, calculated *valve byte*, and sent a character 'a' to the image processing computer to acquire a new picture. The *valve byte* was one byte of information indicating which valves the microcontroller opened or closed while the valve array was traveling over the corresponding weed location in an image.

The *download* function was executed when valve arrays were downloaded from the microcontroller via RS-232 serial communication in order to compare them with those sent from the image processing computer. The *set_param* function was used to adjust three parameters for the operation of the valve opening in connection with encoder reading. The first parameter, *cpc*, was the number of encoder pulses per spray column. The second, *spray_delay*, was the number of spray cells between the camera and the first line of valves, and was used to account for the delay associated with the offset between the camera and the first line of valves. The last, *noz_ofs*, was the delay between the first and the second line of spray valves. The typical value of these parameters were 49, 20, and 6 respectively for commercial field use.

The function, *kb_scan*, was used for scanning the keyboard of the microcontroller and returning the status of keyboard. The function, *kb_decode*, was used to convert

keypad code into a more understandable pattern. The function, *tb_isr*, was a time base interrupt service routine. This function was executed every 1.024 ms when an interrupt was generated by a time base counter and turned off all eight valves after 10 ms spraying.

The function, *ctl_isr*, was used to send *valve bytes* to the nozzles, to determine the travel distance, and to send a character 'a' to the image processing computer when the prototype system traveled the distance of an image size. This function was executed whenever an interrupt was generated by a μ PD71504 counter/timer (NEC, Japan) for every *cpc* (counts per cell) encoder pulses. A new *cpc* count was loaded in the counter at the beginning of the *running* function. When the count down counter decremented from an initial count *cpc* and reached a count 1, an interrupt was generated and the counter interrupt service routine, *ctl_isr*, was executed. Then, new *cpc* count was reloaded to the counter for the next interrupt in the *running* function.

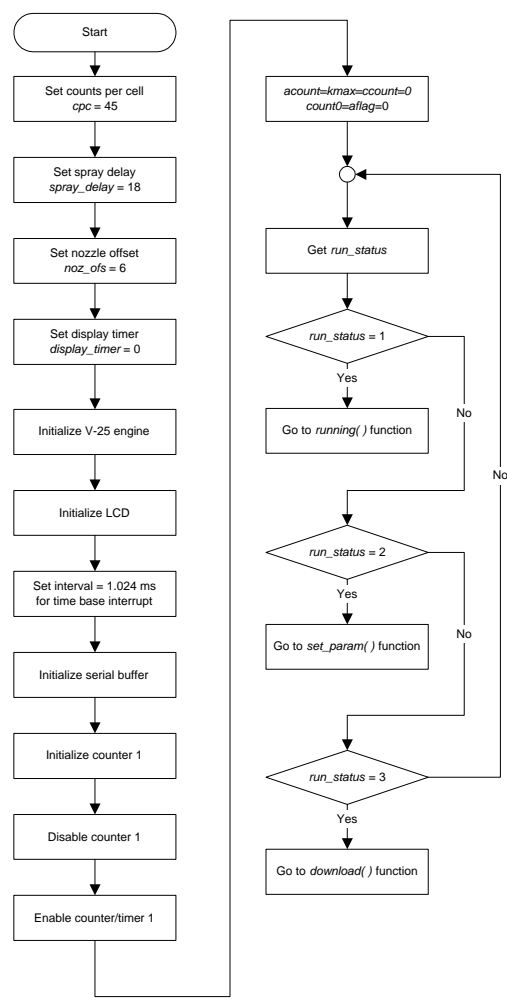
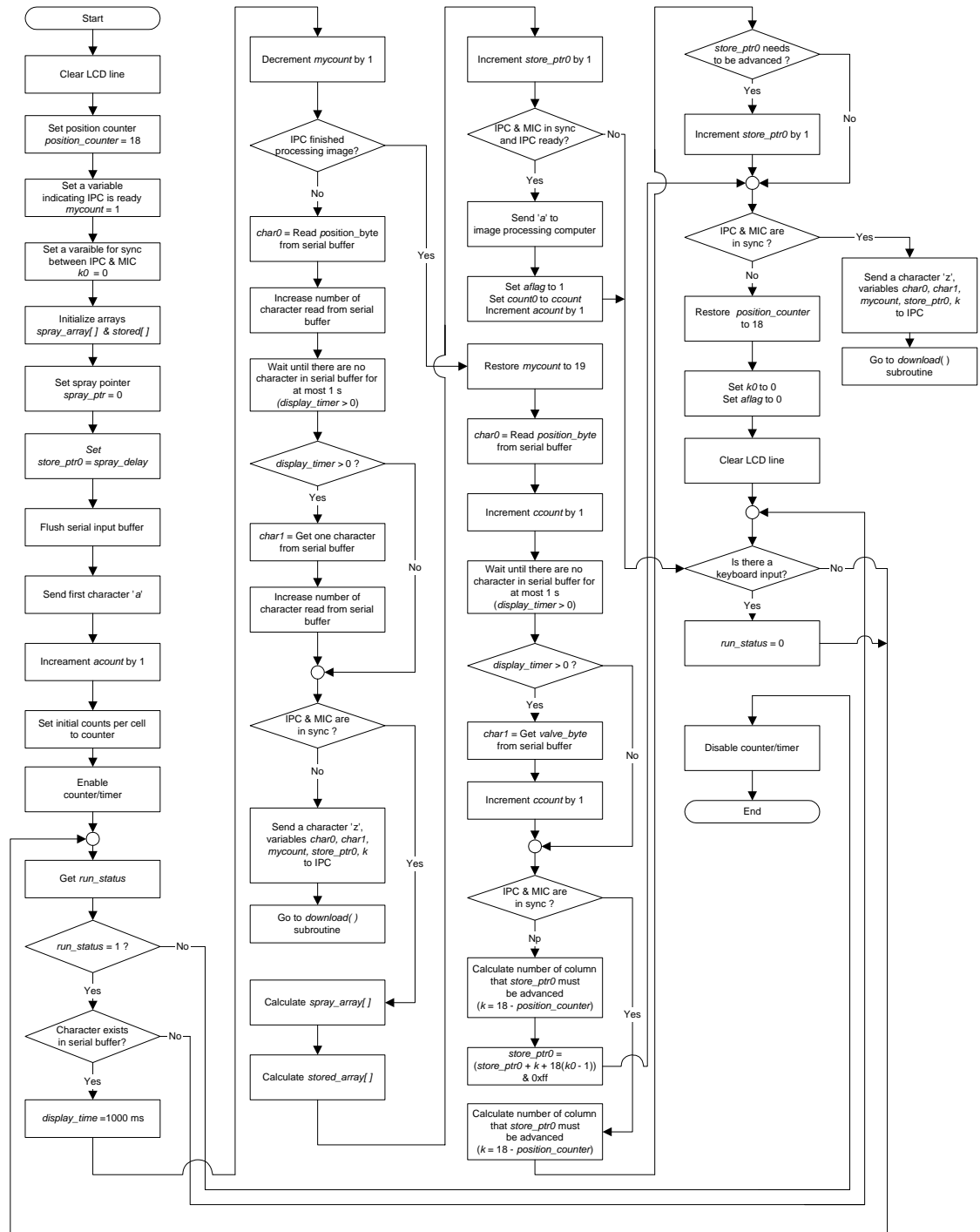
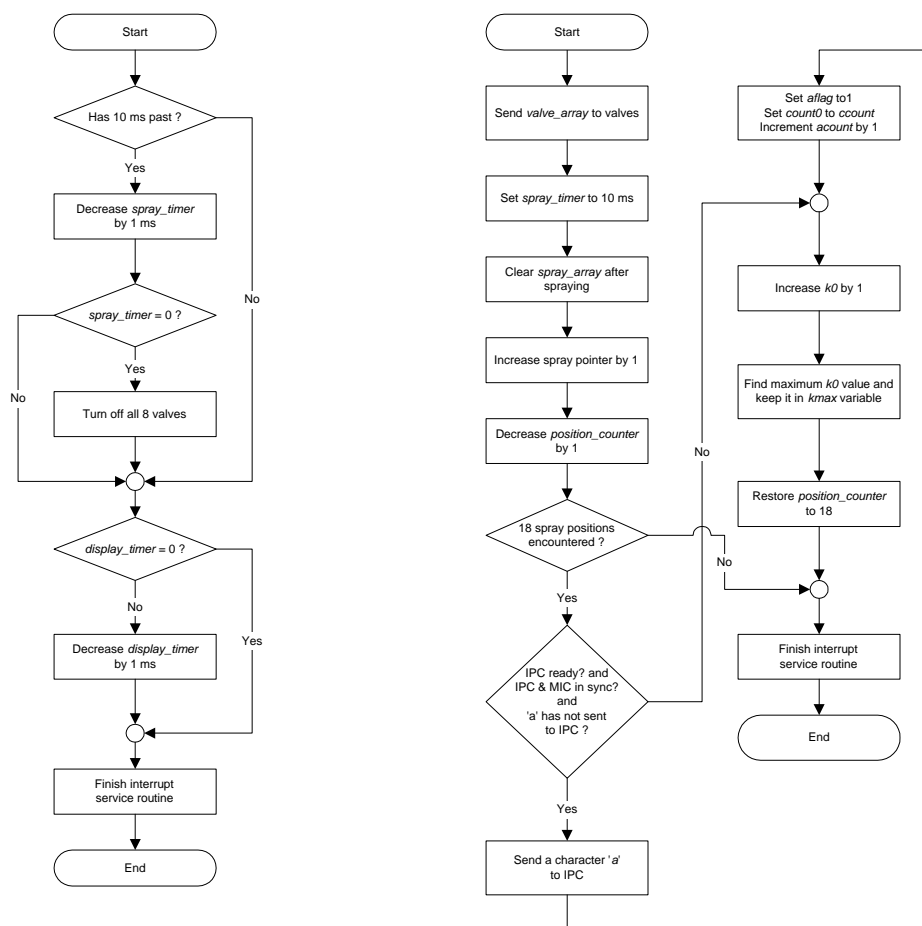


Figure 3.33 Flow chart of `main()` function.

Figure 3.34 Flow chart of *running()* function.

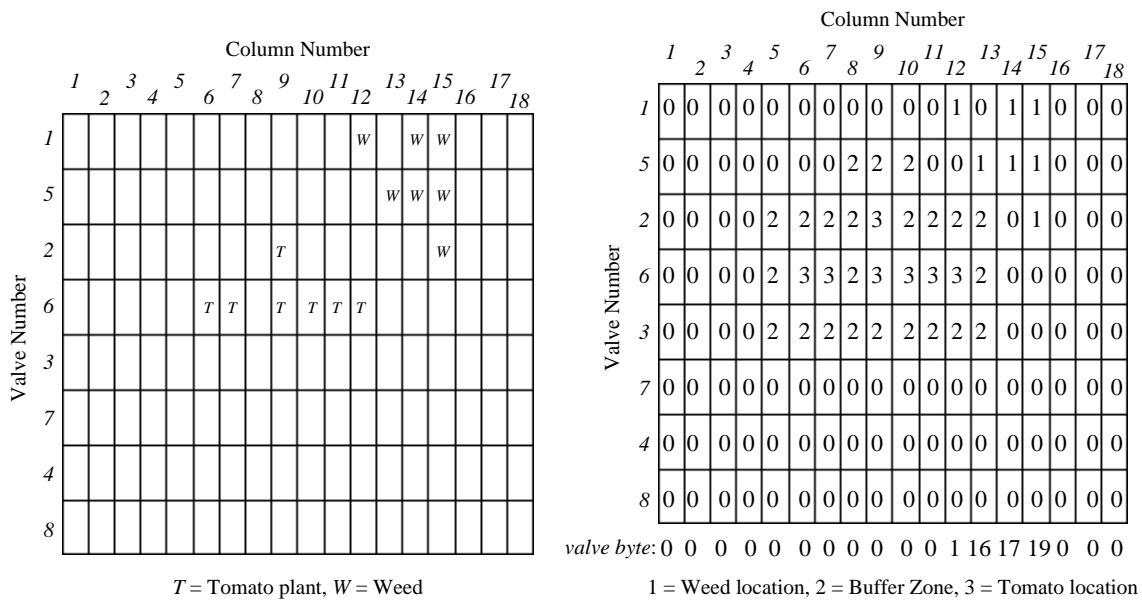


(a) Time base interrupt service routine,
tb_isr()

(b) Counter interrupt service routine,
ct1_isr()

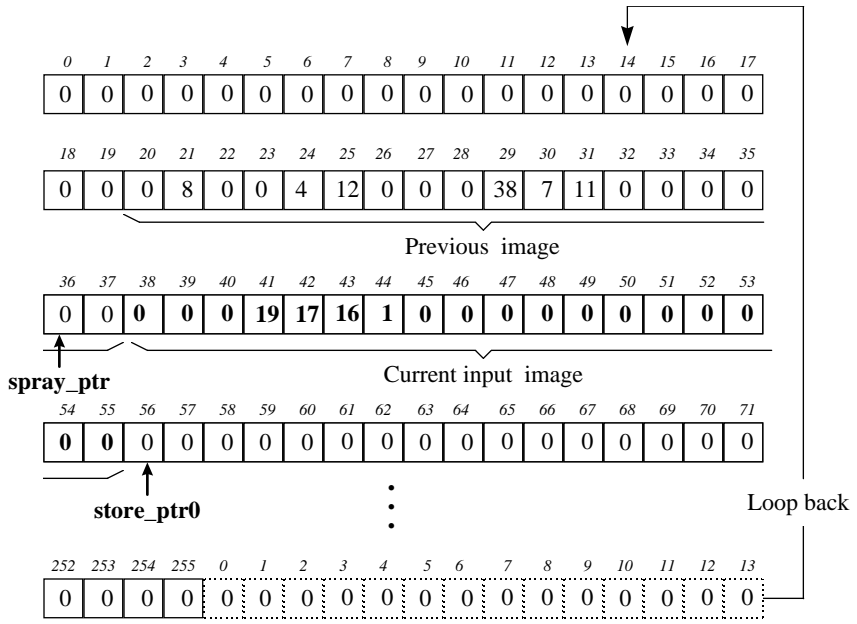
Figure 3.35 Flow chart of interrupt service routines.

The following is an example showing how *valve bytes* were calculated, encoded in the microcontroller and decoded after downloading them to the image processing computer. The input image used here is the same as Figure 3.38 (a), page 122. Figure 3.36 (a) shows an input image, where *T* signifies tomato plants and *W* signifies weeds.



(a) Input image

(b) Valve array



(c) Memory (*spray_array[]*) for valve byte in the microcontroller.

Figure 3.36 Example of calculating and storing valve bytes.

Figure 3.36 (b) shows a preparation step for calculating *valve bytes*. A number ‘1’ is assigned to the cells which contain weeds. A number ‘3’ is assigned to the cells which contain tomato plants. Here a number ‘2’ means protection zone, showing cells where tomato plants are protected from herbicide drift. The *valve byte* for each column is calculated using Eq. (3.44).

$$\text{Valve byte} = v1|(v2\ll 1)|(v3\ll 2)|(v4\ll 3)|(v5\ll 4)|(v6\ll 5)|(v7\ll 6)|(v8\ll 7) \quad (3.44)$$

where $v1 - v8$ are values in each cell for eight valves (rows) in a column.

The resulting *valve bytes* are as follows.

Column no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Valve byte	0	0	0	0	0	0	0	0	0	0	0	1	16	17	19	0	0	0

After calculating *valve bytes*, *position bytes* (column number) and *valve bytes* are sent to the microcontroller.

Figure 3.36 (c) illustrates the relationship of *valve bytes*, *spray_ptr*, and *store_ptr0* in the *spray_array[]* in the microcontroller memory. The variable, *spray_array[]*, is a circular buffer with 256 elements, and the pointers loop back when all elements are used. *Spray_ptr* points to the column of the input image, which is to be sprayed by the precision valve/nozzle, and *store_ptr0* points to the memory location for storing *valve bytes* in the *spray_array[]*.

In the microcontroller, *valve bytes* are re-arranged before being sent to the nozzles, since the valve order in the precision spraying system is valve1, valve 5, valve 2,

valve 6, valve 3, valve 7, valve 4, and valve 8 (see Figure 3.5) and valve numbers 1, 3, 5 and 7 are in the same line, valves 2, 4, 6, and 8 are in the other line. The valves in the second line are opened *noz_ofs* spray cells later than the ones in the first line of valves. The value of *noz_ofs* was 6 in this research. The *valve bytes* are re-calculated and stored in *spray_array[]* as follows.

$$spray_array[i] \dagger = valve\ byte \ \& \ 0x0f \quad (3.45)$$

$$spray_array[i + noz_ofs] \dagger = valve\ byte \ \& \ 0xff \quad (3.46)$$

The resulting *spray_array[]* is as follows.

spray array index	...	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
<i>spray_</i> <i>array</i>	...	0	0	1	0	1	3	0	0	0	16	16	16	0	0	...

The *spray_array[]* is sometimes downloaded from the microcontroller to the image processing computer and used to compare with the *valve bytes* sent from the image processing computer. Suppose *spray_array[]* is downloaded and stored in the *from_sw[]* array, *valve bytes* are decoded from the *from_sw[]* array in the following manner.

$$low4[i] = from_sw[i] \ \& \ 0x0f \quad (3.47)$$

$$hi4[i] = from_sw[i] \ \& \ 0xf0 \quad (3.48)$$

$$valve\ byte[i] = low4[i] + high4[i + noz_ofs] \quad (3.49)$$

The resulting *valve bytes* should be same as the ones originally sent to the microcontroller from the image processing computer.

3.7 Test procedure of precision chemical application system

3.7.1 Performance of encoder

In order to observe the operation of the encoder alone on different ground surfaces, the precision spraying system was set to spray all 8 valves at a predetermined spacing (22.86 cm on soil surfaces & paved roads and 11.43 cm on smooth concrete surfaces) using the image processing computer and the microcontroller. The image processing computer was used only to send the *sync bytes* to the microcontroller and no imaging operation was done during this test. The gage wheel pressure was set to 138 kPa. The distance between each of the two sprayed lines was measured, subtracted from the predetermined spacing and considered as errors.

3.7.2 Spray targeting accuracy without imaging

In order to isolate problems within the scope of the precision spraying system, the prototype was set to spray a predetermined “imaginary” weed pattern without taking any images and with no targets on the ground. The “imaginary” weed pattern was composed of 10 consecutive images, which were replayed from the hard disk. Each image had three or four weeds (targets) in it. Four types of ground surface (medium tomato bed, rough tomato bed, smooth concrete floor, and asphalt) were used to test the system and four repetitions were executed for each surface. The medium bed did not have any relatively large dirt clods or bumps on the bed and was a reasonably smooth soil surface, whereas the rough bed had big (about 10 cm diameter) dirt clods or bumps and was not smooth. The imaginary pattern was sprayed on a 12-image long paper strip (10.16 cm x 121.9 cm) laid on each surface.

After spraying the imaginary weeds, errors were measured by two different methods: Method I and Method II. The Method I was to measure errors from the starting line (beginning of the first image) for each test. The position of each sprayed drop was measured from the starting line of the test and the difference between the position of the sprayed drops and the position of the imaginary weeds was calculated and considered as errors.

Method II measured the spray errors based on their relative position at the time of spraying. The basic idea for this method was that for each image, the reference line for measuring spray drop locations should be set at a distance from the sprayed image equivalent to the distance traveled between the time the image was taken and the time the image was sprayed (i.e., the distance between Line A and Line B in Figure 3.37), to avoid artificially high levels of cumulative encoder wheel error. The positions of the actual spray drops were measured based on the line (Line A in Figure 3.37) where the camera was aligned at the time of spraying (Figure 3.37). A grid-printed transparency was placed over the sprayed paper and used to measure positions of spray drops. Then, the differences between the position of actual spray drops and the position of imaginary weeds were measured and considered as errors.

This procedure was used because one operating cycle of the prototype system started with acquiring an image and ended in spraying weeds in the image. The errors should not be accumulative, but were relative to the time of image acquisition. This method more accurately measured the errors based upon the actual operation of the prototype.

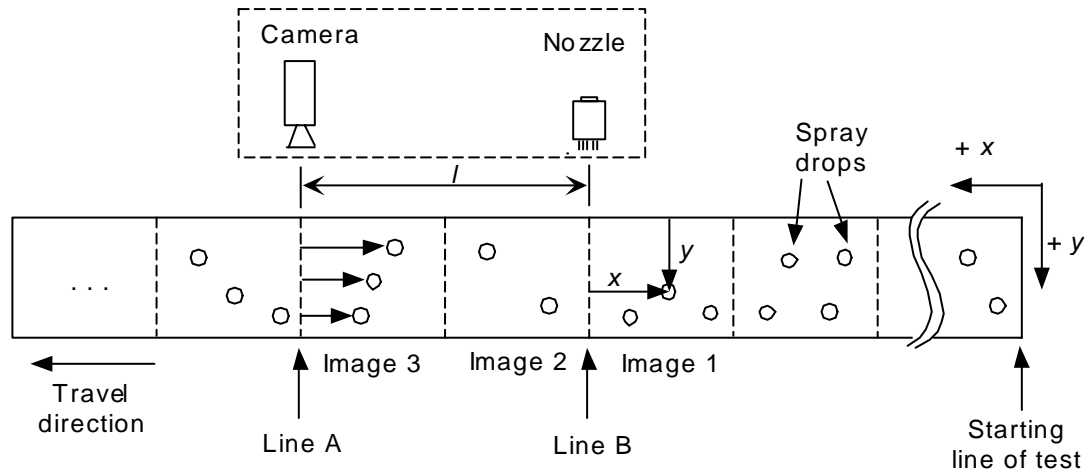


Figure 3.37 Measurement of location of spray drops (Method II).

More specifically, the position of the actual spray drops in Image 1 were measured with the following procedure as shown in Figure 3.37.

- (i) Find base line (Line A) of Image 3 from actual spray drops in Image 3 by averaging x and y locations of all spray drops in Image 3.
- (ii) Find base line (Line B) of Image 1 from Line A (two images apart).
- (iii) Put the grid transparency over the paper and align the transparency with Line A and Line B.
- (iv) Measure x and y locations of each drop in Image 1 from Line B using a calipers.
- (v) Repeat the previous three steps for all spray drops.

3.7.3 Spray targeting with imaging on different ground surfaces

Tests were conducted to evaluate the prototype system as a whole with imaging and spraying together. In order to estimate the spray accuracy of the whole prototype

system, six tests were conducted, three outdoors and three indoors. In these tests both imaging and spraying operations were conducted.

First, the prototype system was tested outdoors on two different tomato beds as well as a paved road using circular green metal targets (thickness 0.16 cm and diameter 2.54 cm) and green circular abrasive foam cleaning material (Scrub, 3M™ Inc.) targets (thickness 0.5 cm and diameter 2.54 cm). The targets were considered as “weeds” and attached to a 10.16 cm wide strip of corrugated fiberboard using double sided tape in order to prevent them from moving. A look-up table was created with a few training images to identify the color of the targets. The targets were laid down on the bed in rows 22.86 cm apart in a line of four or five targets perpendicular to the tractor’s direction of travel. Their centroids were sprayed with a blue dye (Precision Laboratories, Inc. SIGNAL™) by the robotic precision spraying system after they were detected by the computer vision system, while the tractor was moving forward at 0.8 km/h. The distance between the center of the coins and the center of the spray patterns were measured.

In the indoor experiments, a smooth concrete floor was used to represent an ideal surface since it did not have any bumps or dirt clods, which could affect the encoder count. The prototype system was mounted on a cart and pulled manually along the center of the paper strip. The same procedure used outdoors was repeated for the first two indoor experiments except that both 2.54 cm and 1.27 cm diameter paper targets were used as “weeds”. In the last indoor experiment, rectangular targets (0.64 cm x 1.27 cm) which approximated the tomato cotyledon shape and were considered as “tomato cotyledons” were also used, while the circular targets (1.27 cm diameter) were considered as “weeds”. In these tests, paper targets were used since metal targets were

too glossy to be recognized as circular targets and abrasive cleaning material absorbed spray drops, producing difficulties in measuring the distance from the center of the spray drops.

3.8 Procedure for field testing of the prototype system

3.8.1 Speed of the image processing commands and algorithm

Processing time is a major concern in real-time machine vision applications. Since the goal of this research was to develop a real-time robotic weed control system, computationally intensive steps were avoided. The processing time of each image processing step was measured using the computer clock and a 200 MHz Pentium processor was used to process an image. Each execution time was averaged from 1000 execution times. In this test, a frame of a 256 by 240 pixel image representing a 11.43 cm x 10.16 cm field of view was used with 10 tomato cotyledons in the image using only the features of ELG and CMP.

In an effort to reduce the processing time, some objects were not identified but were considered to be weeds if they were at the very top or bottom of an image or if they were too small or too big. Also, the algorithm checked the center and 4 corner points of each spray cell in the processed image in order to determine whether to spray that cell, rather than scanning every pixel of the entire image. Image acquisition time was also saved by acquiring only a *field* instead of a *frame* and subsampling it columnwise (to eliminate distortion).

3.8.2 Field testing of the prototype system

The prototype robotic weed control system was tested in commercial tomato fields in northern California from March to May 1997. The travel speed was about 0.8 km/hr and the tomato plants ranged from just emerging up to the first true leaf stage.

In the actual field test, the following shape features were obtained for each binarized plant leaf: area (AREA), major axis (MJX), minor axis (MNX), centroid (CNTRD), perimeter (PERIM), compactness (CMP), and elongation (ELG). These features were different than those determined in the plant recognition performance section 4.2.2 since the field tests were run before the results of the best feature subset (result of section 4.2.2) were obtained and at that time the hardware did not seem to be fast enough to handle any more than the above features used in the field tests.

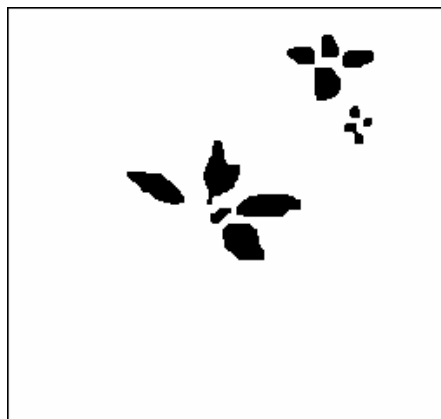
The centroid ('+' sign), major and minor axis, and perimeter from the feature extraction process are shown in Figure 3.38 (c). With these features, plant leaves were identified either as tomato cotyledons or as weeds for non-occluded leaves using a Bayesian classifier. Shown in Figure 3.38 (d), tomato leaves are identified in green and weeds in red. The algorithm does not spray weeds in any cells adjacent to tomato cells in order to protect tomato seedlings from spray drift, Figure 3.38 (e).

Since the timing for real-time operation is essential, a few preprocessing tests were made to reduce the processing time of an image. In an effort to reduce the processing time, objects were not processed and considered to be weeds if they were on the very top or bottom of an image or if they are too small or too big. Also, the algorithm checked only the center and 4 corner points of each spray cell in the classified image in

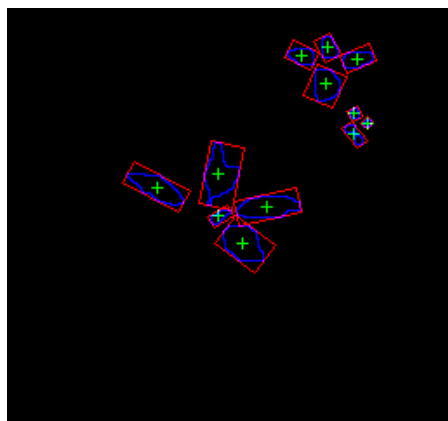
order to determine whether to spray that cell, rather than scanning every pixel of the entire image.



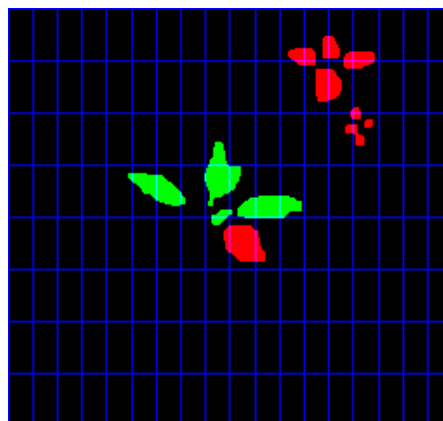
(a) Raw image of a tomato seedling and weeds in a commercial field.



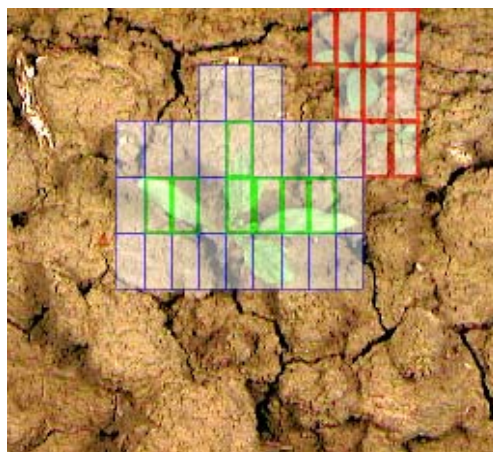
(b) Binary image.



(c) Feature extraction of each object.



(d) Processed image, tomato leaves in green & weeds in red.



(e) Tomato cells (green cell), buffer zone (blue cell), and spray zone (red cell) overlaid on the original image.

Figure 3.38 Process of tomato and weed recognition.

3.9 Cotyledon opening experiment in a field

The orientation of plant leaves has a significant impact on the feasibility of accurately determining their morphological characteristics from a single top view. During field tests in 1996 and 1997, the tomato cotyledons in commercial fields varied in their degree of openness. The ‘openness’ of a cotyledon, or ‘cotyledon angle’ can be defined as an angle between a cotyledon and an extended axis of the stem from their side view (Figure 3.42, page 127). When the images collected during these periods are compared with those collected in 1993 and 1994 by Tian (1995), the 96-97 cotyledons are considerably more closed. When cotyledons are closed, it is difficult to assess their true shape using a single top view increasing the probability that they will be classified as weeds since their feature values (e.g., ELG and CMP) become closer to those of nightshade or grass weeds. A fundamental question as to what makes cotyledons open and close arose and factors such as temperature, water supply, sunlight or variety were considered as possible causes.

A study was designed to evaluate varietal effects on cotyledon orientation. Sixteen tomato varieties (Table 3.20) were selected from the top 50 processing tomato varieties produced in 1995 (Processing Tomato Advisory Board). These varieties were planted in a 200 ft long single row in May 1997 in an experimental field plot on the University of California, Davis campus farm. Each variety was planted with about 2 seeds per hole in 10 ft long block, each hole was 0.2 ft apart and each block was 2 ft apart. At cotyledon stage, video images were taken at different times of the day, i.e., morning, afternoon, and evening.

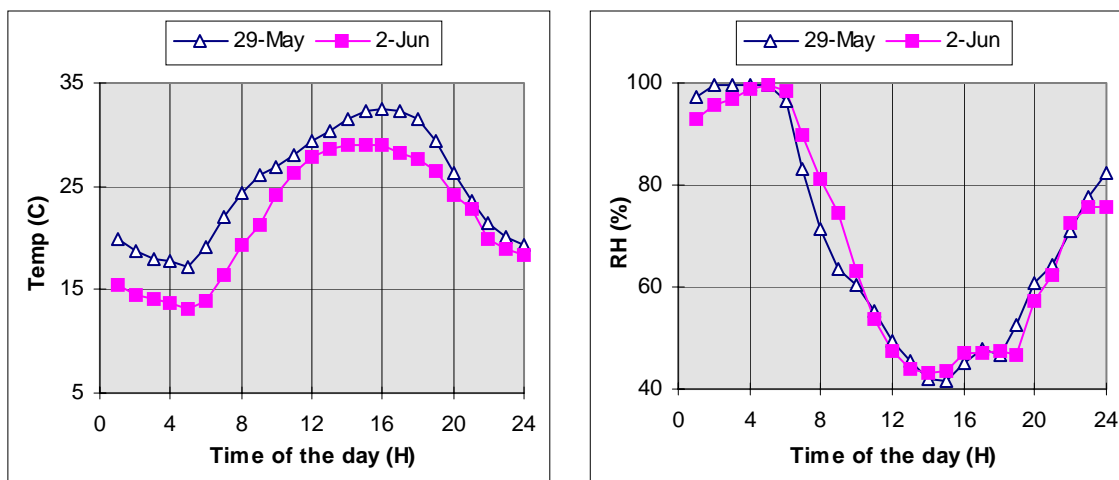
Table 3.20 Sixteen varieties used in the cotyledon opening experiment.

Variety	Assigned letter
HALLEY, BOS 3155	<i>A</i>
HEINZ 8892	<i>B</i>
BRIGADE	<i>C</i>
HEINZ 3044	<i>D</i>
HEINZ 9280	<i>E</i>
PETO NEMA 512	<i>F</i>
CAMPBELL CXD 152	<i>G</i>
FM 1047	<i>H</i>
SUN 5715	<i>I</i>
HP 108	<i>J</i>
ORSETTI, BOS 707	<i>K</i>
HALLEY, BOS 3155	<i>L</i>
FM PEELMECH	<i>M</i>
CAMPBELL, ALTA	<i>N</i>
FM 6203	<i>O</i>
CXD 189	<i>P</i>

Images were taken on two different days (May 29 and June 2, 1997). On the first day, pictures of varieties *A* through *H* were taken at 10 AM and 4 PM, and on the second day pictures of all sixteen varieties were taken at 8 AM, 4 PM and 8 PM (Table 3.21). The number of images for each variety were not same since the cotyledons emerged on different dates. Figure 3.39 shows temperature and relative humidity on these two days as measured at the University of California, Davis weather station.

Table 3.21 Cotyledon imaging schedule.

Date	Time	Variety
Day 1: May 29, 1997	10 AM, 4 PM	<i>A, B, C, D, E, F, G, H</i>
Day 2: June 2, 1997	8 AM, 4 PM, 8 PM	<i>A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P</i>



(a) Temperature.

(b) Relative humidity.

Figure 3.39 Temperature and Relative Humidity on May 29 and June 2, 1997 at the University of California, Davis Weather Station

Two basic cotyledon patterns were observed as illustrated in Figure 3.40 and Figure 3.41. For some varieties (e.g. FM 1047), the cotyledons stayed open from the morning to the late afternoon (Figure 3.40 (a) - (d)) and then started closing at dusk (Figure 3.40 (e)), becoming completely closed during the night (Figure 3.40 (f)). However, this change does not apply to all plants or varieties. HALLEY, BOS 3155 show little change at different times of the day (Figure 3.41).

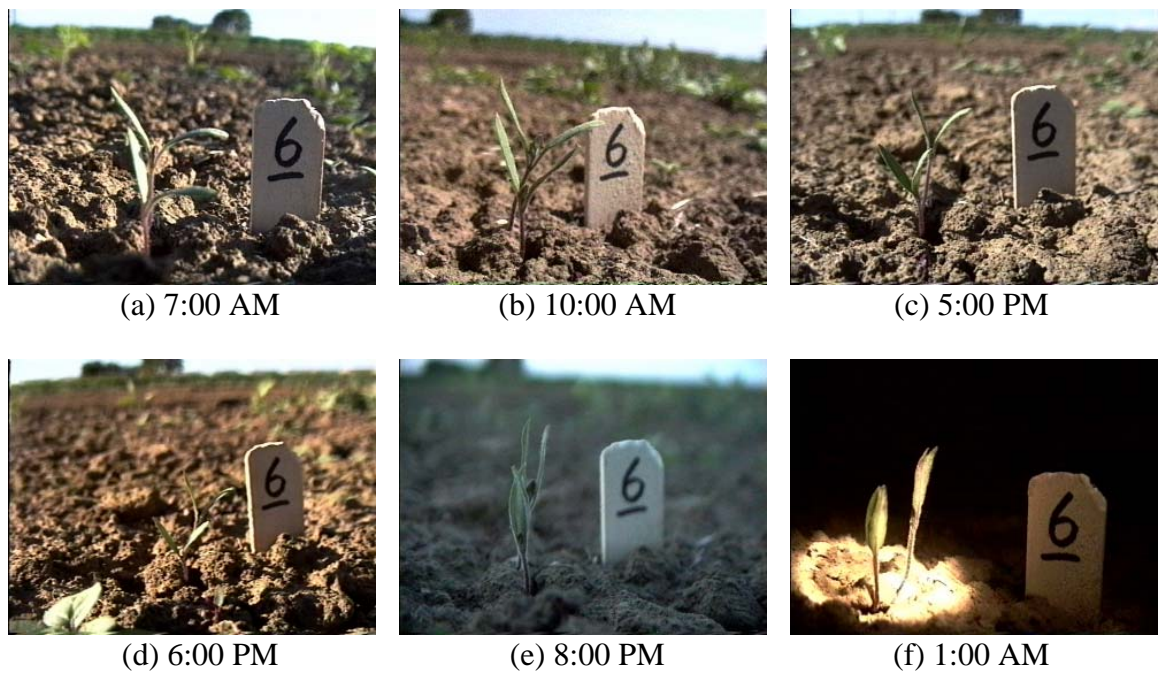


Figure 3.40 Movement of FM 1047 tomato cotyledons at different times of day.

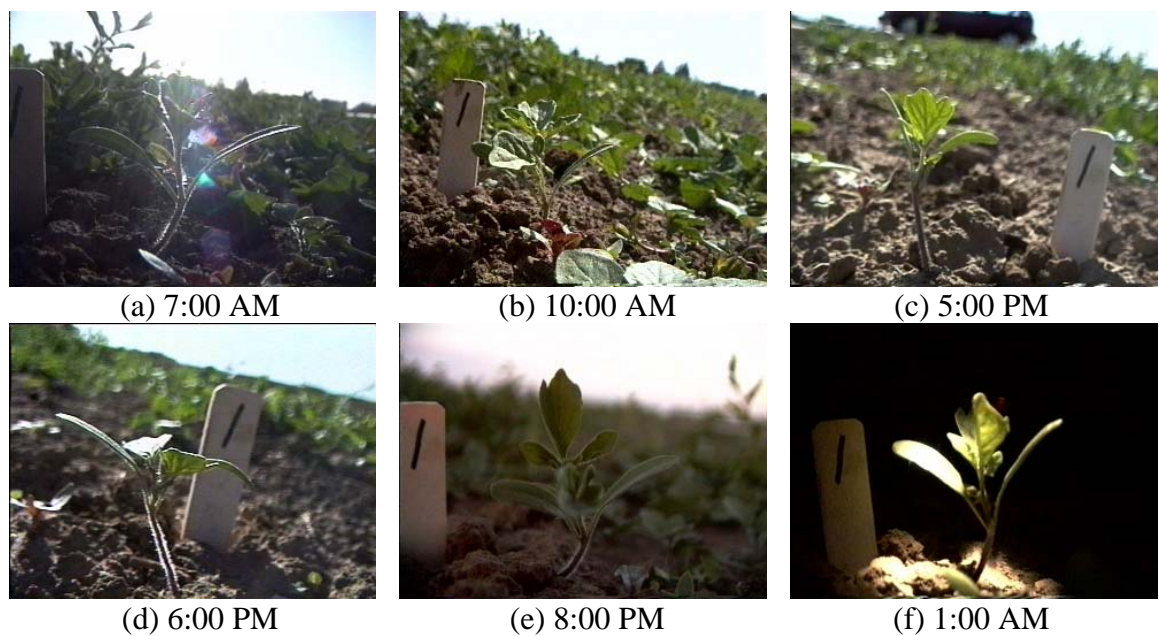


Figure 3.41 Movement of HALLEY, BOS 3155 tomato cotyledons at different times of day.

After images were acquired and digitized from the video tape, the angle of each cotyledon was measured from the side view as in Figure 3.42 using Scion ImagePC (National Institutes of Health, USA. 1997. Modified for Windows by Scion Corporation) software. The angle of each cotyledon's openness was quantized as a dimensionless ratio of the apparent leaf length from the top (a and b) to the true leaf length from the side (l_1 and l_2).

$$\text{Ratio L} = a / l_1 \quad (3.50)$$

$$\text{Ratio R} = b / l_2 \quad (3.51)$$

The openness of a cotyledon (an angle of a cotyledon), γ can be calculated as in Eq. (3.52).

$$\gamma = \sin^{-1} (a / l_1) \quad (3.52)$$

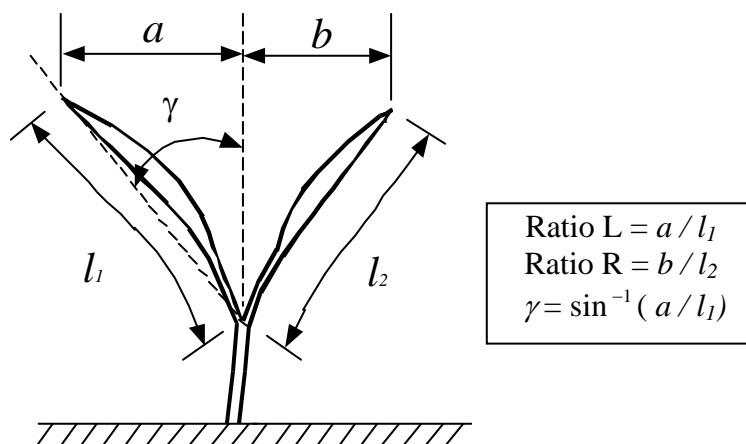


Figure 3.42 Measurement of tomato cotyledon opening.

An analysis of variance and Tukey's means test were conducted using the SAS GLM procedure (SAS Institute Inc., 1993) to determine if different varieties had

significantly different cotyledon angles. A GLM analysis was done to determine if the average cotyledon angle was different between two days at 4 PM for varieties of *A* through *H*. The same analysis was done with the data set of 10AM-Day1 and 8AM-Day2.

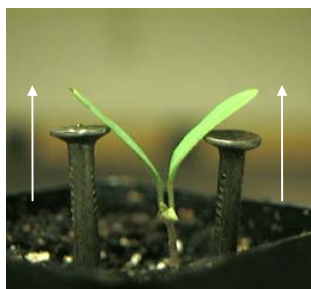


Fig. 3.43 Cotyledon angle of recognition.

To determine at what angle the performance of cotyledon recognition affects machine vision recognition, the cotyledons were raised from a horizontal position to determine the critical angle of recognition. Six tomato cotyledons were used to estimate the critical angle. The

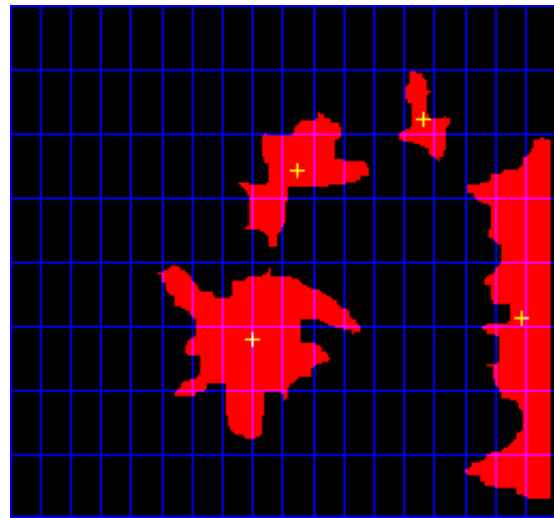
cotyledons were put under the camera and raised slowly and continuously. At the critical angle the cotyledons switch from being recognized as tomatoes and are recognized as weeds by the machine vision system due to their apparent change of shape using only ELG and CMP features, similar to the real-time algorithm used in the actual field test (section 4.5). When cotyledons reach this critical angle of recognition, their side and top pictures were taken. Then, the critical angle of recognition was calculated from the side images as in Figure 3.43.

3.10 Transgenic purple tomato plants

Some of the difficulties in real-time machine vision applications are processing time and occlusion of objects. In order to keep up with the conventional cultivation speed (~ 3.2 km/hr), the processing speed of the current prototype system needs to be increased. The leaf morphology recognition algorithm has difficulty in recognizing tomato plants when plant leaves are overlapped or are at a later growth stage than cotyledons. Figure 3.44 illustrates these problems. In this example, a weed is growing underneath a tomato plant, and they are segmented as one binary object. In addition, since the tomato plant has advanced beyond the cotyledon stage, it is more difficult to recognize due to the complexity of the shape.



(a) Tomato seedling with a weed growing underneath.



(b) Objects recognized as weeds are shown in red.

Figure 3.44 Example scene showing recognition problems when plants overlap.

In an attempt to recognize tomato plants beyond the cotyledon stage in real-time, the feasibility of identifying tomato plants by their color alone was investigated using tomato plants with “purple” foliage. A group of researchers, including Professor John Yoder at UC Davis, developed “purple” tomato plants by introducing the maize anthocyanin regulatory *Lc* gene into the tomato (Goldsbrough et al. 1996). The transgenic purple tomato plants could help solve the occlusion problem and greatly increase the processing speed by eliminating many morphological recognition operations. Figure 3.45 shows an example of a tomato plant with purple foliage.



Figure 3.45 Tomato plant with purple foliage.

Utilizing color alone for distinguishing plants from weeds would provide a number of advantages from an image processing perspective. A great advantage is that

there is no need to extract the morphological features which took about 42.0% of the processing time in the morphologically based algorithm (See Chapter 4.4). This is a great saving in processing time. Using purple tomato plants could also reduce the time required for some of the pre-processing steps such as erosion, dilation, shrink, swell and deletion of the isolated points since accurate leaf shape is not critical.

Another significant advantage is that the performance of the prototype would not be as affected by partial occlusion of plant leaves. Since plants in any other color except “purple” are weeds, we are no longer worried about partially overlapped plant leaves. In other words, the detection of the whole leaf shape is not needed in order to see color. In addition, the color characteristics could reduce problems associated with wind and diurnal change of plant appearance. It is no longer important that plants are laid down or their leaves are not fully open, as long as their color is visible. Plants older than the cotyledon stage could be identified easily and accurately since the shape of occluded leaves don’t need to be recognized to see color.

In order to investigate the feasibility of using color as the sole criteria for recognizing tomatoes, purple tomatoes were planted in May 1997 in a 200 ft long row in an experimental field at the University of California, Davis. After they had grown to the second true leaf stage, top view pictures were taken using a camcorder (Yashica, Model KD-H170). The pictures were digitized using the Sharp image processing hardware as still images and stored in a hard disk. Two color look-up tables (LUT) were created using some training images, one for purple tomato plants and the other for green weed plants. Once the LUTs were made, a computer vision algorithm was developed to recognize the purple tomato plants, Figure 3.46. In order to speed up the process, only

one pre-processing operation (i.e., erosion) was done for each binary image. The recognition performance is described in section 4.5.

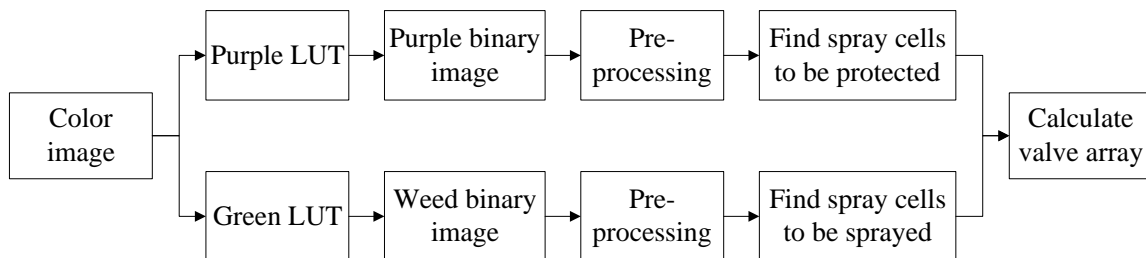


Figure 3.46 Recognition procedure for transgenic purple tomato plants.

4. RESULTS AND DISCUSSION

4.1 LUT performance

LUT performance was evaluated using binary images created using different LUTs based upon different color spaces and using 2 or 3 input classes (i.e., 2 classes: plants and background, or 3 classes: tomato plants, weeds, and background). Fifteen validation images (distinct from those in the training set) were chosen randomly among the images acquired in 5 different commercial processing tomato fields. From each image, a plant-only and a background-only image were created and the number of non-black plant and background pixels, respectively, were counted. The total number of non-black plant pixels in 15 plant-only images was 67333 ($= n_1$) and the total number of non-black background pixels in 15 background-only images was 854267 ($= n_2$).

In this chapter the LUT performance was evaluated by two methods. The first method (Method I) was to use the binary images from LUTs with noise removal operation. The second method (Method II) was to evaluate LUTs with single point noise (one pixel noise in the background) removed from the background-only binary images. This method was used since single point noise could be easily removed by the Sharp board.

There were three types of noise in the background: single point, small cluster, and large cluster (Figure 4.1). The goal of segmentation was to extract only plant pixels from a color image, so the background noise should be removed. The single point noise (Figure 4.1 (b)) was generated when a single pixel with a similar color to a plant pixel in the soil background was surrounded by soil pixels that were not similar in color to a plant pixel. However these individual noise pixels were easily removed with a built-in Sharp

image processing command which removed isolated pixels (`s_delisolp`). Small clusters of noise were defined as a group of soil or background pixels with an area of less than 50 pixels which passed through a LUT as plant pixels (shaded squares in Figure 4.1 (c)). Small clusters of noise were removed by using a combination of the built-in Sharp commands such as median filtering (`s_kmedian`), smoothing (`s_smooth`), binary erosion (`s_berode`), or binary shrink (`s_shrink`).

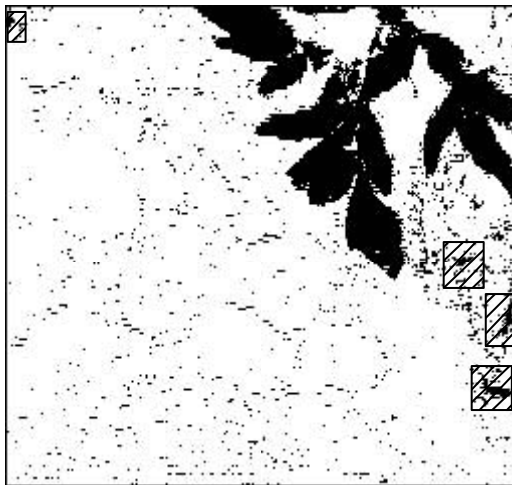
Sometimes a large cluster of noise (larger than plant size, Figure 4.1 (d)) was generated either when the pixel value of the noise was not included in the training set for LUT generation or when sunlight penetrated the cultivation tunnel from beneath the side shields (as in the right edge of Figure 4.1 (a)), however this type of noise was unusual since background pixels from nearly every possible situation were included in the training set for LUT generation and the cultivation tunnel was well shielded from sunlight most of the time. This type of noise could be removed by area thresholding once the objects were labeled. If the area of the noise is greater than the area thresholding, it would be removed. When a large cluster of noise was removed by area thresholding, plants which had greater area than the large noise cluster could also be removed, however this was an unavoidable trade-off in removing a large cluster of noise.



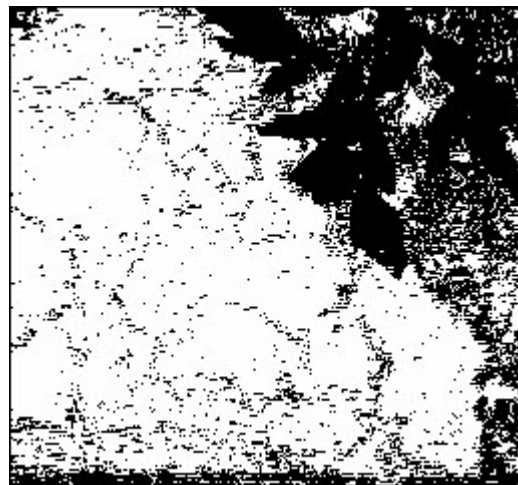
(a) Raw color image.



(b) Single point noise from LutRGBC3.



(c) Noise of small cluster (inside shaded square) from LutRGC3.



(d) Noise of large cluster (right side and bottom of the image) from LutHC3.

Figure 4.1 Types of background noise in binary images.

High levels of background noise had an adverse effect on obtaining accurate plant shapes in binary images and on execution times. Most of the background noise could be removed with 3 or 4 successive erosions based on the degree of noise level. However, when there was a lot of background noise, the noise affected plant shape recognition since plant pixels are also removed when too many successive erosions were conducted.

It is important to keep plant pixels from being removed in a binary image pre-processing step while removing the background noise at the same time.

Method I

Table 4.1 shows LUT performance, ranked by their background error rate. The LUT made with H , S , and I components and three input classes (LutHSIC3) showed superiority in removing noise from the background-only image. The LUT made with r and g color components and three input classes (LutrgC3) produced the most error in binary conversion for background-only images. These facts indicated that the I component helped to remove the background noise since the plane composed of r , g , and b components ($r + g + b = 1$) in rgb color space is perpendicular to the I axis.

Table 4.1 LUTs listed by Background error rate and number of pixels removed by 1 erosion in the background-only image.

Rank	LUT	Background error rate (%)	No. of pixels before erosion	No. of pixels after erosion	Pixels removed (%)
1	LutHSIC3	1.67	1267	0	100.00
2	LutRGBC2	3.46	1345	0	100.00
3	LutRGC2	4.21	2074	27	98.70
4	LutRGBC3	4.21	1593	0	100.00
5	LutHSC3	4.35	5691	103	98.19
6	LutHSIC2	4.96	2178	0	100.00
7	LutRGC3	5.05	2262	30	98.67
8	LutrgC2	9.19	10923	1404	87.15
9	LutHSC2	9.21	10842	905	91.65
10	LutHC2	10.54	12200	2047	83.22
11	LutHC3	10.69	12226	2047	83.26
12	LutrgC3	11.17	11906	1992	83.27

Regarding the background noise, it is interesting to know how many pixels could be removed by a single noise removal operation. Table 4.1 also shows the number of pixels removed by one erosion for the background-only binary image for each LUT. In the background-only binary image, over 83.22% of noise (four of the 12 LUTs had 100.00% removal) were easily removed using only one erosion for all LUTs. Figure 4.2 shows an example of the effect of 1 erosion on the number of pixels removed in background-only and plant-only binary images. This example image is an extreme case since unwanted sunlight came in on the right side of the image. However, if the appropriate look-up-table (LutRGBC3) is used for binarization, the background noise is completely removed as well as keeping most plant pixels (Figure 4.2 (h) and (i)).

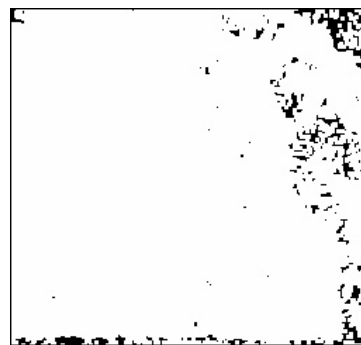
The execution time for the built-in Sharp commands for removing background noise, deletion of single noise (`s_delislp`), binary erosion (`s_berode`), and binary shrinking (`s_shrink`) took about 3 ms, and median filtering (`s_kmedian`) took about 6 ms with an image of 256 x 240 pixels containing about 10 objects. The typical pre-processing steps executed on a binary image (Table 3.3) were composed of 1 deletion of isolated pixels, 1 binary erosion, 1 binary dilation, 4 binary shrinkings, and 3 binary swellings, thus several steps (over 10 ms) could be saved if there was less noise in the segmented image.



(a) Background-only image.



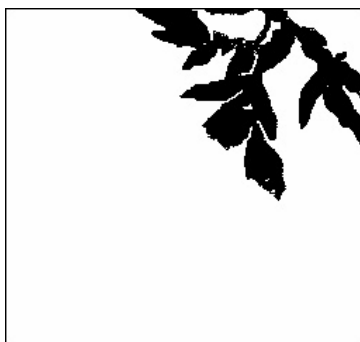
(b) Binary image from LutHC2.



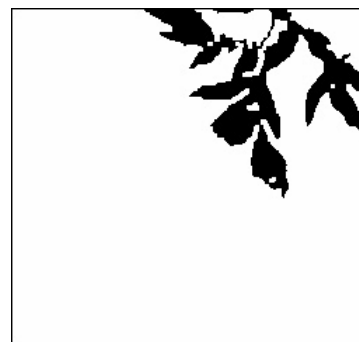
(c) Eroded image of (b).



(d) Plant-only image.



(e) Binary image from LutHC2.



(f) Eroded image of (e).



(g) Whole color image.



(h) Binary image from LutRGBC3.



(i) Eroded image of (h).

Figure 4.2 Effect of 1 erosion on background-only and plant-only binary images.

All in all, background noise was not a big problem since it was relatively easy to eliminate in most circumstances. Plant error rate was more important than background error rate since accuracy in extracting plant pixels from a raw color image was more closely related with plant shape recognition. Therefore, the LUTs were evaluated by their plant error rate even though the better LUTs for plant-only image produced more large clusters of noise than other LUTs as discussed further in this chapter.

Table 4.2 shows the LUTs listed by their plant error rate, the number of plant pixels not passed through LUTs ($= n_1 - n_3$), and the number of background pixels incorrectly passed through LUTs ($= n_4$). It also shows the plant error rate, background error rate and total error rate for each LUT.

Table 4.2 Number of pixels passed through each LUT and their performance listed by their plant error rate.

LUT name	$n_1 - n_3$	n_4	Plant error rate (%)	Background error rate (%)	Total error rate (%)
LutHC3	7719	91306	11.5	10.7	22.2
LutHC2	7973	90030	11.8	10.5	22.4
LutrgC3	8037	95390	11.9	11.2	23.1
LutHSC2	8789	78654	13.1	9.2	22.3
LutrgC2	9153	78480	13.6	9.2	22.8
LutRGBC3	11486	35986	17.0	4.2	21.3
LutRGC3	11510	43110	17.1	5.1	22.1
LutRGBC2	12258	29581	18.2	3.5	21.7
LutHSIC2	12326	42381	18.3	5.0	23.3
LutRGC2	12394	35958	18.4	4.2	22.6
LutHSC3	15147	37167	22.5	4.4	26.9
LutHSIC3	17792	14254	26.4	1.7	28.1

The performance of a LUT could be evaluated by its total error rate since this tells how many pixels were accurately passed through LUT for a given number of input pixels.

The LutRGBC3 produced the least total error rate and the LutHSIC3 produced the most error in binary conversion. However, as discussed before in this chapter, the noise in the background was not a big problem since these pixels were relatively easy to remove in most situations.

The LUT built using only the H component and three input classes (LutHC3) produced the most plant pixels in the plant-only binary images. The LUT with H , S , and I components and three input classes (LutHSIC3) produced the largest plant error rate in the plant-only binary images.

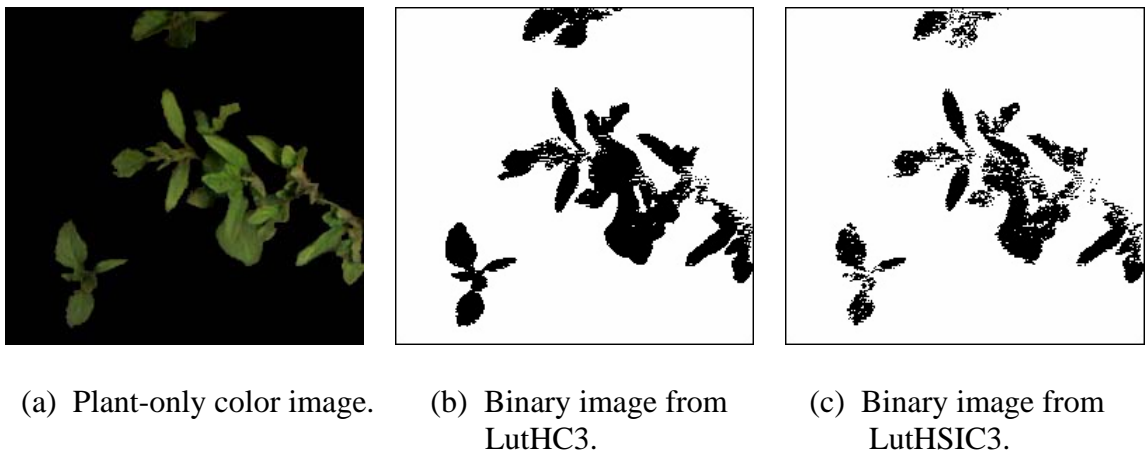


Figure 4.3 Effect of LUTs on a plant-only image.

It is important to find out that in each color space, how many input classes and which color components produced better LUTs and then, which color space was better than others. For example, in implementing a LUT, it is very important to know that LutHSC2 is better than LutHSC3 since these two LUTs have the same color components

but different number of input classes or that LutHSC2 is better than LutRGC2 (same number of input classes but different color components).

In *RGB* color space, when the same number of input classes were used, there was little difference between the LUT made with *R*, *G*, and *B* components (LutRGBC_{*i*}) and the LUT made with *R* and *G* components (LutRGC_{*i*}). There was no difference between the LUT made with *R*, *G*, and *B* components (LutRGBC3) and the LUT made with *R* and *G* components (LutRGC3) when three input classes (i.e., tomato plant, weed and background) were used. When two input classes (plant and background) were used, the LUT made with *R*, *G*, and *B* components (LutRGBC2) was a little better than the LUT made with *R* and *G* components (LutRGC2). Example binary images from these LUTs are shown in Figure 4.4.

The LUTs made with the 3 input classes of tomato plants, weed and background (LutRGBC3 and LutRGC3) produced a lower total error rate than those made with 2 input classes of plants and background (LutRGBC2 and LutRGC2) in *RGB* color space. This fact could indicate that there were slight differences between tomato plants and weeds that were useful in building a LUT. However, if the difference was calculated in terms of the actual average number of pixels per image (Table 4.3), the difference was only 51.5 pixels per image (= 817.2 pixels - 765.7 pixels) between LutRGBC3 and LutRGBC2 and 59.0 pixels per image (= 826.3 pixels - 767.3 pixels) between LutRGC3 and LutRGC2. The difference of 51.5 pixels and 59.0 pixels were only 1.1% and 1.3% of average number of plant pixel per image (= 4488.9 pixels = 67333 pixels / 15 images) and could come from the separation step of plant-only and background-only images.

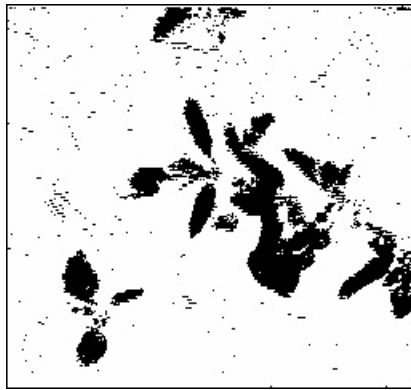
Therefore, LUTs with 3 input classes were a little better than those with 2 input classes, but the difference was negligible in *RGB* color space.

Table 4.3 LUT performance with respect to the number of input classes.

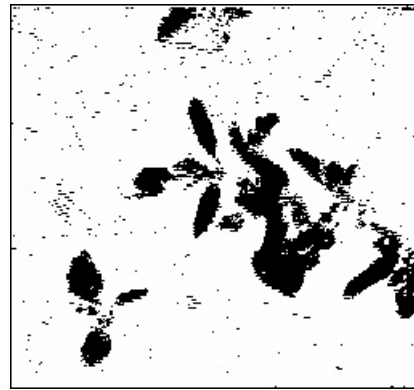
LUT name	No. of plant pixel, n_1	No. of correctly passed pixel, n_3	No. of incorrectly removed pixel, $n_1 - n_3$	No. of incorrectly removed pixel per image, $(n_1 - n_3) / 15$
LutRGBC3	67333	55847	11486	765.7
LutRGBC2	67333	55075	12258	817.2
LutRGC3	67333	55823	11510	767.3
LutRGC2	67333	54939	12394	826.3



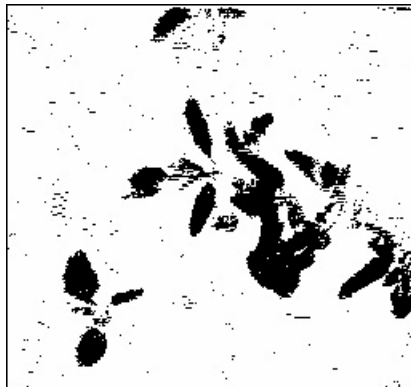
(a) Original color image.



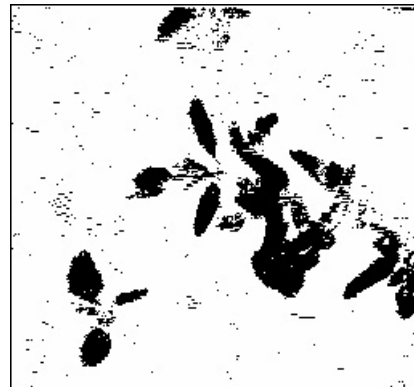
(b) Binary image from LutRGBC2.



(c) Binary image from LutRGBC3.



(d) Binary image from LutRGC2.



(e) Binary image from LutRGC3.

Figure 4.4 Effect of different LUTs on binary image quality in *RGB* color space.

In *HSI* color space (Figure 4.5), the LUTs built using only the *H* component (LutHC3 and LutHC2) were better than those made with *H* and *S* components (LutHSC2 and LutHSC3) and those made with *H*, *S*, and *I* components (LutHSIC2 and LutHSIC3) in terms of plant error rate. Adding the *S* and the *I* components to the *H* component did not produce better results in LUT performance. Adding the *I* component to a LUT made with *H* and *S* components produced a lot more error when two or three input classes were used, i.e., 13.1% plant error rate for LutHSC2 vs. 18.3% plant error rate for LutHSIC2 and 22.5% plant error rate for LutHSC3 vs. 26.4% plant error rate for LutHSIC3. The *I* component did not help the performance of the LUTs in terms of plant error rate. This might be because any change in the camera aperture produced intensity variations among pictures when they were acquired from field to field.

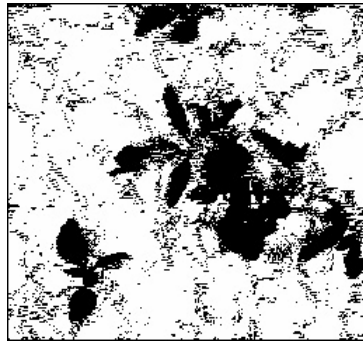
Regarding the number of input classes used in *HSI* color space, the LUTs made with 2 input classes, plants and background (LutHSC2 and LutHSIC2), produced much less plant error than those made with 3 input classes (LutHSC3 and LutHSIC3). More specifically, when the *H* and the *S* components were used to make LUTs, the LUT with 2 input classes (LutHSC2) was better than the one made with 3 input classes (LutHSC3). When the *H*, *S*, and *I* components were used to make LUTs, the LUT made with 2 input classes (LutHSIC2) was also better than the one made with 3 input classes (LutHSIC3). These facts were in disagreement with the results in *RGB* color space. However, Figure 3.14 (d) on page 52 shows there is no separation of plant and background using the *S* and the *I* components. Because *S* and *I* components were essentially noise, the 2 vs. 3 class conclusions based on *S* or *I* components were not meaningful. Thus, if only the LUTs with the *H* component were considered, the total error rate with 3 input classes (LutHC3)

were slightly better than the LUT with 2 input classes (LutHC2), but the difference was probably negligible (22.2% vs. 22.4%), similar to that found in *RGB* color space. In this case the total error rate was used since the Bayes classifier was built in a way that it minimized the total error rate.

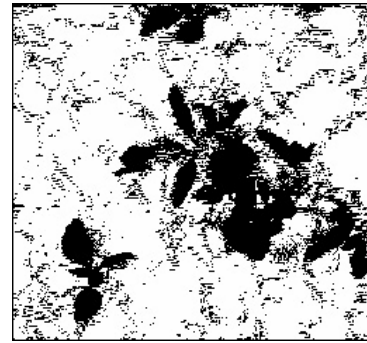
Comparing the binary images in Figure 4.5, the LutHC2 and LutHC3 generated more background noise than LutHSC2, LutHSC3, LutHSIC2, and LutHSIC3. However, as discussed previously, since the background noise could be removed easily and LutHC2 and LutHC3 produced more plant pixels, hue based LUTs were better than the others.



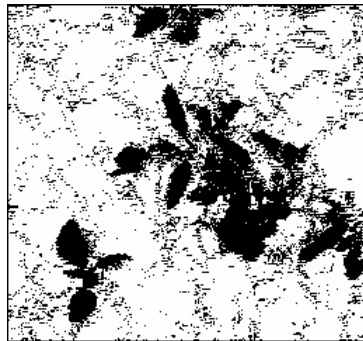
(a) Color image.



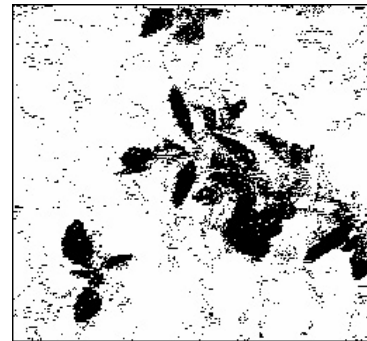
(b) Binary image from LutHC2.



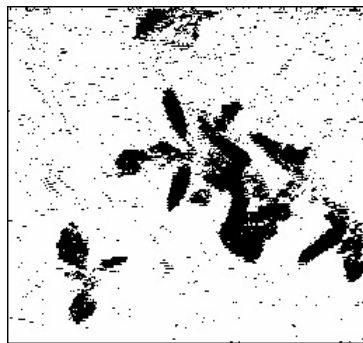
(c) Binary image from LutHC3.



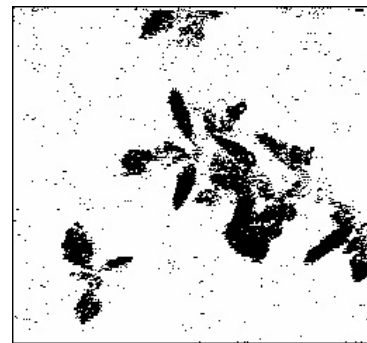
(d) Binary image from LutHSC2.



(e) Binary image from LutHSC3.



(f) Binary image from LutHSIC2.

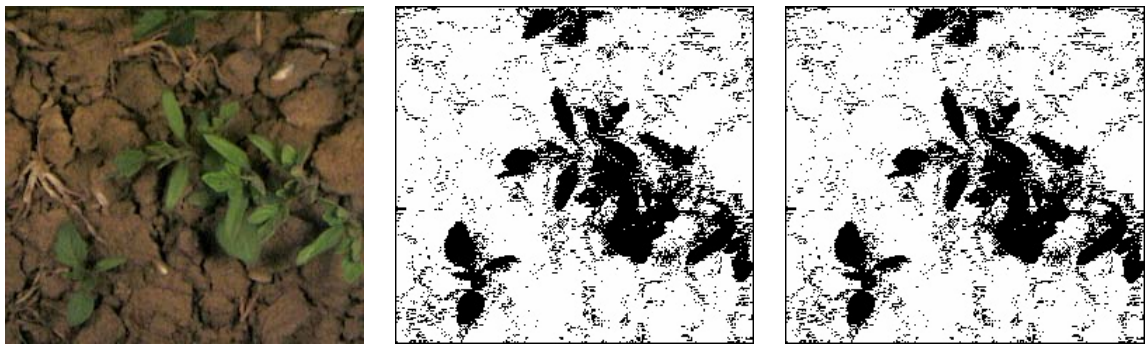


(g) Binary image from LutHSIC3.

Figure 4.5 Effect of H , S , and I component on LUT performance.

In *rgb* color space, the LUT made with 3 input classes (LutrgC3) was a little better than the one made with 2 input classes (LutrgC2). Example binary images in *rgb* color space are shown in Figure 4.6.

Tian (1995) reported that using *r*, *g*, and *b* components produced a better LUT than a LUT in *RGB* color space, since it eliminated the effect of *I* component to LUT. In this research the same result was confirmed in terms of plant error rate since the aperture of the camera was changed when pictures were acquired from field to field and both the object reflectance levels and illumination levels contributed to intensity variations in an image.



(a) Color image.

(b) Binary image from LutrgC2.

(c) Binary image from LutrgC3.

Figure 4.6 Effect of number of input class on LUT performance in *rgb* color space.

In summary, the LUTs made in *HSI* color space were generally better than those made in *rgb* color space and *RGB* color space and the LUTs built in *rgb* color space were better than those in *RGB* color space. In *HSI* color space, the *I* component didn't help the performance of LUTs in terms of plant error rate and this may explain the superior

performance of the LUTs made in *rgb* color space to those made in *RGB* color space. The LUTs built with only the *H* component (LutHC2 and LutHC3) produced the least plant error rates, correctly classifying 77.8% of color pixels.

Method II

The LUTs were also evaluated with single point noise (isolated pixels) removed from the binary images since most of the background-only binary image noise consisted of isolated pixels, which could be easily removed by the image processing system. For the second evaluation of LUTs, the isolated pixels were removed before counting the number of pixels in the segmented plant-only and background-only binary images. The results are listed in Table 4.4, however the results were similar to those from method I.

Table 4.4 LUTs listed by rank for each of the three error categories after isolated points were removed.

Rank	Plant error rate (%)	Background error rate (%)	Total error rate (%)
1	LutHC3 (11.5)	LutHSIC3 (1.0)	LutRGBC3 (20.3)
2	LutHC2 (11.9)	LutRGBC2 (2.6)	LutRGBC2 (20.8)
3	LutrgC3 (12.0)	LutRGBC3 (3.2)	LutRGC3 (21.2)
4	LutHSC2 (13.1)	LutRGC2 (3.3)	LutHSC2 (21.3)
5	LutrgC2 (13.6)	LutHSC3 (3.5)	LutHC3 (21.3)
6	LutRGBC3 (17.1)	LutHSIC2 (3.8)	LutHC2 (21.6)
7	LutRGC3 (17.2)	LutRGC3 (4.0)	LutRGC2 (21.8)
8	LutRGBC2 (18.3)	LutHSC2 (8.2)	LutrgC2 (21.9)
9	LutHSIC2 (18.4)	LutrgC2 (8.3)	LutrgC3 (22.1)
10	LutRGC2 (18.5)	LutHC2 (9.7)	LutHSIC2 (22.1)
11	LutHSC3 (22.6)	LutHC3 (9.9)	LutHSC3 (26.1)
12	LutHSIC3 (26.5)	LutrgC3 (10.1)	LutHSIC3 (27.5)

There were three changes in LUT rank (LutRGBC3 and LutRGC2, LutHSC2 and LutrgC2 in Background error rate, LutHSC2 and LutHC3 in Total error rate), but their difference was almost negligible.

4.2 Plant recognition performance

4.2.1 Cotyledon opening experiment results

There were some limitations in implementing the robotic weed control system in real-time. The first constraint was determining the morphological characteristics of tomato plants in real-time from a single top view. Even when the image quality was good and there was no wind, there was still another difficulty affecting plant recognition performance, diurnal changes in plant appearance. The orientation of plant leaves and changes in leaf position according to the environmental condition had a significant impact on the feasibility of accurately determining their morphological characteristics from a single top view.

A study was designed to evaluate varietal effects on cotyledon orientation using sixteen tomato varieties. The degree of openness of tomato cotyledons varied with environmental stresses such as lack of soil moisture and high temperature and cotyledons responded to environmental conditions by opening or closing. In a few cases there were tomato plants with three cotyledons, which also could make the recognition difficult.

Time of day effect

An analysis of variance and Tukey's means test was done with the 8AM-Day2, 4PM-Day2, and 8PM-Day2 data sets for varieties *A* through *P* to find out if different varieties had significantly different cotyledon angles at different times. These analyses showed that cotyledon angle is not significantly different ($\alpha = 0.05$) between 8AM and 4PM, however, 8PM is significantly different than the earlier times. The maximum mean change in cotyledon angle was 14.3° between 8AM and 8PM, which can be clearly seen

between Figure 3.40 (a) and (e) on page 126. The result from a second analysis with the 10AM-Day1 and 8AM-Day2 data sets showed that the effect of date, variety, and date & variety interaction had no effect on cotyledon opening in the morning. However, only the result with the 10AM-Day1 and 8AM-Day2 data sets showed no varietal or date effects and there was a significant difference later in the day (8PM).

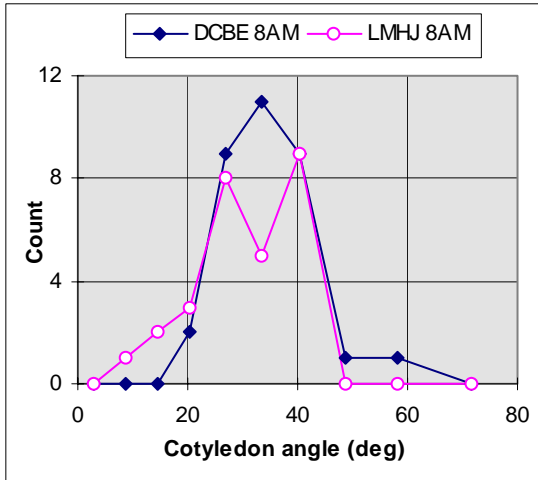
Varietal effect

The following figure shows the effect of variety on cotyledon opening using the average value for each variety on Day 2. The cotyledon angle of those varieties with the same underline are not significantly different. The maximum mean difference of openness was 13.7° between varieties *D* and *P*.

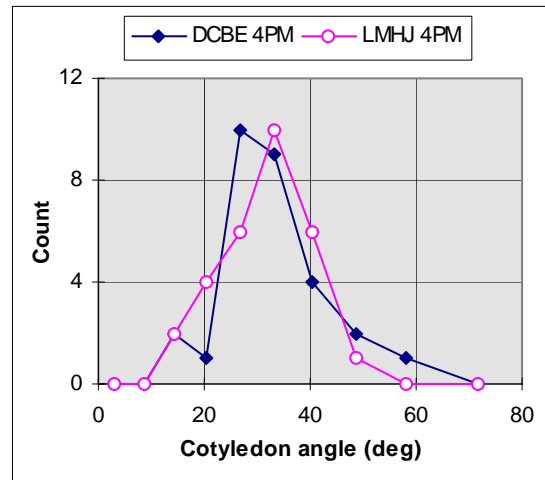
D C G F A B N I E L O J H K M P

Since the majority of the change in cotyledon angle occurred at night, an analysis of variance and Tukey's means test was done with Day2-8PM data set for all sixteen varieties. These results showed that the varieties were generally clustered into two groups: more open and less open. The more open group was composed of varieties *D*, *C*, *B*, and *E* and the less open group was composed of varieties *L*, *M*, *H*, and *J*. Three histograms were drawn for these two groups at different times of day to show the group differences (Fig. 4.7).

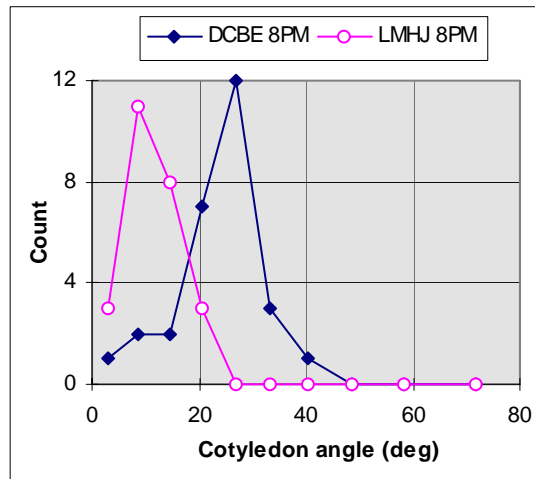
There is not much difference between the more open group and the less open group at 8 AM and 4 PM (Fig. 4.7 (a) and (b)). However, at 8 PM (Fig. 4.7 (c)), the difference between cotyledon angle is greater. The maximum difference observed was 17.3° between varieties *D* and *J*.



(a) 8:00 AM



(b) 4:00 PM



(c) 8:00 PM

Fig. 4.7 Histograms of cotyledon angles at different times of the day.

Daily effect

The results from the analysis with the 4PM-Day1 and 4PM-Day2 data sets showed that the average cotyledon opening is significantly different between these two days. The daily average mean ratio (as defined in Eq. (3.50) and Eq. (3.51)) on Day 2 (0.545) was a little higher than the one on Day 1 (0.489). However, when the mean ratio difference (i.e., 0.056) is converted into a cotyledon angle, it is only 3.2° for a cotyledon (or total angle of 6.4° for a seedling) from the side view. Thus, the difference is not meaningful from a practical point. This small difference might have been caused by temperature difference between two days. Day 1 was about 3.5°C warmer than Day 2 (Fig. 3.39 (a), page 125). It is speculated that the cotyledons close in an attempt to minimize moisture loss on hot days. The variety effect on the opening was not significant ($\alpha = 0.05$) between these two days.

This theory was confirmed in a commercial farm during field testing one morning. We started testing the prototype machine vision system, and the cotyledons were well open, but started closing in the afternoon as the temperature increased. However, at the same time, we observed that the cotyledons were more open in another part of the same field. These plants had just been watered by a single sprinkler set running on that portion of the field. On the other hand, the cotyledons near the prototype system were not yet watered and stayed closed. This behavior confirmed that cotyledon openness is closely related to environmental stresses such as lack of soil moisture and high temperature and that cotyledons respond to these environmental stresses by opening or closing.

The average value of the critical angle of recognition, γ , was found to be 27.5° . When the cotyledon was raised manually as shown in Figure 3.43, the normally long and thin elliptical appearance of the cotyledon started to look shorter and round from a top view. When it reached the critical point (critical angle of recognition), it began to look like a round object which led to being recognized as weed. In this test, the features ELG, CMP, and AREA were used to recognize tomato cotyledons. The range of values for these features were obtained from the data base under non-stressed conditions and if the values of these 3 features of an object were all within these ranges, then it was recognized as tomato cotyledon. Otherwise, the object was recognized as weed. Thus, if the total angle between two cotyledons is smaller than 55.0° , the machine vision system could not recognize tomato cotyledons as tomato leaves.

In summary, the results indicate that the cotyledon angle was significantly different at the same time of a day between two days which differed by 3.5°C in temperature, but the mean difference was not practically meaningful. The cotyledon angle of some varieties does not change from morning to dusk, but starts to close at dusk, becoming completely closed at night. The variety effect on cotyledon angle on Day 2 had a lot of variability in the data set. The maximum average daily difference of cotyledon angle among varieties was 13.7° . At night, some varieties close completely, but some don't. The maximum average difference of cotyledon opening between varieties was 17.3° at 8 PM. The critical angle of recognition was estimated as 27.5° for a tomato cotyledon.

Recognition difficulties caused by diurnal changes or plants lying down due to wind might be solved by incorporating a side view of the plants with the top view.

However, it would take longer and would be more complicated to process two views of the same scene.

4.2.2 Bayesian classifier with features

To identify tomato plants and weeds in real-time, an optimal subset of the 35 features described in sections 2.1, 3.3.3.7 and 3.4 must be selected. The following results show which features were selected using the three methods described in section 3.4.

Method I

The first method was to use canonical discriminant analysis and principal component analysis to choose the best features by removing any useless features and to take advantage of the large feature set, but at the same time to eliminate the problems with multi-collinearity. This method produced the best solution regardless of cost (time).

The first step tested the hypothesis that class means were equal for each input feature. Univariate analyses were conducted with each input feature from the training images in the good group and the results are given in Table 4.5.

Based on the univariate test results for the good training set, the following 8 features were removed since the F -test showed that their class means were not significantly different at the 0.01 significance level.

⇒ Features removed based on univariate analysis:

YCNRD, MAXC, LHW, SUMINV, M_{11} , PRINAXIS, MTMC, CTC

Table 4.5 Univariate test results showing that the class means for 8 features (bold characters) were not significantly different at 0.01 level in the good training set.

Univariate Test Statistics					
F Statistics, Num DF= 2 Den DF= 400					
Variable	Total STD	R-Squared	RSQ/ (1-RSQ)	F	Pr > F
AREA	374.4265	0.167834	0.2017	40.3365	0.0001
YCNRD	56.9603	0.000482	0.0005	0.0964	0.9081
PERIM	55.6139	0.195997	0.2438	48.7553	0.0001
MJX	17.2040	0.227899	0.2952	59.0334	0.0001
MNX	10.2414	0.197973	0.2468	49.3681	0.0001
ELG	0.1494	0.348996	0.5361	107.2178	0.0001
CMP	0.2547	0.171447	0.2069	41.3848	0.0001
MAXC	68.2608	0.010921	0.0110	2.2082	0.1112
MINC	158.1438	0.090821	0.0999	19.9786	0.0001
AVGC	43.2000	0.197440	0.2460	49.2025	0.0001
AVGABSC	34.2269	0.176877	0.2149	42.9769	0.0001
STDEVC	33.6119	0.023177	0.0237	4.7454	0.0092
NEG	4.1903	0.188728	0.2326	46.5264	0.0001
ATL	4.7415	0.177190	0.2153	43.0694	0.0001
PTB	0.0893	0.112024	0.1262	25.2314	0.0001
LHW	0.2416	0.010279	0.0104	2.0772	0.1266
LTP	0.0455	0.271551	0.3728	74.5559	0.0001
SUMINV	0.7020	0.004524	0.0045	0.9090	0.4038
ABSUMINV	0.8667	0.031720	0.0328	6.5519	0.0016
WID	13.5155	0.149675	0.1760	35.2041	0.0001
HET	16.2725	0.172868	0.2090	41.7993	0.0001
M20	66902	0.075565	0.0817	16.3484	0.0001
M02	101692	0.071527	0.0770	15.4074	0.0001
M11	39259	0.011942	0.0121	2.4173	0.0905
PRINAXIS	24.9279	0.003474	0.0035	0.6972	0.4986
ATP	0.1440	0.120266	0.1367	27.3414	0.0001
MTM	0.7647	0.361780	0.5669	113.3715	0.0001
OCCR	0.1139	0.199287	0.2489	49.7775	0.0001
PTP	0.0381	0.190322	0.2351	47.0118	0.0001
MTMC	39.9677	0.005944	0.0060	1.1958	0.3035
ATC	7.5361	0.155176	0.1837	36.7358	0.0001
PTC	1.1128	0.206263	0.2599	51.9726	0.0001
ETC	0.3095	0.288956	0.4064	81.2767	0.0001
CTC	0.4314	0.006631	0.0067	1.3351	0.2643
ECCN	19426728	0.050902	0.0536	10.7264	0.0001

Therefore, the canonical discriminant analysis was conducted with the remaining 27 features. These 27 features will also be used for any further analysis unless stated otherwise.

⇒ Remaining 27 features used for canonical discriminant analysis:

AREA, PERIM, MJX, MNX, ELG, CMP, MINC, AVGC, AVGABSC, STDEVC,
 NEG, ATL, PTB, LTP, ABSUMINV, WID, HET, M_{20} , M_{02} , ATP,
 MTM, OCCR, PTP, ATC, PTC, ETC, ECCN

The result of the canonical discriminant analysis is given in Tables 4.6 and 4.7. From canonical discriminant analysis for the good training set (Table 4.6), the squared distances between classes were 6.6 between class 1 and class 2, 5.6 between class 1 and class 4, and 3.0 between class 2 and class 4. This indicated that the separation of class 1 from the classes 2 and 4 would be easier than the separation of class 2 from class 4. For the bad training set (Table 4.7), the squared distances between classes showed more difficulties since the distances were closer than for the good training set (4.0 between class 1 and class 2, 2.4 between class 1 and class 4, and 1.9 between class 2 and class 4).

The multivariate test for differences between the classes was significant at the 0.0001 level using Wilk's Λ test and other tests (Tables 4.6 and 4.7). Thus, the mean for each input feature was significantly different between classes.

The R^2 , given by the squared canonical correlation in Table 4.10 for the good group, was 0.455 between CAN1 (canonical variable 1) and class variables and 0.363 between CAN2 and class variables and the cumulative proportion of eigenvalues explained was 1.0 (100%) with CAN1 and CAN2. For the bad training set, Table 4.7, the cumulative proportion of eigenvalues also explained 1.0 with CAN1 and CAN2.

Thus, using these two (CAN1 and CAN2) variables, the first 20 features were preliminary selected based on the magnitude of their raw canonical coefficients for each canonical variable from canonical discriminant analysis. These features are listed in Table 4.8 in the order of absolute magnitude of their raw canonical coefficients. These

raw canonical coefficients were produced from the input training data set, which were first standardized (mean = 0 and standard deviation = 1). Then, all features which were used in both CAN1 and CAN2 in the good group as well as in both CAN1 and CAN2 in the bad group (i.e., the features which appeared 4 times in the first 20 features) were selected as the best subset for discriminant analysis (also listed in alphabetical order in Table 4.8). Figure 4.8 shows a scatter plot of the good training samples using the variables of CAN1 and CAN2. In this figure, class 1 is well separated from classes 2 and 4, however there is some overlap between class 2 and class 4. Figure 4.9 shows the same scatter plot for the training set in the bad group, which indicates more overlap between class 1 and class 4 and between class 2 and class 4.

Table 4.6 Canonical discriminant analysis for good training set.

Canonical Discriminant Analysis Pairwise Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \text{COV}^{-1} (\bar{X}_i - \bar{X}_j)$$

Squared Distance to CLASS

From CLASS	1	2	4
1	0	6.59227	5.60150
2	6.59227	0	2.96972
4	5.60150	2.96972	0

Multivariate Statistics and F Approximations

Statistic	S=2 M=12 N=186			Num DF	Den DF	Pr > F
	Value	F				
Wilks' Lambda	0.34707175	9.6606		54	748	0.0001
Pillai's Trace	0.81821360	9.6160		54	750	0.0001
Hotelling-Lawley Trace	1.40502047	9.7050		54	746	0.0001
Roy's Greatest Root	0.83401097	11.5835		27	375	0.0001

NOTE: F Statistic for Roy's Greatest Root is an upper bound.
 NOTE: F Statistic for Wilks' Lambda is exact.

Canonical Discriminant Analysis

	Canonical Correlation	Adjusted Canonical Correlation	Approx Standard Error	Squared Canonical Correlation
1	0.674349	0.643287	0.027195	0.454747
2	0.602882	0.576231	0.031747	0.363467

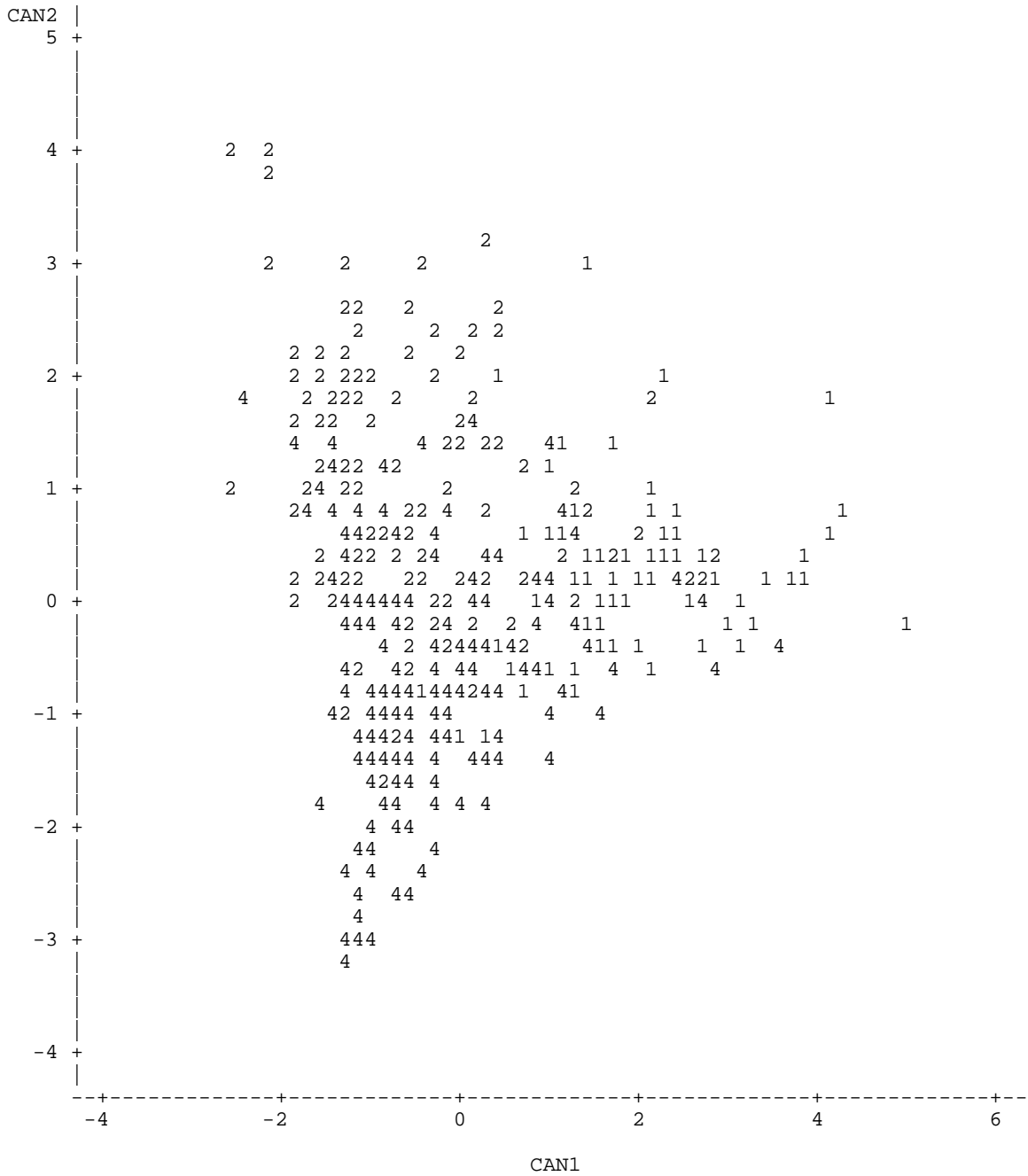
Eigenvalues of INV(E)*H
 = CanRsqr/(1-CanRsqr)

	Eigenvalue	Difference	Proportion	Cumulative
1	0.8340	0.2630	0.5936	0.5936
2	0.5710	.	0.4064	1.0000

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approx F	Num DF	Den DF	Pr > F
1	0.34707175	9.6606	54	748	0.0001
2	0.63653339	8.2357	26	375	0.0001

Plot of CAN2*CAN1. Symbol is value of CLASS.



NOTE: 116 obs hidden.

Figure 4.8 Plot of canonical variables 1 and 2 for good training set.

Table 4.7 Canonical discriminant analysis for bad training set.

Canonical Discriminant Analysis Pairwise Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \text{COV}^{-1} (\bar{X}_i - \bar{X}_j)$$

Squared Distance to CLASS

From CLASS	1	2	4
1	0	4.02095	2.48426
2	4.02095	0	1.90409
4	2.48426	1.90409	0

Multivariate Statistics and F Approximations

Statistic	S=2 M=12 N=155.5			Num DF	Den DF	Pr > F
	Value	F				
Wilks' Lambda	0.48791701	5.0036		54	626	0.0001
Pillai's Trace	0.59463400	4.9207		54	628	0.0001
Hotelling-Lawley Trace	0.88033819	5.0864		54	624	0.0001
Roy's Greatest Root	0.59687950	6.9415		27	314	0.0001

NOTE: F Statistic for Roy's Greatest Root is an upper bound.

NOTE: F Statistic for Wilks' Lambda is exact.

Canonical Discriminant Analysis

	Canonical Correlation	Adjusted Canonical Correlation	Approx Standard Error	Squared Canonical Correlation
1	0.611374	0.568668	0.033912	0.373779
2	0.469952	0.414112	0.042193	0.220855

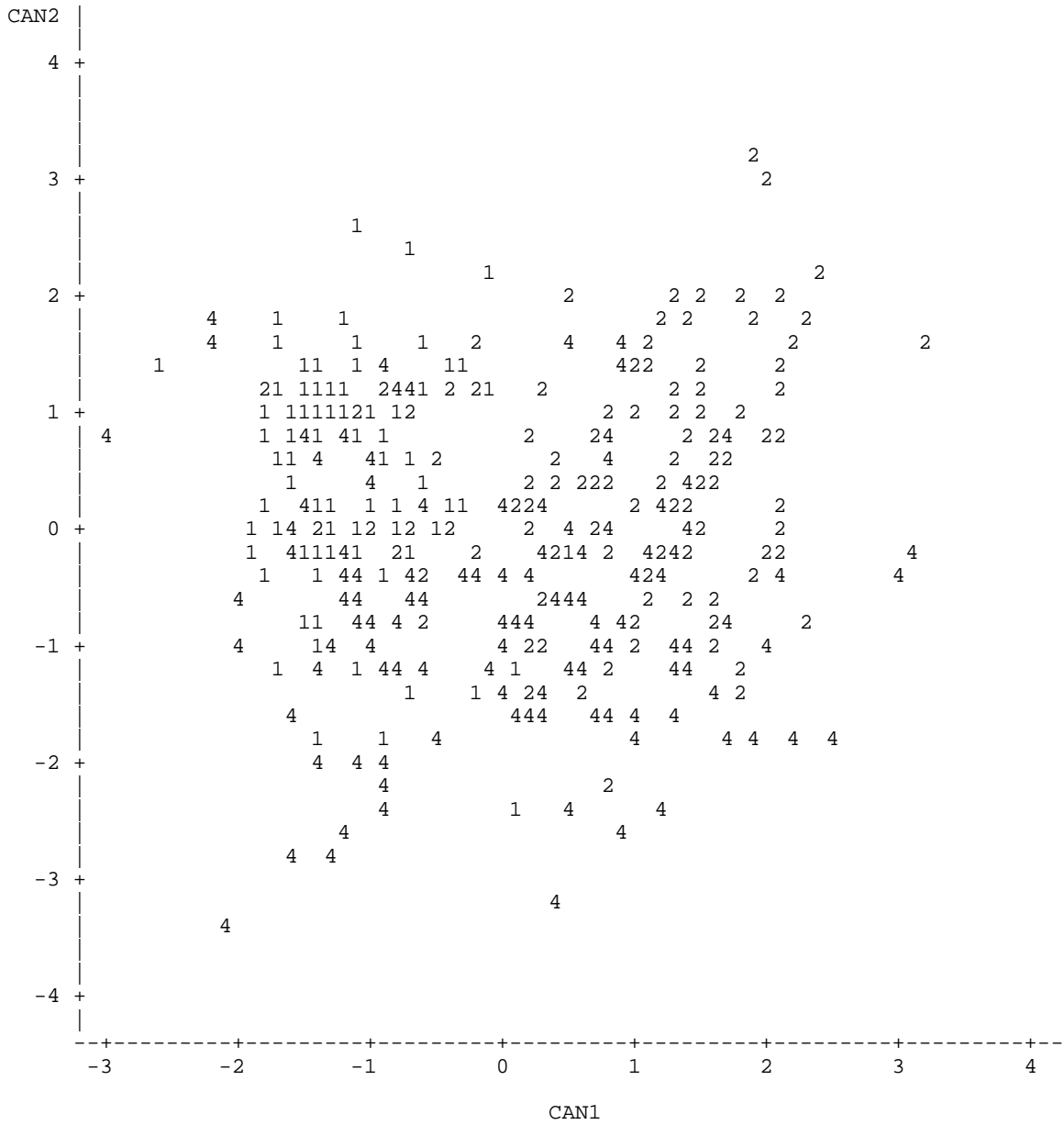
Eigenvalues of INV(E)*H
= CanRsqr/(1-CanRsqr)

	Eigenvalue	Difference	Proportion	Cumulative
1	0.5969	0.3134	0.6780	0.6780
2	0.2835	.	0.3220	1.0000

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approx F	Num DF	Den DF	Pr > F
1	0.48791701	5.0036	54	626	0.0001
2	0.77914468	3.4233	26	314	0.0001

Plot of CAN2*CAN1. Symbol is value of CLASS.



NOTE: 61 obs hidden.

Figure 4.9 Plot of canonical variables 1 and 2 for bad training set.

Table 4.8 Result of canonical discriminant analysis for good and bad training sets.

Good training set		Bad training set		Features Chosen
CAN1	CAN2	CAN1	CAN2	
LTP	LTP	PTP	LTP	AVGABSC
ELG	PTB	LTP	PTB	ABSUMINV
OCCR	ETC	PTB	PTP	ATL
CMP	ATP	ATP	OCCR	ATP
PTB	OCCR	ETC	ELG	CMP
ETC	CMP	OCCR	ATP	ELG
PTP	PTC	ELG	ETC	ETC
ATP	ELG	CMP	PTC	HET
MTM	PTP	MJX	MTM	LTP
PTC	ATL	ABSUMINV	CMP	MNX
MJX	MTM	PTC	ATL	MTM
NEG	NEG	MNX	ATC	NEG
ATL	MNX	ATC	ABSUMINV	OCCR
WID	ABSUMINV	MTM	WID	PERIM
HET	ATC	PERIM	HET	PTB
ABSUMINV	MJX	ATL	NEG	PTC
PERIM	AVGABSC	NEG	PERIM	PTP
MNX	PERIM	WID	MNX	
AVGABSC	HET	AVGABSC	AVGABSC	
AVGC	STDEVC	HET	AREA	

In order to find out how well these selected features worked in classifying tomato plants and weeds, discriminant analyses were conducted with canonical variable scores (CAN1 and CAN2) as variables for both good and bad data sets. New canonical variable scores were created for both good and bad validation sets based on the result of the canonical discriminant analysis on the training sets. With these canonical variable scores, the SAS discriminant analysis procedure, DISCRIM was used to classify tomato plants and weeds. The DISCRIM procedure developed a discriminant criterion to classify each observation into one of the groups using the training data set. The derived discriminant criterion was then applied to the validation set. This procedure evaluated the

performance of a discriminant criterion by estimating the error rates (probabilities of misclassification) in the classification of future observations. In this test, to optimize classification, a proportional *a priori* probability was used for all classes.

Table 4.9 shows the discriminant analysis for the good group and Table 4.10 shows the result for the bad group. With the newly created canonical variable scores from the selected 17 features in Table 4.8, the validation error rates for class 1, 2, and 4 were 21.3%, 37.4% and 16.3%. For the bad group, the validation error rates for class 1, 2, and 4 were 17.1%, 31.5% and 42.0%. The total error rates were 24.9% for the good group and 31.2% for the bad group.

The overall classification error was calculated using the total number of plant leaves which were classified from classes 1 & 2 into class 4 and from class 4 into classes 1 & 2. If a cotyledon in class 1 was assigned to class 2 or a true leaf in class 2 was assigned to class 1, they were not considered as errors, since they were both tomato plants. In the training set of the good group in Table 4.9, for example, 15 tomato cotyledons (class 1), 41 tomato true leaves (class 2), and 37 weeds (class 4) were incorrectly assigned to other classes. Thus, the total error in this example was

$$\frac{(15 + 41 + 37)}{403} \times 100 = 23.08 \%$$

Table 4.9 Discriminant analysis result with canonical variable scores calculated from the result of canonical discriminant analysis for good training and validation sets.

Good training set: CAN1 and CAN2

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	59 75.64	4 5.13	15 19.23	78 100.00
2	10 7.87	76 59.84	41 32.28	127 100.00
4	16 8.08	21 10.61	161 81.31	198 100.00
Total	85	101	217	403
Percent	21.09	25.06	53.85	100.00
Priors	0.1935	0.3151	0.4913	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.2436	0.4016	0.1869	0.2308

Good validation set: CAN1 and CAN2

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	114 76.00	4 2.67	32 21.33	150 100.00
2	7 3.91	105 58.66	67 37.43	179 100.00
4	7 3.47	26 12.87	169 83.66	202 100.00
Total	128	135	268	531
Percent	24.11	25.42	50.47	100.00
Priors	0.1935	0.3151	0.4913	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.2400	0.4134	0.1634	0.2486

Table 4.10 Discriminant analysis result with canonical variable scores calculated from the result of canonical discriminant analysis for bad training and validation sets.

Bad training set: CAN1 and CAN2

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	70 77.78	4 4.44	16 17.78	90 100.00
2	15 13.16	72 63.16	27 23.68	114 100.00
4	31 22.46	28 20.29	79 57.25	138 100.00
Total	116	104	122	342
Percent	33.92	30.41	35.67	100.00
Priors	0.2632	0.3333	0.4035	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.2222	0.3684	0.4275	0.2982

Bad validation set: CAN1 and CAN2

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	96 78.05	6 4.88	21 17.07	123 100.00
2	12 10.81	64 57.66	35 31.53	111 100.00
4	35 22.29	31 19.75	91 57.96	157 100.00
Total	143	101	147	391
Percent	36.57	25.83	37.60	100.00
Priors	0.2632	0.3333	0.4035	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.2195	0.4234	0.4204	0.3120

In principal component analysis (Table 4.11), the eigenvalues indicated that, for the good group, the first five principal components provided a good summary of the good group, accounting for 88.6% of the standardized variance and similarly the first five principal components explained 88.0% of the variance for the bad group.

Table 4.11 Principal component analysis for the good and bad training sets, showing that the first 5 principal components explained 88% of the standardized variance.

Principal Component Analysis				
Good training set: Eigenvalues of the Correlation Matrix				
	Eigenvalue	Difference	Proportion	Cumulative
PRIN1	14.4419	9.88619	0.534884	0.534884
PRIN2	4.5557	2.00638	0.168729	0.703613
PRIN3	2.5493	1.20150	0.094419	0.798032
PRIN4	1.3478	0.32973	0.049919	0.847950
PRIN5	1.0181	0.27111	0.037706	0.885657
PRIN6	0.7470	0.17854	0.027666	0.913322
PRIN7	0.5684	0.10136	0.021053	0.934375
PRIN8	0.4671	0.12971	0.017299	0.951674
PRIN9	0.3374	0.07415	0.012495	0.964169
PRIN10	0.2632	0.10799	0.009749	0.973918
PRIN11	0.1552	0.02474	0.005749	0.979667
PRIN12	0.1305	0.03862	0.004833	0.984499
PRIN13	0.0919	0.01352	0.003402	0.987901
PRIN14	0.0783	0.01197	0.002902	0.990803
PRIN15	0.0664	.	0.002458	0.993261
Bad training: Eigenvalues of the Correlation Matrix				
	Eigenvalue	Difference	Proportion	Cumulative
PRIN1	13.6573	8.77577	0.505826	0.505826
PRIN2	4.8815	2.14777	0.180797	0.686623
PRIN3	2.7337	1.23948	0.101250	0.787873
PRIN4	1.4943	0.49821	0.055343	0.843216
PRIN5	0.9961	0.29617	0.036891	0.880107
PRIN6	0.6999	0.10662	0.025922	0.906029
PRIN7	0.5933	0.12650	0.021973	0.928002
PRIN8	0.4668	0.06794	0.017288	0.945289
PRIN9	0.3988	0.15542	0.014772	0.960061
PRIN10	0.2434	0.02657	0.009015	0.969076
PRIN11	0.2168	0.05000	0.008031	0.977108
PRIN12	0.1668	0.06648	0.006179	0.983287
PRIN13	0.1004	0.01254	0.003717	0.987004
PRIN14	0.0878	0.01543	0.003253	0.990257
PRIN15	0.0724	.	0.002681	0.992937

It is a common practice to limit the number of principal components used to those which explain about 70 - 80% of the standardized variance (Azari, 1998). Thus, in order to choose more important features, the top 20 features from the first five components (PRIN1 - PRIN5) were preliminarily selected based on their eigenvalues. The features selected are listed in Table 4.12 for the good group in order of their magnitude of their eigenvalues.

Table 4.12 Selected features from principal component analysis for good training set based on their magnitude of eigenvalues.

PRIN1	PRIN2	PRIN3	PRIN4	PRIN5
PERIM	ELG	STDEVC	ECCN	ABSUMINV
MJX	MTM	MINC	AVGABSC	M20
AREA	ETC	CMP	M ₀₂	WID
NEG	LTP	ATC	AVGC	HET
MNX	PTP	ATL	ABSUMINV	ECCN
PTC	AVGABSC	ATP	WID	MINC
HET	PTB	OCCR	ATP	STDEVC
AVGC	ATP	PTB	ATL	AVGABSC
WID	MNX	PTC	NEG	OCCR
ATC	OCCR	AREA	PTB	PTB
ATL	CMP	M ₀₂	HET	LTP
M ₀₂	MJX	ELG	ELG	AREA
CMP	PTC	MTM	MTM	M ₀₂
OCCR	ATL	AVGABSC	MNX	ATC
M20	AVGC	PTP	STDEVC	PTP
PTB	ECCN	M20	OCCR	CMP
PTP	MINC	MNX	M20	MTM
MINC	ABSUMINV	HET	PERIM	NEG
ATP	HET	ECCN	AREA	ELG
AVGABSC	ATC	AVGC	LTP	ATL

Then, the features which appeared at least three times in Table 4.12 were selected as the most important subset (listed in alphabetical order):

ABSUMINV, AREA, ATC, ATL, ATP, AVGABSC, AVGC, CMP, ECCN, ELG,
HET, LTP, M₀₂, M₂₀, MINC, MNX, MTM, NEG, OCCR, PTB,
PTC, PTP, STDEVC, and WID.

A discriminant analysis was conducted with the first five principal components (PRIN1-PRIN5) added to the good validation data set based on principal component analysis using the good training data set, Table 4.13. In this test, proportional *a priori* probability was also used to optimize classification. The error rates were 6.0% for class 1, 41% for class 2, and 23.8% for class 4. Comparing the result with the one of the discriminant analysis with CAN1 and CAN2 for the good validation set (Table 4.9), the principal component scores showed superiority for classifying class 1 (92.7% vs. 76.0%) and the canonical variable scores showed less error for class 2 (37.4% vs. 41.3%) and class 4 (16.3% vs. 23.8%) than the principal component scores.

Table 4.13 Discriminant analysis result with the principal component scores obtained from the principal component analysis for the good group.

Good training set: PRIN1 - PRIN5

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	64 82.05	5 6.41	9 11.54	78 100.00
2	13 10.24	67 52.76	47 37.01	127 100.00
4	23 11.62	35 17.68	140 70.71	198 100.00
Total	100	107	196	403
Percent	24.81	26.55	48.64	100.00
Priors	0.1935	0.3151	0.4913	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.1795	0.4724	0.2929	0.2829

Good validation set: PRIN1 - PRIN5

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	139 92.67	2 1.33	9 6.00	150 100.00
2	23 12.85	82 45.81	74 41.34	179 100.00
4	26 12.87	22 10.89	154 76.24	202 100.00
Total	188	106	237	531
Percent	35.40	19.96	44.63	100.00
Priors	0.1935	0.3151	0.4913	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.0733	0.5419	0.2376	0.2467

Table 4.14 Selected features from principal component analysis for bad training set based on their magnitude of eigenvalues.

PRIN1	PRIN2	PRIN3	PRIN4	PRIN5
PERIM	ETC	STDEVC	ECCN	ABSUMINV
MJX	ELG	CMP	AVGABSC	M20
AREA	MTM	ATP	M ₀₂	WID
NEG	LTP	ATC	AVGC	M ₀₂
MNX	PTP	MINC	M20	HET
PTC	PTB	ATL	ATL	OCCR
HET	AVGABSC	M ₀₂	ABSUMINV	MINC
AVGC	ATP	MTM	STDEVC	ECCN
WID	MNX	AREA	NEG	ATP
ATC	MJX	OCCR	MNX	STDEVC
ATL	OCCR	ELG	PTB	ATL
CMP	PTC	PTC	WID	MTM
OCCR	MINC	PTB	ELG	AREA
MINC	ATL	ECCN	MTM	NEG
PTB	STDEVC	M20	ETC	AVGC
PTP	ATC	PTP	ATP	ELG
M ₀₂	ECCN	MNX	PTP	CMP
M20	CMP	AVGC	PERIM	LTP
AVGABSC	HET	HET	HET	ATC
ATP	M ₀₂	NEG	ATC	PTP

The same procedure was used for bad training set (Table 4.14) and the following features were selected:

AREA, ATC, ATL, ATP, AVGABSC, AVGC, CMP, ECCN, ELG, HET,

M₀₂, M₂₀, MINC, MNX, MTM, NEG, OCCR, PTB, PTC, PTP,

STDEVC, and WID

A discriminant analysis was also conducted with the principal component scores obtained with the bad group, Table 4.15. The error rates were 2.4% for class 1, 21.6% for class 2 and 75.2% for class 4. Comparing the result with the one from the discriminant analysis with the canonical variable scores, the principal component scores showed only 2.4% error for class 1 whereas the canonical variable scores showed 17.1% error. The

principal component scores also showed less error for class 2 (21.6% vs. 31.5%) than the canonical variable scores. However, the canonical variable scores produced less error for class 4 than the principal component scores (42.0% vs. 75.2%) for the bad validation set.

Table 4.15 Discriminant analysis result with the principal component scores obtained from the principal component analysis for the bad group.

Bad training set: PRIN1 - PRIN5

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	84 93.33	1 1.11	5 5.56	90 100.00
2	22 19.30	69 60.53	23 20.18	114 100.00
4	61 44.20	33 23.91	44 31.88	138 100.00
Total	167	103	72	342
Percent	48.83	30.12	21.05	100.00
Priors	0.2632	0.3333	0.4035	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.0667	0.3947	0.6812	0.3567

Bad validation set: PRIN1 - PRIN5

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	115 93.50	5 4.07	3 2.44	123 100.00
2	22 19.82	65 58.56	24 21.62	111 100.00
4	81 51.59	37 23.57	39 24.84	157 100.00
Total	218	107	66	391
Percent	55.75	27.37	16.88	100.00
Priors	0.2632	0.3333	0.4035	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.0650	0.4144	0.7516	0.3018

Therefore, from the canonical discriminant analysis (Tables 4.8) and the principal component analyses (Table 4.12 and 4.14), the following 13 features were selected because they were listed in both the canonical discriminant analysis and the principal component analysis feature subsets.

⇒ **Final best subset of features for Method I:**

**AVGABSC, ATL, ATP, PTB, CMP, ELG, HET, MNX, MTM, NEG,
OCCR, PTC, and PTP.**

With these 13 input features, a discriminant analysis was conducted to classify tomato plants and weeds. In this analysis, a serial-two-step Bayes classifier was used in order to get higher performance for class 2. Suppose Classifier14 is the Bayes classifier trained on the training data set containing only classes 1 and 4. The basic idea of this two step Bayes classifier was to first classify all objects as either class 1 or class 4 using the Classifier14. Then, the class 4 objects were re-classified as either class 2 or class 4 using a second Bayes classifier trained on the training data set containing the correctly classified class 4 objects using the Classifier14 and the class 2 objects classified as class 4 using the Classifier14 in the training data set.

The results are shown in Table 4.16 for the good group and in Table 4.17 for the bad group. As illustrated in Table 4.16, the classifier with 13 features worked better in identifying class 1 (tomato cotyledon) than in identifying the other two classes (class 2 and 4) for both good and bad validation sets. The classification rates were over 92% for tomato cotyledons (class 1) for both groups, and a little over 40% for tomato true leaves for both groups. The identification rate for the weed class was 77.7% for the good group

and 45.2% for the bad group. This indicated that the weed class in bad group was not easy to identify. The overall error was 25.0% for the good group and 35.6% for the bad group.

Table 4.16 Result of discriminant analysis with the selected features from Method I for the good group.

Good group, Training set:

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	71 91.03	0 0.00	7 8.97	78 100.00
2	22 17.32	61 48.03	44 34.65	127 100.00
4	29 15.15	16 8.08	153 77.27	198 100.00
Total	122	77	204	403
Percent	30.27	19.11	50.62	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0897	0.5197	0.2273	0.2382

Good group, Validation set:

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	147 98.00	0 0.00	3 2.00	150 100.00
2	24 13.41	70 39.11	85 47.49	179 100.00
4	26 12.87	20 9.90	156 77.23	202 100.00
Total	197	90	244	531
Percent	37.10	16.95	45.95	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0200	0.6089	0.2277	0.2524

Table 4.17 Result of discriminant analysis with the selected features from Method I for the bad group.

Bad group, Training set:

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	83 92.22	0 0.00	7 7.78	90 100.00
2	19 16.67	68 59.65	27 23.68	114 100.00
4	43 31.16	22 15.94	73 52.90	138 100.00
Total	145	90	107	342
Percent	42.40	26.32	31.29	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0778	0.4035	0.4710	0.2895

Bad group, Validation set:

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	114 92.68	0 0.00	9 7.32	123 100.00
2	15 13.51	52 46.85	44 39.64	111 100.00
4	64 40.76	22 14.01	71 45.22	157 100.00
Total	193	74	124	391
Percent	49.36	18.93	31.71	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.0732	0.5315	0.5478	0.3555

Method II

The second method was to try to reduce the number of features to the most important subset for real-time use by correlation analysis and linear regression model selection based on the R^2 criteria. In this method, the number of features were limited to 4 for real-time implementation. The multi-colinearity among features was solved using correlation analysis.

The average feature execution times for 10 tomato cotyledons are listed in Table 4.18. Most of the curvature related features took less time to calculate than other non-curvature related features, however the longer the perimeter (i.e., bigger object), the longer the curvature would take to calculate. LTP was the most time-consuming feature.

The basic idea was first to find the minimum feature subset that gives good classification of tomato cotyledons vs. weeds in the good image set, then to find additional features for tomato true leaves vs. weeds so that the overall classification could be improved.

Table 4.18 Average time to calculate each feature for an image with 10 tomato cotyledons.

Feature	Time (ms)
ABSUMINV	2.32
AVGABSC	2.32
AVGC	2.32
MAXC	2.32
NEG	2.32
MINC	2.32
STDEVC	2.32
SUMINV	2.32
MTMC	2.37
AREA	3.22
YCNTRD	3.30
ATC	5.59
HET & WID	6.49
LHW	6.53
PERIM	7.65
M ₂₀	7.70
M ₀₂	7.70
M ₁₁	8.19
ATP	9.76
PTC	10.02
CMP	10.92
MNX	12.76
CTC	13.29
PTB	14.19
PTP	14.20
MJX	14.56
ELG	14.91
MTM	14.91
PRINAXIS	15.95
ETC	17.28
ATL	17.83
OCCR	18.13
ECCN	19.16
LTP	22.26

The results from the linear regression analysis conducted with the features as independent variables and class numbers (1 = tomato cotyledons and 0 = weeds) as dependent variables are given in Table 4.19. This table shows only partial results (the 5 best combinations at each number of independent variables) since there were too many feature combinations with the same R^2 value to list.

Table 4.19 Selected features for classes 1 and 4 by R^2 criteria from linear regression using both good and bad training sets.

R^2	Good training set	Time (ms)	R^2	Bad training set	Time (ms)
0.49	MNX, LTP, PTC, ETC	62.32	0.29	ELG, CMP, AVGC, PTC	12.34
0.49	CMP, PTB, LTP, OCCR	65.50	0.29	ELG, CMP, MINC, PTC	38.17
0.48	PTB, LTP, WID, M20	50.64	0.29	ELG, CMP, MINC, ECCN	47.31
0.48	CMP, LTP, WID, OCCR	57.80	0.29	ELG, CMP, STDEVC, ECCN	47.31
0.48	ELG, PTB, LTP, WID	57.85	0.29	MNX, ELG, CMP, PTC	48.61
0.47	AVGC, PTB, LTP	38.77	0.28	ELG, CMP, STDEVC	28.15
0.47	ELG, AVGC, LTP	39.49	0.28	STDEVC, LTP, OCCR	42.71
0.47	PTB, LTP, WID	42.95	0.28	ELG, CMP, ECCN	44.99
0.47	ELG, LTP, WID	43.66	0.27	MINC, LTP, PTC	34.65
0.47	CMP, LTP, OCCR	51.31	0.27	ELG, CMP, PTC	35.85
0.46	CMP, LTP	33.18	0.26	ELG, CMP	25.83
0.46	ELG, LTP	37.17	0.26	LTP, PTC	32.28
0.45	ELG	14.91	0.26	LTP, ETC	39.54
0.45	ELG, NEG	17.23	0.26	LTP, OCCR	40.39
0.45	ELG, HET	21.40	0.26	LTP, ECCN	41.42
0.45	ELG, ATP	24.64	0.24	LTP	22.86
0.37	LTP	22.26	0.16	MNX	12.76
0.35	ETC	17.28	0.15	ELG	14.91
0.09	MNX	12.76	0.15	OCCR	18.13
0.08	OCCR	18.13	0.10	PTB	14.19

With these combinations of features, discriminant analyses were conducted and the following set of features in Table 4.20 produced good discriminatory power with over

90% recognition of tomato cotyledons and over 80% classification of weeds in the good validation set.

The same procedure was done with class 2 (tomato true leaves) vs. class 4 (weeds). Table 4.21 shows the result of linear regression by R^2 criteria and Table 4.22 shows the feature combinations in the good group which had recognition rates greater than 60% for tomato true leaves and weeds.

Table 4.20 Feature subsets with classification rates over 90% for tomato cotyledons and over 80% for weeds at the same time in the good validation set.

Feature	Time (ms)	Feature	Time (ms)
ELG	14.91	ELG, LTP, WID	43.66
ELG, ABSUMINV	17.23	CMP, LTP, OCCR	51.31
ELG, AVGABSC	17.23	ELG, PTB, LTP	51.36
ELG, AVGC	17.23	MNX, ELG, HET, ETC	51.44
ELG, MINC	17.28	CMP, MINC, LTP, OCCR	53.63
ELG, HET	21.40	CMP, AVGC, LTP, OCCR	53.63
ELG, WID	21.40	CMP, AVGABSC, LTP, OCCR	53.63
LTP	22.86	ELG, AVGC, PTB, LTP	53.68
ELG, NEG, WID	23.72	PTB, LTP, ETC	53.73
ELG, ATP	24.64	LTP, WID, PTC, ETC	56.05
ELG, PTC	24.93	AVGC, PTB, LTP, OCCR	56.90
ELG, CMP	25.83	CMP, LTP, WID, OCCR	57.80
ELG, PTB	29.10	ELG, PTB, LTP, WID	57.86
ELG, OCCR	33.01	PTB, LTP, WID, ETC	60.22
CMP, LTP	33.18	PTB, LTP, HET, ETC	60.22
ELG, NEG, WID, ATP	33.48	CMP, LTP, ATP, OCCR	61.07
ELG, ECCN	34.07	PTB, LTP, WID, OCCR	61.07
ELG, LTP	37.17	MNX, LTP, PTC, ETC	62.32
ELG, HET, ETC	38.68	MNX, CMP, LTP, OCCR	64.07
AVGC, PTB, LTP	38.77	CMP, PTB, LTP, OCCR	65.50
ELG, AVGC, LTP	39.49	ELG, CMP, LTP, OCCR	66.22
PTB, LTP, WID	42.94	CMP, LTP, OCCR, ETC	68.59

Table 4.21 Selected features for classes 2 and 4 by R^2 criteria from linear regression using both good and bad training sets.

R^2	Good training set	Time (ms)	R^2	Bad training set	Time (ms)
0.34	NEG, M ₀₂ , PTC, ETC	37.32	0.19	AREA, MINC, AVGABSC, M20	15.56
0.33	AVGABSC, NEG, M ₀₂ , ETC	29.62	0.19	AREA, AVGABSC, WID, M ₂₀	19.13
0.33	NEG, M ₀₂ , OCCR, ETC	45.43	0.19	AREA, AVGC, WID, M20	19.73
0.33	ELG, M ₀₂ , OCCR, PTC	50.76	0.19	AREA, AVGABSC, M20, M ₀₂	20.94
0.33	M ₀₂ , OCCR, PTC, ETC	53.13	0.19	AREA, AVGABSC, M20, ATP	23.00
0.32	OCCR, ETC	35.41	0.18	AREA, AVGABSC, M ₂₀	13.24
0.32	AVGC, OCCR, ETC	37.73	0.18	AREA, AVGC, M ₂₀	13.24
0.32	AVGABSC, OCCR, ETC	37.73	0.18	AREA, WID, M ₂₀	17.41
0.32	M ₀₂ , OCCR, ETC	43.11	0.17	AREA, AVGABSC, ECCN	24.70
0.32	PTB, OCCR, ETC	49.60	0.17	AVGABSC, WID, M20	16.51
0.32	LTP, OCCR, ETC	57.67	0.16	MINC, AVGABSC	4.64
0.30	AVGC, ETC	19.60	0.16	AVGC, STDEV	4.64
0.29	NEG, ETC	19.60	0.16	AVGC, AVGABSC	4.64
0.29	CMP, ETC	28.20	0.16	AREA, AVGABSC	5.54
0.29	PTB, ETC	31.47	0.16	MNX, AVGABSC	15.08
0.22	PTC	10.02	0.15	AVGC	2.32
0.22	ETC	17.28	0.13	AVGABSC	2.32
0.21	AVGC	2.32	0.11	AREA	3.22
0.20	AVGABSC	2.32	0.11	WID	6.49
0.19	CMP	10.92	0.10	HET	6.49

Table 4.22 Feature subsets with classification rates over 60% for tomato true leaves and over 60% for weeds simultaneously for the good validation set.

Feature	Time (ms)	Feature	Time (ms)
AVGABSC, NEG	4.64	ABSUMINV, OCCR, ETC	37.73
CMP	10.92	AREA, OCCR, ETC	38.63
ELG, AVGC	17.23	LTP, ETC	39.54
AVGC, ETC	19.60	MNX, NEG, M ₀₂ , ETC	40.06
NEG, ETC	19.60	OCCR, ATC, ETC	41.00
MINC, ETC	19.60	WID, OCCR, ETC	41.90
STDEVC, ETC	19.60	HET, OCCR, ETC	41.90
CMP, PTC	20.94	ELG, NEG, M ₀₂ , ETC	42.21
WID, ETC	23.77	M ₀₂ , OCCR, ETC	43.11
ATP, ETC	27.04	M ₂₀ , OCCR, ETC	43.11
NEG, M ₀₂ , ETC	27.30	ATP, OCCR, ETC	45.17
MNX, ELG	27.67	NEG, M ₀₂ , OCCR, ETC	45.43
CMP, ETC	28.20	OCCR, PTC, ETC	45.43
AVGABSC, NEG, M ₀₂ , ETC	29.62	AREA, M ₀₂ , OCCR, ETC	46.33
AVGC, NEG, M ₀₂ , ETC	29.62	CMP, OCCR, ETC	46.33
MNX, ETC	30.04	MNX, OCCR, ETC	48.17
PTB, ETC	31.47	WID, M ₂₀ , OCCR, ETC	49.60
NEG, HET, M ₀₂ , ETC	33.79	HET, M ₀₂ , OCCR, ETC	49.60
ELG, NEG, M ₀₂ , PTC	34.95	PTB, OCCR, ETC	49.61
OCCR, ETC	35.41	M ₀₂ , OCCR, ATC, ETC	48.70
MINC, OCCR, ETC	37.73	ELG, OCCR, ETC	50.32
STDEVC, OCCR, ETC	37.73	ELG, M ₀₂ , OCCR, PTC	50.76
AVGC, OCCR, ETC	37.73	M ₀₂ , OCCR, PTC, ETC	53.13
AVGABSC, OCCR, ETC	37.73	PTB, M ₀₂ , OCCR, ETC	57.30
NEG, OCCR, ETC	37.73	LTP, OCCR, ETC	57.67

Therefore, the feature combinations which were common in both Table 4.20 and Table 4.22 were selected for further discriminant analysis. These feature subsets are listed in Table 4.23 and the bold faced features are the common features .

Table 4.23 Common feature subsets from both linear regression results of classes 1 & 4 (Table 4.19) and classes 2 & 4 (Table 4.21) (bold faced features are common features in both Table 4.19 and Table 4.21).

1 common feature	2 common features	2 common features
ELG	OCCR, ETC	ETC, OCCR, ABSUMINV
ELG, MNX	ETC, MNX	ETC, OCCR, STDEVC
ELG, M₀₂, OCCR, PTC	ELG, AVGC	ETC, OCCR, AVGC
ELG, NEG, M₀₂, ETC	ELG, OCCR	ETC, OCCR, AVGABSC
ELG, NEG, M₀₂, PTC	ELG, ETC, HET	ETC, OCCR, M₀₂
ELG, PTB, LTP	ETC, HET, PTB, LTP	ETC, OCCR, WID
LTP	ELG, NEG, WID	ETC, OCCR, PTC
LTP, ETC	ELG, NEG, WID, ATP	ETC, OCCR, ELG
LTP, ETC, OCCR	MNX, CMP, LTP, OCCR	ETC, OCCR, MNX
CMP	ETC, PTB	ETC, OCCR, MINC
CMP, LTP	ETC, PTB, LTP	ETC, OCCR, M20
CMP, ETC	CMP, LTP, ATP, OCCR	ETC, OCCR, CMP
CMP, PTC	LTP, WID, PTC, ETC	ETC, OCCR, HET
CMP, PTB, LTP, OCCR	CMP, OCCR, LTP	AVGC, PTB, LTP, OCCR
CMP, AVGC, LTP, OCCR	ETC, PTC, MNX, LTP	CMP, MINC, LTP, OCCR
ELG, CMP, LTP, OCCR	ETC, MNX, NEG, M₀₂	
	ETC, MNX, ELG, HET	
	ETC, OCCR, NEG	
3 common features	ETC, OCCR, AREA	
CMP, OCCR, ETC	ETC, OCCR, ATC	
CMP, OCCR, ETC, LTP	ETC, OCCR, ATP	
CMP, WID, OCCR	CMP, AVGABSC, LTP,	
CMP, LTP, WID, OCCR	OCCR	

With the selected feature combinations in Table 4.23, discriminant analyses were conducted using the SAS DISCRIM procedure with three classes (tomato cotyledon, tomato true leaf, and weed) for both good and bad groups. The *a priori* probabilities used were 0.1 for class 1, 0.1 for class 2, and 0.8 for class 4. The following list of feature combinations in Table 4.24 showed classification rates of over 70% for class 1 and over 80% for class 4 simultaneously.

Table 4.24 Selected feature sets which produced over 70% recognition for class 1 and over 80% for class 4 simultaneously in the good group from discriminant analysis.

Selected features from discriminant analysis	Calculation Time (ms)	Total error rate (%)	
		Good	Bad
ELG, NEG, M ₀₂ , PTC	34.95	30.32	46.04
ELG, M ₀₂ , OCCR, PTC	50.76	33.90	40.92
ETC, MNX, NEG, M ₀₂	52.52	29.19	45.27
ELG, NEG, M ₀₂ , ETC	54.67	29.94	42.71
MNX, CMP, LTP, OCCR	64.07	33.52	40.92
ELG, CMP, LTP, OCCR	66.22	30.32	37.60
ETC, PTC, MNX, LTP	72.98	28.25	41.18

Based on the discriminant analysis results, the feature subsets: [ELG, NEG, M₀₂, ETC], [ELG, NEG, M₀₂, PTC], [ELG, NEG, M₀₂, PTC], [ETC, PTC, MNX, LTP], and [ETC, MNX, NEG, M₀₂] showed poor classification for class 4 (weeds) in bad group, ranging from 50% to 66%. Between the two remaining feature subsets, [ELG, CMP, LTP, OCCR] and [MNX, CMP, LTP, OCCR], the feature subset [MNX, CMP, LTP, OCCR] showed better classification (4% more for the good group and 1% more for the bad group) for class 4 (weeds) in both good and bad groups than [ELG, CMP, LTP, OCCR]. Therefore, the following feature subset was chosen as the best subset from the Method II. The average calculation time (10 cotyledons) for this feature set was 64.07 ms.

⇒ **Best subset from Method II: MNX, CMP, LTP, and OCCR.**

The result of discriminant analysis with this feature subset is shown in Table 4.25 for the good group and Table 4.26 for the bad group. In this analysis, a serial-two-step Bayes classifier was also used in order to get higher performance for class 2.

For the good group, the error rates for classes 1, 2, and 4 were 12.67%, 61.45%, and 5.45%. For true leaves, the error rate was only when they were classified as weeds since classes 1 and 2 were both tomato plants. For the bad group, the error rates for classes 1, 2, and 4 were 12.20%, 91.00%, and 28.03%. The error rate for tomato true leaves in the bad group was extremely high, which resulted from their shape being very similar to weeds. The total error rates were 26.37% for the good group and 40.92% for the bad group.

Table 4.25 Discriminant analysis result with the best subset [MNX, CMP, LTP, OCCR] from the method II for the good group.

Good training set: MNX, CMP, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	57 73.08	0 0.00	21 26.92	78 100.00
2	16 12.60	36 28.35	75 59.06	127 100.00
4	16 8.08	8 4.04	174 87.88	198 100.00
Total	89	44	270	403
Percent	22.08	10.92	67.00	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.2692	0.7165	0.1212	0.2978

Good Validation set: MNX, CMP, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	131 87.33	0 0.00	19 12.67	150 100.00
2	22 12.29	47 26.26	110 61.45	179 100.00
4	7 3.47	4 1.98	191 94.55	202 100.00
Total	160	51	320	531
Percent	30.13	9.60	60.26	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.1267	0.7374	0.0545	0.2637

Table 4.26 Discriminant analysis result with the best subset [MNX, CMP, LTP, OCCR] from the method II for the bad group.

Bad training set: MNX, CMP, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	73 81.11	0 0.00	17 18.89	90 100.00
2	16 14.04	4 3.51	94 82.46	114 100.00
4	37 26.81	0 0.00	101 73.19	138 100.00
Total	126	4	212	342
Percent	36.84	1.17	61.99	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.1889	0.9649	0.3801	0.4327

Bad Validation set: MNX, CMP, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	108 87.80	0 0.00	15 12.20	123 100.00
2	10 9.01	1 0.90	101 91.00	111 100.00
4	44 28.03	0 0.00	113 71.97	157 100.00
Total	162	1	229	391
Percent	41.43	0.26	58.57	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.1220	0.9974	0.2803	0.4092

Method III

This method was used to find the best feature subsets for real-time field use by trial and error. Due to time constraints, it was not possible to try all possible combinations of 27 features. Thus, the features in Table 4.27 were selected to conduct further discriminant analysis based on the results from principal component analysis, canonical discriminant analysis and the stepwise discriminant analysis.

Table 4.27 Feature subsets used in discriminant analysis in Method III.

Features used	Calculation time (ms)
ELG, CMP, ATP	35.59
MNX, M ₀₂ , ETC	37.74
ELG, PTC, OCCR	43.06
AREA, LTP, OCCR	43.61
MNX, M ₀₂ , ETC, PTC	47.76
MNX, LTP, OCCR	53.15
ELG, LTP, OCCR	55.30
MNX, LTP, OCCR, AVGABSC	55.47
AREA, LTP, OCCR, PTB	57.80
LTP, ETC, MNX, ATC	57.89
ELG, PTC, OCCR, ETC	60.34
CMP, LTP, OCCR, ETC	68.59
ELG, LTP, OCCR, ETC	72.58

Based on discriminant analysis results with the above feature sets, the following sets were selected since they produced classification rates of over 80% for class 1 and over 90% for class 4 for the good group.

[AREA, LTP, OCCR], [AREA, LTP, OCCR, PTB], [MNX, LTP, OCCR],
[MNX, LTP, OCCR, AVGABSC], and [LTP, ETC, MNX, ATC]

The set [MNX, LTP, OCCR] produced poor classification rate (54%) of class 1 in the bad group and the set [LTP, ETC, MNX, ATC] had a 67% classification rate of class 4 in the

bad group. Thus these two sets were eliminated. The remaining three subsets [AREA, LTP, OCCR], [AREA, LTP, OCCR, PTB] and [MNX, LTP, OCCR] produced very low classification rates for true leaves, ranging from 9% to 17% for the good group and from 1% to 6% for the bad group. However, among the remaining three subsets, the set [AREA, LTP, OCCR] had the highest classification rates for class 4 in both good and bad groups while maintaining the same level of classification rates for class 1 in both groups. Therefore, this set was chosen as the best set in the method III. The discriminant result with this best feature subset is given in Table 4.28. In this analysis, a serial-two-step Bayes classifier was also used. The calculation time for this subset was 43.61 ms.

⇒ **Best subset from Method III: AREA, LTP, and OCCR**

For the good group, the error rates for classes 1, 2, and 4 were 24.0%, 77.7%, and 5.0%. For the bad group, the error rates were 48.0%, 94.6%, and 8.9% for classes 1, 2, and 4 respectively. This feature subset also showed difficulty in correctly classifying class 2 in the bad group. The total error rates were 34.8% for the good group and 45.5% for the bad group.

Table 4.28 Classification result with the best feature subset from the method III for good training and validation sets.

Good training set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	55 70.51	0 0.00	23 29.49	78 100.00
2	8 6.30	25 19.69	94 74.02	127 100.00
4	10 5.05	2 1.01	186 93.94	198 100.00
Total	73	27	303	403
Percent	18.11	6.70	75.19	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.2949	0.8031	0.0606	0.3201

Good validation set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	114 76.00	0 0.00	36 24.00	150 100.00
2	12 6.70	28 15.64	139 77.65	179 100.00
4	9 4.46	1 0.50	192 95.05	202 100.00
Total	135	29	367	531
Percent	25.42	5.46	69.11	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:

	1	2	4	Total
Rate	0.2400	0.8436	0.0495	0.3484

Table 4.29 Classification result with the best feature subset from the method III
for bad training and validation sets.

Bad training set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	40 44.44	0 0.00	50 55.56	90 100.00
2	7 6.14	6 5.26	101 88.60	114 100.00
4	15 10.87	2 1.45	121 87.68	138 100.00
Total	62	8	272	342
Percent	18.13	2.24	79.53	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.5556	0.9474	0.1232	0.4912

Bad validation set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	64 52.03	0 0.00	59 47.97	123 100.00
2	3 2.70	3 2.70	105 94.59	111 100.00
4	13 8.28	1 0.64	143 91.08	157 100.00
Total	80	4	307	391
Percent	20.46	1.02	78.52	100.00
Priors	0.1000	0.1000	0.8000	
Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.4797	0.9730	0.0892	0.4552

Final selection of the best feature subsets

Finally, the results from Method I, II, and III were summarized as classification error rates in Table 4.30 in order to select the overall best subset among the 27 features. Comparing the results from the three methods, the feature subset from method I showed superiority in classifying class 1 and class 2, however showed very poor results for class 4 in the bad group.

Table 4.30 Classification error rate (%) for each class and total error rate (E_T , %) from the discriminant analysis results from the methods I, II, and III. (where C_i is i -th class.)

Method	Selected features	Time (ms)	Good validation set				Bad validation set			
			C1 (%)	C2 (%)	C4 (%)	E_T (%)	C1 (%)	C2 (%)	C4 (%)	E_T (%)
I	AVGABSC, ATL, ATP, PTB, CMP, ELG, HET, MNX, MTM, NEG, OCCR, PTC, PTP	148.76	2.0	47.5	22.8	25.2	7.3	39.6	54.8	35.6
II	MNX, CMP, LTP, OCCR	64.07	12.7	61.5	5.5	26.4	12.2	91.0	28.0	40.9
III	AREA, LTP, OCCR	43.61	24.0	77.7	4.9	34.8	48.0	94.6	8.9	45.5

The feature subset from method III showed lower error rates for class 4 in both good and bad groups than those from the method II, and higher error rates for classes 1 and 2. Since additional weed control needs to be done if the prototype system does not kill all the weeds, weed recognition is more important than tomato plant recognition.

Also tomato growers usually plant more tomato seeds to ensure a good stand and thin the stand after first cultivation. In terms of calculation time for feature subsets, the subset from method III took the least time (43.61 ms) among three subsets. Therefore, the subset of [AREA, LTP, OCCR] was selected as the overall best subset for identifying tomato plants and weeds:

⇒ **Final best subset from Method I, II, and III:**

AREA, LTP, and OCCR

With this final best subset, a discriminant analysis was conducted with 4 classes (classes 1, 2, 3, and 4) by introducing the third group (class 3) after feature selection to show the classification results for all classes. In this analysis, the classifier was trained with the classes 1, 2, and 4 in the good training data set and was validated with all 4 classes since the classifier was designed to classify classes 1, 2, and 4. Instead of using a separate classifier for the good and bad image quality groups, a classifier trained on the good training data set was used for both the good and the bad validation data sets since the robot could not know to which group the image belongs in order to use the good vs. the bad group classifier. The same *a priori* probabilities were used: class 1 = 0.1, class 2 = 0.1 and class 4 = 0.8. The results are given in Table 4.31 for the good group and Table 4.32 for the bad group.

For the good group, 80.7% of class 1 (tomato cotyledons) and 95.5% of class 4 (weeds) were correctly classified, however the recognition rates for class 2 (21.2% of tomato true leaves were correctly classified as tomato plants) and class 3 (12.9% of the third class were correctly classified as tomato plants) were very low, indicating that the

recognition of those classes would be very difficult. In terms of classification error rate, class 2 had 78.8% and class 3 had 87.2%.

Table 4.31 Classification result for 4 classes in good group using the best three features (AREA, LTP, and OCCR).

Good group, training set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:				
From CLASS	1	2	4	Total
1	55 70.51	3 3.85	20 25.64	78 100.00
2	8 6.30	24 18.90	95 74.80	127 100.00
4	12 6.06	3 1.52	183 92.42	198 100.00
Total	75	30	298	403
Percent	18.61	7.44	73.95	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:				
	1	2	4	Total
Rate	0.2949	0.8110	0.0758	0.3226

Good group, validation set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:

From CLASS	1	2	4	Total
1	121 80.67	0 0.00	29 19.33	150 100.00
2	14 7.82	24 13.41	141 78.77	179 100.00
3	45 7.05	37 5.80	556 87.15	638 100.00
4	9 4.46	0 0.00	193 95.54	202 100.00
Total	189	61	919	1169
Percent	16.17	5.22	78.61	100.00
Priors	0.1000	0.1000	0.8000	

Error Count Estimates for CLASS:					
	1	2	3	4	Total
Rate	0.1933	0.8659	.	0.0446	0.6287

Table 4.32 Classification result for 4 classes in bad group using the best three features (AREA, LTP, and OCCR).

Bad group, validation set: AREA, LTP, OCCR

Number of Observations and Percent Classified into CLASS:					
From CLASS	1	2	4	Total	
1	72 58.54	0 0.00	51 41.46	123 100.00	
2	2 1.80	8 7.21	101 90.99	111 100.00	
3	34 5.13	25 3.77	604 91.10	663 100.00	
4	10 6.37	1 0.64	146 92.99	157 100.00	
Total	118	34	902	1054	
Percent	11.20	3.23	85.58	100.00	
Priors	0.1000	0.1000	0.8000		
Error Count Estimates for CLASS:					
	1	2	3	4	Total
Rate	0.4146	0.9279	.	0.0701	0.7277

For bad group, the result showed similar trend with reasonably high classification rates for the classes 1 (58.5%) and 4 (92.9%) and extremely low rates for classes 2 and 3. The classification error rates were 90.9% for class 2 and 91.1% for class 3. These high error rates for classes 2 and 4 came mainly from their shape being very similar to weeds. Sometimes true leaves didn't have distinct notches on their boundaries, and showed the similar convex shape of weeds. Even when true leaves had clear notches, their shape was frequently similar to the one of overlapped weeds or to concave weeds. Most of the third group was composed of curled, laid down, or upright tomato leaves, thus they couldn't satisfy the feature criteria of tomato leaves. However, although the classification rates for true leaves and the third group were very low in both groups, the result was still useful

since most of the weeds were correctly recognized for cultivation and tomatoes were usually over planted. The optimum window in time when the current technology could be used would be very early cultivation before the cotyledons fall off or become hidden by true leaves. However, further studies are required for true leaves and the third group to enhance the classification rate since in both good and bad groups the recognition rate for both classes were very low.

With these selected features, the image processing algorithm took 0.341 s to distinguish 10 tomato cotyledons in the image for one frame of a 256 x 240 pixel image representing a 11.43 cm x 10.16 cm field of view, Table 4.33. Thus, the prototype could travel at a continuous rate of 1.21 km/h.

Table 4.33 Execution time for each image processing step.

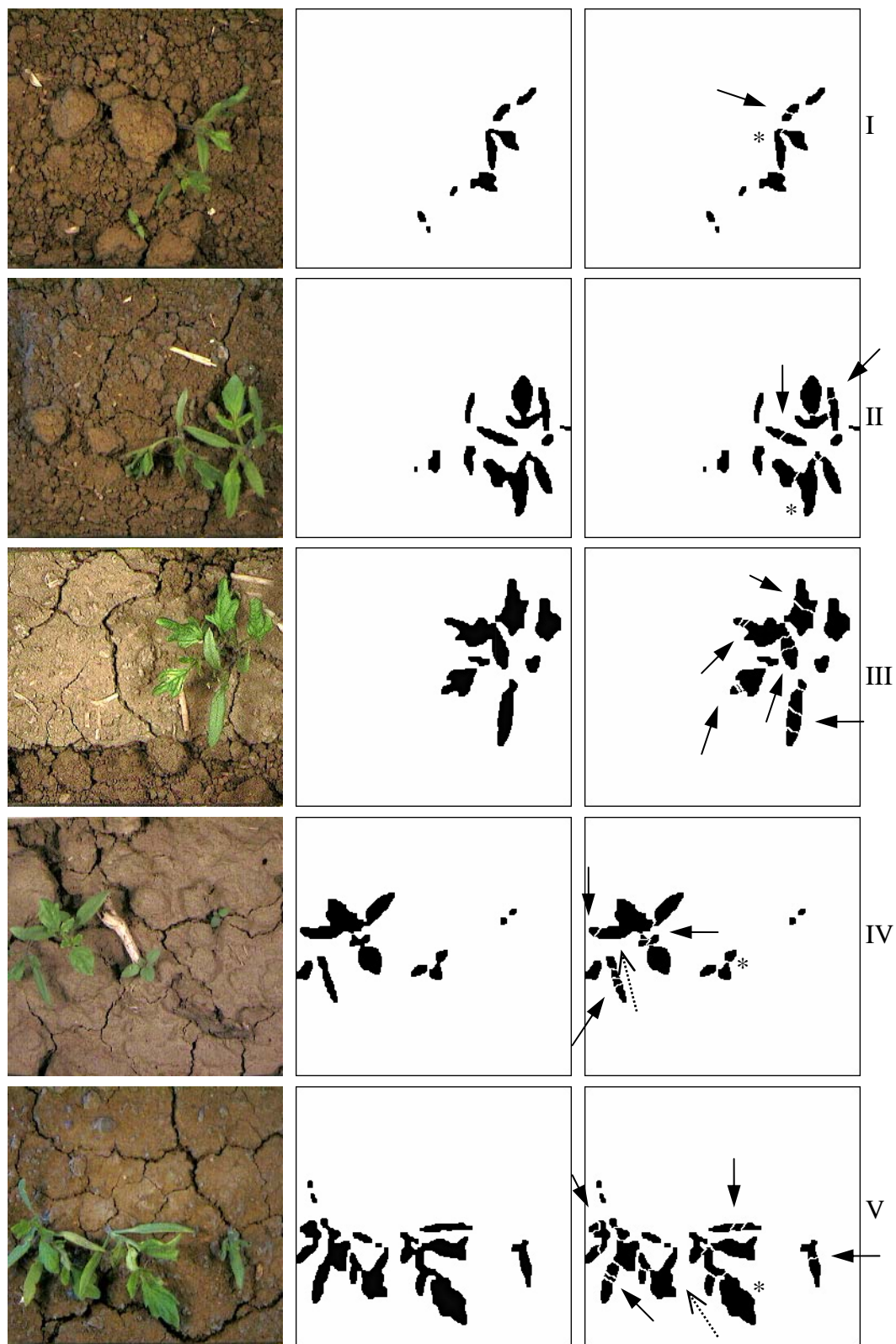
Image processing step	Execution time (ms)	Percent of Total time
Prepare image acquisition	0.02	0.01
Acquire color image (one field)	16.76	4.87
Transfer and subsample acquired image	27.21	7.91
Check synchronization of image processing computer and spray controller	2.08	0.60
Check image buffer overflow	1.19	0.35
Binarize	2.92	0.85
Morphology analysis	32.04	9.31
Label objects	9.89	2.87
Extract features	141.28	42.00
Make decision with a Bayesian classifier	0.94	0.27
Find tomato & weed locations	58.10	16.88
Send tomato & weed locations to spray controller	37.44	10.88
Miscellaneous commands	11.03	3.20
Total time	340.90	100.00

4.2.3 Separation of touching leaves: Watershed method

Occlusion has been one of the most difficult obstacles in machine vision recognition of outdoor scenes since occluded objects are difficult to identify. Multiple occluded objects appear as one object in the segmented binary image, producing an unusual set of feature values. In this chapter, the watershed algorithm (Vincent and Soille, 1991) was implemented to cut apart the occluded leaves, as a preprocessing step to improve object identification.

The implementation of the original watershed algorithm did not work very well and tended to produce excessive cutting. As discussed in chapter 3.3.3.8, this is due to the fact that every regional minima becomes the center of a catchment basin. Some examples of watershed over cutting are shown in Figure 4.10 (c). The black arrows in the figures in this chapter indicate either excessive cutting (\blackrightarrow) or improper object separation ($\cdots\blackrightarrow$). A star (*) indicates a properly cut leaf. The original algorithm produced 15 over cut leaves, 2 uncut leaves and 4 properly cut leaves.

The original watershed algorithm was modified in five different ways to reduce excessive cutting. When occluded objects, excessively cut by the original algorithm, were properly separated by a modified watershed algorithm, the plant leaves were labeled with numbers in the figures in this chapter.



(a) Color images.

(b) Binary images.

(c) Over cutting.

Figure 4.10 Examples of over cutting or improper separation (black arrows) by the original watershed algorithm.

Figure 4.11 shows the results of the watershed algorithm modified with *opening*. Many of the instances of over cutting were avoided (labeled as 1 - 9 in Figure 4.11) by applying the *opening* operation since it connected local minima into a larger connected component. However, some objects were still excessively separated and the *opening* operation could not remove all of the instances of over cutting.

Let m_1 be the number of occluded leaves over cut, m_2 be the number of uncut leaves, m_3 be the number of properly cut leaves and m_4 be the adjusted number of correctly cut leaves ($= m_3 - m_2$). Considering there were 15 over cut leaves, 2 uncut leaves and 4 properly cut leaves by the original algorithm (Figure 4.10 (c)), m_4 would be 2 for the original algorithm. Then, the excessive cutting improvement from a modification can be calculated as the adjusted number of correctly cut leaves minus 2, i.e.,

$$\text{Improvement} = \frac{m_4 - 2}{15} \times 100 (\%). \quad (4.1)$$

The *opening* modification resulted in 7 over cut leaves, 2 uncut leaves, and 12 properly cut leaves in the 5 images in Figure 4.11. The improvement from this modification was calculated as

$$\frac{10 - 2}{15} \times 100 = 53.3 (\%).$$

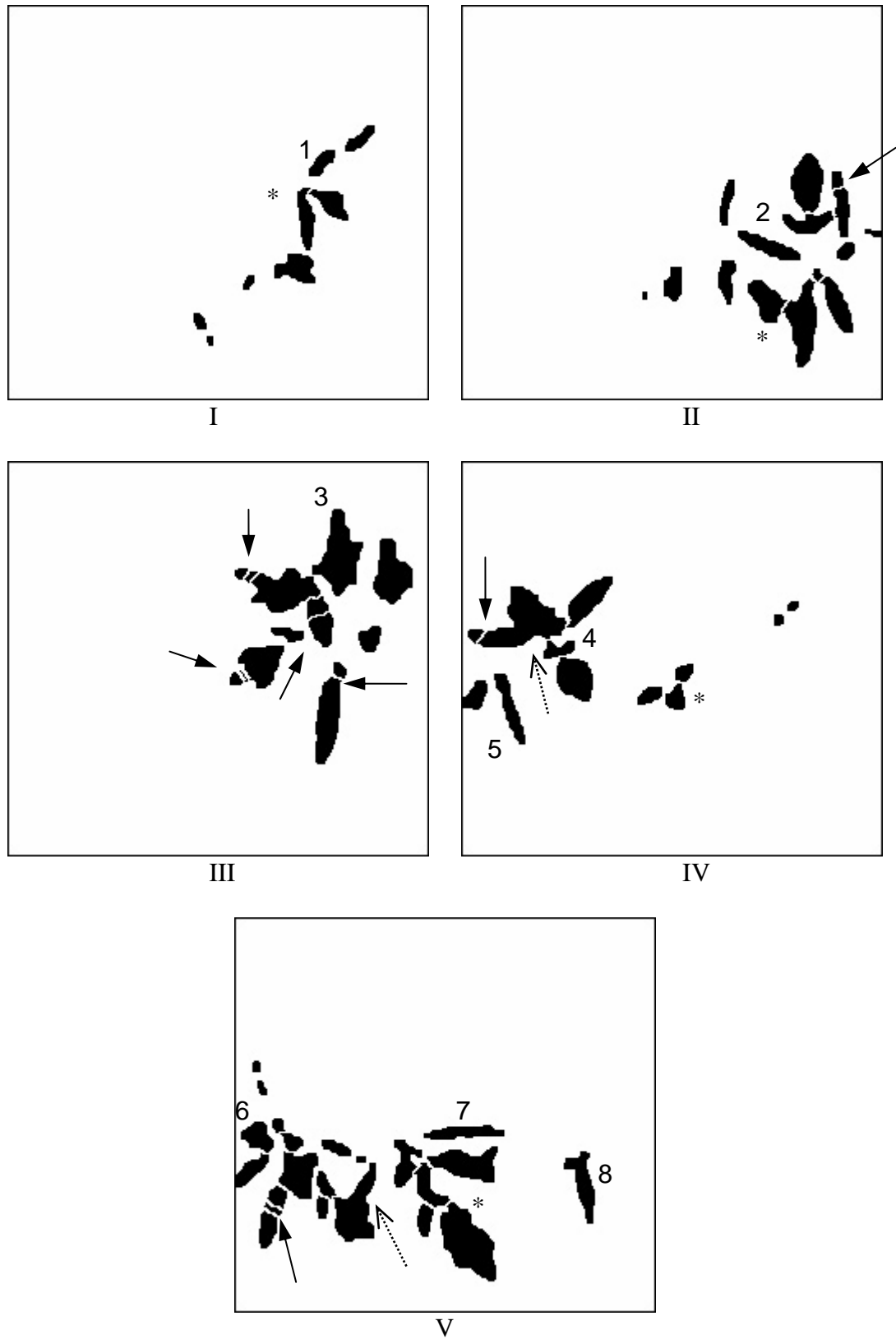


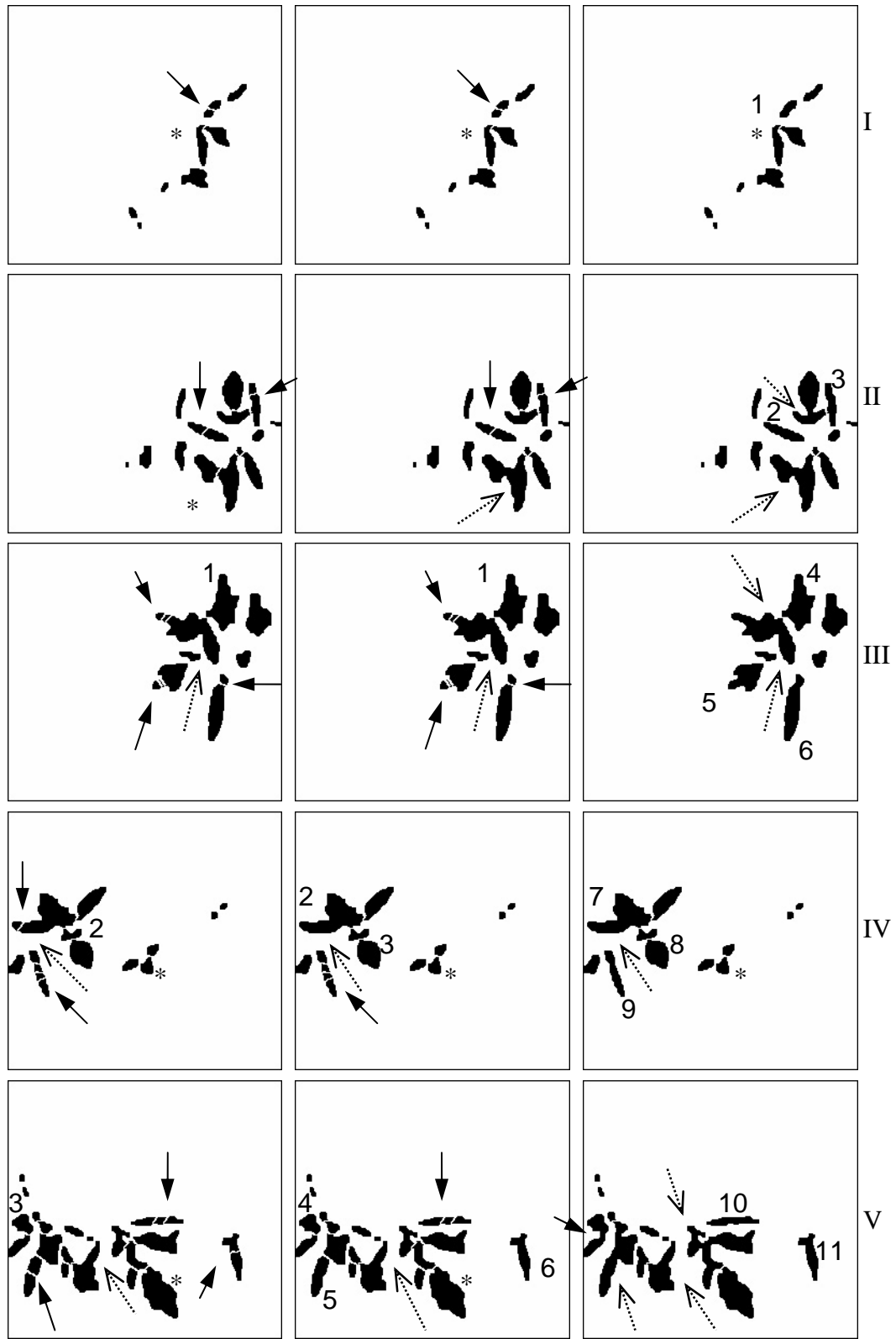
Figure 4.11 Results of watershed algorithm modified with the *opening* operation.

Figure 4.12 shows the result of the pre-flooding algorithm using a flooding level up to heights of 251, 252 and 253 respectively for each of the 5 images in Figure 4.10. The typical range of pixel values for the local minima in cotyledons after distance transformation was around 250 since they were usually long and thin. Other leaves including true leaves and occluded leaves had local minima as low as about 230, depending on their size. If the pre-flooding level was too low, the operation was ineffective in reducing over cutting. On the contrary, if the pre-flooding level was too high, the objects were not cut properly. For example, the pre-flooding level of 251 produced 11 over cut objects, 3 uncut leaves, and 7 properly cut leaves (Figure 4.12 (a)), whereas the pre-flooding level of 253 produced no over cut leaves, and 12 properly cut leaves with 9 uncut leaves (Figure 4.12 (c)). The adjusted number of correctly cut leaves after pre-flooding were 4, 5, and 3 for pre-flooding levels of 251, 252, and 253 respectively. Thus, using Eq. (4.1), the improvement from the modification with pre-flooding levels of 251, 252, and 253 were 13.3%, 20.0%, and 6.7% respectively, considering 15 over cut leaves produced by the original algorithm. The appropriate level of pre-flooding varied with object shape.

The spatial resolution might affect the pre-flooding technique because with a spatial resolution greater than the current resolution, 0.45 mm/pixel (= 114.3 mm / 256 pixel), there would be a greater range of pre-flooding levels to work with. There might be a minimum spatial resolution for pre-flooding to work properly on objects of a certain size.

In applying the watershed algorithm, over cutting due to noise needs to be considered. The noise could come from the digitization of an image and a low spatial

resolution image would not represent long and thin objects very well. For example, in some cases the presence or absence of a single pixel along the boundary of a cotyledon (refer to Figure 3.26 (c), page 73) could lead to proper or non-proper cutting by the watershed algorithm since the single pixel could disconnect the local minima into more than two disconnected regions. As Russ (1990) pointed out, the watershed algorithm has two implicit assumptions (of a convex object and sufficiently small overlap between objects), the convex object assumption did not always apply to overlapped plant leaves particularly when boundary noise causes local concavities. This is probably why the opening (smoothing) technique had the best performance of the modifications attempted. However, more study would be needed to investigate the relationship between the pre-flooding technique and the spatial resolution.



(a) Pre-flood level: 251. (b) Pre-flood level: 252. (c) Pre-flood level: 253.

Figure 4.12 Result of watershed algorithm modified by pre-flooding.

The third modification was to determine when the watershed algorithm needs to be applied based on the features of AREA, ELG, and CMP. Figure 4.13 shows the images resulting from the use of this feature criteria. This modification produced 6 over cut leaves, 3 uncut leaves, and 12 properly cut leaves, which led a 46.7% excessive cutting improvement, giving more appropriate cutting results than using the pre-flooding modifications. Comparing the performance of this modification with the one from *opening* modification, there was only one more correctly cut leaf from *opening* modification than from the feature criteria modification. Thus, these two modifications could be considered to have a similar level of performance (improvement). In order to validate the performance of this modification, more samples would be needed to obtain more accurate feature ranges for applying the watershed algorithm.

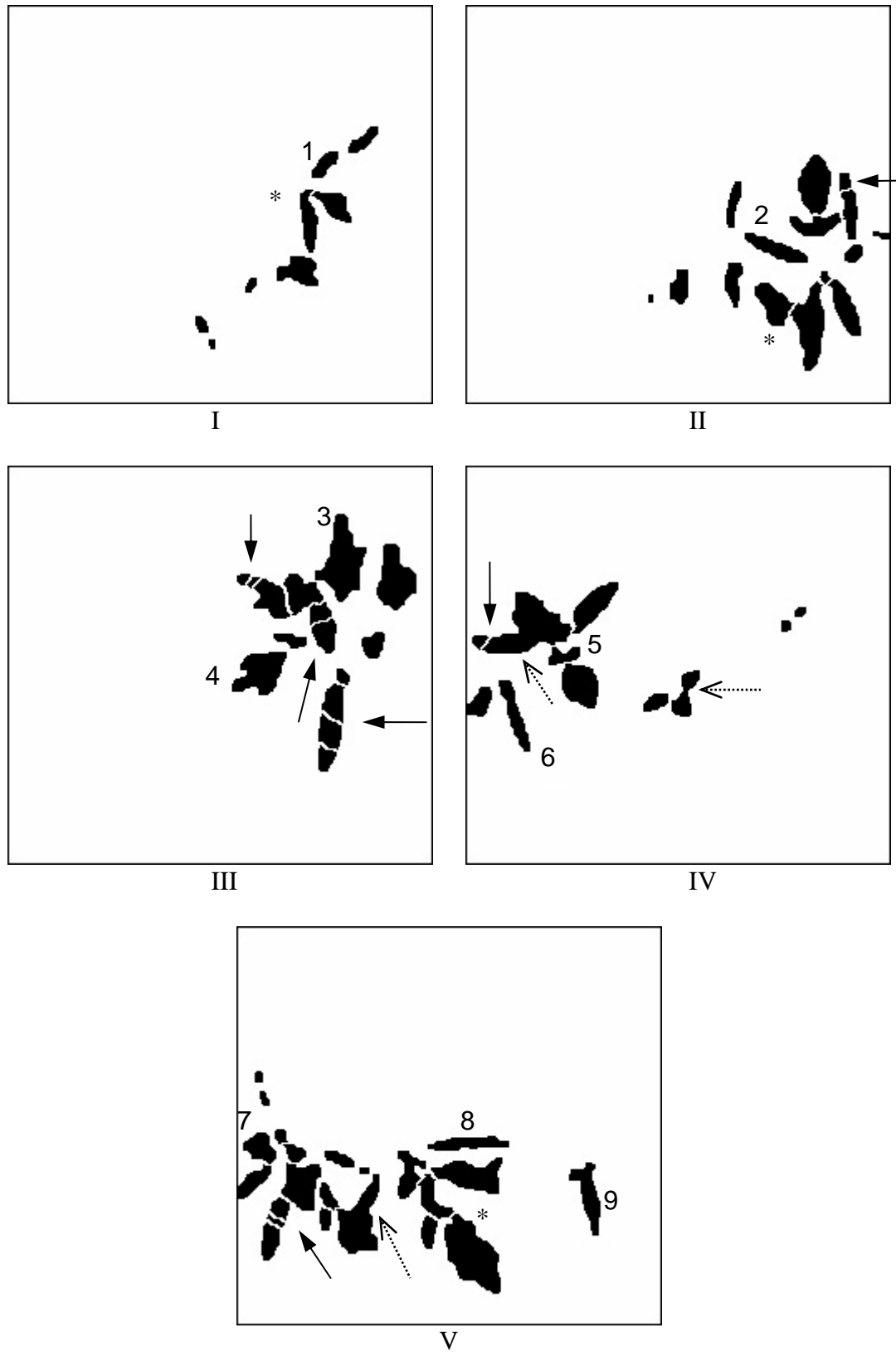


Figure 4.13 Result of watershed algorithm modified by the feature criteria.

Figure 4.14 shows the results of using the concavity criteria to determine the watershed algorithm application. Like the modification with the feature criteria, this modification determined selective application of the watershed method to objects. Some leaves could benefit by the criteria, be hurt (skipped or not properly cut) by the modifications or fixed by *opening* or pre-flooding. This modification produced 11 over cut leaves, 2 uncut leaves and 8 properly cut leaves. The performance of this modification (26.7% improvement) was a lot worse than the modification with the feature criteria, Table 4.34.

Table 4.34 Comparison of feature criteria and concavity criteria with the number of leaves benefiting from, hurt by, could be fixed by *opening* or pre-flooding by the modifications.

Number of leaves	Feature criteria	Concavity criteria
Benefited from modification	9	4
Hurt by modification	1	0
Over cut leaves could be fixed by <i>opening</i> or pre-flooding	5	8

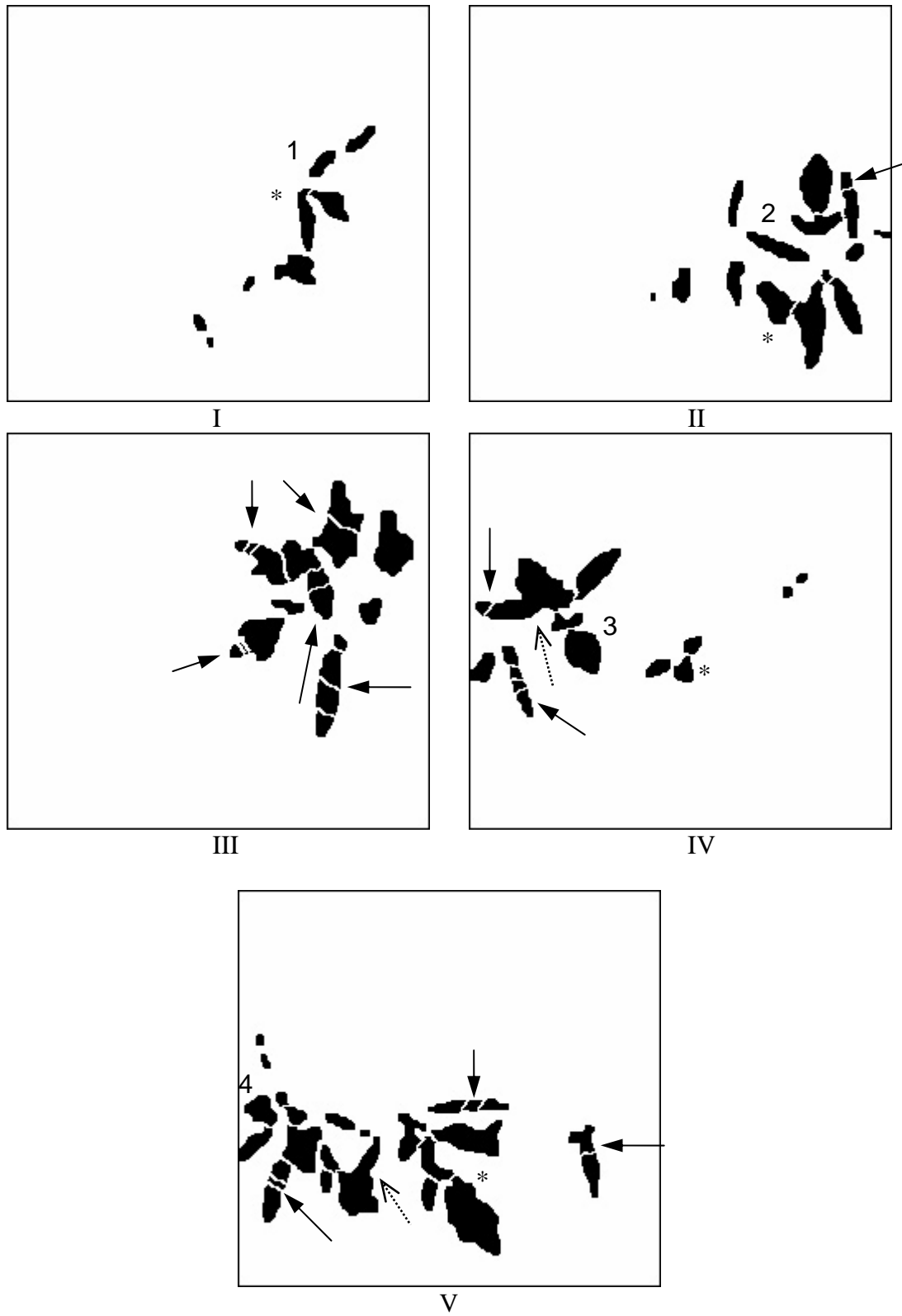


Figure 4.14 Result of applying number of concavities to the watershed algorithm.

Finally, the modification with *opening* and the feature criteria were applied together to optimize cutting. Figure 4.15 shows the results of this combined modification of the watershed algorithm which reduced over cutting by 62.5% (reducing over cut leaves from 16 to 6). However this modification still produced over cut objects with the same improvement rate of 43.8% as the modifications with *opening* operation and feature criteria.

Comparing the results in Figure 4.11 - 4.15 from all methods applied so far (*opening* operation, pre-flooding, feature criteria, number of concavities, and *opening* and feature criteria) with the original algorithm, all modifications produced improved results when compared to the original watershed algorithm. The performance of the modifications are summarized in Table 4.35.

Table 4.35 Performance of different modified watershed algorithms, compared to the original method.

Method	No. of occluded leaves over cut (m1)	No. of occluded leaves uncut (m2)	No. of occluded leaves properly cut (m3)	Adjusted no. of correctly cut leaves (m4=m3-m2)	Improvement from modification $((m4-2) / 15 \times 100, \%)$
W0	15	2	4	2	0.0
W1	7	2	12	10	53.3
W2 - 251	11	3	7	4	13.3
W2 - 252	8	4	9	5	20.0
W2 - 253	0	9	12	3	6.7
W3	6	3	12	9	46.7
W4	11	2	8	6	26.7
W5	6	3	12	9	46.7

Although the five modifications could be divided into two groups, those fixing the watershed algorithm (W1 and W2) and those applying the algorithm selectively (W3 and W4), their performance was listed in the same table for simplification.

The performance of the *opening* modification is a lot better (53.3% improvement) than the pre-flooding technique (6.7% - 20.0% improvement). In the pre-flooding technique, the pre-flooding level of 252 was a little better than the levels of 251 and 253. The performance of the feature criteria modification was a lot better than the modification with the concavity criteria. The combined modification (W5 algorithm) of *opening* operation and feature criteria showed same improvement as when they were applied separately. Overall, as stated before, W1, W3 and W5 algorithms showed the same level of performance (only one correctly cut leaf difference). Thus, the best modifications were the application of *opening*, feature criteria and the combined modification of *opening* and feature criteria.

However, no one method produced perfect cutting results. The watershed algorithm might not be ideal for separating occluded plant leaves when the local minima was composed of more than two disconnected regions. In this case watershed algorithm required additional information or modification to produce more accurate results. The watershed algorithm (Vincent and Soille (1991)) seemed to have a basic problem of over cutting when the local minima was composed of two or more disconnected regions.

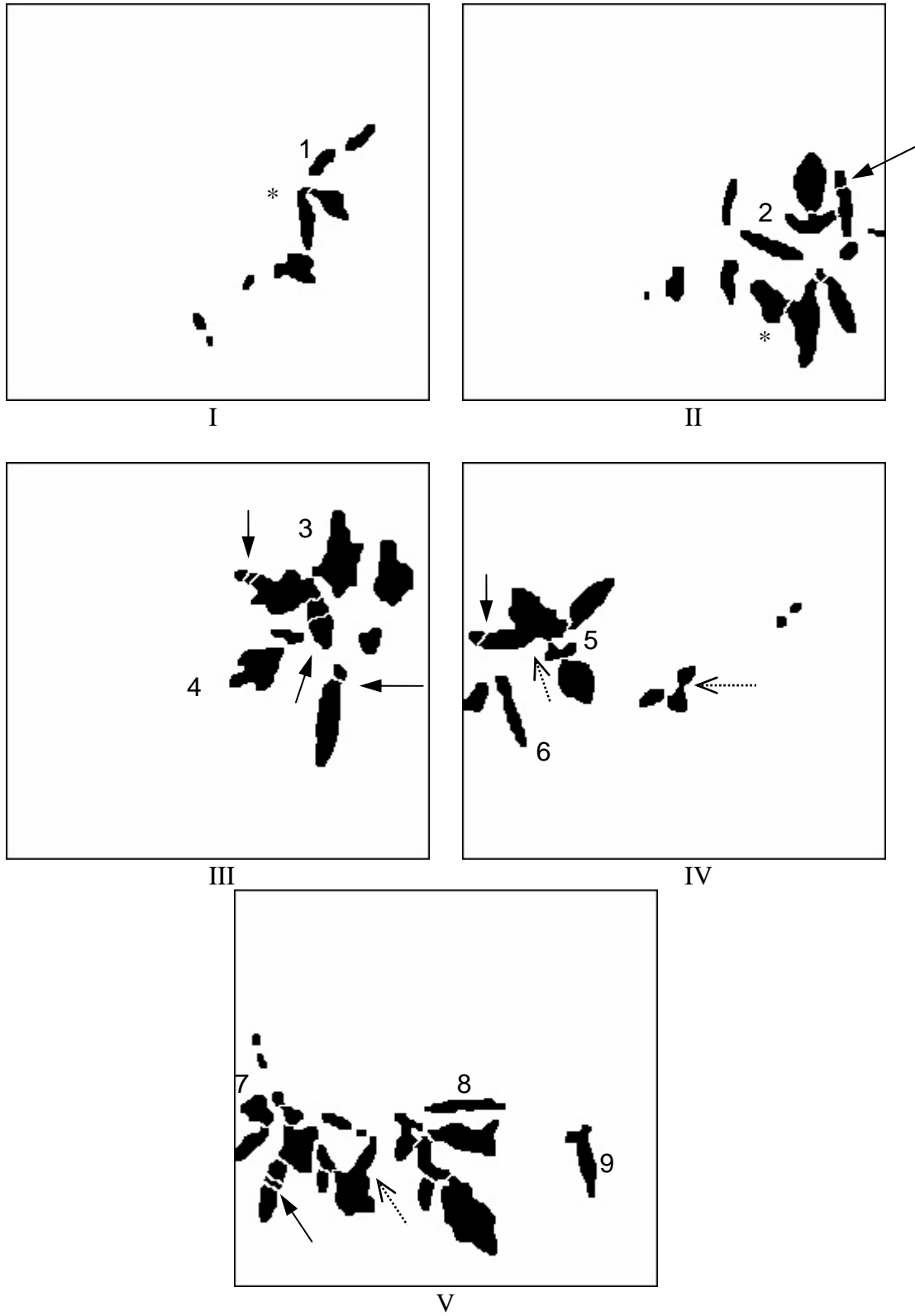


Figure 4.15 Result of application of both *opening* and the feature criteria to the original watershed algorithm.

Since the objective of this research was to build a real-time weed control system, the execution time for the original and the modified algorithms was roughly measured, Table 4.36, just to show that these methods could not be implemented in real-time using a 200 MHz Pentium Pro CPU. The execution time for the watershed algorithm was measured from the step of making the distance function of the occluded binary leaf to the step of obtaining the separated leaves. In this task, the second image (II) from the top in Figure 4.10 was used as the test image since the number of leaves in that image was typical of these in the study. Each result listed in Table 4.36 was the average of 10 executions using a 200 MHz Pentium Pro processor.

Table 4.36 Execution time of original and modified watershed algorithms.

Modification	Description	Execution time (sec)
W0	Original algorithm	16.69
W1	<i>Opening</i> operation	19.68
W2 - 252	Pre-flooding	17.73
W3	Feature criteria	14.79
W4	Concavity criteria	14.64
W5	<i>Opening</i> & feature criteria	16.70

The modifications with feature criteria (W3) and concavity criteria (W4) took the least time since the watershed subroutine was not applied to some objects, which were non-occluded. Modifications with the *opening* operation (W1) took the most time since sorting of pixel values and matching pixel locations with each pixel address were repeated as many as three times. Modifications with *opening* operation and feature criteria combined (W5) took almost same time as the original algorithm since in the original algorithm the watershed was applied to all objects, whereas the watershed and

opening operation were only applied to those objects which satisfied the feature conditions in the modification with *opening* and feature criteria combined.

Despite its time limitation, the watershed algorithm could help in identifying occluded objects more correctly. Figure 4.16 shows some examples of correctly identified objects before and after the watershed algorithm (modified with feature criteria and concavity criteria combined with pre-flooding up to 252) was applied. In all of the five test images, 15 tomato leaves were correctly recognized after the watershed algorithm application, which were incorrectly recognized as 7 occluded leaves without the watershed algorithm. For the recognition subroutine, the following rule of ELG and CMP were applied, which were obtained from the database previously built.

If $0.261 < ELG < 0.875$ and $0.450 < CMP < 1.073$, then identify as tomato plant.

Otherwise, identify as weeds.

In order to estimate the impact of the watershed algorithm on plant classification rate, 12 field images were selected randomly, not including the previous 5 test images. For each image, the normal plant identification algorithm with the above rule and the modified watershed algorithm with feature criteria and concavity criteria combined with pre-flooding up to 252 were executed and the number of correctly recognized and incorrectly recognized objects were counted, Table 4.37 - 4.38.

For occluded cotyledons, the normal algorithm correctly identified 28.6% of them, however, with the modified watershed algorithm 61.9% of them were correctly identified. For occluded true leaves, the identification rate increased from 11.8% to

52.9% after the application of the modified watershed algorithm. Thus, if the watershed algorithm could be implemented in real-time, it could help to identify plants more correctly.

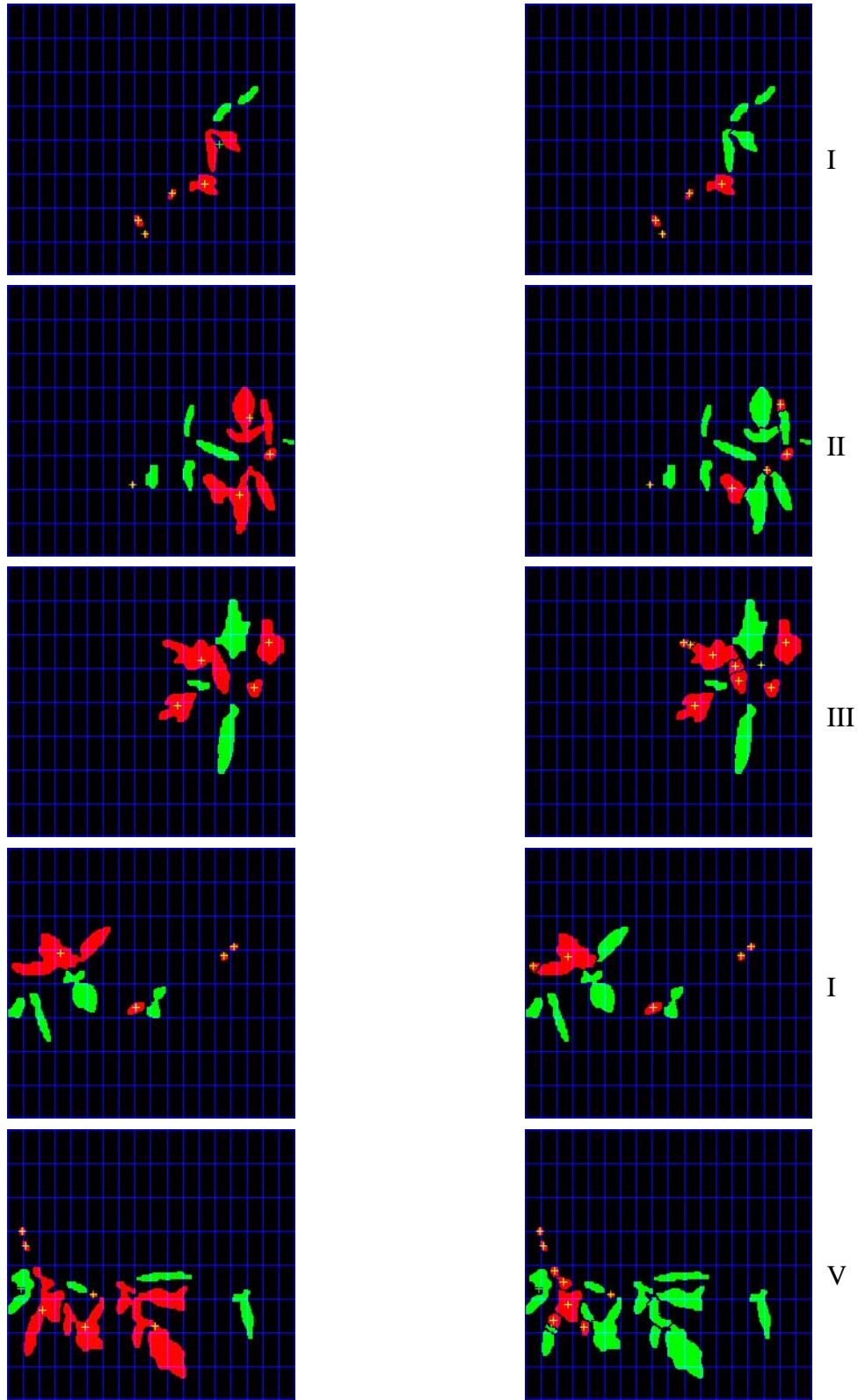
Table 4.37 Recognition results with normal identification algorithm and without watershed application. (C = Correctly identified, I = Incorrectly identified)

	Cotyledon				True leaf				Weed			
	Non-occluded		Occluded		Non-occluded		Occluded		Non-occluded		Occluded	
	C	I	C	I	C	I	C	I	C	I	C	I
No. of leaves	17	1	6	15	10	10	4	30	10	1	2	0
%	94.4	5.6	28.6	71.4	50.0	50.0	11.8	88.2	90.9	9.1	100.0	0.0

Table 4.38 Recognition results with modified watershed algorithm (feature criteria, concavity criteria with pre-flooding up to 252). (C = Correctly identified, I = Incorrectly identified)

	Cotyledon				True leaf				Weed			
	Non-occluded		Occluded		Non-occluded		Occluded		Non-occluded		Occluded	
	C	I	C	I	C	I	C	I	C	I	C	I
No. of leaves	17	1	13	8	10	10	18	16	10	1	2	0
%	94.4	5.6	61.9	38.1	50.0	50.0	52.9	47.1	90.9	9.1	100.0	0.0

The watershed algorithm may not be feasible for real-time implementation for now due to the over cutting problem and time limitations due to intensive computation with current computing technology. However, in the near future it may be possible to implement it in real-time making it suitable for use in a real field application. Thus, further research would be appropriate in order to improve the separation accuracy and to achieve faster execution.



(a) Without watershed algorithm.

(b) With watershed algorithm.

Figure 4.17 Comparison of recognition result without and with watershed algorithm.

4.3 Performance of precision chemical application system

4.3.1 Displacement sensor performance

Tests were conducted in order to observe the operation of the displacement sensor alone on different ground surfaces (soil surface, paved road, and smooth concrete surface). The precision spraying system was set to spray all 8 valves at a predetermined spacing. The distance between each of the two sprayed lines was measured, subtracted from the predetermined spacing and considered as errors.

The absolute values of the position errors were taken and their mean and standard deviations were summarized, Table 4.39. The mean position error on the smooth concrete surface was smaller than those from the soil surface or the paved road. The standard deviation of the position error from the paved road and the smooth concrete surface were smaller than that of the soil surface. This shows that the displacement sensor worked more consistently on a smoother surface. The drops sprayed on the paved road and smooth concrete surface seemed to be better aligned and more uniform than those sprayed on the soil surface. The overall mean error of the precision spraying system without any image processing operation was 0.15 cm and the standard deviation was 0.19 cm.

Table 4.39 Performance of displacement sensor on different ground surfaces.

	Soil surface	Paved road	Smooth concrete surface	Total
No. of readings	73	57	228	358
Mean (cm)	0.23	0.15	0.12	0.15
Std. Deviation (cm)	0.19	0.14	0.15	0.19

Many sources of error were observed during field tests of the displacement sensor. Even with the gage wheel, the displacement sensor did not always work consistently (i.e., did not generate the same number of pulses for the same distance). This was due to the fact that there were bumps, clods, or irrigator's footsteps on the bed where the gage wheel traveled, or the compactability of the soil was different from one field to another. The circumference of the gage wheel, which affected the number of pulses generated from the encoder, also varied with tire pressure and load exerted on the gage wheel by the toolbar and equipment. One example of toolbar effect on imaging and spraying was the circumference change of the gage wheel by lifting or lowering the toolbar. For every field test the encoder was calibrated for a pre-measured distance. The pre-measured distance was then divided by the number of encoder counts and distance per count was obtained to calculate the three parameters of the microcontroller. Then, prior to every field test, the toolbar was lifted from the ground to set the encoder to its initial image acquisition position by turning the gage wheel manually. Then, the toolbar was put down. However, the circumference of gage wheel changed after it was put down on the ground since it was not put down exactly in the same way.

The prototype was tested with full weight and light weight exerted on the gage wheel on the paved road. When the toolbar was lowered fully (full weight), the precision spraying system sprayed 34 times in 750.57 cm, which resulted in 22.08 cm distance between spray drops. When the toolbar was only lowered until the gage wheel just touched the paved road (light weight), the precision spraying system sprayed 32 times in 746.76 cm, which resulted in 23.34 cm distance between spray drops. There was 1.26 cm

(= 23.24 cm - 22.08 cm) difference for the same distance sensed depending on the weight exerted on the gage wheel.

The encoder resolution was about 0.13 mm per count, a small change of the circumference of the gage wheel would affect the performance of the prototype greatly. When this happened the image size was not 11.43 cm wide anymore and the precision spraying system would spray the wrong location for the weeds even though the computer vision system correctly identified weed locations.

4.3.2 Spray targeting accuracy without imaging

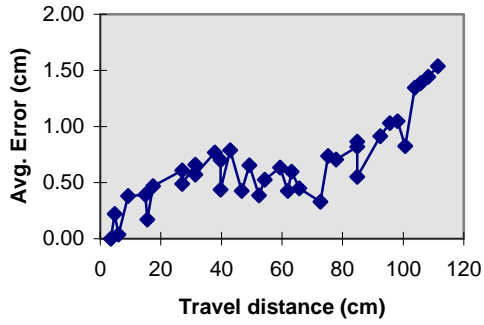
In order to isolate problems within the scope of the precision spraying system, the prototype was set to spray a predetermined “imaginary” weed pattern without taking any images and with no targets on the ground. Four types of ground surface were used to test the system with two different error measuring methods. The error were from the center of the target to the center of the spray drops and even when there was an error, it did not mean that the target was missed, only that the spray was not centered on the target.

Errors measured by Method I

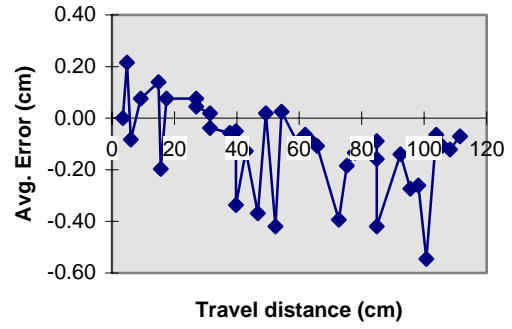
Method I measured spray errors referenced from the starting line (beginning of the first image) of each test. Figures 4.17 - 4.18 and Table 4.40 show the average error from the spraying-only experiment on four different surfaces using analysis method I. Note that the errors accumulated as the travel distance increased (Figure 4.17 (a) and (c) and Figure 4.18 (a) and (c)). This was due to the measurement method (Method I), which

leads inevitably to the accumulation of errors since incremental motion information is used over time, i.e., dead reckoning (Borenstein et al. 1996). Even though the encoder was calibrated prior to each test, the accumulation of error was unavoidable since calibration itself may not be accurate due to unknown effective radius of tire or slippage of the gage wheel. In addition to gage wheel slippage, there may be other sources of errors for dead reckoning which are systematic: unequal wheel diameter, misalignment of wheel, finite encoder resolution, finite encoder sampling rate, and which are non-systematic: travel over uneven floors or unexpected objects on the ground surface.

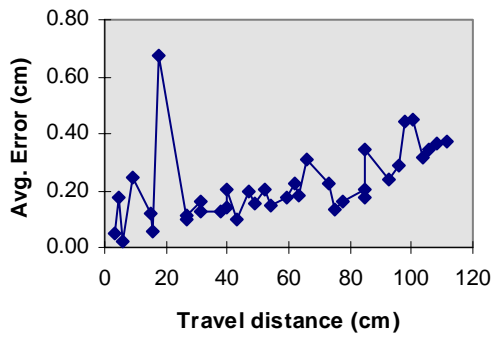
However, method I was not appropriate for measuring the error of this experiment since the nozzles were located 17.0 cm (about 1½ images) behind the center of the camera and the nozzles needed to travel only 17.0 cm before they were activated. If the displacement sensor was inaccurate in this 17.0 cm distance, then the spray would miss the target. Thus, as long as the encoder wheel was accurate in the 17.0 cm distance, the prototype would spray accurately. The average error in y direction indicated that how straight the prototype traveled during the tests.



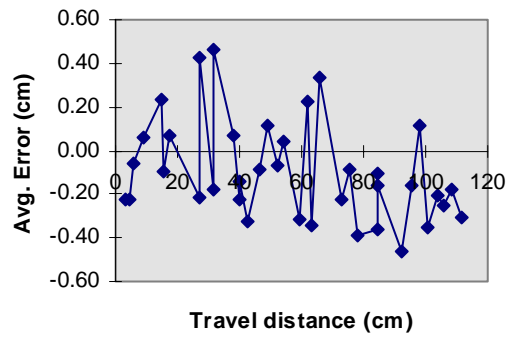
(a) Smooth concrete floor: x - direction.



(b) Smooth concrete floor: y - direction.

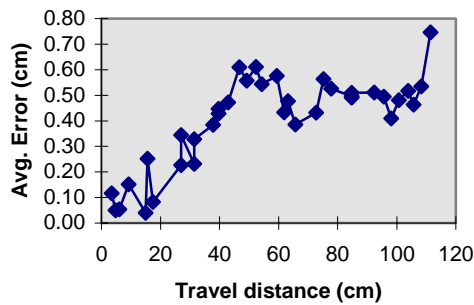


(c) Asphalt: x - direction.

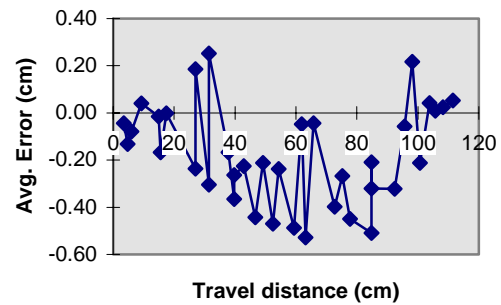


(d) Asphalt: y - direction.

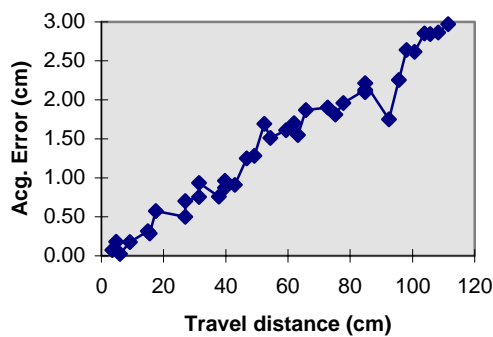
Figure 4.17 Average spray errors measured by Method I on concrete floor and asphalt.



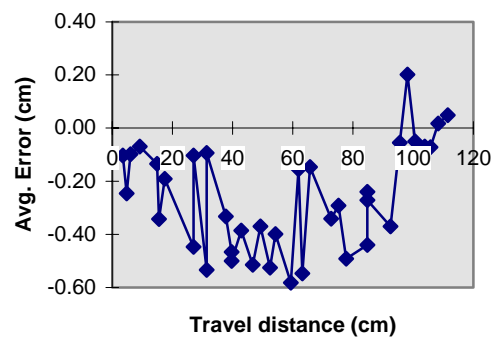
(a) Medium tomato bed: x - direction.



(b) Medium tomato bed: y - direction.



(c) Rough tomato bed: x - direction.



(d) Rough tomato bed: y - direction.

Figure 4.18 Average spray errors measured by Method I on medium and rough beds.

Table 4.40 shows the average spray error and standard deviation on four surfaces calculated in the x (travel) direction, the y direction (perpendicular to travel direction), and x & y directions simultaneously ($= \sum \sqrt{x^2 + y^2}$). The average error in the x (travel) direction increased in the following surface order: asphalt \rightarrow medium bed \rightarrow smooth concrete surface \rightarrow rough bed. Surprisingly, the medium bed had a smaller average error than the smooth concrete floor. Except for the smooth concrete surface, the results confirmed that the displacement sensor worked better on a smoother surface.

Table 4.40 Average error and standard deviation on different surfaces by Method I.

(unit: cm)	Avg. Error			Std. Dev. of Error		
Surface	<i>x</i>	<i>y</i>	(<i>x</i> & <i>y</i>)	<i>x</i>	<i>y</i>	(<i>x</i> & <i>y</i>)
Smooth concrete floor	0.658	-0.116	0.704	0.363	0.169	0.352
Medium bed	0.405	-0.173	0.541	0.178	0.208	0.189
Rough bed	1.443	-0.262	1.530	0.887	0.199	0.848
Asphalt	0.218	-0.097	0.349	0.130	0.228	0.141

Errors measured by Method II

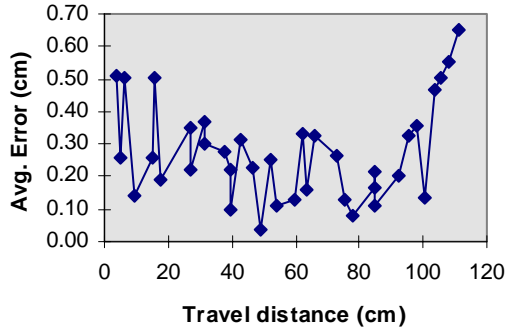
Method II measured spray errors referenced from the relative position at the time of spraying. The positions of the actual spray drops were measured based on the line (Line A in Figure 3.37) where the camera was aligned at the time of spraying (Figure 3.37). Table 4.41 shows the average spray error and standard deviation of errors measured by Method II on different ground surfaces. Again the average spray error and standard deviation on four surfaces were calculated in the *x* (travel) direction, the *y* direction (perpendicular to travel direction), and *x* & *y* directions simultaneously ($= \sum \sqrt{x^2 + y^2}$). The error level was reduced by almost 50% of that from Method I for all surfaces.

Table 4.41 Average error and standard deviation on different surfaces by Method II.

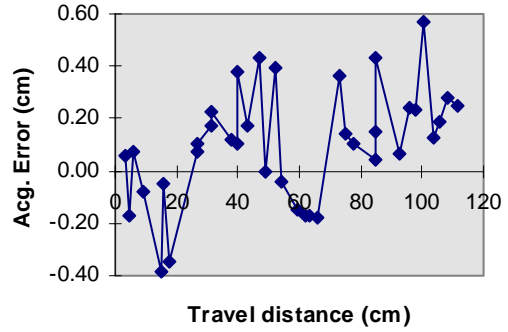
(unit: cm)	Avg. Error			Std. Dev. of Error		
Surface	<i>x</i>	<i>y</i>	(<i>x</i> & <i>y</i>)	<i>x</i>	<i>y</i>	(<i>x</i> & <i>y</i>)
Smooth concrete floor	0.277	0.101	0.388	0.150	0.216	0.155
Medium bed	0.281	0.016	0.363	0.133	0.187	0.127
Rough bed	0.663	0.121	0.759	0.203	0.292	0.237
Asphalt	0.162	-0.024	0.273	0.087	0.205	0.115

Comparing the average spray errors from the fixed spacing test in Table 4.39, the average spray errors of spraying an imaginary weed pattern were generally similar to those of displacement sensor except on the smooth concrete floor. The average spray errors on the soil surface in the x direction were 0.23 cm for the fixed spacing test and 0.28 cm for spraying an imaginary weed pattern. On the asphalt, the average spray errors were 0.15 cm for the fixed spacing test and 0.16 cm for spraying an imaginary weed pattern. However, on the smooth concrete floor, the average spraying error for spraying an imaginary weed pattern (0.28 cm) was more than twice the one for the fixed spacing test (0.12 cm). This might be because errors were measured for each imaginary weed location for spraying imaginary weed patterns whereas the error was measured based on only one spray drop location for the fixed spacing test.

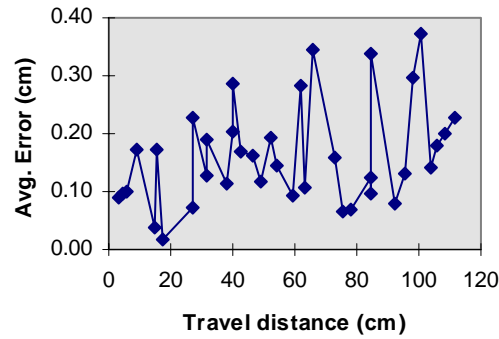
When analyzed using Method II, the level of errors were smaller than with Method I, but the errors still seemed to be accumulated along the travel distance (Figure 4.19 (a) and (c), and Figure 4.20 (a) and (c)).



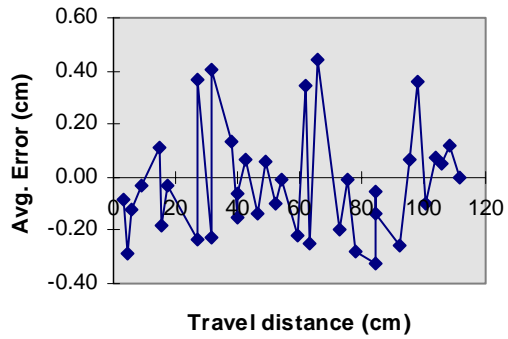
(a) Smooth concrete floor: x - direction.



(b) Smooth concrete floor: y - direction.

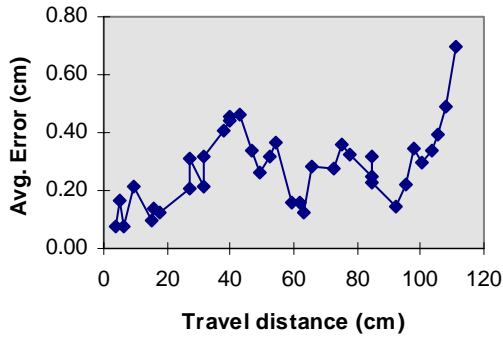


(c) Asphalt: x - direction.

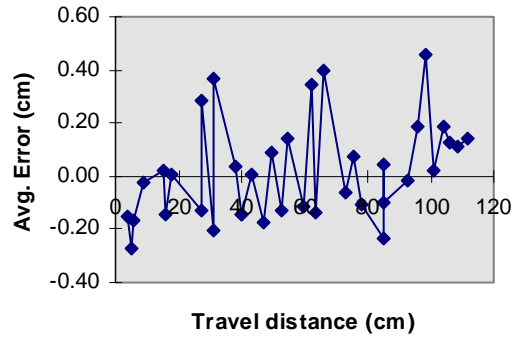


(d) Asphalt: y - direction.

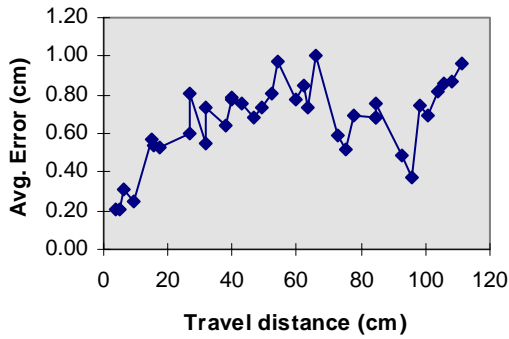
Figure 4.19 Average spray errors measured by Method II on concrete floor and asphalt.



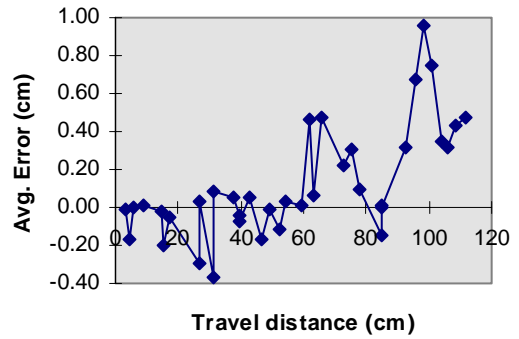
(a) Medium tomato bed: x - direction.



(b) Medium tomato bed: y - direction.



(c) Rough tomato bed: x - direction.



(d) Rough tomato bed: y - direction.

Figure 4.20 Average spray errors measured by Method II on medium and rough beds.

In order to see if the errors measured from 4 different surfaces linearly increased with the travel distance, a linear regression analysis was conducted with the travel distance as an independent variable and error as a dependent variable and a *t*-test was performed to see if the slope was significantly different from 0 with a significance level of 0.05. The results are listed in Table 4.42.

Table 4.42 Result of linear regression between the travel distance & errors and a *t*-test for the slope to see if it was significantly different from 0 (where NS: not significantly different and *: significantly different with a significance level of 0.05).

Surface	Term	Estimate	Std. error	<i>t</i> ratio	Prob. > <i>t</i>
Smooth concrete floor	Slope	0.000384	0.000758	0.51	0.6155 ^{NS}
	Intercept	0.255	0.0496	5.14	< 0.0001*
Asphalt	Slope	0.000891	0.000415	2.14	0.0391*
	Intercept	0.112	0.027225	4.11	0.0002*
Medium bed	Slope	0.00193	0.00059	3.26	0.0025*
	Intercept	0.172	0.038684	4.44	< 0.0001*
Rough bed	Slope	0.00323	0.000877	3.68	0.0008*
	Intercept	0.481	0.057436	8.37	< 0.0001*

It turned out that only the slope of the errors from smooth concrete floor was not significantly different from 0 and the slopes from other three surfaces were significantly different from 0. Thus, the errors did not accumulate on the smooth concrete surfaces, but linearly increased on asphalt, medium bed, and rough beds along the travel distance.

4.3.3 Spray targeting with imaging on different ground surfaces

In order to estimate the spray accuracy of the prototype system when both imaging and spraying were conducted, five tests were conducted, three outdoors and two indoors. Test results of the precision spraying system with green targets on different ground surfaces are shown in Table 4.43. On tomato beds, the average errors between the center of the targets and the spray patterns were 1.36 cm for the abrasive cleaning material and 0.66 cm for the metal targets and the standard deviations were 0.71 cm and 0.49 cm respectively. The larger error and standard deviation of abrasive cleaning material might be due to an error when the microcontroller parameters were set for spraying targets. It was also difficult to measure the error with the abrasive cleaning material because it absorbed the spray. On the paved road, the average spray error and the standard deviation of error were 0.79 cm and 0.52 cm respectively.

Table 4.43 Precision spraying system performance with imaging.

Surface	Tomato Bed	Tomato bed	Paved road	Concrete floor	Concrete floor
Target size	2.54 cm dia.	2.54 cm dia.	2.54 cm dia.	2.54 cm dia.	1.27 cm dia.
Target material	Green abrasive cleaning material	Green metal	Green paper	Green paper	Green paper
No. of targets	67	99	190	83	285
Avg. error (cm)	1.36	0.66	0.79	0.51	0.52
Std. Dev. of error (cm)	0.71	0.49	0.52	0.21	0.32

One difficulty in this test was to drive the tractor straight. The locations of the spray pattern relative to the center of the target (Figure 4.21) were recorded as shown in Figure 4.21 for the paved road and concrete floor tests. On the paved road, the spray struck the right side of the targets, indicating that the spraying system traveled to the right, not straight. When the traveling path of the precision spraying system was not same as the traveling path of the camera, the location supplied by the computer vision algorithm could not be accurately sprayed due to the difference in path.

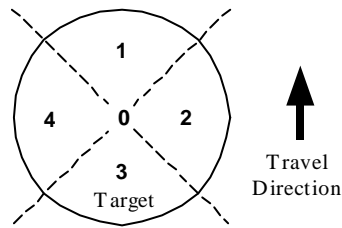


Figure 4.21 Direction of spray drops from the center of coin.

In hallway tests, the targets were sprayed a little later (direction of 1) with 2.54 cm diameter targets and a little earlier (direction of 3) with 1.27 cm diameter targets. This fact indicated that the encoder was not well calibrated.

On the tomato beds, cone wheels were used to guide the toolbar mounted the prototype system. On the paved road, however, there were no guides to keep the tractor straight. Thus, the average error and the standard deviation were a little higher on the paved road than on the tomato beds with metal targets. The average error and the standard deviation for the indoor concrete floor tests were the smallest of all tests. This indicates that the precision spraying system worked better on a smoother surface.

Table 4.44 Directions of spray from the precision spray system.

Surface	Paved road	Hallway	Hallway
Target size	2.54 cm dia.	2.54 cm dia.	1.27 cm dia.
Target material	Green paper	Green paper	Green paper
No. of targets	153	83	285
% of direction 0	7.8	0.0	2.5
% of direction 1	20.3	23.8	40.0
% of direction 2	48.4	26.2	17.5
% of direction 3	11.1	32.1	21.1
% of direction 4	12.4	16.7	18.9

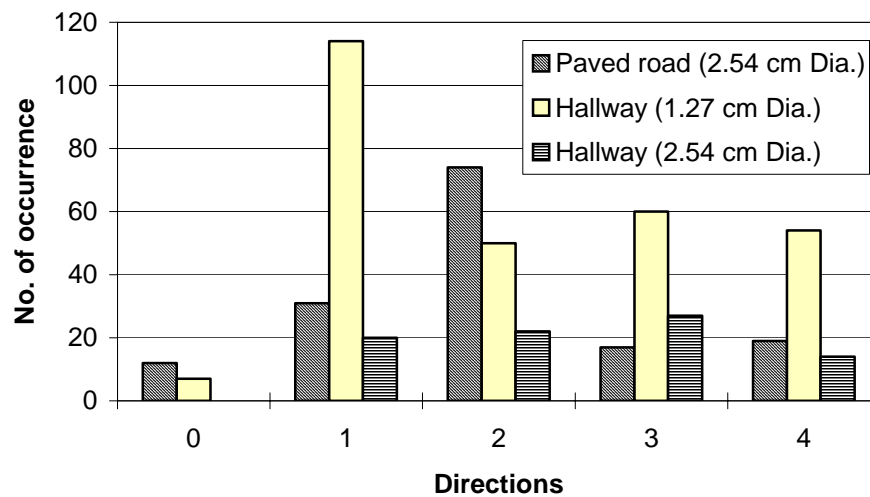


Figure 4.22 Directions of spray from the precision spray system.

The following is one example of imaging & spraying from the tests executed for the performance of the precision spraying system. This test was conducted with 1.27 cm diameter green paper coins in order to determine the effect of taking images on the performance of the precision spraying system. Figure 4.23 shows the errors in the x and y directions and indicates that the error in the travel direction (x -direction) did not increase with travel distance. The error did not accumulate even though the distance was measured by the incremental encoder (dead reckoning) over a distance of 120 cm. This

was because the maximum distance possible for error accumulation was only the length of 17.0 cm (about 1½ images) which was the distance from the camera to the nozzle. The average error and the standard deviation are shown in Table 4.45.

Table 4.45 Average and standard deviation of spraying error with imaging.

(unit: cm)	x - direction	y - direction	x & y direction
Avg. error	0.321	0.389	0.547
Std. Dev. of error	0.219	0.236	0.240

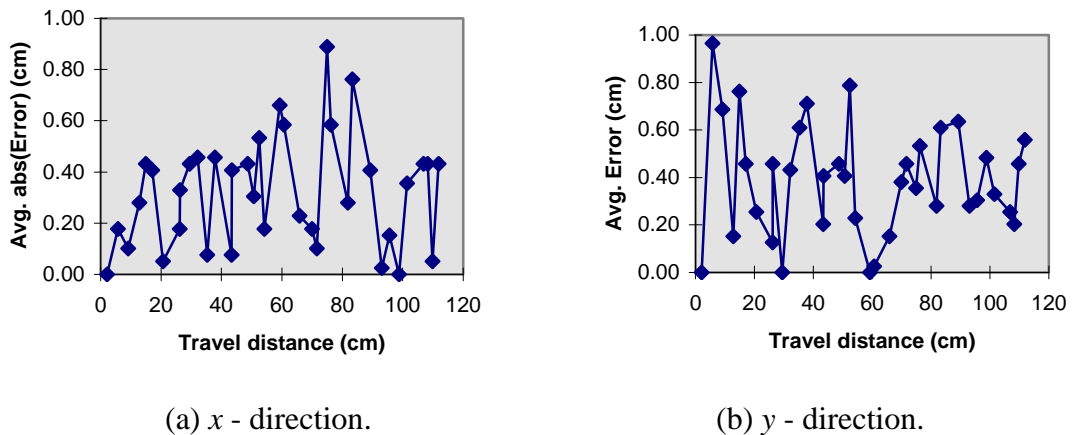


Figure 4.23 Spraying errors with imaging.

Indoor tests of the prototype system with rectangular (“tomato cotyledon”) and circular (“weeds”) targets

The precision spraying system was tested indoors on a smooth concrete floor with rectangular targets (“tomatoes”) and circular targets (“weeds”) made of green paper in order to demonstrate the performance in ideal indoor conditions, and results are summarized in Table 4.46.

Table 4.46 Test results of precision spraying system on concrete floor.

Trial	Travel distance (m)	No. of rectangles (cotyledon)	No. of rectangles sprayed	No. of circles (weed)	No. of circles not sprayed
1	1.14	23	0	26	0
2	1.14	6	4	39	0
3	1.14	22	0	19	0
Total	3.42	51	4	84	0
Total error rate			7.8 %		0 %

All circular targets were sprayed correctly and four of the rectangular targets were sprayed incorrectly due to LUT error. This was due to the fact that the aperture of the camera was not open sufficiently and the image was too dark, thus the prototype could not recognize the rectangular shape in the binary image. The overall error for the rectangular targets was 7.8% and there was no error in spraying the circular targets. Except for the error from the LUT, the prototype worked well on a smooth surface with distinct well separated objects.

Comparison of spraying-only test with imaging & spraying test

Comparing the errors between the two different indoor tests of spraying-only and imaging & spraying tests, the error level from the encoder should be the same since the tests were conducted indoors on the same smooth concrete surface, i.e., the effect of the encoder remained same during the tests. The average error on the concrete floor from the spraying-only test (0.388 cm in Table 4.41) was a little smaller than the one from imaging & spraying tests (0.51 cm in Table 4.43). The standard deviations of error between the two tests were about same magnitude (0.21 cm for imaging & spraying and 0.155 cm for spraying-only). The main difference between these two tests was the acquisition and processing of images. The increase in error may be due to inaccuracies in knowing exactly when the image was acquired or in synchronizing the imaging acquisition with the spray microcontroller.

In the outdoor tests, many sources of error were observed. First of all, the displacement sensor did not always work consistently (i.e., it did not generate the same number of the pulses for the same distance). There is also an intrinsic error due to the physical distance between the nozzle centers (1.27 cm). If a target is located in the corner of a spray cell, the spray error (the distance between the center of the target and the spray drop) could be 7.10 mm even though the computer vision system correctly identifies the center of the coin, since the spray nozzle would target the spray at the center of the cell.

The process of sending data to the microcontroller and spraying were same for those tests. The difference in error might come from the prototype software, which took images and processed them. The types of errors from the prototype software might be the synchronization of communication between the image processing computer and the

microcontroller, or any possible image acquisition delays. Other sources of error might be different test settings (inappropriate parameters for the microcontroller, or calibration error) and environment conditions (roughness of soil surface, tire pressure) between the tests.

Comparing the test results between indoor test and outdoor test, the average difference of indoor tests between imaging & spraying and spraying-only was smaller than the one from outdoor tests. This signifies that the prototype could work more consistently on a smoother surface regardless of imaging & spraying or spraying-only tasks.

4.4 Speed of the prototype system and field testing

Speed of the image processing algorithm

Processing time is a major concern in real-time machine vision applications. Since the goal of this research was to develop a real-time robotic weed control system, computationally intensive steps were avoided. Table 4.47 shows the average execution times for each image processing step. For one frame of a 256 by 240 pixel image representing a 11.43 cm x 10.16 cm field of view, the image processing algorithm took 0.344 s to distinguish 10 tomato cotyledons in the image using only the features of ELG and CMP. Using this algorithm the prototype cultivator could travel at a continuous rate of 1.20 km/h.

Table 4.47 Execution time for each image processing step.

Image processing step	Execution time (ms)	Percent of Total time
Prepare image acquisition	0.02	0.01
Acquire color image (one field)	16.76	4.87
Transfer and subsample acquired image	27.21	7.91
Check synchronization of image processing computer and spray controller	2.08	0.60
Check image buffer overflow	1.19	0.35
Binarize	2.92	0.85
Morphology analysis	32.04	9.31
Label objects	9.89	2.87
Extract features	144.58	42.00
Make decision with a Bayesian classifier	0.94	0.27
Find tomato & weed locations	58.10	16.88
Send tomato & weed locations to spray controller	37.44	10.88
Miscellaneous commands	11.03	3.20
Total time	344.20	100.00

Higher travel speed could be achieved simply by dedicating more image processing units to extract the morphological features from the leaves in parallel, since the execution time was dependent on the number of objects in an image and feature extraction took about 42% of the time to process one image, Table 4.47.

Field testing of the prototype system

The results of a test of the prototype robotic weed control system performed in a commercial processing tomato field in northern California in early May 1997 are shown in Table 4.48. The travel speed was about 0.8 km/hr and the tomato plants ranged from just emerging up to the first true leaf stage. The total number of tomato plants and weeds in this field test were 520 and 21 respectively.

Table 4.48 Field test results of the robotic weed control system.

Trial	Travel distance (m)	No. of tomatoes	No. of tomatoes sprayed	No. of weeds	No. of weeds not sprayed
1	7.62	80	18	1	0
2	7.62	119	32	2	2
3	7.62	106	27	4	2
4	7.62	118	29	2	2
5	7.62	97	20	12	5
Total	38.10	520	126	21	11
Total error rate			24.2 %		52.4 %

When implementing the prototype robotic weed control system, serial communication between the image processing computer and the spray microcontroller was a problem, especially when the tractor travel speed was faster than the image processing speed. For this reason, the valve control subroutine was modified as described

in Chapter 3.6.4 such that some of the input images would not be processed to keep up with the tractor speed when the tractor was moving faster than the image processing speed. However, few images were skipped since most of the input images had 10 leaves or less in them.

The overall system results from the field test conducted in a row where most of the tomato plants were in the cotyledon stage show that about 24% of the tomatoes were sprayed and about 52% of the weeds were not sprayed. The low percentage of weeds sprayed was due to several factors.

Some weeds were near tomatoes, so they were not sprayed due to the protection zone around the tomato leaves. There were some grass weeds, which looked very similar to tomato cotyledons. Some weeds may have been outside of the camera's field of view, but they were counted as processed objects since it was very difficult to determine the exact boundary of the camera's view. In addition, some tomato plants were hidden by weeds, so they were identified as weeds and sprayed. Figure 4.24 shows some of the successful precision spray application to weeds by the robotic weed control system.



Figure 4.24 Successful precision application of spray on weeds by the robotic weed control system.

There were difficulties in acquiring images out in the fields. Even with the uniform illumination device there were some difficulties, including very bright sun-light, wind, and diurnal changes of plant appearance. It was not very easy to adjust the focus and aperture of the camera by looking at the computer monitor outdoors due to very bright sunlight causing glare on the monitor. The quality of some images, which were considered focused and well illuminated out in the field, were not good when they were examined afterwards in the lab. Wind also had great impact on imaging and plant recognition performance. It was an especially windy spring in northern California in 1997. Most of the plants were lying down along the direction of wind, and in many cases it was impossible for the prototype system to recognize plants from a single top view.

Real-time implementation of morphological pattern recognition has a trade-off with accuracy using the machine vision technology described in this study. Computationally intensive steps could not be used due to time constraints even though they would produce more accurate results. A segmentation method for partially occluded objects would greatly increase the plant recognition rate, but it could not be used if it was very computationally intensive and not one of the built-in functions of the Sharp image processing boards. Tomato plants larger and older than the cotyledon stage have more complex shape characteristics and require more computation steps in order to be recognized. The displacement measurement was also a great factor in real-time implementation since timing for image acquisition and valve control were done based on travel distance. Bright sun-light would be less of a problem if some kind of shadow such as a tarp is used.

4.5 Recognition of transgenic purple tomato plant

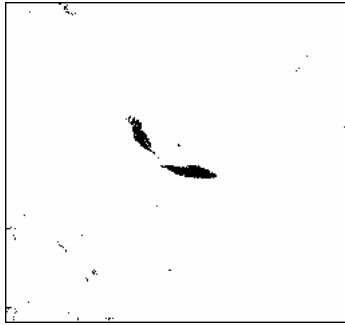
To overcome some of the difficulties of using leaf morphology to recognize tomato plants especially when the plant leaves are overlapped or are at a larger growth stage than cotyledons, the feasibility of identifying tomato plants by their color alone was investigated using tomato plants with “purple” foliage.

The following figure shows an example of the purple tomato processing algorithm. Fig. 4.25 (d) is the result of one 4 connected erosion of the binary image (b), and (e) is the result of a delete-isolated points operation on image (c). Fig. 4.25 (g) shows the raw color image overlaid with spray cells to be sprayed and protected. The red cells are to be sprayed, the green cells are the center of the purple tomato leaves, which are not sprayed and protected, and the blue cells are the buffer zone to protect purple tomato plants from spray drift.

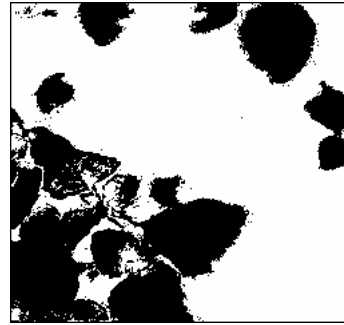
Table 4.49 shows the processing time for the transgenic purple tomato plant recognition algorithm. In this implementation, image acquisition was the most time consuming operation, which took 33.9% of the total processing time (16.76 ms for acquiring one field, and 27.21 ms for transferring and subsampling by column). The next most time consuming step was to send plant locations to the microcontroller (28.9%). In this trial, a baud rate of 9600 was used, however if the baud rate is increased, more time could be saved.



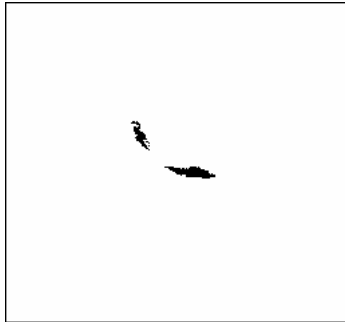
(a) Raw color image of a transgenic purple tomato plant and weeds.



(b) Raw binary image for purple plants.



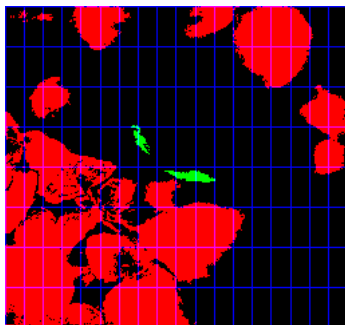
(c) Raw binary image for green weeds.



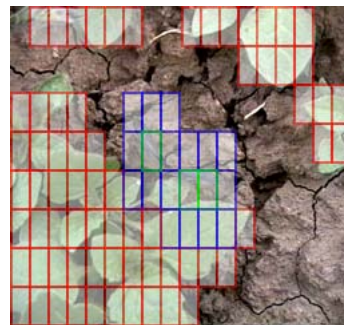
(d) Processed purple binary image.



(e) Processed green binary image.



(f) Transgenic tomato plant recognized in green and weeds shown in red on a spray cell grids.



(g) Center of protection zone (□), protection zone (□), and spray zone (□) overlaid on original color image.

Figure 4.25 Example of transgenic purple tomato plant recognition.

Table 4.49 Execution time for each image processing step.

Image processing step	Execution time (ms)	Percent of total time
Prepare image acquisition	0.02	0.02
Acquire image (one field)	16.76	12.92
Transfer and subsample acquired image	27.21	20.98
Check synchronization of image processing computer and microcontroller	2.08	1.60
Check image buffer overflow	1.19	0.92
Binarize purple tomatoes	2.92	2.25
Pre-process purple binary image	3.20	2.47
Binarize green weeds	2.92	2.25
Pre-process weed binary image	3.20	2.47
Find cells to be sprayed and protected	21.72	16.75
Send tomato & weed locations to microcontroller	37.44	28.87
Miscellaneous commands	11.03	8.50
Total time	129.69	100.00

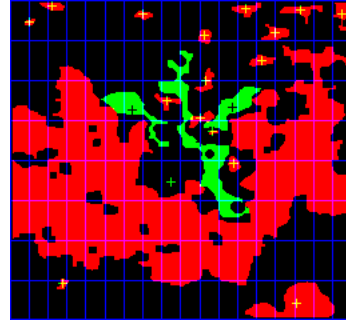
To find spray cells for tomato plants and weeds, only two points (instead of 5 as in the normal algorithm) in a spray cell were used to determine whether to spray the cell. To make the algorithm that determines which spray cells contain weeds smarter, the process of checking cells needs to be stopped as soon as a weed is detected in the cell or the starting point of the cells could be at the centroid of a weed in determining which cells to check in order to try and skip some cells. Compared with the results for the morphology based algorithm, only the image acquisition and sending valve array routines remained the same, Table 4.49. The whole process took 129.69 ms for a field of view of 11.43 cm by 10.16 cm, thus the prototype system could travel at speeds up to 3.17 km/hr.

Figure 4.26 shows example results for images containing purple tomato plants. The images in Figure 4.26 (a)-(d) illustrate very complex scenes, in which purple tomato plants were recognized correctly even though the tomato plants and weeds were highly

overlapped. Figure 4.26 (e) - (f) shows an example of a weed growing underneath of a tomato plant. In this example, the tomato cotyledon was damaged but was still recognized correctly without any difficulty even though the leaf pattern was distorted, while Figure 3.22 (b) showed a similar image with a normal green tomato that could not be recognized by the leaf morphology algorithm.



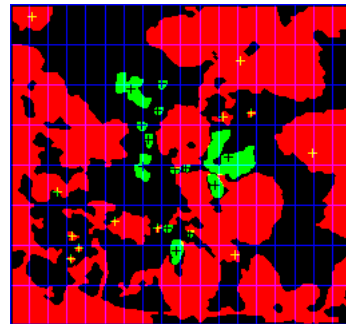
(a) Raw color image.



(b) Recognition result.



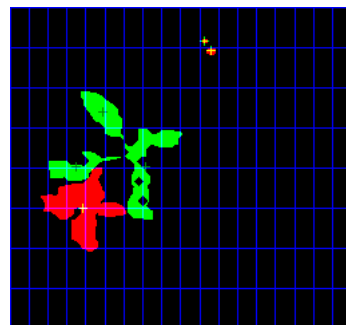
(c) Raw color image.



(d) Recognition result.



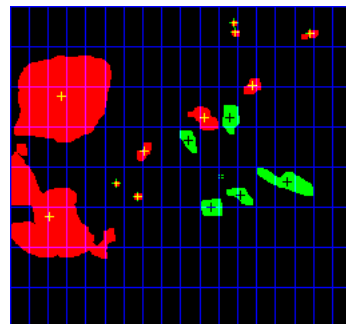
(e) Raw color image.



(f) Recognition result.



(g) Raw color image.



(h) Recognition result.

Figure 4.26 Example scenes showing transgenic purple tomato plant recognition. In (b), (d), (f), and (h) the tomato plants are shown in green and weeds in red.

5. SUMMARY AND CONCLUSIONS

5.1 Real-time prototype system

A *real-time* intelligent robotic weed control system was developed and tested for non-occluded plant leaves in commercial processing tomato fields for selective spraying of in-row weeds using a machine vision system and a precision chemical application system. The machine vision system was composed of the SHARP image processing boards, a color video camera, a multifunction I/O board, a Pentium Pro 200 MHz CPU, and plant recognition algorithms. The precision chemical application system was composed of a microcontroller, a manifold, a specially designed accumulator, 8 solenoid valves and micro-spray nozzles, valve control circuits, a valve control software. A Bayesian decision rule and a real-time LUT conversion board (AUXLUT card) were utilized in segmenting color images and thus made real-time implementation possible.

The image processing algorithm took 0.344 s to identify 10 tomato cotyledons in the image using only the features of ELG and CMP, for one frame of a 256 by 240 pixel image representing a 11.43 cm x 10.16 cm field of view. Thus, the prototype cultivator could travel at a continuous rate of 1.20 km/h.

The prototype robotic weed control system was tested in commercial tomato fields in northern California from March to May 1997. Overall the robotic weed control system correctly identified and did not spray 75.8% of the tomato plants and correctly sprayed 47.6% of the weeds.

5.2 Performance of precision chemical application system

Tests were conducted in order to observe the operation of the displacement sensor alone on different ground surfaces (soil surface, paved road, and smooth concrete surface). The overall mean error of the precision spraying system without any image processing operation was 0.15 cm and the standard deviation was 0.19 cm.

In order to isolate problems within the scope of the precision spraying system, the prototype was set to spray a predetermined “imaginary” weed pattern without taking any images and with no targets on the ground using four types of ground. The errors were measured based on the relative position at the time of spraying. The average error in the travel direction increased in the following surface order: asphalt → smooth concrete surface → medium bed → rough bed, ranging from 0.162 cm to 0.663 cm. The standard deviation of the error was ranged from 0.087 cm to 0.203 cm.

In order to estimate the spray accuracy of the prototype system, five tests were conducted with real targets, three outdoors and two indoors. In these tests both imaging and spraying operations were conducted. On tomato beds, the average errors between the center of the targets and the spray patterns were 1.36 cm for the abrasive cleaning material and 0.66 cm for the metal targets and the standard deviations were 0.71 cm and 0.49 cm respectively. On the paved road, the average error and the standard deviation of error were 0.79 cm and 0.52 cm respectively. On concrete floor, the average error and the standard deviation of error were 0.51 ~ 0.52 cm and 0.21 ~ 0.32 cm respectively.

The average error on the concrete floor error from the spraying-only test (0.39 cm) was a little smaller than the one from imaging & spraying tests (0.51 cm). The standard deviations of error between the two tests were about same magnitude (0.21 cm

for imaging & spraying and 0.16 cm for spraying-only). Comparing the outdoor tests on the tomato bed, paved road and asphalt surfaces, the average error from the spraying-only tests was smaller than that from the imaging & spraying tests.

Comparing the test results between the indoor test and the outdoor test, the average difference of indoor tests between imaging & spraying and spraying-only was smaller than the one from outdoor tests. This signifies that the prototype could work more consistently on a smoother surface regardless of imaging & spraying or spraying-only tasks.

5.3 LUT performance

The LUTs made in *HSI* color space were generally better than those made in *rgb* color space and *RGB* color space and the LUTs built in *rgb* color space were better than those in *RGB* color space. In *HSI* color space, the LUTs built using only the *H* component (LutHC3 and LutHC2) were better than those made with *H* and *S* components (LutHSC2 and LutHSC3) and those made with *H*, *S*, and *I* components (LutHSIC2 and LutHSIC3). Adding the *S* and the *I* components to the *H* component did not produce better results in LUT performance. Overall, LUT's built only with hue gave the best performance, correctly classifying 77.8% of color pixels.

There were three types of noise in the background: single point, small cluster, and big cluster. These types of noise were easily removed in most situations. Thus, the performance of a LUT was evaluated by the plant error rate. The execution time for removing background noise was directly proportional to the noise in the segmented image, however several steps (over 10 ms) of removing background noise could be saved

if there was less noise in the segmented image. The LUTs were also evaluated with single point noise (isolated pixels) removed from the binary images. However, the results were similar to the previous ones.

5.4 Plant recognition performance

All images were divided into two groups of good and bad image quality based on the focus, camera aperture, wind, cotyledon opening, state of maturity, and occlusion. From each group, a training set and a validation set were created in order to estimate the plant recognition performance by the image processing algorithm. For the good image group, a total of 117 images were used in the training set and 157 images were used in the validation set. For the bad group, a total of 129 images and 133 images were used respectively.

Three methods were used to select the best feature subsets for real-time identification of tomato plants and weeds. The first method was to use canonical discriminant analysis and principal component analysis and the subset [AVGABSC, ATL, ATP, PTB, CMP, ELG, HET, MNX, MTM, NEG, OCCR, PTC, and PTP] was selected as the best set. The classification rates for the validation data set were 98.0% for tomato cotyledons for the good image group and 92.7% for the bad image group. The classification rates for the tomato true leaves were 52.5% for the good image group and 60.4% for the bad image group. The identification rate for the weed class in the validation data set was 77.2% for the good image group and 45.2% for the bad image group.

The second method was to try to reduce the number of features to the most important subset for real-time field by correlation analysis use and linear regression model selection based on the R^2 criteria. By this method, the set [MNX, CMP, LTP, and OCCR] was selected as the best set. The calculation time for this set was 64.1 ms. This feature subset correctly classified 87.3% of tomato cotyledons, 38.6% of tomato true leaves, and 94.6% of weeds in the good image group of the validation data set, and 87.8% of tomato cotyledons, 9.9% of tomato true leaves and 72.0% of weeds in the image bad group of the validation data set.

The final method selected the set [AREA, LTP and OCCR] by trial and error. This set was also selected as the final best set for real-time field use. The calculation time for this feature subset was 43.6 ms, allowing a travel speed of 1.21 km/h. In the final validation tests with all 4 classes, this best subset correctly classified 80.7% of tomato cotyledons, 21.2% of tomato true leaves, and 95.5% of weeds in the good image group of the validation data set and 58.5% of tomato cotyledons, 9.0% of true leaves, and 93.0% of weeds in the bad image group of the validation data set. The third group of tomato plant leaves, which were curled, occluded, eaten by bugs, and partially hidden by the edge of the image, was introduced and classified with the best feature subset. This feature set also correctly classified 12.9% of the tomato third group in the good image group and 8.9% in the bad image group of the validation data sets.

5.5 Separation of touching leaves: Watershed method

Occlusion has been one of the most difficult obstacles in machine vision recognition of outdoor scenes since occluded objects are difficult to identify. Multiple

occluded objects appear as one object in the segmented binary image, producing an unusual set of feature values. In order to overcome these difficulties, the watershed algorithm was implemented to cut apart the occluded leaves, as a preprocessing step to improve object identification. Since the original watershed algorithm did not work very well and tended to produce excessive cutting, five different modifications were implemented to reduce the over cutting problem: modification with an *opening* operation (W1), modification with pre-flooding (W2), modification with a feature criteria (W3), modification with a concavity criteria (W4), and modification with a combined *opening* and feature criteria (W5).

The performance of W1, W3 and W5 were a lot better (53.3% improvement for W1 and 46.7% improvement for W3 & W5) than W2 and W4 modification (6.7% - 26.7% improvement). In the pre-flooding technique, the pre-flooding levels of 252 and 253 were a little better than the level of 251. However, the execution time for W3 and W4 modifications (14.6 - 14.8 s) were the fastest among the modifications. The W1 modification took the most time (19.7 s).

In order to estimate the impact of the watershed algorithm on plant classification rate, 12 field images were selected randomly and the modification of feature criteria and concavity criteria with pre-flooding up to 252 was applied to the objects. There was a 33.3% improvement for occluded tomato cotyledons in classification rate, and 41.1% improvement for occluded tomato true leaves. Some of the occluded leaves in the third group of tomato plants might also be benefited from these modifications.

5.6 Diurnal changes in plant appearance

A study was designed to evaluate varietal effects on cotyledon orientation using sixteen tomato varieties. The results indicate that the cotyledon angle was significantly different between two different days (that were 3.5°C different in the maximum daily temperature) - qualify the difference between the days at the same time of day, but the mean difference was not practically meaningful. The cotyledon angle of some varieties did not change from morning to dusk, but started to close at dusk, becoming completely closed at night. The variety effect on cotyledon angle had a lot of variability in the data set. The maximum average daily difference of cotyledon angle among varieties was 13.7°. At night, some varieties closed completely, but some didn't. The maximum average difference of cotyledon opening among varieties was 17.3° at 8 PM. The critical angle of recognition was estimated as 27.5° for a tomato cotyledon.

5.7 Recognition of transgenic purple tomato plant

To overcome some of the difficulties of the leaf morphology recognition algorithm in recognizing tomato plants when the leaves are overlapped or are at larger growth stage than cotyledons, the feasibility of identifying tomato plants by their color alone was investigated using tomato plants with "purple" foliage. This test showed quite promising results. In this test, purple tomato plants were correctly recognized even though the tomato plants were growing in the middle of a group of weed plants. This technique of using color alone could make it possible to use a 2-D image in a situation where if morphology were used would probably require 3-D imaging. The whole

recognition process took 129.69 ms for a field of view of 11.43 cm by 10.16 cm, thus the prototype system could travel at speeds up to 3.17 km/hr. Comparing the current conventional cultivating speed (~ 3.2 km/hr) in processing tomato farming, this technique is very encouraging as a future weed control method.

6. RECOMMENDATIONS FOR FUTURE WORK

For better recognition of tomato true leaves and the tomato third group, machine vision systems need to be able to identify features in a similar way as humans identify them. A person could distinguish tomato plants from weeds perfectly and very easily if he/she is trained even for a very short period, regardless of tomato plants being occluded, curled, lying down, bigger than cotyledon stage or eaten by insects. A possible way to increase the recognition performance is to incorporate a side view with the top view. This option could provide more information about plant characteristics and thus would lead to more correct recognition of tomato plants and weeds. In this research, the images were taken from 13 different fields in two years for the recognition of tomato plants and weeds, which brought in a lot of variability in plant shape. In some sense, the author believes that this great variability in plant shape may lower the recognition performance of tomato true leaves and the third group. Thus, it is recommended that it would be better to work with a smaller set of images (for example, images taken in one field), even though the recognition process would need to be repeated from field to field or for different times. This option would be possible if the feature selection procedure is automated.

For better displacement sensing, another technique needs to be developed instead of using wheel-driven encoders. For example, Stone and Kranzler (1992) proposed an image-based ground speed measuring system using a linear imaging device, which showed promising results. Occlusion of plant leaves is also a significant problem in distinguishing tomato leaves in the uncontrolled outdoor environment of an agricultural

field and an accurate real-time algorithm for separating partially occluded leaves at the point of occlusion is needed.

REFERENCES

- Ahmad, I. S., J. F. Reid and M. R. Paulsen.** 1996. Textural feature extraction of fungal-damaged soybean images. ASAE Paper No. 96-3047, St. Joseph, MI, USA.
- Alchanatis, L. and S. W. Searcy.** 1995. High speed inspection of carrots with a pipelined image processing system. ASAE Paper No. 95-3170, St. Joseph, MI, USA.
- Al-Jonobi, A. A. and G. Kranzler.** 1994. Machine vision inspection of date fruits. ASAE Paper No. 94-3575, St. Joseph, MI, USA.
- Anderson, N. W. and K. A. Wendorf.** 1993. Analyzing plant images in the HLS color space. ASAE Paper No. 93-3593, St. Joseph, MI, USA.
- Asada, H. and M. Brady.** 1986. The curvature primal sketch. IEEE Transactions on pattern analysis and machine intelligence, PAMI-8(1): 2-14.
- Ballard, D. H. and C. M. Brown.** 1982. *Computer vision*. Englewood Cliffs, NJ, Prentice-Hall, Inc.
- Bers, L.** 1969. *Calculus*. Vol. 2. Holt, Rinehart and Winston, Inc.
- Beucher, S.** 1998. Personal communication. Scientific staff. Centre de Morphologie Math—matique, Ecole des Mines, France.
- Beucher, S.** 1990. Segmentation tools in mathematical morphology. Proc. of SPIE in Image algebra and morphological image processing, 1350: 70-84.

- Beucher, S. and F. Meyer.** 1993. Chapter 12. The morphological approach to segmentation: The watershed transformation. In *Mathematical morphology in image processing*. Edited by Dougherty, E. Marcel Dekker, Inc., New York.
- Bond, W.** 1992. Non-chemical approaches to weed control in horticulture. *Phytoparasitica*. Israel journal of plant protection science. 20(Supplement): 77S-81S.
- Borenstein, J., Everett, H. R. and L. Feng.** 1996. *Navigating mobile robots*. A. K. Peters, Ltd., Wellesley, MA.
- Brivot, R. and J. A. Marchant.** 1996 Segmentation of plants and weeds for a precision crop protection robot using infrared images. *IEE Proc.- Vision, Image and Signal Process.* 143(2):118-124.
- Burks, T. and S. A. Shearer.** 1995. Ground cover classification in row crop images using color co-occurrence texture analysis. ASAE Paper No. 95-3569, St. Joseph, MI, USA.
- California Weed Conference.** 1989. *Principles of weed control in California*. 2nd. ed. Thompson Publication.
- Casent, D., A. Talukder, W. Cox, H-T. Chang, and D. Weber.** 1996. Detection and segmentation of multiple touching product inspection items. *Proc. SPIE in Optics in Agriculture, Forestry, and Biological Processing II*, 2907: 205-216.
- Cooperative Extension Service.** 1995. Non-chemical weed control methods. Bulletin 1118. The University of Georgia College of Agricultural and Environmental Sciences, Athens.

- Crowe, T. G. and M. J. Delwiche.** 1996a. Real-time detection of fruit-Part I: Design concepts and development of prototype hardware. *Transactions of the ASAE.* 39(6): 2299-2308.
- Crowe, T. G. and M. J. Delwiche.** 1996b. Real-time detection of fruit-Part II: An algorithm and performance of a prototype system. *Transactions of the ASAE.* 39(6): 2309-2317.
- Digabel, H. and C. Lantu—joul.** 1978. Iterative algorithms. *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine, Caen, France, Oct. 1977, J. L. Chermant, Ed. Stuttgart, West Germany: Riederer Verlag, 1978: 85-99.*
- Dougherty, E. R.** 1993. *Mathematical morphology in image processing.* Edited by E. R. Dougherty. Marcel Dekker, Inc.
- Dougherty, E. R.** 1992. *An introduction to morphological image processing.* SPIE, Bellingham, Washington, USA.
- Duda, R. O. and P. E. Hart.** 1973. *Pattern classification and scene analysis.* John Wiley & Sons.
- Eggleston, P.** 1995. Scientific image analysis development tools. *Advanced Imaging.* October: 48-53.
- Felton, W. L., A. F. Doss, P. G. Nash, and K. R. McCloy.** 1991. A microprocessor controlled technology to selectively spot spray weeds. *Proc. of Symposium on automated agriculture for the 21st century.* ASAE: 427-432, St. Joseph, MI, USA.
- Felton, W. L. and K. R. McCloy.** 1992. Spot spraying. *Agricultural Engineering,* November: 9-12.

- Ferguson, W. and A. Padula.** 1994. Economic effects of banning methyl bromide for soil fumigation. Resources and Technology Division, Economic Research Service, U.S. Department of Agriculture. Agricultural Economic Report Number 677: 1-11.
- Ford, G.** 1994. EEC207: Pattern recognition and classification (Lecture Notes). Department of Electrical Engineering and Computer Science, University of California, Davis.
- Franz, E., M., R. Gebhardt, and K. B. Unklesbay.** 1991a. Shape description of completely visible and partially occluded leaves for identifying plants in digital images. Transactions of the ASAE 34(2):673-681.
- Franz, E., M., R. Gebhardt, and K. B. Unklesbay.** 1991b. The use of local spectral properties of leaves as an aid for identifying weed seedlings in digital images. Transactions of the ASAE 34(2): 682-687.
- Gao, X., J. Tan and D. Gerrard.** 1995. Image segmentation in 3-Dimensional color space. ASAE Paper No. 953607, St. Joseph, MI, USA.
- Goldsbrough, A. P., Y. Tong, and J. I. Yoder.** 1996. *Lc* as a non-destructive visual reporter and transposition excision marker gene for tomato. *The Plant Journal*, 9(6): 927-933.
- Gonzalez, R. C. and R. E. Woods.** 1993. *Digital image processing*. Addison-Wesley publishing company, Inc.
- Guyer, D. E., G. E. Miles, M. M. Schreiber, O. R. Mitchell, and V. C. Vanderbilt.** 1986. Machine vision and image processing for plant identification. Transactions of the ASAE 29(6):1500-1507.

- Haggar, R. J., C. J. Stent, and S. Isaac.** A prototype hand-held patch sprayer for killing weeds, activated by spectral differences in crop/weed canopies. *J. Ag. Eng. Res.* 28: 349-358, (1983).
- Haney, L., C. Precetti, and H. Gibson.** 1994. Color matching of wood with a real-time machine vision system. ASAE Paper No. 94-3579, St. Joseph, MI, USA.
- Hooper, A. W., G. O. Harries, and B. Ambler.** 1976. A photoelectric sensor for distinguishing between plant material and soil. *J. Ag. Eng. Res.* 21: 145-155.
- Horn, B. K. P.** 1992. *Robot vision.* The MIT Press.
- Horsch, R. B., R. T. Fraley, S. G. Rogers, P. R. Sanders, A. Lloyd, and H. Hoffman.** 1984. Inheritance of functional foreign genes in plants. *Science*, 223: 496-498.
- Isaac, E. J. and R. C. Singleton.** 1956. Sorting by address calculation. *Journal of the association for computing machinery*, 3: 169-174.
- Jain, A. K.** 1989. *Fundamentals of digital image processing.* Prentice-Hall, Inc.
- Jia, J., G. W. Krutz, and H. G. Gibson.** 1990. Corn plant locating by image processing. *SPIE Optics in Agriculture* 1379:246-253.
- Jiang, C. and R. C. Derksen.** 1993. Morphological image processing for spray evaluation. ASAE Paper No. 933599, St. Joseph, MI, USA.
- Lan, Y., Q. Fang, M. F. Kocher, and M. A. Hanna.** 1996. Detection of fissures in grains using machine vision. ASAE Paper No. 96-3048, St. Joseph, MI, USA.
- Lass, L. W. and R. H. Callihan.** 1993. GPS and GIS for weed surveys and management. *Weed technology*, 7: 249-254.

- Lee, W. S., D. C. Slaughter, and D. K. Giles.** 1997. Robotic weed control system for tomatoes using machine vision and precision chemical application. ASAE Paper No. 97-3093, St. Joseph, MI, USA.
- Lester, J. M., H. A. Williams, B. A. Weintraub and J. F. Brenner.** 1978. Two graph searching techniques for boundary finding in white blood cell images. *Computers in biology and medicine* 8(4): 293-308.
- Liao, K., M. R. Paulsen and J. F. Reid.** 1994. Real-time detection of colour and surface defects of maize kernels using machine vision. *J. of Agricultural Engineering Research* 59(4):263-271.
- Luo, X., D. S. Jayas and N. R. Bulley.** 1997. Identification of damaged kernels in wheat using a color machine vision system. ASAE Paper No. 973099, St. Joseph, MI, USA.
- Marchant, J. A., C. M. Onyango, and M. J. Street.** 1990. Computer vision for potato inspection without singulation. *Computers and Electronics in Agriculture*. 4: pp. 235-244.
- Matthews, J.** 1996. Senate backs farmers' use of toxic pesticide. *The Sacramento Bee*. Feb. 23: pp. A1, A14.
- Merritt, S. J., G. E. Meyer, K. Von Bargaen, and D. A. Mortensen.** 1994. Reflectance sensor and control system for spot spraying. ASAE Paper No. 94-1057, St. Joseph, MI, USA.
- Meyer, F.** 1990. Skeletons and watershed lines in digital spaces. *Proc. of SPIE in Image algebra and morphological image processing*, 1350: 85-102.

- Meyer, F. and S. Beucher.** 1990. Morphological segmentation. *J. of visual communication and image representation*, 1(1): 21-46.
- Miller, B. K. and M. J. Delwiche.** 1991. Peach defect detection with machine vision. *Transactions of the ASAE* 34(6): 2588-2597.
- Moltó, E., J. Blasco, N. Aleixos, J. Carrión, and F. Juste.** 1997. A machine vision system for robotic weeding of row crops. *Proc. 5th Int. Symp. Fruit, Nut, and Vegetable Production Engineering*. Davis, CA., USA.
- Moltó, E., J. Blasco, N. Aleixos, J. Carrión, and F. Juste.** 1996. Machine vision discrimination of weeds in horticultural crops. *AGENG 96*. Madrid. Report N. 96G-037.
- Novini, A. R.** 1992. Fundamentals of machine vision inspection in glass and metal container industries. *Proceedings of the 1992 Conference, Food Processing Automation II*. ASAE, St. Joseph, Michigan.
- O’Gorman, L.** 1988. An analysis of feature detectability from curvature estimation. *Proceedings of Computer vision and pattern recognition*: 235-240.
- Orbert, C. L., E. W. Bengtsson, and B. G. Nordin.** 1993. “Watershed segmentation of binary images using distance transformations,” in *Nonlinear Image Processing IV*, E. R. Dougherty, J. Astola, H. G. Longbotham, Editors, *Proc. SPIE* 1902: 159-170.
- Parish, S.** 1990. A review of non-chemical weed control techniques. *Biological Agriculture and Horticulture*, 7:117-137.
- Prather, T. S. and R. H. Callihan.** 1993. Weed eradication using geographic information systems. *Weed technology*, 7: 265-269.

- Précetti, C. J. and G. W. Krutz.** 1993. Building a color classifier. ASAE Paper No. 93-3003, St. Joseph, MI, USA.
- Rencher, A. C.** 1995. *Methods of multivariate analysis*. John Wiley & Sons, Inc.
- Russ, J. C.** 1990. *Computer-assisted microscopy*. Plenum press, New York: 153-161.
- Sarkar, N. and R. R. Wolfe.** 1985. Feature extraction techniques for sorting tomatoes by computer vision. Transactions of the ASAE 28(3): 970-974, 979.
- SAS Institute Inc.** 1993. *SAS/STAT User's Guide*. Version. 6, 4th Ed. Vol. 1 & 2. Cary, NC.
- Sharp.** 1993. *Image processing board GPB-2 Manual*. Sharp digital information products, Inc.
- Shatadal, P., D. S. Jayas, and N. R. Bulley.** 1995. Digital image analysis for software separation and classification of touching grains: I. Disconnect algorithm. Transactions of the ASAE 38(2): 635-643.
- Shatadal, P., D. S. Jayas, and N. R. Bulley.** 1995. Digital image analysis for software separation and classification of touching grains: II. Classification. Transactions of the ASAE 38(2): 645-649.
- Shatadal, P., D. S. Jayas, and N. R. Bulley.** 1991. Fourier and spatial domain analysis of image texture. Automated Agriculture for the 21st Century. Proceedings of the National Symposium, Chicago, 16-17 December: pp. 36-41.
- Shearer, S. A. and R. G. Holmes.** 1990. Plant identification using color co-occurrence matrices. Transactions of the ASAE 33(6): 2037-2044.
- Shearer, S. A. and F. A. Payne.** 1990. Color and defect sorting of bell peppers using machine vision. Transactions of the ASAE 33(6): 2045-2050.

- Shiraishi, M., and H. Sumiya.** 1996. Plant identification from leaves using quasi-sensor fusion. *J. of Manufacturing Sci. and Eng., ASME* 118: 382-387.
- Slaughter, D. C., P. Chen and R. G. Curley.** 1997. Computer vision guidance system for precision cultivation. ASAE Paper No. 97-1079, St. Joseph, MI, USA.
- Slaughter, D. C., P. Chen, R. F. Norris and R. G. Curley.** 1996. Development of a robotic system for a non-chemical weed control. A research proposal submitted to UC IPM (University of California Integrated Pest Management) Program.
- Slaughter, D. C., R. Curley, P. Chen, and C. Brooks.** 1992. Development of a robotic system for non-chemical weed control. Proceeding 44th Annual California Weed Conference. Red Lion Hotel, Sacramento, CA.
- Slaughter, D. C. and R. C. Harrell.** 1989. Discriminating fruit for robotic harvest using color in natural outdoor scenes. *Transactions of the ASAE* 32(2): 757-763.
- Slaughter, D. C. and R. C. Harrell.** 1987. Color vision in robotic fruit harvesting. *Transactions of the ASAE* 30(4): 1144-1148.
- Soille, P. and L. Vincent.** 1990. Determining watersheds in digital pictures via flooding simulations. *Proc. of SPIE in Visual communications and image processing '90*, 1360: 240-250.
- Stafford, J. V., J. M. Le Bars and B. Ambler.** 1996. A hand-held data logger with integral GPS for producing weed maps by field walking. *Computers and electronics in agriculture*, 14: 235-247.
- Stone, M. L. and G. A. Kranzler.** 1992. Image-based ground velocity measurement. *Transactions of the ASAE* 35(5): 1729-1735.

- Tao, Y., C. T. Morrow, P. H. Heinemann, and H. J. Sommer III.** 1995. Fourier-based separation technique for shape grading of potatoes using machine vision. . Transactions of the ASAE 38(3): 949-957.
- Tarbell, K. and J. F. Reid.** 1989. Characterizing corn growth and development using computer vision. ASAE Paper No. 89-7509, St. Joseph, MI, USA.
- Tauzer, C. J.** 1995. *Development of a target activated offset sprayer using machine vision for plant detection.* M.S. Thesis. Department of Biological and Agricultural Engineering, University of California, Davis.
- Tauzer, C. J., K. Giles, and D. C. Slaughter.** 1994. Machine vision control of offset highway spraying. ASAE Paper No. 94-1507, St. Joseph, MI, USA.
- Thompson, J. F., J. V. Stafford, and B. Ambler.** 1990. Weed detection in cereal crops. ASAE Paper No. 907516, St. Joseph, MI, USA.
- Tian, L.** 1995. *Knowledge based machine vision system for outdoor plant identification.* Ph.D. dissertation. Department of Biological and Agricultural Engineering. University of California, Davis.
- Tian, L., M. Su, and E. Wassink.** 1997. Development of a machine-vision controlled precision sprayer. ASAE Paper No. 973055, St. Joseph, MI, USA.
- Tian, L. and D. C. Slaughter.** 1993. Computer vision identification of tomato seedlings in natural outdoor scenes. ASAE Paper No. 93-3608, St. Joseph, MI, USA.
- University of California Statewide Pest Management Project.** 1985. Integrated pest management for tomatoes. Second edition. Division of agriculture and natural resources, Publication 3274.

- USDA and NASS.** 1997. *Agricultural statistics 1997*. United States government printing office, Washington D. C.
- USDA, NASS and ERS.** 1995. *Agricultural chemical usage. Vegetables. 1994 Summary*. United States government printing office, Washington D. C.
- Vincent, L. and P. Soille.** 1991. Watersheds in digital space: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13(6):583-598.
- Vincent, L. and S. Beucher.** 1989. The morphological approach to segmentation: An introduction. School of Mines, Paris, France, Internal report. CMM.
- Visser, R. and A. J. M. Timmermans.** 1996. WEED-IT: a new selective weed control system. *Proceedings of SPIE, Optics in Agriculture, Forestry, and Biological Processing II* 2907: 120-129.
- Webster, T. M. and J. Cardina.** 1997. Accuracy of Global Positioning System (GPS) for weed mapping. *Weed technology*, 11: 782-786.
- Whittaker, A. D., G. E. Miles, O. R. Mitchell, and L. D. Gaultney.** 1987. Fruit location in a partially occluded image. *Transactions of the ASAE*. 30(3):591-596.
- Wilson, J. P., W. P. Inskeep, P. P. Rubright, D. Cooksey, J. S. Jacobson, and R. D. Snyder.** 1993. Coupling geographic information systems and models for weed control and groundwater protection. *Weed technology*, 7: 255-264.
- Woebbecke, D. M., G. E. Meyer, K. Von Bargen, and D. A. Mortensen.** 1995a. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the ASAE* 38(1): 259-269.

- Woebbecke, D. M., G. E. Meyer, K. Von Bargaen, and D. A. Mortensen.** 1995b. Shape features for identifying young weeds using image analysis. *Transactions of the ASAE* 38(1): 271-281.
- Worring, M. and A. W. M. Smeulders.** 1993. Digital curvature estimation. *CVGIP: Image understanding*, 58(3): 366-382.
- Zhang, N. and C. Chaisattapagon.** 1995. Effective criteria for weed identification in wheat fields using machine vision. *Transactions of the ASAE* 38(3): 965-974.

APPENDIX

Source code

- (1) Program to make a LUT (rgblut.c).
- (2) Program to make HSI pixel data file in ASCII with a computer mouse (hsipixel.c).

```

/*****
*
*   Filename: rgblut.c
*   Author   : Won Suk Lee
*   Date     : November 3, 1995
*
*****
*
*   This program generates LUT with the inputs of red (0-255),
*   green (0-255) and blue (0-255) using Bayesian classifier.
*   The red input is reduced to 5 bits (0-31) and the green input is
*   reduced to 6 bits(0-63), and blue input is reduced to 5 bits(0-31).
*   Then, the three inputs are converted into hue, saturation, and
*   intensity. These HSI values are used as an input to Bayesian
*   classifier.
*   This program calculates n possibilities for n classes and compare
*   those. Then if possibility of cotyledon or plant is biggest, the
*   LUT output is 255. Otherwise output is 0 (for background).
*
*****
*   Revision History
*   Date      Name      Description
*   12/8/97  WSL       Added to generate a file of address and r,g,b values.
*                   File name is "address.rgb".
*****/

#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <errno.h>
#include <fcntl.h>
#include <conio.h>
#include <sys\types.h>
#include <sys\stat.h>
#include "gpb2.h"
#include "elibdef.h"
#include "errcode2.h"
#include "windw2.h"
#include <math.h>

double bayesian(double *x, double *mu, long double *cov, int n);
double detr(long double *a);
long double minv(long double *mp, long *l, long *m, long n);
void diff(double *a, double *b, double *result, int row, int col);
void transpose(double *a, double *result, int row, int col);
void multiply(double *a, double *b, double *result, int row, int mid,
int col);
double get_max(double a, double b, double c, double d, double e, double
f, double g);

void main(void)
{
    long double a[3][3], b[3][3];
    long double x11,x12,x13,x21,x22,x23,x31,x32,x33;
    long l[3], m[3];
    int ii, jj;
    int red2, green2, blue2;
    double x[3][1],p_coty,p_true,p_weed,p_bak1,p_bak2,p_bak3,p_bak4,
p_max;
    double mu_coty[3][1],mu_true[3][1],mu_weed[3][1];

```



```

double mu_bak1[3][1],mu_bak2[3][1],mu_bak3[3][1],mu_bak4[3][1];
long double cov_coty[3][3],cov_true[3][3],cov_weed[3][3];
long double cov_bak1[3][3],cov_bak2[3][3],cov_bak3[3][3],
    cov_bak4[3][3];
long double cov_cotyx[3][3],cov_truex[3][3],cov_weedx[3][3];
long double cov_bak1x[3][3],cov_bak2x[3][3],cov_bak3x[3][3],
    cov_bak4x[3][3];
long double cov_cotyp[3][3],cov_truep[3][3]; // for purple tomatoes
double mu_cotyp[3][1],mu_truep[3][1];
long double cov_cotypx[3][3],cov_truepx[3][3];
double p_cotyp, p_truep;
FILE *fout;
char file_name[30], in_char;
int output;
double hue,intensity,saturation,red,green,blue,minrgb,factor;
unsigned long address;
unsigned char buffer[_FILE_BUFF_SIZE]; /* _FILE_BUFF_SIZE=512 in
    'elibdef.h' */

int fp,i,j,status; /* fp = file handle */
INTPTR p; /* pointer to image file header */
/* INTPTR is a far integer pointer defined in 'windw2.h'*/
int mm,class;
unsigned int ihue, isat, iint;
double coty, back;
FILE *outfp;
double std1, std2, std3; // std. deviation of HSI of cotyledon
double rhm, rhp, rsm, rsp, rim, rip;
char addr_file[]={"address.rgb"};
FILE *addr_ptr;

system("cls");

printf("Would you like to make a new LUT? (y/n)\n");
in_char = tolower( getch());
if( in_char == 'n')
    exit(0);
printf("Enter LUT image file name (*.img) :");
scanf("%s", file_name);
printf("Making LUT ... ");

/* make address file associated RGB value combination */
if((addr_ptr=fopen(addr_file,"w")) == NULL)
{
    printf("No %s FILE FOUND!\n", addr_file);
    exit(-1);
}

/* sample initial values for mean and covariance for each class */
/*****
/* lut516c2.img: For 516.img on 5/16/97 */
mu_coty[0][0]= 51.87202925; mu_coty[1][0]=      82.79798903;
mu_coty[2][0]=  114.57952468;
cov_coty[0][0]=  67.20629547L; cov_coty[0][1]=  18.11771236L;
cov_coty[0][2]=  31.76784038L;
cov_coty[1][0]=  18.11771236L; cov_coty[1][1]=  391.35469188L;
cov_coty[1][2]= -223.47363973L;
cov_coty[2][0]=  31.76784038L; cov_coty[2][1]= -223.47363973L;
cov_coty[2][2]=  393.90359520L;

```

```

std1=8.19794459; std2=19.78268667; std3=19.84700469;

mu_bak1[0][0]=      27.11690141; mu_bak1[1][0]=  82.47042254;
mu_bak1[2][0]=    111.79577465;
cov_bak1[0][0]=    53.46304257L; cov_bak1[0][1]=   51.93223544L;
cov_bak1[0][2]=   -3.44576770L;
cov_bak1[1][0]=    51.93223544L; cov_bak1[1][1]=  350.53156400L;
cov_bak1[1][2]=  -154.11535787L;
cov_bak1[2][0]=   -3.44576770L; cov_bak1[2][1]= -154.11535787L;
cov_bak1[2][2]=   606.32635730L;

/*****/

/* open output image file */
if((fp=open(file_name, O_WRONLY|O_CREAT|O_TRUNC, S_IREAD|S_IWRITE))==0)
{
    printf("error creating '%s'\n", file_name);
    exit(0);
}

/* write header data to image file */
/* _FILE_HEAD_SIZE=128 in 'elibdef.h' */
for (i = 0; i < _FILE_HEAD_SIZE; i++)
    buffer[i] = 0;

buffer[0] = _ID_CHAR; /* _ID_CHAR = 'M' in 'elibdef.h' */
p = (INTPTR)(buffer + 1);

*p++ = 0; /* cols */
*p++ = 0; /* rows */
*p++ = 512; /* width */
*p++ = 128; /* height */
*p++ = 8; /* pixel size 8 bits */
*p++ = TO_BOTTOM; /* scan direction is to the bottom, TO_BOTTOM = 0 in
'elibdef.h' */
*p++ = 1; /* number of planes/banks of data */

/* write header data to image file */
status = SUCCESS;
if (write(fp, buffer, _FILE_HEAD_SIZE) == _FILE_HEAD_SIZE)
{
    for (j = 0, address = 0; j < 128; j++)
    {
        for (i = 0; i < _FILE_BUFF_SIZE; i++, address++)
        {
            red = (double)( ((address << 17) >> 31) << 7)
                | ((address << 20) >> 31) << 6)
                | ((address << 23) >> 31) << 5)
                | ((address << 26) >> 31) << 4)
                | ((address << 29) >> 31) << 3) );
            green = (double)( ((address << 16) >> 31) << 7)
                | ((address << 19) >> 31) << 6)
                | ((address << 22) >> 31) << 5)
                | ((address << 25) >> 31) << 4)
                | ((address << 28) >> 31) << 3)
                | ((address << 31) >> 31) << 2) );
            blue = (double)( ((address << 18) >> 31) << 7)
                | ((address << 21) >> 31) << 6)
                | ((address << 24) >> 31) << 5)
                | ((address << 27) >> 31) << 4)
                | ((address << 30) >> 31) << 3) );

```

```

fprintf(addr_ptr,"%6ld, %7.1f, %7.1f, %7.1f,",\
        address, red,green,blue);
red = ((int)(red / 8.0)) * 8.0;
green = ((int)(green / 4.0)) * 4.0;
blue = ((int)(blue / 8.0)) * 8.0;

/* convert RGB into HSI */
/* intensity rounded */
intensity = ((red + green + blue) / 3.0 * 255.0/249.0) + 0.5;

/* saturation rounded */
minrgb = min(red, green);
minrgb = min(minrgb, blue);

if (intensity < 16.0)
    saturation = 0.0;
else
    saturation = (255.0 * (1.0 - (minrgb / intensity))) + 0.5;

/* hue rounded */
if ((saturation < 16.0) || (intensity < 16.0))
    hue = 0.0;
else
{
    if (blue > green)
        factor = 180.0;
    else
        factor = 0.0;

    hue = 90.0-(atan((2.0*red-green-blue)/(green- blue+0.01)
        /sqrt(3.0)))*(180.0/3.14)+factor+1.5;
}

hue = hue * 255.0/360.0;
x[0][0]=(double)((unsigned int)hue);
x[1][0]=(double)((unsigned int)saturation);
x[2][0]=(double)((unsigned int)intensity);

for(ii=0; ii<3; ii++)
    for(jj=0; jj<3; jj++)
    {
        /* Feed correct covariance value each time since those
        values are changed after 'bayesian' function. */
        cov_cotyx[ii][jj]=cov_coty[ii][jj];
        cov_truex[ii][jj]=cov_true[ii][jj];
        cov_weedx[ii][jj]=cov_weed[ii][jj];
        cov_bak1x[ii][jj]=cov_bak1[ii][jj];
        cov_bak2x[ii][jj]=cov_bak2[ii][jj];
        cov_bak3x[ii][jj]=cov_bak3[ii][jj];
        cov_bak4x[ii][jj]=cov_bak4[ii][jj];
    }
/* calculate probability of each class */
p_coty=bayesian((double *)x,(double *)mu_coty,
    (long double*)cov_cotyx,3);
//p_true=bayesian((double *)x,(double *)mu_true,
//    (long double *)cov_truex,3);
//p_weed=bayesian((double *)x,(double *)mu_weed,
//    (long double *)cov_weedx,3);
p_bak1=bayesian((double *)x,(double *)mu_bak1,
    (long double *)cov_bak1x,3);

```

```

//p_bak2=bayesian((double *)x,(double *)mu_bak2,
//                (long double *)cov_bak2x,3);
//p_bak3=bayesian((double *)x,(double *)mu_bak3,
//                (long double *)cov_bak3x,3);
//p_bak4=bayesian((double *)x,(double *)mu_bak4,
//                (long double *)cov_bak4x,3);

p_max=get_max(p_coty, 0.0, 0.0, p_bak1, 0.0, 0.0, 0.0);

if((p_max == p_coty)|| (p_max == p_weed))
{ class=1; output=255;}
else
{ class=2; output=0;}

if( red==255.0 && green==255.0 && blue==255.0)
{ class=2; output=0;}

rhm=mu_coty[0][0]-2.0*std1; rhp=mu_coty[0][0]+2.0*std1;
rsm=mu_coty[1][0]-2.0*std2; rsp=mu_coty[1][0]+2.0*std2;
rim=mu_coty[2][0]-2.0*std3; rip=mu_coty[2][0]+2.0*std3;
if( (x[0][0]<rhm) || (x[0][0]>rhp) || \
    (x[1][0]<rsm) || (x[1][0]>rsp) )
{ class=2; output=0;}

fprintf(addr_ptr,"%3d\n", output);
buffer[i] = (unsigned int)output;
}
if (write(fp, buffer, _FILE_BUFF_SIZE) != _FILE_BUFF_SIZE)
{
    status = FILE_ERR;
    break;
}
}
}
else
    status = FILE_ERR;

close(fp);          /* close lut file */

if (status == FILE_ERR) /* delete file if error populating */
{
    unlink(file_name);
    printf("Error! Lut file is not built.\n");
    exit(0);
}
else
    printf("LUT data is stored in '%s'.\a\n", file_name);
_fcloseall();
exit(0);
}

/*****
Function:double bayesian(double *x, double *mu, long double *cov,int n)

It returns the conditional probability of Bayes' classifier
using HSI values, mean and covariance.

Inputs:
x   (3x1) : matrix of hue, sat and int values

```

```

mu (3x1) : mean matrix of hue, sat, and int values
cov (3x3) : covariance matrix of hue, sat, and int
n       : number of rows of matrix x

*****/

double bayesian(double *x, double *mu, long double *cov, int n)
{
    double p, pi, mdiff[3][1], fcov[3][3], tdiff[1][3];
    long l[3], mm[3];
    double value, determinant, mult1[1][3], mult2, m, num, den;
    int i, j;
    long double det;

pi=3.141592654;
m=3.0;

determinant=detr(cov);
den=pow((2.0*pi),(m/2.0)) * sqrt(determinant);
pow((2.0*pi),(m/2.0),m,determinant);
det=minv(cov, l, mm, 3L);

for(i=0; i<n; i++) // convert cov from long double to double
    for(j=0; j<n; j++)
        fcov[i][j] = (double)*(cov+ i*n + j));

diff((double *)x,(double *)mu,(double *)mdiff,3,1);

transpose((double *)mdiff,(double *)tdiff,3,1);
multiply((double *)tdiff,(double *)fcov,(double *)mult1,1,n,n);
multiply((double *)mult1,(double *)mdiff,&mult2,1,n,1);
num=exp(-0.5*mult2);

p=num/den;
return p;
}

/*****
Function : double detr(long double *a)

Calculates determinant of matrix a (size x size).
In this case, size is equal to 3.

*****/

#define mat(name, r, c) (*(name + r*3 + c))
double detr(long double *a)
{
    double value;

value=mat(a,0,0)*((mat(a,1,1)*mat(a,2,2))-(mat(a,1,2)*mat(a,2,1)))\
    -mat(a,1,0)*((mat(a,0,1)*mat(a,2,2))-(mat(a,0,2)*mat(a,2,1)))\
    +mat(a,2,0)*((mat(a,0,1)*mat(a,1,2))-(mat(a,0,2)*mat(a,1,1)));

return value;
}

/* MINV:C; 50/4 *****/
/* FUNCTION: Subroutine

```

```

                long double minv (mp,l,m,n)

-----*/
/* Special Information:
PROGRAM TO INVERT A NxN MATRIX OF RANK N          11/21/86

ORIGINAL MATRIX IS CORRUPTED - RETURNS ZERO IF ORIGINAL NOT
OF RANK N OTHERWISE THE DETERMINANT OF THE ORIGINAL MATRIX
IS RETURNED
GAUSS-JORDAN METHOD IS USED TO COMPUTE INVERSE */

/* mp is a pointer to a n*n array of doubles containing the
   original matrix to be inverted. Inverse matrix is returned in mp.
   l and m are pointers to vectors of length n
   and are used as scratch working space. */

/*-----*/
/*      Author                      Creation Date
/* David Slaughter                    11/21/86
/*-----*/
/* Revision
   History By      Date              Description
/*-----*/
/* Version
/* 1.00           David      11/21/87 Creation
/* 1.10           Phil       04/28/87 Update filed with this header

*****/

//#pragma segment Main
long double minv(mp,l,m,n)
long double *mp;
long *l, *m, n;
{
    register long i,j,k,h;
    long double d,bigm,hold;

    d = 1.0;
    for(k=0L; k<n; k++)
    {
        *(l+k) = k;
        *(m+k) = k;
        bigm = *(mp + (n*k) + k);
        for(j=k; j<n; j++) /* find largest element */
        {
            for(i=k; i<n; i++)
                if(fabs(bigm) < fabs(*(mp + (i*n) + j)))
                {
                    bigm = *(mp + (i*n) + j);
                    *(l+k) = i;
                    *(m+k) = j;
                }
        }
        if(*(l+k) > k) /* interchange rows */
        {
            h = *(l+k);
            for(j=0L; j<n; j++)
            {
                hold = -(mp + (k*n) + j);
                *(mp + (k*n) + j) = *(mp + (h*n) + j);
            }
        }
    }
}

```

```

        *(mp + (h*n) + j) = hold;
    }
}
if(*(m+k) > k)          /* interchange c0Lumns */
{
    h = *(m+k);
    for(i=0L; i<n; i++)
    {
        i *= n;
        hold = -* (mp+i+k);
        *(mp+i+k) = *(mp+i+h);
        *(mp+i+h) = hold;
        i /= n;
    }
}
if(bigm == 0.0) return (bigm);    /* matrix is SINGULAR */
for(i=0L; i<n; i++)              /* divide c0Lumn by minus pivot (bigm)*/
    if(i != k) *(mp + (i*n) + k) /= -bigm;
for(i=0L; i<n; i++)              /* reduce matrix */
{
    hold = *(mp + (i*n) + k);
    for(j=0L; j<n; j++)
        if((i!=k) && (j!=k))
            *(mp + (i*n) + j) = (*(mp+(k*n)+j)*hold) + *(mp+(i*n)+j);
}
for(j=0L; j<n; j++)              /* divide row by pivot */
    if (j!=k) *(mp + (k*n) + j) /= bigm;
d *= bigm;                       /* product of pivots */
*(mp + (k*n) + k) = 1.0/bigm;    /* replace pivot by reciprocal */
}
for(k=n-2L; k>-1L; k--)          /* final row and c0Lumn interchange */
{
    if( *(l+k) > k)
    {
        h = *(l+k);
        for(i=0L; i<n; i++)
        {
            i *= n;
            hold = *(mp+i+k);
            *(mp+i+k) = -* (mp+i+h);
            *(mp+i+h) = hold;
            i /= n;
        }
    }
}
if( *(m+k) > k)
{
    h = *(m+k);
    for(j=0L; j<n; j++)
    {
        hold = *(mp + (k*n) + j);
        *(mp + (k*n) + j) = -* (mp + (h*n) + j);
        *(mp + (h*n) + j) = hold;
    }
}
}
return(d);
}
/*****
Function : void diff(double *a, double *b, double *result, int row,
                  int col)

```

```

Input:  matrix a (rowxcol)
        matrix b (rowxcol)

Output: matrix result (rowxcol)

*****/

#define matrix(name, r, c) (*(name + r*col + c))
void diff(double *a, double *b, double *result, int row, int col)
{
    int i,j;

    for(i=0; i<row; i++)
        for(j=0; j<col; j++)
            matrix(result,i,j) = matrix(a,i,j) - matrix(b,i,j);
}

/*****
Function : void transpose(double *a, double *result, int row, int col)

It returns transpose matrix of the original matrix.
Input : matrix a (3x1)
Output: matrix result (1x3)

*****/

void transpose(double *a, double *result, int row, int col)
{
    int i,j;

    for(i=0; i<row; i++)
        for(j=0; j<col; j++)
            *(result+ j*row +i) = *(a+ i*col + j);
}

/*****
Function : void multiply(double *a,double *b,double *result,int row,
                        int mid,int col)

It returns multiplication of the two input matrices.

Inputs:
a (rowxmid) : input matrix
b (midxcol) : input matrix

Output: reslut (rowxcol)

*****/

void multiply(double *a, double *b, double *result, int row, int mid,
             int col)
{
    int i,j,k;
    double sum=0.;

    for(i=0; i<row; i++)
        {
            for(j=0; j<col; j++)
                {

```



```

        for(k=0; k<mid; k++)
        {
            sum += (*(a+ i*mid + k)) * (*(b+ k*col +j));
        }
        (*(result + i*col +j)) = sum;
        sum=0.;
    }
}

/*****
Function : double get_max(p1,p2,p3,p4,p5,p6,p7)

It returns maximum value among 7 inputs (p1 - p7).
*****/

double get_max(double p1,double p2,double p3,double p4,double p5,double
p6,double p7)
{
    double max1,max2,max3,max4,max5,max6;

    max1=__max(p1,p2);
    max2=__max(p3,max1);
    max3=__max(p4,max2);
    max4=__max(p5,max3);
    max5=__max(p6,max4);
    max6=__max(p7,max5);

    return max6;
}

/***** E * N * D *****/

```

```

/*****
*
* Filename: hsipixel.c
* Author   : Won Suk Lee
* Date     : October 25, 1995
*
* Description: This program loads color images from the hard disk and
*              converts them into HSI using the AUXLUT in 4 quadrants.
*              Then, pixel values for HUE, SAT and INT are obtained by
*              drawing a box in the color image with a mouse.
*              Those HUE, SAT and INT pixel values are stored in
*              filename+H.asc, filename+S.asc, and filename+I.asc,
*              respectively.
*
*****
* Revision History
* Date      Name  Description
* 5/30/96   WSL   added for-loop for multiple images.
* 7/25/96   WSL   added image size checking and subsampling routine
* 10/3/96   WSL   modified to use this program for different length of
*                image file names. (Before it worked with only 3
*                common char.)
* 7/7/97    WSL   Fixed error for null-pointer assignment ('inf_ptr'
*                was problem) and restoring graphics mode after
*                finishing program.
*****/

#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <graph.h>
#include <time.h>
#include "gpb2.h"
#include "windw2.h"
#include <dos.h>
#include <math.h>
#include <string.h>
#include <memory.h>
#include <process.h>
#include <errno.h>
#include <sys\types.h>
#include <sys\stat.h>

#define P1 1
#define P2 2
#define P3 3
#define B1 1
#define B2 2
#define B3 3
#define B4 4

short old_mode;          /* old video mode */
int getpixel(char *file_name, int *common);

main( argc, argv)
int argc;
char **argv;
{
int status, i;
char in_char, file_name[30], ch;

```

```

int wait_sec, stop_flag;          /* run-time parameter flags */
struct videoconfig vc;           /* video configuration */
short rows, save_mode;          /* number of text rows */
short s_mon_color;              /* color index for single monitor */
int vga;
int pb[] = {0,1,0, 0,1,0, 0,1,0, 0,1}; /* parameter block */
FILE *image_ptr;                /* image file pointer */
int x1, x2, y1, y2;            /* pixel location */
int xx;                          /* number of files */
struct _stat filestat;
int common;                      /* number of common characters in image file name */
/* initialize defaults */
stop_flag = TRUE;                /* wait for carriage return */
wait_sec = 4;                    /* number of seconds to wait */
s_mon_color = 13;

/* process command line arguments */
argv++; argc--;
while( argc > 0)
{
    if( strcmp( "/s", *argv) == 0)
    {
        /* set single monitor color */
        argv++; argc--;
        if( argc > 0 )
        {
            s_mon_color = atoi( *argv);
            argv++; argc--;
        }
    }
    else if( strcmp( "/t", *argv) == 0)
    {
        /* set wait time interval */
        argv++; argc--;
        stop_flag = FALSE;
        if( argc > 0 )
        {
            wait_sec = atoi( *argv);
            argv++; argc--;
        }
    }
    else
    {
        printf("Unknown flag: %s\n", *argv);
        exit(0);
    }
}

/* initialize GPB */
s_gpbinit(0);
s_clearall();
printf("Enter NUMBER of common character in filename: ");
scanf("%d", &common);

/* initialize graphics */
_getvideoconfig( &vc);
old_mode = vc.mode;

for(xx=0; xx<3000; xx++)
{
    /* get input image file */
    s_gpbroi( 0, 0, 0, 0, 512, 480);
}

```

```

system("cls");
printf("Enter RGB image file name (Enter 999 to exit): ");
scanf("%s", file_name);

/* initialize graphics */
_getvideoconfig( &vc);
save_mode = vc.mode;
vga = TRUE;
rows = _setvideomode( _VRES16COLOR);
if( rows == 0)
{
    rows = _setvideomode( save_mode);
    printf("Can't run in single monitor mode\n");
    vga = FALSE;
}
if( vga)
{
    _settextcolor( 14); /* text color is light yellow */
    _remappalette( s_mon_color, _BLACK); /* for single monitor display */
    _setcolor( s_mon_color);
    _rectangle( _GFILLINTERIOR, 0,0, 639,479);
    _setbkcolor( _BLUE); /* background color is blue */
}
if(file_name[0] == '9')
{
    _setvideomode( old_mode);
    exit(0);
}
/* Get file statistics. */
if( _stat(file_name, &filestat) != 0)
    printf( "Error in getting file size!\n");
if((image_ptr=fopen(file_name, "r")) == NULL)
{
    printf("No %s file found! Press any key to exit.\n", file_name);
    getch();
    _setvideomode( old_mode);
    exit(0);
}
else
{
    /* load color image */
    s_gpbroi(0,0,0,0,256,240);
    if(s_loadcimg(file_name, P2,B1,P2,B2,P2,B3, FALSE) != SUCCESS)
    {
        printf("Error in loading %s image!\n", file_name);
        _setvideomode( old_mode);
        exit(0);
    }
    if (filestat.st_size > 200000) /* 512 x 480 */
    {
        s_gpbroi(0,0,0,0,512,480);
        s_loadcimg(file_name, P1,B2,P1,B3,P1,B1, FALSE);
        if((status=s_reduce( P1,B2, P2,B1, 0,0,0, P3,B4, 2,2))!= SUCCESS)
            printf("Error in reducing RED\n"); // subsampling
        if((status=s_reduce( P1,B3, P2,B2, 0,0,0, P3,B4, 2,2))!= SUCCESS)
            printf("Error in reducing GREEN\n");
        if((status=s_reduce( P1,B1, P2,B3, 0,0,0, P3,B4, 2,2))!= SUCCESS)
            printf("Error in reducing BLUE\n");
    }
}
}

```

```

fclose(image_ptr);

/* initialize auxlut */
s_gpbroi( 0, 0, 0, 0, 512, 128);
if((status = s_ldauxlt( "rbhue.img", 0, AUX_LDRD_LUT1B2)) != SUCCESS)
    {printf("error: %d ldauxlt -- rbhue.img", status); exit(0);}
if((status = s_cpyauxlt( P3,B3, AUX_READ_LUT1)) != SUCCESS)
    {printf("error: %d cpyauxlt read -- Hue", status); exit(0);}
if((status = s_ldauxlt( "rgbsat.img", 0, AUX_LDRD_LUT2B2)) != SUCCESS)
    {printf("error: %d ldauxlt -- rgbsat.img", status); exit(0);}
if((status = s_cpyauxlt( P3,B1, AUX_READ_LUT2)) != SUCCESS)
    {printf("error: %d cpyauxlt read -- Sat", status); exit(0);}
if((status = s_ldauxlt( "rgbint.img", 0, AUX_LDRD_LUT2B2)) != SUCCESS)
    {printf("error: %d ldauxlt -- rgbint.img", status); exit(0);}
if((status = s_cpyauxlt( P3,B2, AUX_READ_LUT2)) != SUCCESS)
    {printf("error: %d cpyauxlt read -- Int", status); exit(0);}
_settextposition( 1, 19);          _outtext(" Red-Green-Blue ");
_settextposition( 1, 56);          _outtext(" Hue ");
_settextposition( rows/2+1, 21);    _outtext(" Saturation ");
_settextposition( rows/2+1, 53);    _outtext(" Intensity ");
if( stop_flag)
{
    _settextposition( rows, 1);
    _outtext("Press any key to continue");
}

/* display buffers as red, green, blue respectively */
s_gpbroi( 0, 0, 0, 0, 256, 240);
s_coldisp( P2,B1, P2,B2, P2,B3);
s_gpbdelay( 15);                  /* wait for display */

/* do conversion to Hue */
s_gpbroi( 0, 0, 0, 0, 256, 240);
if((status=s_rgbauxlt(P2,B1, P2,B2, P2,B3, AUX_RGLUT1)) != SUCCESS)
    {printf("error: %d rgbauxlt -- LUT1", status); exit(0);}
if((status=s_thrucpy(0, AUXLUT, P2,B4, 0,0, 0,0, 0,0,0)) != SUCCESS)
    {printf("error: %d thrucpy -- LUT1", status); exit(0);}

/* display Hue */
s_gpbroi( 0, 0, 0, 0, 256, 240);
s_thrucpy( P2,B4, P1,B1, 0,0, 0,0, 0,0,0);
s_gpbroi( 0, 0, 256, 0, 256, 240);
s_disp( P1, B1, 'X');
s_gpbdelay(15);                  /* wait for display */

/* do conversion to Sat */
s_gpbroi( 0, 0, 0, 0, 512, 128);
if((status = s_cpyauxlt( P3,B1, AUX_LOAD_LUT2)) != SUCCESS)
    {printf("error: %d cpyauxlt -- Sat", status); exit(0);}
s_gpbroi( 0, 0, 0, 0, 256, 240);
if((status=s_rgbauxlt(P2,B1, P2,B2, P2,B3, AUX_RGLUT2)) != SUCCESS)
    {printf("error: %d rgbauxlt -- LUT2", status); exit(0);}
if((status=s_thrucpy(0, AUXLUT, P2,B4, 0,0, 0,0, 0,0,0)) != SUCCESS)
    {printf("error: %d thrucpy -- LUT2", status); exit(0);}

/* display Sat */
s_gpbroi( 0, 0, 0, 0, 256, 240);
s_thrucpy( P2,B4, P1,B2, 0,0, 0,0, 0,0,0);
s_gpbroi( 0, 0, 0, 240, 256, 240);
s_disp( P1, B2, 'X');

```

```

s_gpbdelay( 15);                               /* wait for display */

/* do conversion to Int */
s_gpbroi( 0, 0, 0, 0, 512, 128);
if((status=s_cpyauxlt(P3,B2, AUX_LOAD_LUT2)) != SUCCESS)
    {printf("error: %d cpyauxlt -- Int", status);  exit(0);}
s_gpbroi( 0, 0, 0, 0, 256, 240);
if((status=s_rgbauxlt(P2,B1, P2,B2, P2,B3, AUX_RGBLUT2)) != SUCCESS)
    {printf("error: %d rgbauxlt -- LUT2", status);  exit(0);}
if((status=s_thrucpy(0, AUXLUT, P2,B4, 0,0, 0,0, 0,0,0)) != SUCCESS)
    {printf("error: %d thrucpy -- LUT2", status);  exit(0);}

/* display Int */
s_gpbroi( 0, 0, 0, 0, 256, 240);
s_thrucpy( P2,B4, P1,B3, 0,0, 0,0, 0,0,0);
s_gpbroi( 0, 0, 256, 240, 256, 240);
s_disp( P1, B3, 'X');
s_gpbdelay( 15);                               /* wait for display */

getpixel(file_name, &common);    // get pixel values
} // end of xx loop
_setvideomode( old_mode);
exit(0);
} // END OF MAIN

```

```

/*****
pixelbx1.c      (ROI=1/4)

```

This program prompts for an image file in the hard disk to be displayed and then allows the mouse to be used to read pixel locations and values in that image by drawing a bounding box. Then, it stores the pixel value in the output file as an ASCII.

```

/*****
/* The following is used for mouse control via MS-DOS Interrupt 0x33: *
*
*   #include <dos.h>
*   union REGS iReg, oReg;
*   iReg.x.ax = number of function to be called;
*   iReg.x.bx = value of second parameter;
*   iReg.x.cx = value of third parameter;
*   iReg.x.dx = value of fourth parameter;
*   int86( 0x33, &iReg, &oReg);
*   oReg.x.ax, oReg.x.bx, oReg.x.cx, oReg.x.dx contain return values
*
*   Information on additional mouse functions can be found in:
*   "Microsoft Mouse Programmer's Reference", Microsoft Press, 1991.
*****/

```

```

int getpixel(char *file_name, int *num)
{
    char response[10], buffer[82];
    struct videoconfig vc;    /* video configuration */
    short save_mode;        /* old video mode */
    short rows;            /* number of text rows */
    short s_mon_color;     /* color index for single monitor */
    union REGS iReg, oReg;  /* for mouse control */
    int x1, x2, y1, y2;    /* pixel location */
    int ival;              /* pixel value */
    int hue_plane, hue_bank; /* hue plane & bank number */
    int sat_plane, sat_bank; /* sat plane & bank number */

```

```

int int_plane, int_bank; /* int plane & bank number */
char *gets();
/* pointer to input image file & output pixel value file */
FILE *inf_ptr,*huef_ptr,*satf_ptr,*intf_ptr;
char out_name[30], color_img[30], in_char;
char hue_name[30], sat_name[30], int_name[30];
int flag, i, j, m, n, k, l; /* flag for drawing box
/* pixel buffer */
unsigned char hue_pixel[10000],sat_pixel[10000],int_pixel[10000];
int *hueptr, *satptr, *intptr, no_pixel;// pointers to pixel buffer
int width, height; // width & height of ROI
int release; // no. of mouse button releasae
int index; // no. of data point in a line on data file
int sourcex,sourcexy, destx, desty; // ROI source(x,y) & dest(x,y)
unsigned long sum;
const char *ch;
int common;

/* initialize default values */
s_mon_color=13;
flag=0;
hueptr=(int *)hue_pixel;
satptr=(int *)sat_pixel;
intptr=(int *)int_pixel;
release=sum=index=0;
common=*num;

/* initialize graphics */
_getvideoconfig( &vc);
/* set high resolution video mode for mouse control
resolution (need 512x480) & single monitor mode */
rows = _setvideomode( _VRES16COLOR);
if(rows == 0)
{
rows = _setvideomode( save_mode);
printf("Can't run in single monitor mode\n");
}
_settextcolor( 14); /* text color is light yellow */
_remappalette( s_mon_color, _BLACK); /* for single monitor display */
_setcolor( s_mon_color);
_rectangle( _GFILLINTERIOR, 0,0, 639,479);
_setbkcolor( _BLUE); /* background color is blue */

_settextposition( rows,1);
_outtext("Press any key to continue");
_settextposition( 1, 1);
sprintf(buffer, "Press and drag left mouse button for pixel value\n");
_outtext(buffer);

/* make an output file name as "dem*." */
n=strlen(file_name); // counts no. of characters in file name
m=n-4; // 4 means 4 char for (.img)
l=n-(common+4); // only number portion of filename
printf("common=%d\n", common);
// initialize character buffers
for(k=0; k<=(m-1); k++)
{
hue_name[k]=' ';
sat_name[k]=' ';
int_name[k]=' ';
}

```

```

        if(k==(m-1))
        {
            hue_name[k]='\0';
            sat_name[k]='\0';
            int_name[k]='\0';
        }
    }

    // copies first (m-1) character to hue_name from file_name
    memcpy(hue_name, file_name,m-1); /* name -> tom, tom, tom */
    memcpy(sat_name, file_name,m-1);
    memcpy(int_name, file_name,m-1);

    strcat(hue_name, "h" );          /* name -> tomh, toms, tomi */
    strcat(sat_name, "s" );
    strcat(int_name, "i" );

    for(k=common; k<(1+common); k++)
    {
        ch=&file_name[k];
        strncat(hue_name, ch, 1);    /* name -> tomh#, toms#, tomi# */
        strncat(sat_name, ch, 1);
        strncat(int_name, ch, 1);
    }
    _settextposition( 3, 1);
    sprintf(buffer, "Pixel data will be saved in %s (HUE
                    transformed),\n", hue_name);
    _outtext(buffer);
    _settextposition( 4, 1);
    sprintf(buffer, "                    %s (SAT
                    transformed),\n", sat_name);
    _outtext(buffer);
    _settextposition( 5, 1);
    sprintf(buffer, "                    and %s (INT transformed),
                    respectively.\n", int_name);
    _outtext(buffer);
    _settextposition( 7, 1);
    sprintf(buffer, " Execution status : %s", file_name);
    _outtext(buffer);
    _settextwindow( 8, 1, rows-3, 80);
    _settextposition( 1, 1);

    /* File did not exist. Create it for writing. */
    if( (huef_ptr = fopen( hue_name, "w" )) == NULL )
    {
        printf("File creation error!\n");
        exit( 0 );
    }
    if( (satf_ptr = fopen( sat_name, "w" )) == NULL )
    {
        printf("File creation error!\n");
        exit( 0 );
    }
    if( (intf_ptr = fopen( int_name, "w" )) == NULL )
    {
        printf("File creation error!\n");
        exit( 0 );
    }
}

```



```

/* arrange values for cursor and mouse */
s_selcur( 2); /* select cursor type */
s_clrdisp( 'X');
iReg.x.ax = 0; /* mouse reset & status */
int86( 0x33, &iReg, &oReg);
if( oReg.x.ax != -1)
    printf("Mouse hardware and/or software is not installed
properly.\n");
iReg.x.ax = 7; /* set horizontal mouse range */
iReg.x.cx = 0; /* left */
iReg.x.dx = 256; /* right */
int86( 0x33, &iReg, &oReg);
iReg.x.ax = 8; /* set vertical mouse range */
iReg.x.cx = 0; /* top */
iReg.x.dx = 240; /* bottom */
int86( 0x33, &iReg, &oReg);
iReg.x.ax = 15; /* change mouse speed */
iReg.x.cx = 8; /* horizontal mickeys/pixel (default=8) */
iReg.x.dx = 8; /* vertical mickeys/pixel (default=16) */
int86( 0x33, &iReg, &oReg);
hue_plane=P1; hue_bank =B1;
sat_plane=P1; sat_bank =B2;
int_plane=P1; int_bank =B3;

/* main loop */
while(1)
{
    /* display image */
    /* display buffers as red, green, blue respectively */
    s_gpbroi( 0, 0, 0, 0, 256, 240);
    s_coldisp( P2,B1, P2,B2, P2,B3);
    s_gpbdelay( 10); /* wait for display */

    s_gpbroi(0,0,256,0,256,240); /* display HUE */
    s_disp( hue_plane, hue_bank, 'X');
    s_gpbdelay( 10);

    s_gpbroi(0,0,0,240,256,240); /* display SAT */
    s_disp( sat_plane, sat_bank, 'X');
    s_gpbdelay( 10);

    s_gpbroi(0,0,256,240,256,240); /* display INT */
    s_disp( int_plane, int_bank, 'X');
    s_gpbdelay( 10);
    // restore ROI
    s_gpbroi(0,0,0,0,256,240);
    if( kbhit()) /* exit -- cleanup before leaving */
    {
        getch();
        fclose(huef_ptr); /* close output file
        fclose(satf_ptr);
        fclose(intf_ptr);
        sprintf( buffer, "\nTotal pixel=%ld. Press any key to
        continue...\n", sum);
        _outtext( buffer);
        getch();
        iReg.x.ax = 0; /* mouse reset & status */
        int86( 0x33, &iReg, &oReg);
        _setvideomode( old_mode); /* reset video mode */
        s_selcur( 0); /* turn off cursor */
    }
}

```

```

    return TRUE;
}
iReg.x.ax = 3;      /* determine mouse position & button state */
int86( 0x33, &iReg, &oReg);
x1 = oReg.x.cx;    y1 = oReg.x.dx;
switch( oReg.x.bx)
{
    case 0:        /* neither button pressed */
        break;
    case 1:        /* left button pressed */
        release = 0;
        while(release == 0)
        {
            iReg.x.ax=3; /* determine mouse position & button state */
            int86( 0x33, &iReg, &oReg);
            x2 = oReg.x.cx; y2 = oReg.x.dx;
            s_putcur(x2, y2);          // update cursor
            iReg.x.ax = 6; /* get button release information */
            iReg.x.bx = 0; /* for left button */
            int86( 0x33, &iReg, &oReg);
            if(oReg.x.bx == 1)
                release=1;
            x2 = oReg.x.cx; y2 = oReg.x.dx; // get mouse position
        }

        if(x2 >= x1)
        {
            sourcecx=x1;
            destx=x2;
        }
        else
        {
            sourcecx=x2;
            destx=x1;
        }
        if(y2 >= y1)
        {
            sourcecy=y1;
            desty=y2;
        }
        else
        {
            sourcecy=y2;
            desty=y1;
        }

        // #5 if
        if( release > 0)
        {
            // #4 if          /* corrected on 10/25/95 */
            if(s_bboxroi(P2,B1, 'B', sourcecx, sourcecy, destx, desty, 255)
                == SUCCESS)
            {
                s_bboxroi(hue_plane, hue_bank, 'B', sourcecx, sourcecy,
                    destx, desty, 255);
                s_bboxroi(sat_plane, sat_bank, 'B', sourcecx, sourcecy,
                    destx, desty, 255);
                s_bboxroi(int_plane, int_bank, 'B', sourcecx, sourcecy,
                    destx, desty, 255);
                width=abs(x1-x2)-2;      height=abs(y1-y2)-2;
            }
        }
    }
}

```

```

sprintf(buffer, "width*height= %d\n", width*height);
_outtext(buffer);
// #3 if          /* corrected 10/25/95 */
if( (width*height) > 11000)
{
    sprintf( buffer, "\nROI is too big.(Width=%5d,
                    Height=%5d)\n", width,height);
    _outtext( buffer);
    sprintf( buffer, "Total pixel=%ld. Press any key to
                    continue...\n", sum);
    _outtext( buffer);
    //fclose(inf_ptr);          // close input file
    fclose(huef_ptr);          // close output file
    fclose(satf_ptr);
    fclose(intf_ptr);
    iReg.x.ax = 0;              /* mouse reset & status */
    int86( 0x33, &iReg, &oReg);
    getch();
    _setvideomode( old_mode);  /* reset video mode */
    s_selcur( 0);              /* turn off cursor */
    return TRUE;
}
// #3 else
else
{
    // #2 if          /* corrected 10/25/95 */
    if(s_gpbroi(sourcecx+1,sourcecy+1,0,0,width,height) ==
        SUCCESS)
    {
        // #1 if
        if( width > 2 && height > 0)
        {
            s_getpxblk(hue_plane, hue_bank, hueptr, 0);
            s_getpxblk(sat_plane, sat_bank, satptr, 0);
            s_getpxblk(int_plane, int_bank, intptr, 0);
            no_pixel=width*height; // no. of pixels
            sum += no_pixel;        /* corrected 10/25/95 */
            sprintf(buffer, "ROI:source(%3d,%3d) ->
                            dest(%3d,%3d): width=%3d, height=%3d\n",\
                            sourcecx+1,sourcecy+1,destx-1,desty-1,width,
                            height);
            _outtext( buffer);
            for(i=0; i<no_pixel; i++)
            {
                fprintf(huef_ptr, "%4d\n",(int)hue_pixel[i]);
                fprintf(satf_ptr, "%4d\n",(int)sat_pixel[i]);
                fprintf(intf_ptr, "%4d\n",(int)int_pixel[i]);
                // if((int)hue_pixel[i]==0) printf("\a");
            }
            sprintf( buffer, "Getting pixel value
                            completed.\n");
            _outtext( buffer);
        }
        // #1 else
        else
        {
            sprintf( buffer, "\nROI selected
                            inappropriately.\n");
            _outtext( buffer);
        }
    }
}

```

```

    }
    // #2 else
    else
    {
        sprintf( buffer, "\nGPB ROI Error!\n");
        _outtext( buffer);
        sprintf( buffer, "Total pixel=%ld. Press any key to
            continue...\n", sum);
        _outtext( buffer);
        //fclose(inf_ptr);          // close input file
        fclose(huef_ptr);          // close output file
        fclose(satf_ptr);
        fclose(intf_ptr);
        iReg.x.ax = 0;              /* mouse reset & status */
        int86( 0x33, &iReg, &oReg);
        getch();
        _setvideomode( old_mode); /* reset video mode */
        s_selcur( 0);              /* turn off cursor */
        return TRUE;
    }
}
// #4 else
else
{
    sprintf( buffer, "\nFailed to draw a bounding box!\n");
    _outtext( buffer);
    sprintf( buffer, "Total pixel=%ld. Press any key to
        continue...\n", sum);
    _outtext( buffer);
}
}
// #5 else
else
{
    sprintf( buffer, "Mouse function 6 error.\n");
    _outtext( buffer);
}
break;
case 2:          /* right button pressed */
    break;
case 3:          /* both buttons pressed */
    break;
default:         /* unexpected number of buttons pressed */
    break;
} // end of switch

s_gpbroi(0,0, 0,0, 256,240);    // restore gpbroi

/* update cursor position */
x1 = oReg.x.cx;    y1 = oReg.x.dx;
s_putcur( x1, y1);
}
return TRUE;
}

/***** E * N * D *****/

```