Exploration and Comparison of Image-based Techniques for Strawberry Detection

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Yongxin Liu

September, 2020

COMMITTEE MEMBERSHIP

TITLE:	Exploration and Comparison of Image-based Techniques for Strawberry Detection
AUTHOR:	Yongxin Liu
DATE SUBMITTED:	September, 2002
COMMITTEE CHAIR:	Xiaozheng (Jane) Zhang, Ph.D. Professor of Electrical Engineering, Adviser
COMMITTEE MEMBER:	Xiao-Hua (Helen) Yu, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Jonathan Ventura, Ph.D. Assistant Professor of Computer Science and Software Engineering

ABSTRACT

Exploration and Comparison of Image-based Techniques for Strawberry Detection

Yongxin Liu

Strawberry is an important cash crop in California, and its supply accounts for 80% of the US market [2]. However, in current practice, strawberries are picked manually, which is very labor-intensive and time-consuming. In addition, the farmers need to hire an appropriate number of laborers to harvest the berries based on the estimated volume. When overestimating the yield, it will cause a waste of human resources, while underestimating the yield will cause the loss of the strawberry harvest [3]. Therefore, accurately estimating harvest volume in the field is important to farmers. This paper focuses on an image-based solution to detect strawberries in the field by using the traditional computer vision technique and deep learning method.

When strawberries are in different growth stages, there are considerable differences in their color. Therefore, various color spaces are first studied in this work, and the most effective color components are used in detecting strawberries and differentiating mature and immature strawberries.

In some color channels such as the R color channel from the RGB color model, Hue color channel from the HSV color model, 'a' color channel from the Lab color model, the pixels belonging to ripe strawberries are clearly distinguished from the background pixels. Thus, the color-based K-mean cluster algorithm to detect red strawberries will be exploited. Finally, it achieves a 90.5% truth-positive rate for detecting red strawberries.

For detecting the unripe strawberry, this thesis first trained the Support Vector Machine classifier based on the HOG feature. After optimizing the classifier through hard negative mining, the truth-positive rate reached 81.11%.

Finally, when exploring the deep learning model, two detectors based on different pre-trained models were trained using TensorFlow Object Detection API with the acceleration of Amazon Web Services' GPU instance. When detecting in a single strawberry plant image, they have achieved truth-

positive rates of 89.2% and 92.3%, respectively; while in the strawberry field image with multiple plants, they have reached 85.5% and 86.3%.

Keywords: Histogram of Oriented Gradients, Support Vector Machine, Transfer Learning, Deep Convolutional Neural Networks

ACKNOWLEDGMENTS

Time flies, my master's career has come to an end. These three years are long and short, full of sweet and bitter, and more rewarding and growing.

Foremost, I would like to express my heartfelt thanks to my advisor, Dr. Jane Zhang, for her has always given me patient guidance and support for my thesis and my master's study in Cal Poly.

Besides, I sincerely thank Dr. Helen Yu and Dr. Jonathan Ventura for serving on my thesis.

Finally, a special thank to my parents for their endless support and encouragement.

Table of Contents

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

Introduction

As strawberry is an important cash crop in California, and there is a demand for farmers to estimate harvest volume in the field accurately. The application of strawberry detection will address the need for farmers to estimate crop yields. Then farmers can reasonably arrange labor and storage based on the estimation, which will ultimately achieve the goal of saving costs.

With the continuous breakthroughs in Artificial Intelligence, especially the significant advances in perceptual intelligence technologies such as speech recognition, natural language processing, object detection recognition, these technologies have been ubiquitous in our daily lives. For example, the virtual agent, Smart security system, and multilingual translation. Object detection is a popular topic in computer vision and digital image processing, which is widely used in many fields such as vehicle navigation [22], intelligent video surveillance [23], and smart manufacturing [24]. It has important practical significance to reduce the consumption of human capital. Therefore, the related technologies are mature enough to implement strawberry detection.

In modern agriculture, there is an extensive body of work carried out in the field of object detection during the past few decades. The related applications include fruit recognition [25], plant disease detection [26], detection of aphids in wheat fields [27], and strawberry detection [28].

This paper aims to detect strawberry in the field and is inspired by the related object detection work. Anisha M Nayak [25] proposed a method that has four stages for building a fruit recognition system: The first step is to preprocess the image dataset, then to crop the object out from each image by image segmentation. After extracting the feature such as color and histogram of oriented gradients (HOG), the final step is to train the SVM Support Vector Machine (SVM) classifier. As N. Dalal mentioned in Histograms of oriented gradients for human detection [7], the HOG feature is invariant when light condition changes due to the HOG descriptors operate on local cell, while such change appears in lager spatial regions. Thus, the HOG feature is a strong basis to perform strawberry detection later in the thesis.

In addition to traditional computer vision technologies, a lot of related work also uses deep neural network models. Nikolas Lamb [28] presented a state-of-the-art Single-shooting Detection (SSD) neural network framework as a fruit detector. Based on a three-layer convolutional neural network, he alters the network in various ways to increase the detection speed and precision. When the input images are with the size of [360 640], he achieves a 0.877 average accuracy. S. Ghosal described [26] described a transfer learning method when training his classifier to classify the rice leaf diseases. His model is based on the pretrained VGG-16 model. The final accuracy achieves 92.46%.

This thesis provides image-based solutions for strawberry estimation. Instead of counting strawberries manually, which is time-consuming, computer vision algorithms will automatically analyze the images taken from the fields, detect and count the number of strawberries. Once the sensors in the strawberry fields have collected the photos for the entire field, the photos will be sent into trained detection models to detect red and green strawberries, respectively. The final system will accurately detect and count all the strawberries in an image.

The red strawberry and green strawberry will be treated as two different objects and detected separately later in this thesis.

First, we will exploit the color-based K-Mean Cluster algorithm to detect red strawberries.

Next we explore two approaches for green strawberry detection. One is to use traditional computer vision methods of Support Vector Machine based HOG features. Another one is to train neural network models based on transfer learning.

During the whole process, the first challenge was to collect data and organize data. The collection process lasted for seven months from January to July 2018. The strawberry pictures had been taken twice a week from the strawberry farm on Cal Poly campus managed by the Cal Poly strawberry center. The time for each collection is fixed between 10 am and 12 pm. Besides, each image is processed manually before being used for training, such as cropping the single strawberry out or drawing bounding boxes around strawberries. Approximately 700 images have been collected over the course of seven months. Sample images are shown below. (Figure1)

Figure 1 Collected Data

The ultimate goal is to help farmers estimate yields by quickly detecting and counting the ripe strawberries. Furthermore, finding green strawberries can be useful for harvest forecasting which allows farmers for better planning. In the project, we aim to develop computer vision algorithms to accurately and reliably detect all red and green strawberries in an image in real time.

Chapter 2

Image Segmentation

Chapter 2 describes the process of red strawberries detection. After performing different color channels analysis to select suitable color models, the K-Mean Clustering algorithm is used to perform image segmentation to find red strawberries in images.

2.1 Color Space Analysis

2.1.1 Introduction

This chapter focuses on detecting ripe strawberries. Ripe strawberries have a red color that is distinct from their background that consists of green leaves, stems, buds, and gray plastic cloth. Thus, color is useful information to help detect red strawberries. Before actually performing image segmentation, the first step is to analyze different color models to determine suitable color image channels.

When performing digital image processing, we often use different image models to represent the same image. As Rafael C. Gonzalez explained, ‘‘The purpose of a color model (also called color space or color system) is to facilitate the specification of colors in some standard, generally accepted way’’ [1]. A suitable image channel means a channel in which the histograms of strawberry and non-strawberry do not overlap or overlap only slightly. Therefore, analyzing the histograms of strawberry images in the RGB, HSV, and Lab image models will help find out those color channels.

First, the pure strawberry pixels and the pixels belonging to the background are obtained by cropping the strawberry out from the strawberry image along its contour.

Figure 2 shows these cropped results. After repeating this process in twenty images, the photographs of strawberries have been isolated from the background.

Figure 2 The strawberries were separated from their backgrounds

For each color component, the color distributions of the background and strawberries are examined in different color channels. The histogram for all the strawberry pixels across all 20 sample images are compared with the histogram of all background pixels across the 20 background images.

2.1.2 RGB Color Space

The most widely used color space in computer technology is the RGB color space. It consists of three color spaces - red, green, and blue [1].

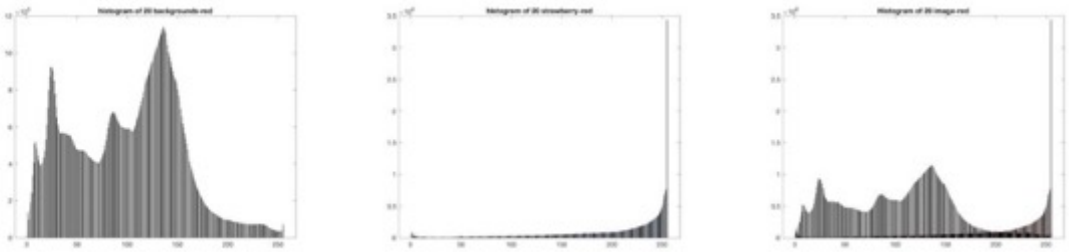

Figure 3 Red histogram of non-strawberry pixels (Left) Red Histogram of strawberry pixels (Middle) Red Histogram of all pixels (Right)

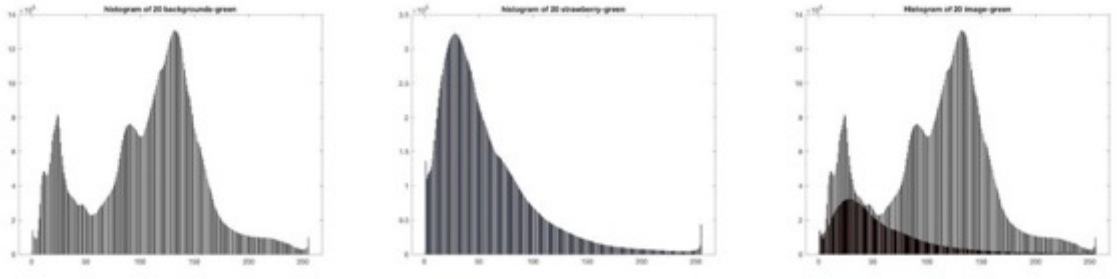

Figure 4 Green histogram of non-strawberry pixels (Left) Green Histogram of strawberry pixels (Middle) Green Histogram of all pixels (Right)

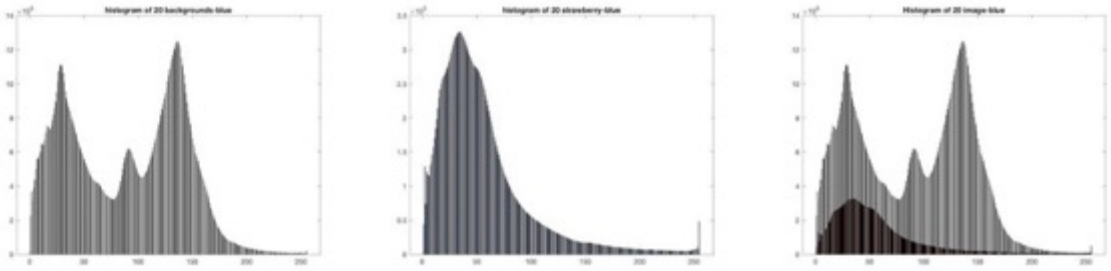

Figure 5 Blue histogram of non-strawberry pixels (Left) Green Histogram of strawberry pixels (Middle) Green Histogram of all pixels (Right)

Upon inspecting the histogram findings in Figures 3, 4, and 5, the advantages of color separation become apparent. As we would only like to consider ripe, red strawberries, the red color space is the most vital to classification. The bimodal nature of the histogram clearly shows the distribution of strawberries above the 200 red mark and background below the 200 red mark. Based on this observation, the value of 200 in the red channel was used as the threshold value to separate strawberry

from its background. The red channel of RGB will be used later for image segmentation. Besides, the red channel of RGB is the right candidate for later image segmentation.

2.1.3 Normalized RGB Color Space

Even though the RGB color model is commonly used in image processing, it is easily affected by changes in the illumination. As M. Vanrell mentioned in 'color normalization based on background information'[14], it is important to remove all intensity values from the image while preserving color values. Therefore, color normalization has been used for object recognition on colored images in the field of robotics, bioinformatics and general artificial intelligence. [14].

According to the Equation 2.1.1, Equation 2.1.2, and Equation 2.1.1, processing the three channels of all RGB, respectively.

$$r = \frac{R}{(R+G+B)} \tag{2.1.1}$$

$$b = \frac{B}{(R+G+B)} \tag{2.1.2}$$

$$g = \frac{G}{(R+G+B)} \tag{2.1.3}$$

Then drawing the histogram of different color planes as before. (Figure6, Figure7, and Figure8).

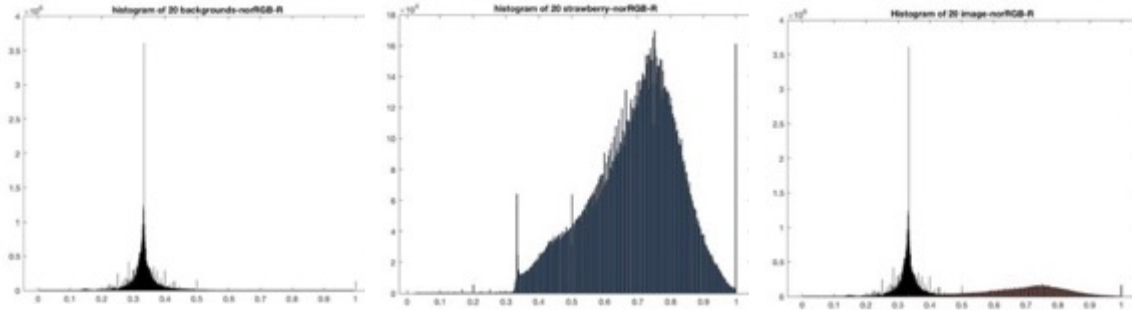

Figure 6 Normalized Red histogram of non-strawberry pixels (Left) Red Histogram of strawberry pixels (Middle) Red Histogram of all pixels (Right)

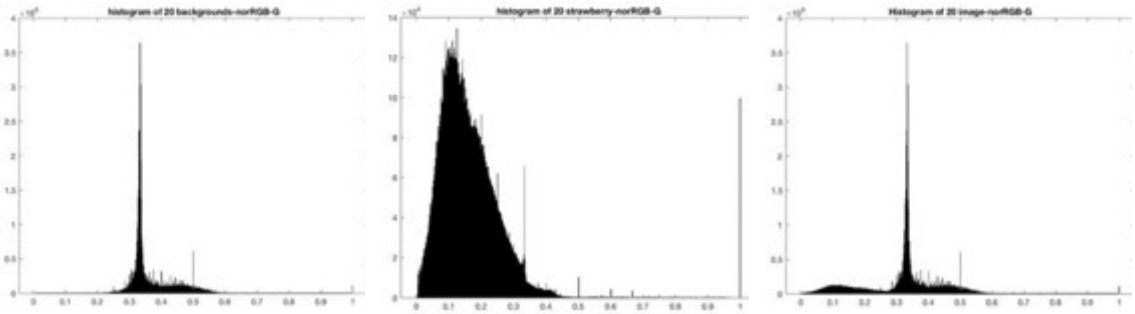

Figure 7 Normalized Green histogram of non-strawberry pixels (Left) Red Histogram of strawberry pixels (Middle) Red Histogram of all pixels (Right)

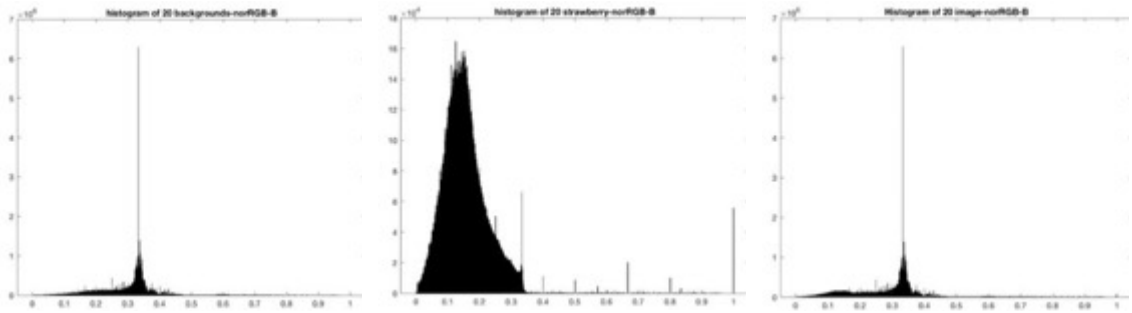

Figure 8 Normalized Blue histogram of non-strawberry pixels (Left) Red Histogram of strawberry pixels (Middle) Red Histogram of all pixels (Right)

Normalized RGB color model offers another suitable candidate, the normalized r channel, in which strawberry pixels are distinct from its background. This color model will be used for image segmentation later.

2.1.3 HSV Color Space

HSV (Hue-Saturation-Value) color space is another commonly used color space in image processing. It starts from the human visual system and uses Hue, Saturation (Saturation or Chroma), and Brightness (Intensity or Brightness) [1]. Figure 9 shows a conical space model which can describe the HSV color space.

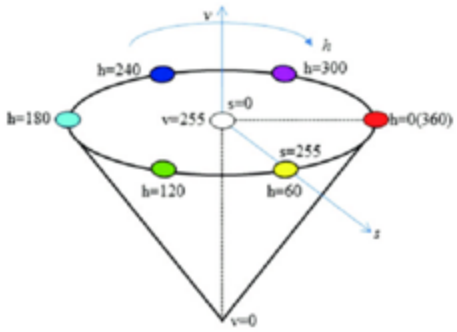

Figure 9 HSV Color Model

The hue 'H' is represented by angle, and the saturation 'S' is the radius length from the axis of the HSV color space to the color point. The closer the distance of the color point from the axis, the more light of color. The intensity 'I' is represented by the height in the direction of the axis. The axis of the cone describes the gray level. The minimum intensity is black, and the maximum intensity is white.

Similar to the method used for the RGB space, with an additional transform done to convert the RGB images into the HSV color space, histograms of the strawberry and background part have been collected.

As shown in Figure10, Figure11 and Figure12, Hue channel shows the ability to identify the vivid red of the berries easily. However, Saturation and Intensity channels are not as useful since the histograms of two kinds of pixels totally overlap together.

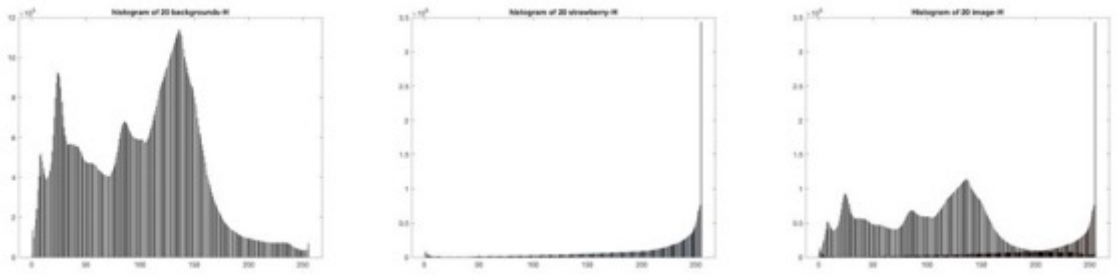

Figure 10 Hue Histogram of non-strawberry pixels (Left)Hue Histogram of strawberry pixels (Middle) Hue Histogram of all pixels (Right)

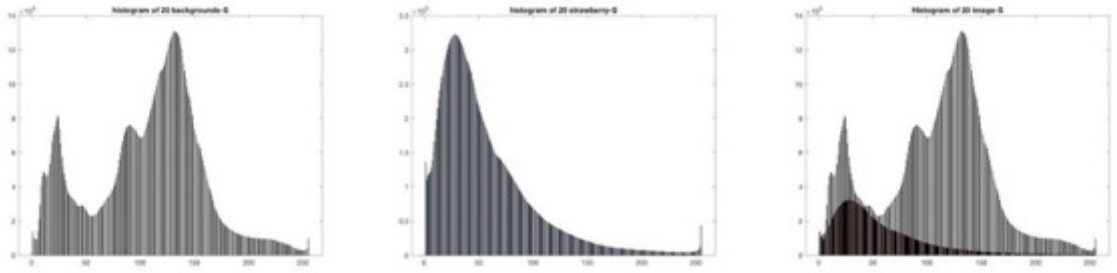

Figure 11 Saturation histogram of non-strawberry pixels (Left) Saturation Histogram of strawberry pixels (Middle) Saturation Histogram of all pixels (Right)

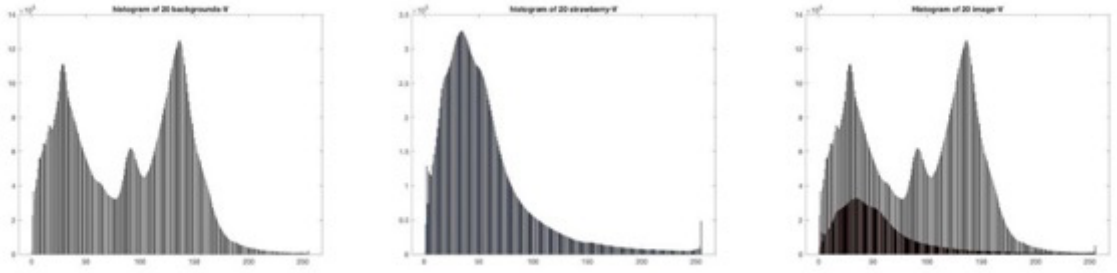

Figure 12 Intensity Histogram of non-strawberry pixels (Left) Intensity Histogram of strawberry pixels (Middle) Intensity Histogram of all pixels (Right)

2.1.4 Lab Color Space

Lab color model is composed of a brightness channel (L) and two-color channels: 'L' stands for brightness, 'a' represents the component from green to red. 'b' represents the component from blue to yellow[1]. After drawing the histogram of each space of the Lab, which were shown in Figure 13, Figure 14, and Figure 15, we can see the pixels from strawberries and background totally overlap in 'L' channel. In ‘b’ channel, part of the pixels from two classes are overlapping. However, 'a' component shows a great distinction between two different sets. So 'a' color channel is a good candidate to build the feature space later and it will be utilized later.

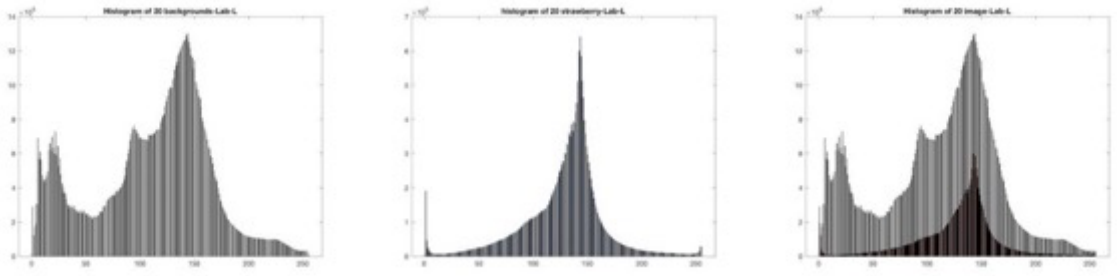

Figure 13 Lightness histogram of non-strawberry pixels (Left) Lightness Histogram of strawberry pixels (Middle) Lightness Histogram of all pixels (Right)

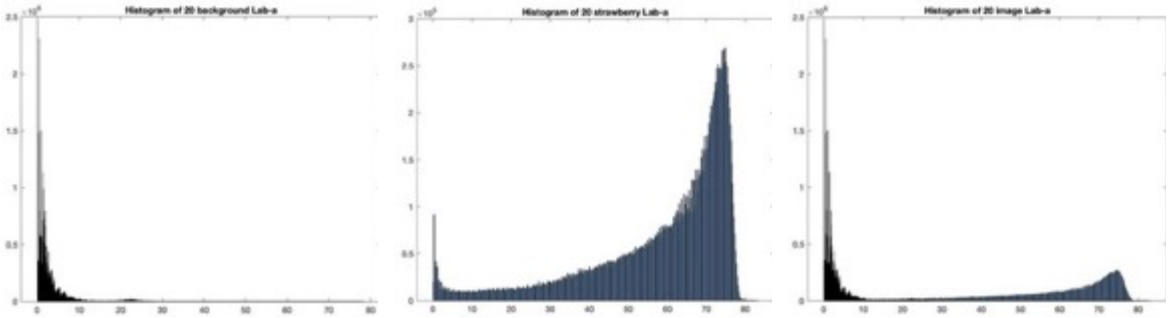

Figure 14 Green-Red (a) histogram of non-strawberry pixels (Left) Green-Red (a) Histogram of strawberry pixels (Middle) Green-Red (a) Histogram of all pixels (Right)

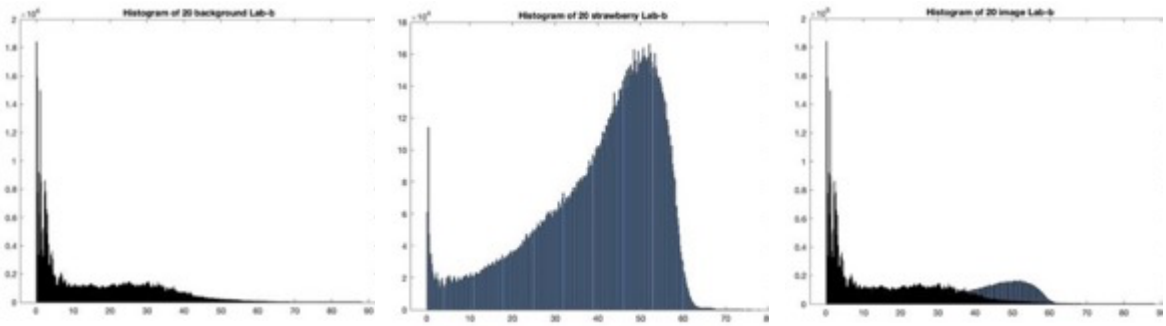

Figure 15 Blue-Yellow (b) histogram of non-strawberry pixels (Left) Blue-Yellow (b) Histogram of strawberry pixels (Middle) Blue-Yellow (b) Histogram of all pixels (Right)

Through the analysis of the four-color models, the R channel from the RGB model, the normalized r from normalized RGB model, the Hue channel from the HSV model, and ‘a’ color channels from the Lab model are good candidates for image segmentation for red strawberry. They will function as a coordinate system of feature space for future image segmentation.

2.2 Simple Threshold

In the previous section, we found that 200 is a critical threshold value to distinguish strawberry and non-strawberry pixels by observing the red channel. Thus, this value will be used for the simple image threshold.

Figure 16 Original input image (Left) Red channel of input image (Right)

Figure 16 shows the experimental image and its red channel. Noticeably, the strawberry in the red channel picture has a higher brightness, which means the pixels in this region have a higher intensity. Then setting 200 as the threshold value, all the pixels in the red channel image would have new values base the Function 2.2.1:

$$P(x,y)_{new} \begin{cases} 1, & P(x,y)_{old} \geq 200 \\ 0, & P(x,y)_{old} < 200 \end{cases} \quad (2.2.1)$$

Pixels with intensity less than 200 are set to 0; otherwise, they are set to 1. Figure 17 shows the binary picture for the following image thresholding.

Figure 17 Binary mask image

Figure 18 is the thresholding result. Even though all three strawberries appear in the resulting image, plenty of unwanted non-strawberry content still remains.

Figure 18 Thresholding resulting image

When using MATLAB built-in function regionpropsto draw bounding boxes, numerous undesirable white boxes remain in the image. Figure 19 shows this lousy result.

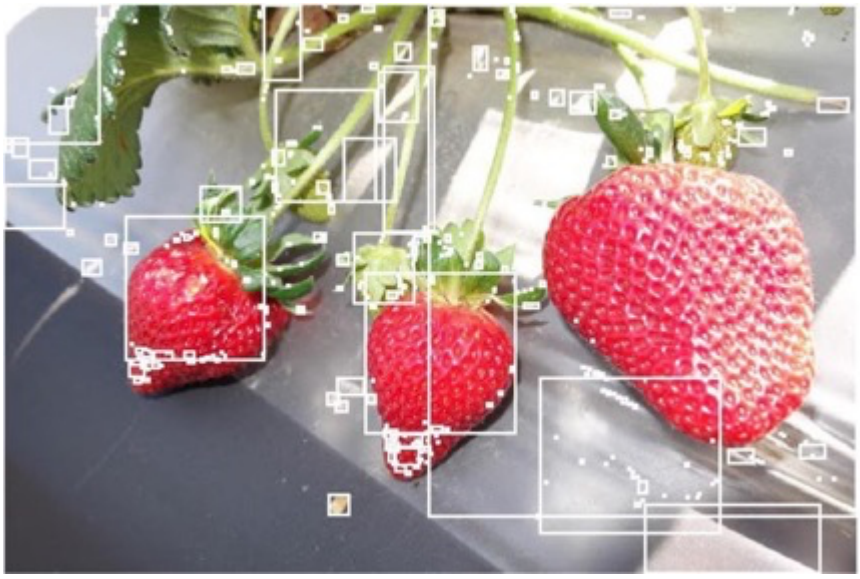

Figure 19 Detection resulting image

To improve this poor result, the morphological image technique is applied to the binary mask image to remove the undesired background part.

First, to erode the binary mask image. Figure 20 shows the eroded image. Comparing Figure 20 with Figure 17, many background parts have been removed.

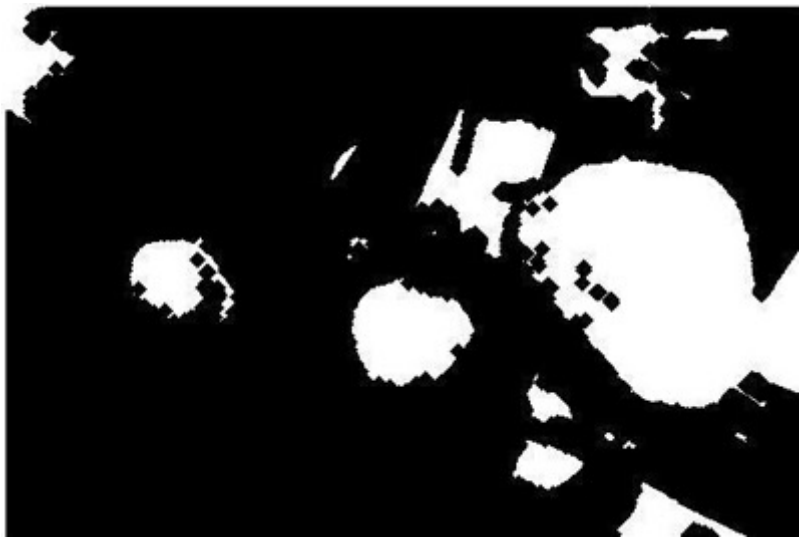

Figure 20 Binary mask image with erosion

In order to fill the small holes inside the strawberry area, the image dilation is applied to Figure 20, as shown in Figure 21. Finally, the third binary mask image is shown below.

Figure 21 Binary mask image with erosion and dilation

The final result is shown in Figure 22.

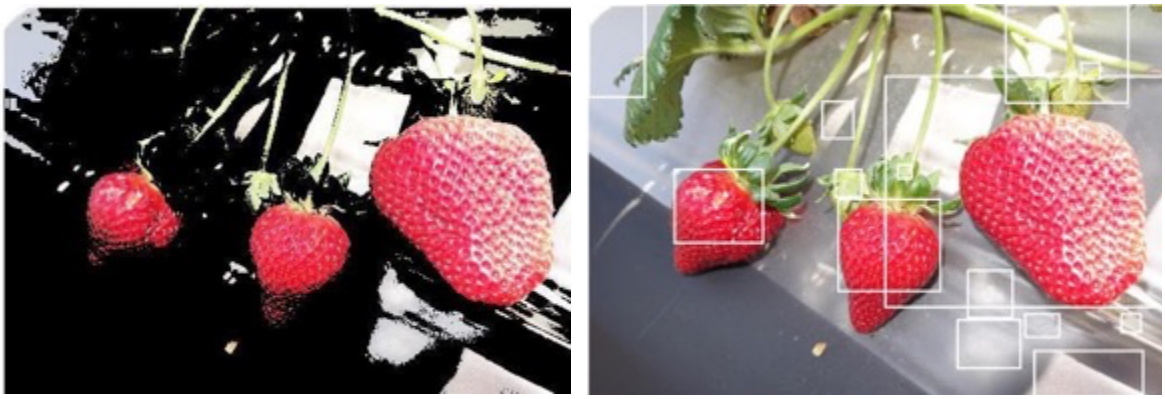

Figure 22 Strawberry detection result with image erosion and dilation

It does look better than Figure 18 and Figure 19. However, this red strawberry detection result is still not satisfactory. The reason being all light objects with R>200 will be detected regardless of the actual color. So in this case, only a single threshold value to split the image into two parts will not be able to segment the pure red strawberry out, and the use of hue or other color channels would help.

Therefore, experiments with more color channels combined with the other image segmentation method will be done in the next section.

2.3 K-Mean Clustering

2.3.1 Algorithm review

K-means algorithm is a clustering algorithm. The so-called clustering means that according to the principle of similarity, objects with high similarity belong to the same cluster in the feature space.[6] Otherwise, they are assigned into clusters of different classes. The k-means algorithm takes distance as the standard of similarity measurement between objects: the shorter the distance between objects, the higher the similarity between them, and the more likely they are in the same class cluster.

The k-means algorithm uses Euclidean distance to calculate the distance between data objects. In Euclidean space, Function (2.3.1) calculates the distance between the point X = (x1,…,xn) and Y = (y1,…,yn):

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \tag{2.3.1}$$

The given training samples are {x(1),…,x(m)}, where x(i) is an n-dimensional vector. Here those training samples are the pixels from the input image and the value of n depends on how many color channels will be used to construct the Euclidean space. The following are steps to implement K-Mean:

Step1: K points were randomly selected as clustering centroids, 1, 2, … ,k ϵ Rn, K is the number of the clusters.

Step2: For each pixel x(i), calculate the class to which it should belong based on Function (2.3.2). Where c(i) is the class x(i) belongs to, it goes from 1 to K.

$$c^{(i)} = arg\ min_j \|x^{(i)} - \mu_j\|^2 \tag{2.3.2}$$

Step3: Once all the pixels have been clustered, calculate the center of each cluster using equation (2.3.3).

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}} \tag{2.3.3}$$

Step4: Repeat step2 and step3 until all the cluster centroids do not change or the number it changed less than one predefined value.

Equation 2.3.4 calculates the sum of the squares of the distance from each sample point to its corresponding center. J is calculated every time after steps 2 and 3. Then compare the new J with the old one. Once the difference is less than the predefined threshold value, the iteration is terminated.

$$J(c,\mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2 \quad (2.3.4)$$

These are the concepts and steps of K-mean. In the next section, this algorithm will be implemented by using the red color channel from the RGB model, the normalized ‘r’ color channel, the H color channel from the HSV model, ‘a’ color channel from the Lab model, and the combination of different color channels. At the same time, different K values will be used to compare the corresponding experiment results.

2.3.2 Experiments

This section will conduct the following experiments to explore the suitable values of K for different color channels:

Experiment1: Use the red channel from the RGB model as the feature space.

In the previous section, simple image thresholding based on the red channel results in poor results. Thus, setting K equals to 2 may not make a difference from the simple thresholding as before. Therefore, the initial value of K here is set as 3 and increased to 5 successively.

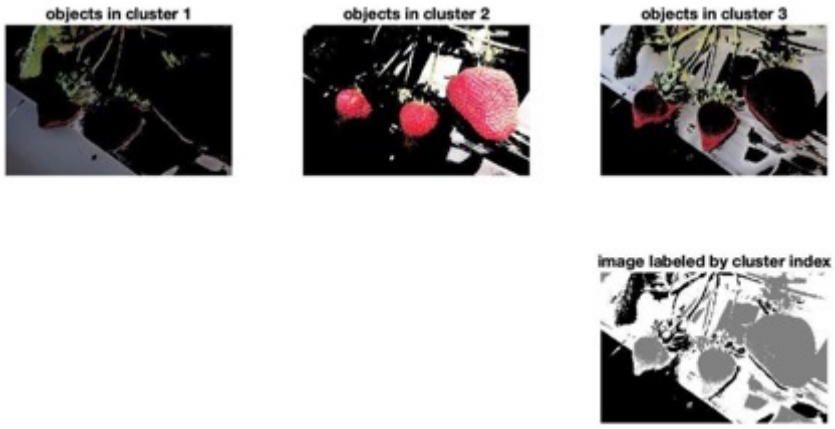

Figure 23 K-Mean application on Red channel (K = 3)

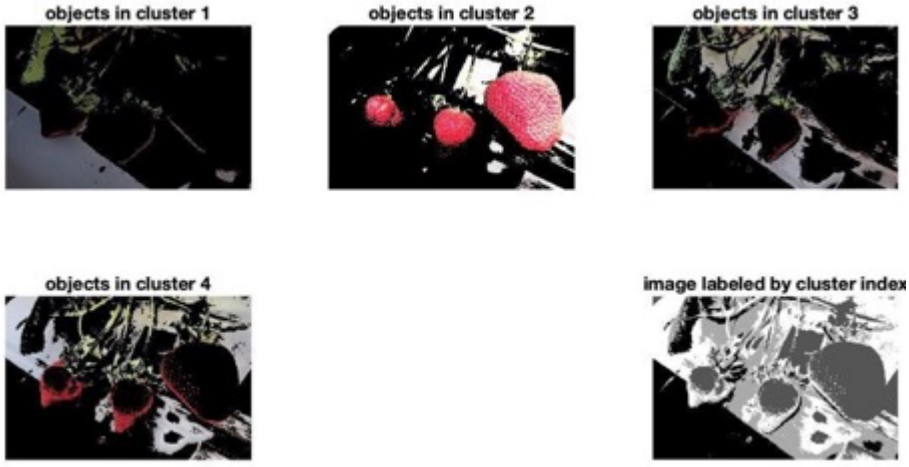

Figure 24 K-Mean application on Red channel (K = 4)

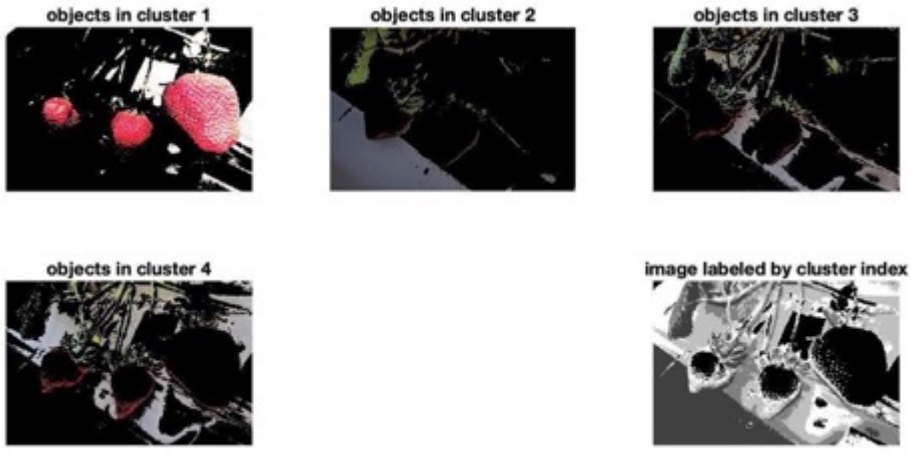

Figure 25 K-Mean application on Red channel (K = 5)

Figure23, Figure24, and Figure25 show the segmentation results with the different K-value. No matter how to change K-value, segmentation results are still poor. Therefore, the red channel is not a suitable space for separating red strawberries.

Experiment 2: Use normalized the r channel from the normalized RGB model as the feature space.

The initial value of K is set as 2 and increases by an increment of 1 until K = 5.

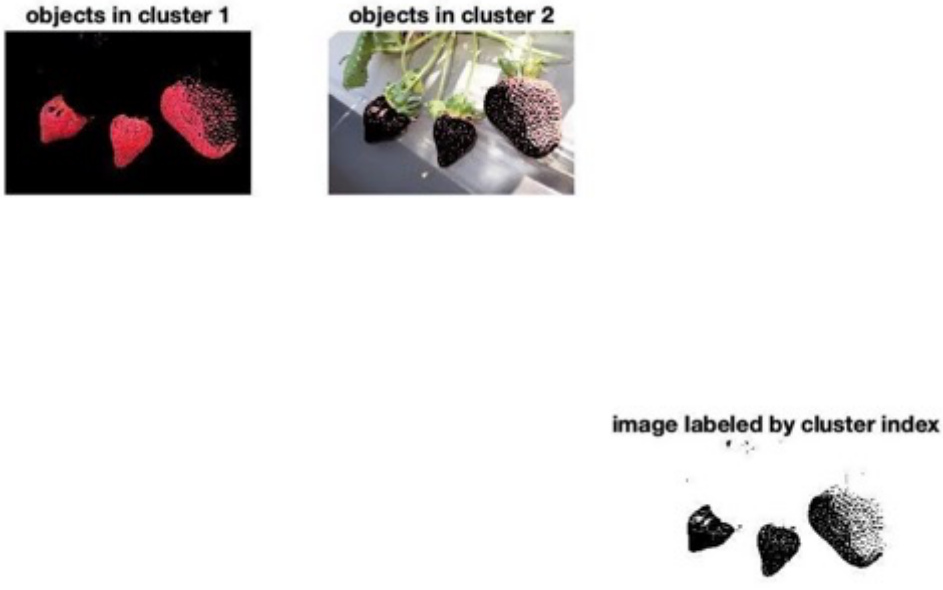

Figure 26 K-Mean application on Normalized Red channel (K = 2)

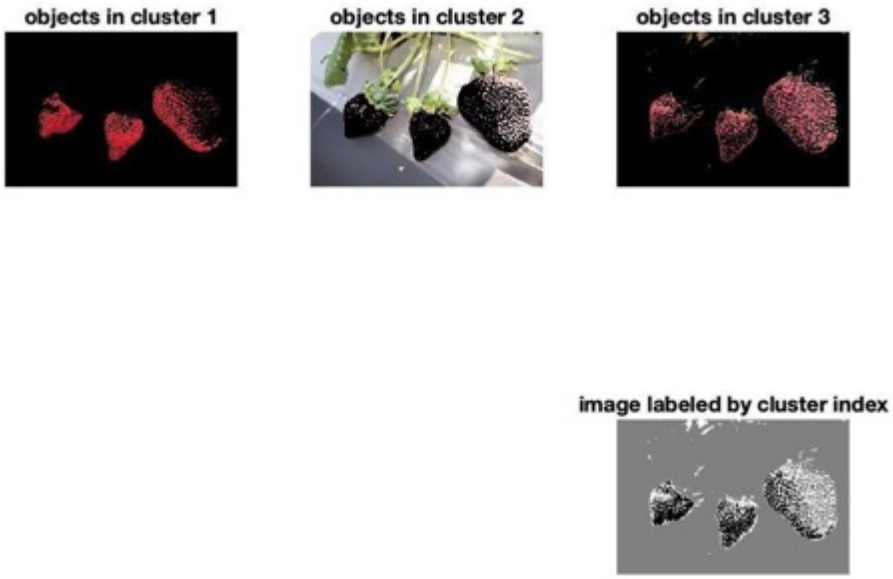

Figure 27 K-Mean application on Normalized Red channel (K = 3)

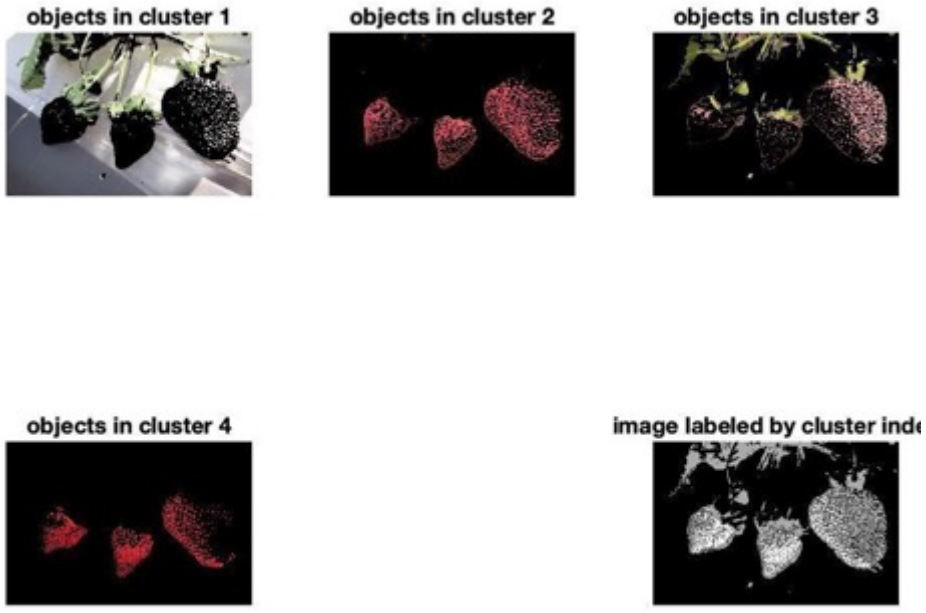

Figure 28 K-Mean application on Normalized Red channel (K = 4)

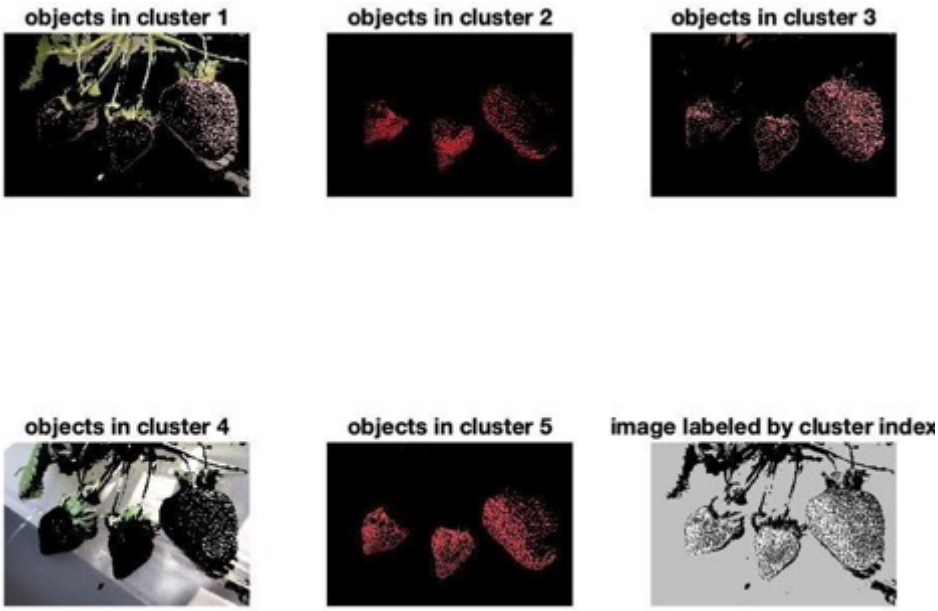

Figure 29 K-Mean application on Normalized Red channel (K = 5)

When setting K=2 (Figure26) has already presented a nice result where red strawberries have been extracted without any background left in cluster1. However, continuing increasing K- value does not bring out any improvement. As shown in Figure 27, Figure 28, and Figure29, the clusters containing red berries are getting blurrier. Therefore, the most suitable K-value for normalized r channels is 2.

Experiment 3: Use Hue channel from the HSV model as the feature space.

The initial value of K here is set as 2 and increased to 5 successively.

Figure 30 K-Mean application on Hue channel (K = 2)

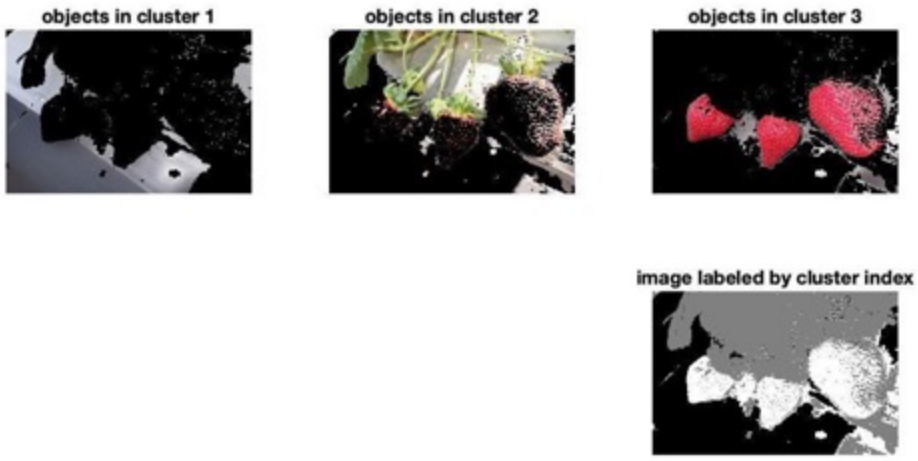

Figure 31 K-Mean application on Hue channel (K = 3)

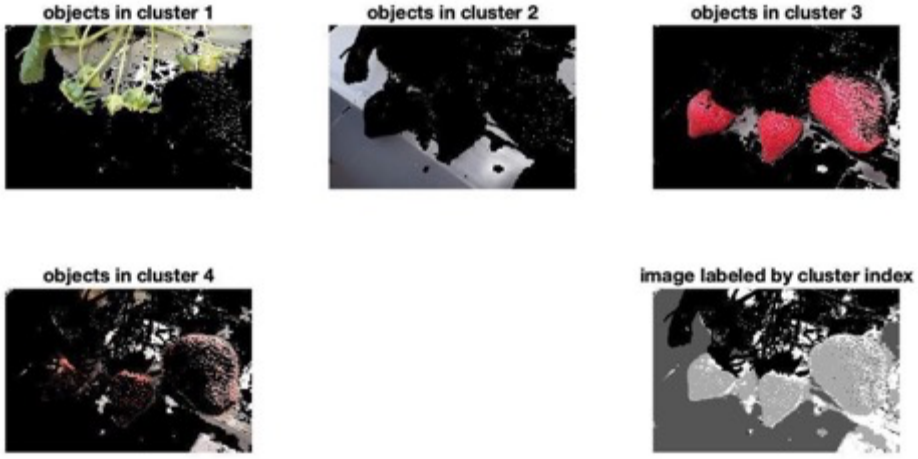

Figure 32 K-Mean application on Hue channel (K = 4)

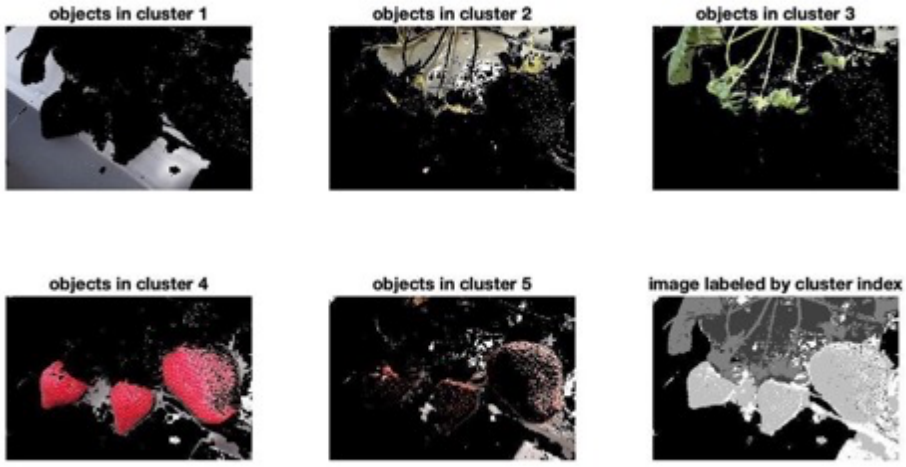

Figure 33 K-Mean application on Hue channel (K = 5)

By observing Figure 30, Figure 31, Figure 32 and Figure 33, no matter how the K value was adjusted, none of the clusters was able to isolate the image of strawberries only. Thus, unlike the normalized r, the K-mean cluster algorithm cannot rely on the Hue channel exclusively to separate the red strawberry from its background.

Experiment 4: Use ‘a’ channel from the Lab model as the feature space.

The initial value of K here is set as 2 and increased to 5 successively.

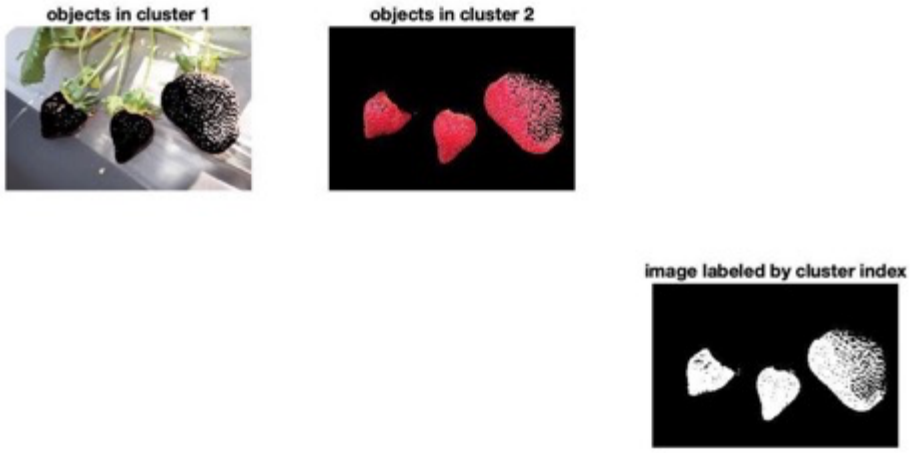

Figure 34 K-Mean application on ‘a’ channel (K = 2)

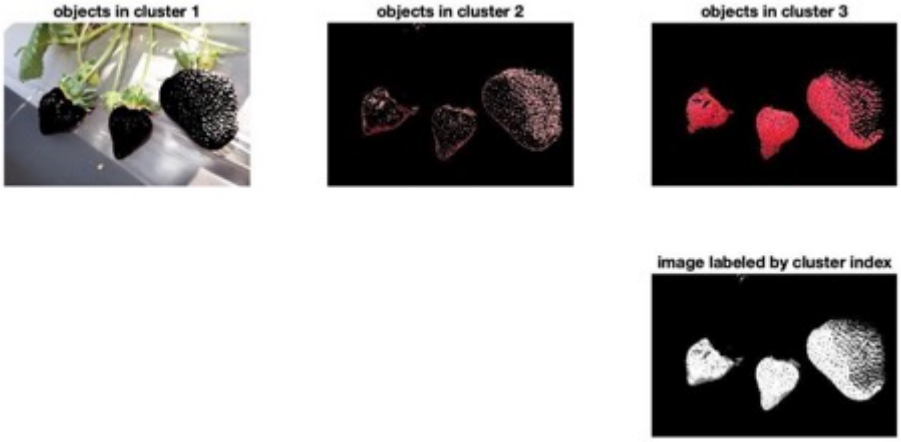

Figure 35 K-Mean application on ‘a’ channel (K = 3)

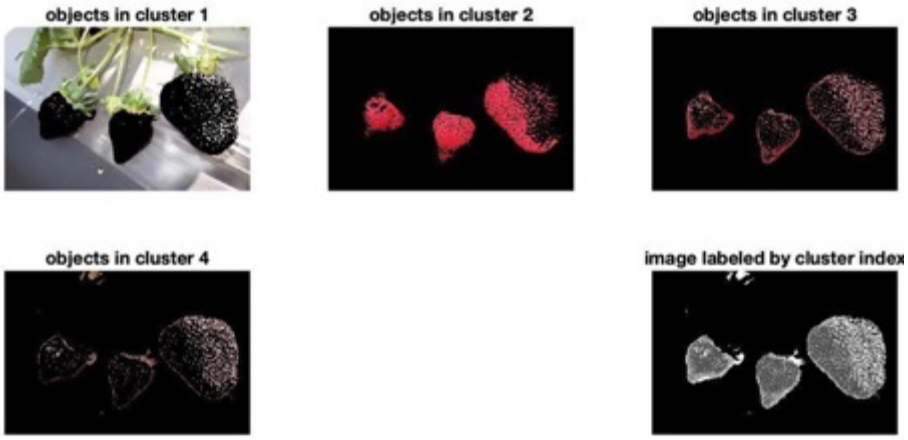

Figure 36 K-Mean application on 'a' channel (K = 4)

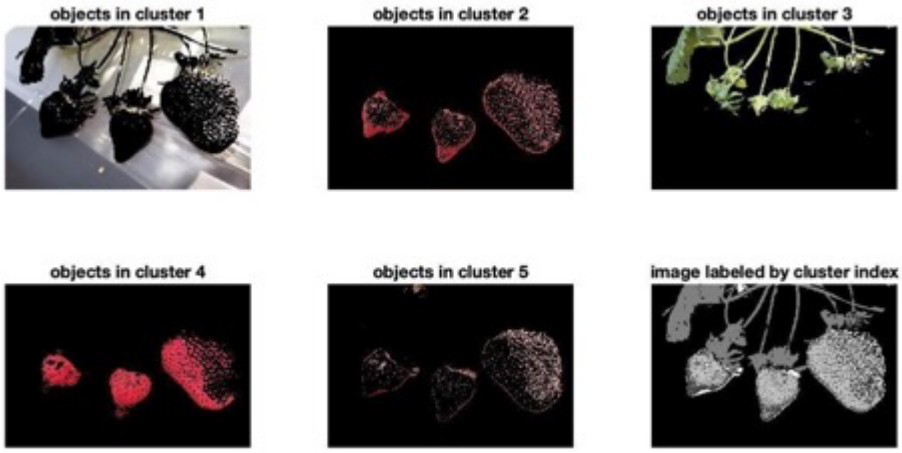

Figure 37 K-Mean application on 'a' channel (K = 5)

Same as the normalized r plane, only 'a' channel from Lab color space could support image segmentation, and the best K-value is 2(Figure 34). Larger K-values undermine the segmented results (Figure35, Figure36, and Figure 37).

Experiment 5: Use the combination of normalized r, Hue and 'a' color channels to build the feature space

As we excluded the feasibility of the red color plane from the RGB image based on the previous experiments, the normalized r, Hue, and 'a' color planes are combined to build the feature space. The value of K is initially set as 2 and increased to 5 successively.

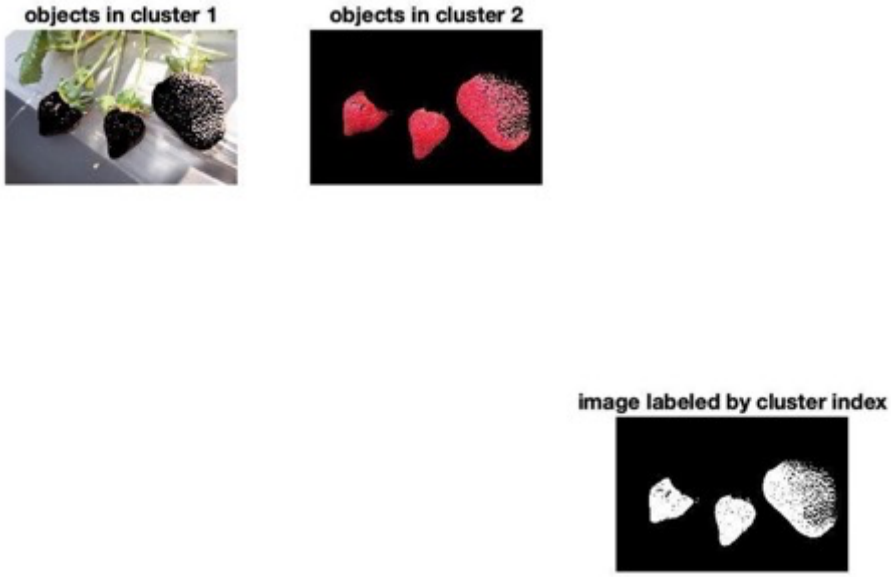

Figure 38 K-Mean application on normalized red, Hue and 'a' channel (K = 2)

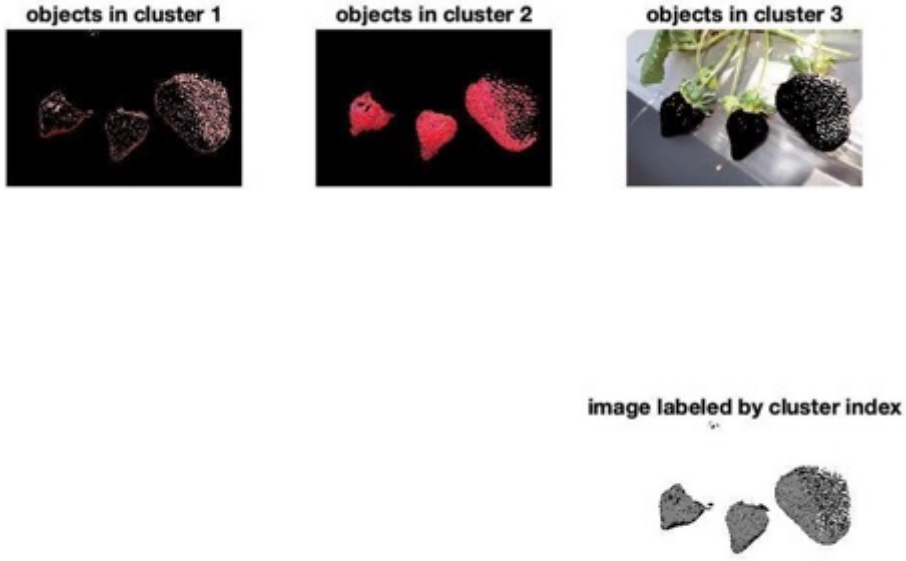

Figure 39 K-Mean application on normalized red, Hue and 'a' channel (K = 3)

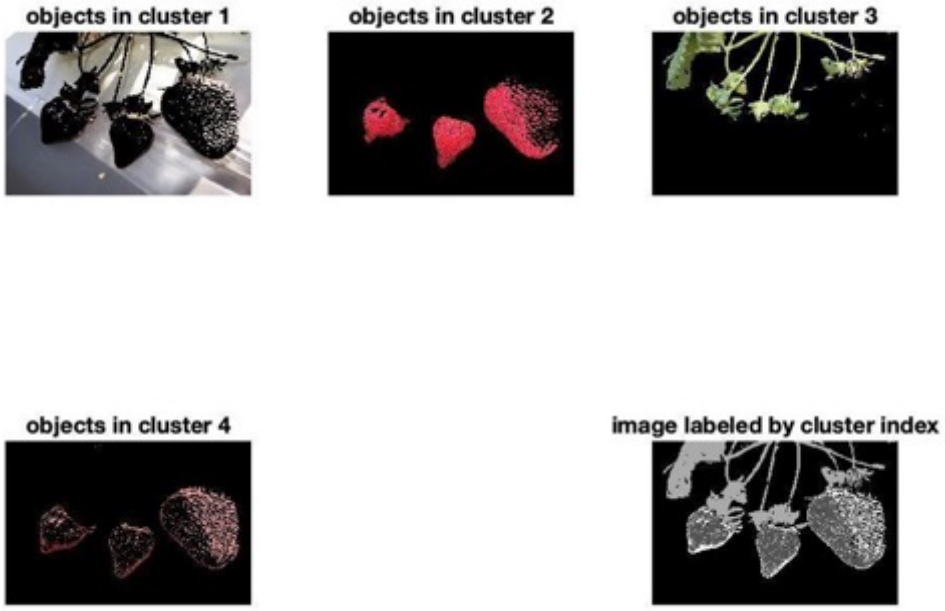

Figure 40 K-Mean application on normalized red, Hue and 'a' channel (K = 4)

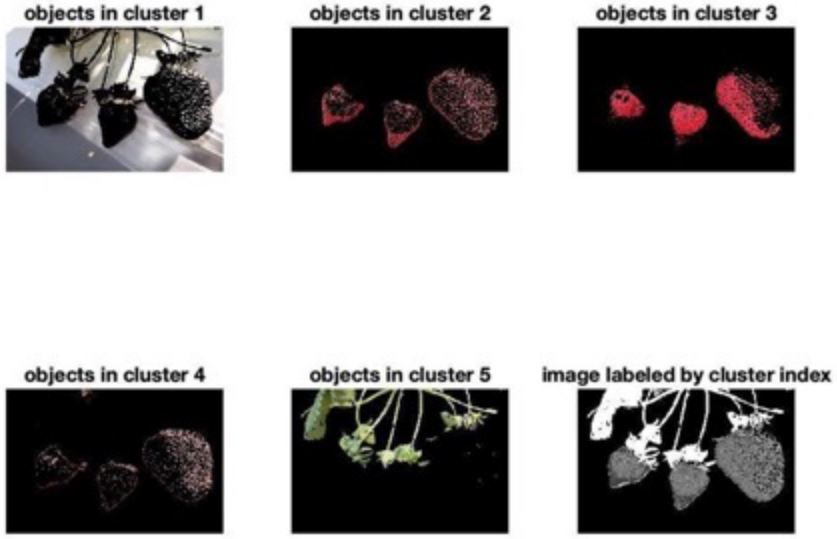

Figure 41 K-Mean application on normalized red, Hue and 'a' channel (K = 5)

As shown in Figure 38, Figure39, Figure40, and Figure41, a combination of color spaces can be utilized to distinguish the pure red strawberry from the background. Similarly, when K-value is 2, the separation result is optimal.

2.3.3 Testing

40 test images which contain 337 red strawberries in total are used to test the effectiveness of k-means clustering in red strawberry detection.

Based on all previous experiments, the normalized r channel, the 'a' channel, and the combination of normalized r, 'a', and Hue color planes all support red strawberry segmentation. The best k value is 2.

According to this conclusion, image segmentation is applied based on three kinds of feature spaces on all testing images respectively. Then counting the detection results, which was shown in Table1.

Table 1 Image Segmentation Results in Different Feature Spaces

Color Plane	Total Number	Truth Positive	False Positive	False negative
Normalized r	337	305	33	22
'a'	337	305	33	22
Normalized r, 'a', and Hue	337	306	35	21

All the detectors show the sensitivity to the red color, and there are not vast differences in their detection results (Figure42)

Figure 42 Truth Positive Samples based on normalized r(left),

Truth Positive Samples based on 'a'(middle),

Truth Positive Samples based on normalized r, 'a', and Hue (right)

Even if the light condition changes, strawberries in the shadow can still be detected (Figure42, Figure44, and Figure45).

Figure 43 Strawberry in Shadow (Normalized r)

Figure 44 Strawberry in Shadow('a')

Figure 45 Strawberry in Shadow (Normalized r, 'a', and Hue)

The false-positive samples in the table are mainly from red leaves and partially red in immature strawberries (Figure46).

Figure 46 False-Positive Samples

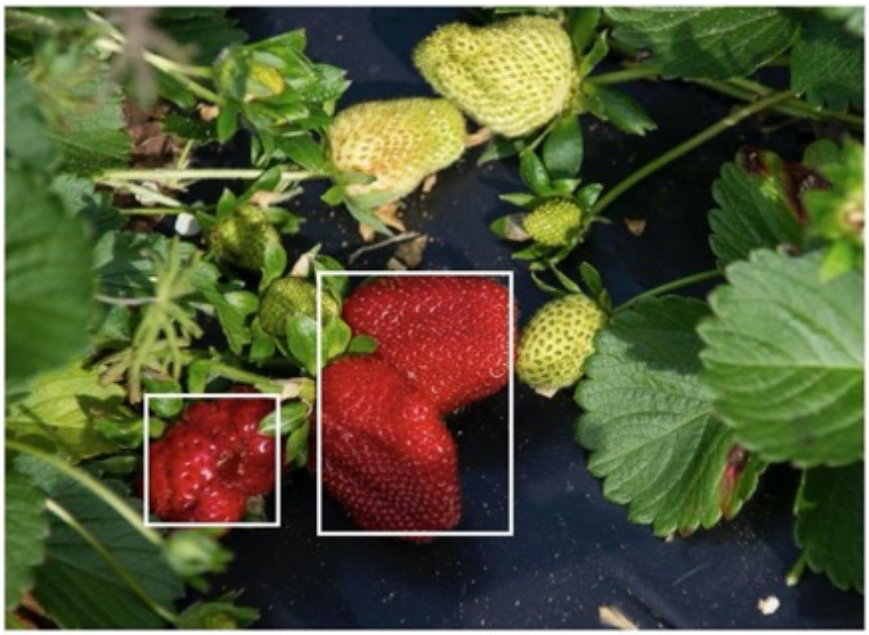

Figure 47 False-Negative Sample

As shown in Figure 47, when two strawberries are close to each other, the detector will often treat them as one strawberry, which is also the main reason for the false-negative samples in Table1. As shown in Figure 47, when two strawberries are close to each other, the detector will often treat them as one strawberry, which is also the main reason for the false-negative samples in Table1. The future work

for improving the detection results on red strawberry will rely on other information, such as strawberry's shape and texture.

In the following section, this thesis will focus on exploring and comparing methods for green strawberry detection.

Chapter 3

Green Strawberry Detection Based on Classical Computer Vision Methods

During strawberry's growth, the time for the strawberry to turn red from green is often short, generally less than a week. Therefore, knowing the number of green strawberries is also of great significance for forecasting strawberries' harvest. In this Chapter, the focus of detection will shift from red strawberries to green strawberries. This chapter uses the traditional computer vision algorithm: Support Vector Machine (SVM) based on Histogram of Oriented (HOG) feature to detect green strawberries.

3.1 Introduction

In this chapter, HOG and SVM will be combined for strawberry detection. To get a higher truth-positive rate, parameters have been adjusted for extracting HOG features and training SVM. Besides, two types of hard negative mining were applied to reduce the false-positive rate.

3.2 Algorithm Review

Histogram of Oriented Gradient (HOG) feature is a feature descriptor used for object detection in computer vision and image processing. It constructs features by calculating and counting the gradient direction histogram of the local area of the image. The HOG feature, combined with the Support Vector Machine classifier, has been widely used in object recognition and detection [7]. French researcher Dalal proposed the method of HOG + SVM for pedestrian detection at the 2005 CVPR [7].

The processes of HOG feature extraction are as follows:

Step1: Convert the input color image to grayscale image.

Step2: Applying gamma transformation to reduce the impact of local shadows and lighting changes.

Step3: Calculate the horizontal and vertical gradients of each pixel in the image. The derivative operation can not only capture the contour and some texture information but also further weaken the influence of lighting. The gradient of pixels (x, y) is:

$$G_x(x,y) = I(x+1,y) - I(x-1,y)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \quad (3.2.1)$$

Where $G_x(x,y)$, $G_y(x,y)$and $I(x,y)$ represent the horizontal gradient, vertical gradient, and intensity of point (x, y) in the input image.

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

$$\alpha(x,y) = tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \quad (3.2.2)$$

The magnitude and direction of the gradient at point (x, y) are Gx,y and x,y.

Step4: We divide the image into several "cells," for example, each cell contains 8 * 8 pixels. And we use 9 bin histograms to count the gradient information of these 8 * 8 pixels.

Step5: Every 4 cells form a block and the feature descriptors of all cells in a block are connected in series to obtain the HOG feature descriptor of the block. The descriptor for each block will be normalized then so that it is not affected by changes in light. Equation 3.2.3 shows the way to calculate the number of blocks in an image, where blocks overlap in Figure48 is the stride.

$$block_{num} = \left(\frac{Image_{width} - block_{width}}{stride} + 1\right)\left(\frac{Image_{high} - block_{high}}{stride} + 1\right) \quad (3.2.3)$$

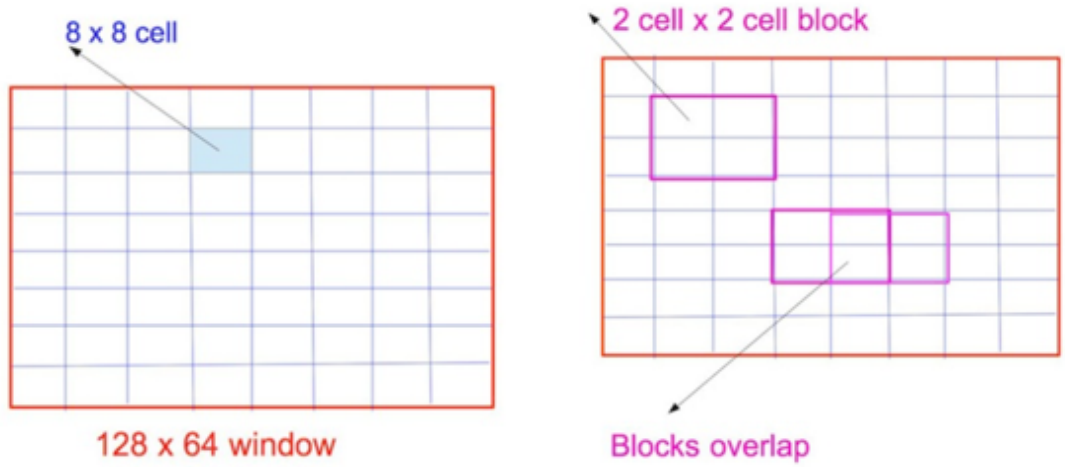

Figure 48 Relationship between cell and block

Thus, given an image with size of [128 64] (Figure 47), it contains 105 blocks.

Step6: Concatenating the HOG feature descriptors of all blocks in the image to get a large vector representing the HOG feature descriptor of the whole image. As the HOG feature descriptor extracted from each block will have a length of 36, the dimensionality of the HOG vector extracted from the image (Figure47) is 3780.

Once the HOG feature vectors are extracted for every training image, they will be sent to train the support vector machine (SVM) classifier.

3.3 Image dataset

All images in the database are taken from the Cal Poly strawberry center.

The positive training samples are the single strawberries cropped out from the original photo. Three kinds of mirror transformations for each initial positive image (Figure 49) will help to increase the number of positive images. Therefore, the number of positive samples is increased to four times the original. Ultimately, the number of positive samples was 2400 and the number of negative samples was 8000.

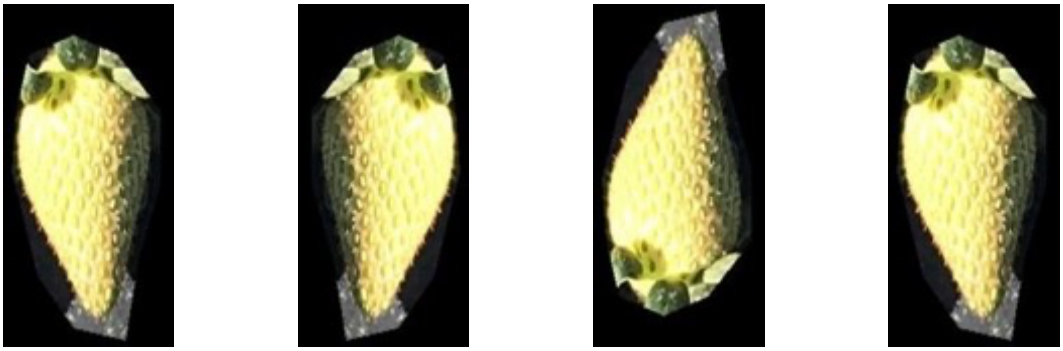

Figure 49 Positive sample and its mirror transformations

To facilitate the extraction of HOG features, two sizes of pictures were created when generating a picture library: [128, 64] and [256, 128]. Each contains the same number of positive and negative samples.

3.4 Implementation and experiments

When extracting HOG features from all training pictures, some important parameters need to be determined first: the size of the input picture and the size of the cell.

As mentioned before, the image database contains two sizes of pictures: [64, 128] and [128, 256]. First, fix the size of the picture (Figure50) to [64, 128], then set the cell size to [4, 4] [8, 8] and [16, 16], and extract the HOG features from the picture respectively. The Figure51 shows this result.

Figure 50 Positive sample with size [128, 64]

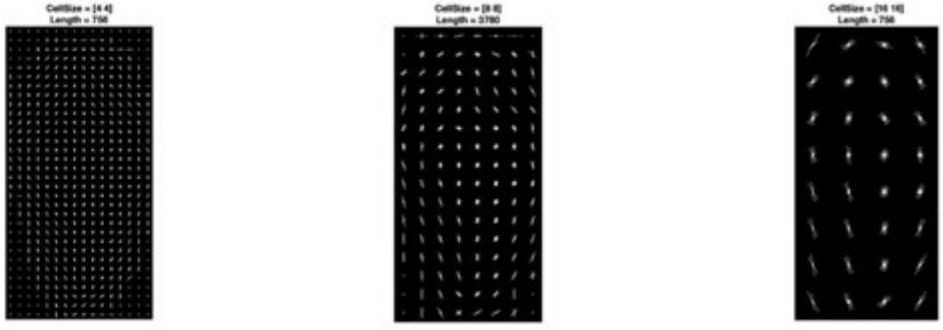

Figure 51 *HOG feature image with cell size [4 4] (Lift)*

HOG feature image with cell size [8 8] (Middle)

HOG feature image with cell size [16 16] (Right)

Compare the HOG feature images in Figure 51. Obviously, the information offered by the image with cell size [16 16] is too scarce to be considered. Compare the cell size [4 4] and [8 8]. Both pictures provide clear strawberry shapes. However, the left one contains too much information, which will increase the time to train the classifier later. Thus, setting [8 8] as the cell size will be the best choice.

Change the size of the input picture size to [256, 128], repeat the above steps,

Figure 52 Positive sample with size [256, 128]

Figure 53 *HOG feature image with cell size [4 4] (Lift)*

HOG feature image with cell size [8 8] (Middle)

HOG feature image with cell size [16 16] (Right)

The three HOG feature pictures in Figure 53 all provide a clear strawberry shape. Out of the consideration to save training time, the cell size is set as [16, 16].

Now there are two options for the cell size: Input [128, 64] image with cell size [8, 8] or input [256 128] image with cell size [16 16].

Then these two sizes will be used to train the classifier and determine which one is better by the classification result.

Table 2 Hog feature parameter comparison

	Image size [128, 64]		Image size [256, 128]	
Cell Size	Truth Positive Rate	False Positive Rate	Truth Positive Rate	False Positive Rate
[8 8]	0.92	0.122	N/A	N/A
[16 16]	N/A	N/A	0.88	0.129

The image database is randomly divided into the training set and the testing set based on the ratio of 9: 1. Table 2 shows the classification results on the testing set. The input images of [128, 64] with the cell size of [8 8] have a higher truth positive rate and a lower false positive rate. Therefore, in the subsequent training, this combination will be used to extract HOG features.

The whole process of implementing HOG features and SVM classifier is shown in the following flowchart:

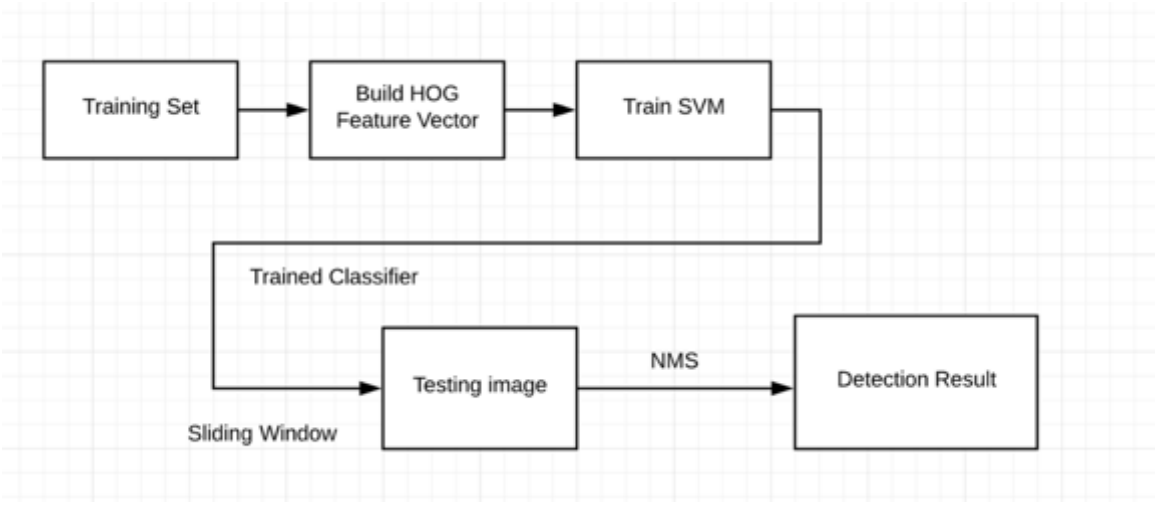

Figure 54 Object Detection Process

During the process of analyzing and comparing HOG features, a well-trained classifier was used. This classifier will be used directly on object detection. As shown in the flow chart above, given a test image, sliding windows of different sizes will go through the entire image. HOG features will be extracted from those window boxes and then sent to the trained classifier to determine which window box contains the object.

There are two methods to create a sliding window, we will compare them and choose the more suitable one:

The first one is to keep the size of the sliding window, moving step length along the horizontal and vertical direction, while changing the size of the test image whenever the sliding window traverses the entire test picture.

Figure 55 Changed testing image (scale = 0.2)

Figure 56 Changed testing image (scale = 0.3.5)

Figure 57 Changed testing image (scale = 0.5)

Figure 58 Changed testing image (scale = 0.65)

Figure 59 Changed testing image (scale = 0.8)

Now, to change the size of the test picture according to the ratio of 0.2 (Figure 55), 0.35 (Figure 56), 0.50 (Figure 57),0.65 (Figure 58), 0.8 (Figure 59) of the original test image successfully. Apparently, this method of changing the size of the picture with a fixed window size does not work well. As can be seen from Figure54, the size of the sliding window is much larger than the strawberry in the picture. When the classifier determines that this window contains an object, it cannot draw the bounding box accurately around the object.

Then, we consider the second type of sliding windows. On the contrary, another method keeps the size of the test picture unchanged while changing the size of the sliding window and moving steps length in the horizontal and vertical directions.

Take the size of the sliding window based on the ratio of 0.08 (Figure 60), 0.1 (Figure 61), and 0.12 (Figure 62) of the testing image, successfully. The horizontal and vertical moving steps length are always proportional to the width and length of the sliding window.

Figure 60 Changed sliding window (scale = 0.08)

Figure 61 Changed sliding window (scale = 0.1)

Figure 62 Changed sliding window (scale = 0.12)

We can tell that these kinds of sliding windows have met the needs of all different size strawberries in the figures.

Thus, the second type of sliding window that tuning the size of the sliding windows will be utilized for object detection then. Besides, if there are larger or smaller strawberries that cannot fit in the current sliding windows, just tune the ratio mentioned before to adjust the window size.

Combining the sliding window and the trained classifier, we perform object detection on the test image. Figure 63 shows the detection results.

Figure 63 Detection result by HOG and SVM

The results shown in Figure 62 are disappointing. The reason is that after extracting the features of the sliding window and classifying it by the classifier, each window will get a classification label and

score. However, oftentimes more than one window will be classified as true and multiple overlapping of bounding boxes are observed.

According to the last step of the flowchart in Figure 53, non-maximum suppression is used to remove unwanted bounding boxes.

Non-maximum suppression (NMS) is widely used in object detection. Its purpose is to eliminate redundant frames and find the best position for the detected object [10].

Suppose that there are 6 candidate boxes, which are sorted according to the classification probability of the classifier, and the probability from small to large is A, B, C, D, E, and F, respectively. The steps of non-maximum suppression are as follows:

1. Starting from the box F that has a maximum probability, determine whether the overlap IOU of A~E and F is greater than a certain predefined threshold value.

IOU refers to intersection-over-union, the overlap rate of the target window, and the original marked window [8].

Function (3.2.1) shows how to compute IOU:

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.4.1}$$

2. Assuming that IOU of B, D, and F exceeds the threshold value, then throw away B and D (because the threshold is exceeded, it shows that D and F or B and F already overlap a large part, then we reserve the box F is enough. The remaining small areas of B and D are redundant. F can be used to represent an object, so retain F and discard B, D), and mark the box F as a true bounding box.

3. Among the remaining boxes A, C, and E, since box E has the highest probability, it will function as the box F. Compute the IOU between E and A, C. If the overlap is greater than the threshold value, then throw it away, and mark E as another a true bounding box.

4. Repeat this process until there is no box left. Then all the marked bounding boxes are the true boxes we need.

Figure 64 Detection result by HOG and SVM with NMS

After processing the resulting image with the non-maximum suppression, many unnecessary bounding boxes in the picture have been removed. Figure63 shows the new detection results.

The detected red strawberries are not our target in this section, they can be removed easily by using color information. However, 8 false-positive bounding boxes mistakenly classified non-strawberry as strawberries. Since the truth positive rate of the classifier is not that low, I don't want to retrain a new

classifier. So, how to improve this classifier without training a new classifier? The use of hard negative mining will be integrated.

Sung and Poggio first introduced the concept of hard negative mining [9] to reduce the false-positive rate. Hard negative samples refer to negative samples that are easily misclassified into positive samples. After the trained classifier has classified the positive samples and negative samples from the test images, put the hard negative samples into the negative sample set, and then send the negative sample back to continue to train the classifier.

Before applying the hard negative to advance the classifier, we need a new type of data, ground truth image. It refers to the image with the objects that have been manually marked, as shown in Figure65.

Figure 65 Ground truth image

First, randomly take samples in the ground truth image. When the overlap rate between these samples and ground truth exceeds 0.5, it is considered to be a positive sample. Otherwise, it is a negative sample.

There are two kinds of hard negative samples. The first one is the false sample that is wrongly classified as positive and overlaps with the object. As a result, it will contain some parts of the object while the IOU is less than 0.5.

The red box in Figure 66 shows this kind of hard negative sample.

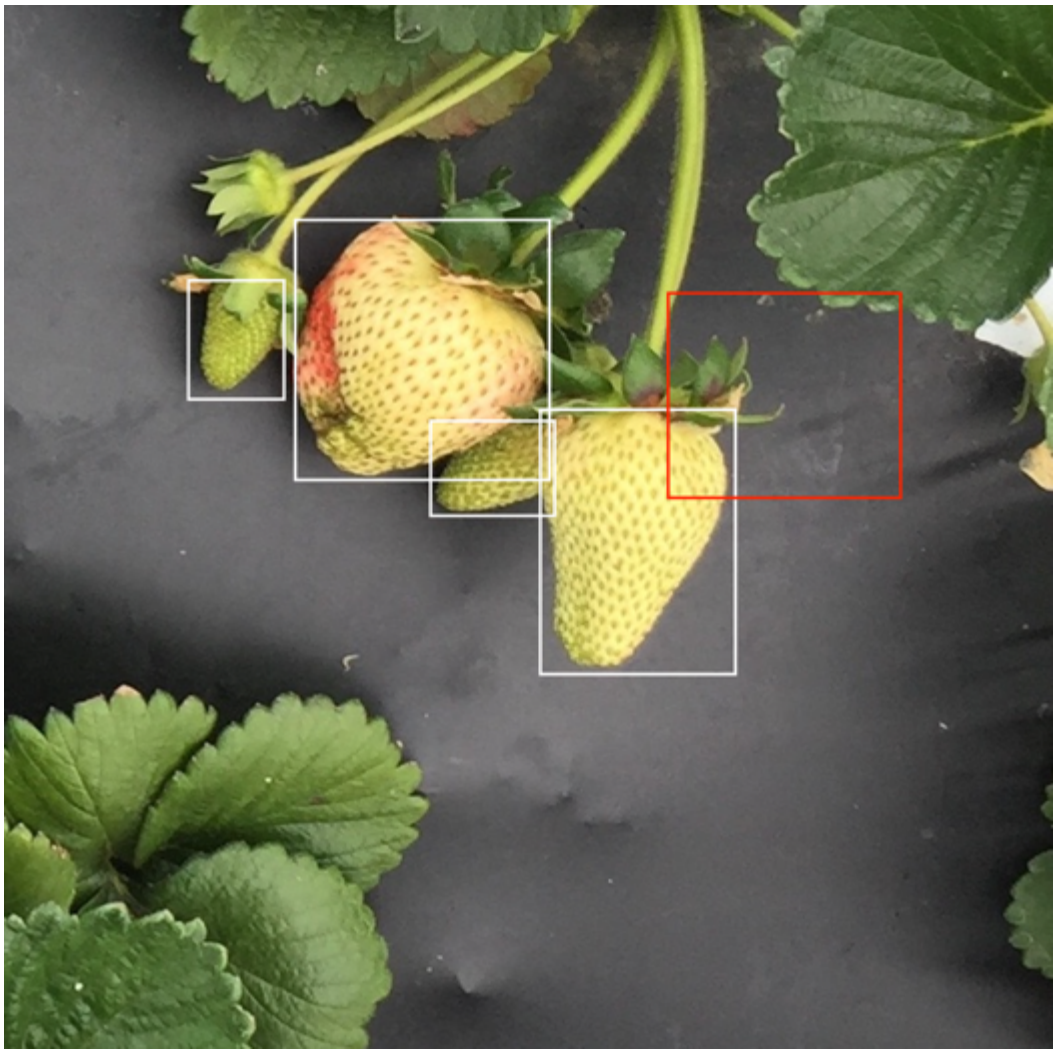

Figure 66 Hard negative (first type)

In addition to the labels, all samples classified by the classifier will also get probabilities that show how likely these samples are the object we want. Thus, the second hard negative is the false positive sample with a high probability. And they can easily mislead the classifier with that high probability.

In the following part, we will attempt to improve the classifier by exploiting these two kinds of hard negative samples.

In order to obtain a sufficient number of hard negative, another 200 ground truth images have been created. 50 samples are randomly selected from each image and sent them to the classifier for classification. Once got all the new false positive samples from the classification results, first to calculate the IOU between each false positive sample and its corresponding ground truth. Then the false-positive samples with a non-zero IOU are treated as hard negative. Finally, we extract the HOG feature vectors from all hard negatives samples and add them into the classifier's training sample to retrain the SVM classifier.

Figure67 shows the detection result with the new classifier.

Figure 67 Detection result with first type of hard negative mining

The number of false-positive samples in the images has reduced from 8 to 2, which proves the effectiveness of hard negative mining. However, the number of positive samples was also significantly reduced.

There are two speculated reasons for the decrease in the truth positive rate: One is that as a large number of the hard negative are added to the classifier training set, the number of positive samples in the classifier training set is much smaller than the number of negative samples, resulting in a poor distribution of the training set. Another reason is the particularity of strawberries. Strawberry texture and shapes are the keys to the HOG+SVM detector. When some hard negatives contain part of strawberries, they may confuse the classifier.

Now, we will try to use the second type of hard negative sample to improve the classifier. Same as before, 50 samples are randomly selected from each ground truth image and classified. Then we collected all the new false positive samples and sorted them according to their probability. The first 800 false-positive samples are treated as hard negatives. We put them into the negative training sample set and then sent them back to retrain the classifier. Simultaneously, 500 additional truth-positive samples are randomly selected and put into the positive training sample set.

Figure 68 Detection result with second type of hard negative mining

Compared with the previous detection results, the newly promoted classifier maintained a similar truth positive rate and significantly reduced the false positive rate. Thus, we will use the classifier that has been improved by the second hard negative sample to do the final test.

Table 3 Performance of HOG + SVM detector

Total Number	Truth Positive	Truth Positive Rate	False Positive
534	433	81.11%	127

There are 45 testing pictures with a total of 534 green strawberries to exam the detector. The final result is shown in Table 3. However, the SVM classifier did not achieve a decent detection result even though the hard-negative mining has improved its performance, and the small green detection (Figure 69) is the most prominent problem that impacts the detection result.

Figure 69 shows the green strawberries in the different growth stages. The big green has a texture that is more similar to a ripe strawberry. And it will not take a long time for large green to grow to maturity. In contrast, small green has a tinier texture. That difference in texture may cause the classifier to ignore the small green.

Figure 69 Small Green (Left),Big Green(Right)

Therefore, we will explore new detection models based on deep convolutional neural networks in the next section.

Chapter 4

Green Strawberry Detection Based on Deep Learning

4.1 Introduction

In the previous chapter, the traditional computer vision method, HOG and SVM, provided a 81.11% precise rate when detecting green strawberries. However, this detector's performance is not satisfactory, In the following part, we will attempt to improve the truth-positive rate by exploring and training deep learning models.

In this chapter, the TensorFlow object detection Application programming interface (API) will be the tool to train the detector for green strawberry detection. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models [15], which is based on transfer learning.

Transfer Learning

As Pan and Yang have defined in their paper[16]: Given a source domain Ds and learning task Ts, a target domain DT and learning task TT, transfer learning aims to help improve the learning of the target predictive function fT(·) in DT using the knowledge in Ds and Ts, where Ds≠DT, or Ts≠TT.

Generally speaking, transfer learning is to use existing knowledge to obtain new knowledge. Because it is too costly to learn directly from the target domain from scratch, we turn to using the existing relevant experience to assist in learning new knowledge as soon as possible.

The core issue of transfer learning is connecting the new knowledge to already existing content and finding the similarities between them.

To perform transfer learning, we first train a base network on a base dataset and then reuse the learned features on the target network. This process will tend to work if the features are general, that is, suitable to both base and target tasks, instead of being specific to the base task.

Although many details of how deep learning models work are still a mystery, it has been realized that the lower convolutional layers capture low-level image features, such as edges (Figure70), which are all common to both base and target tasks.

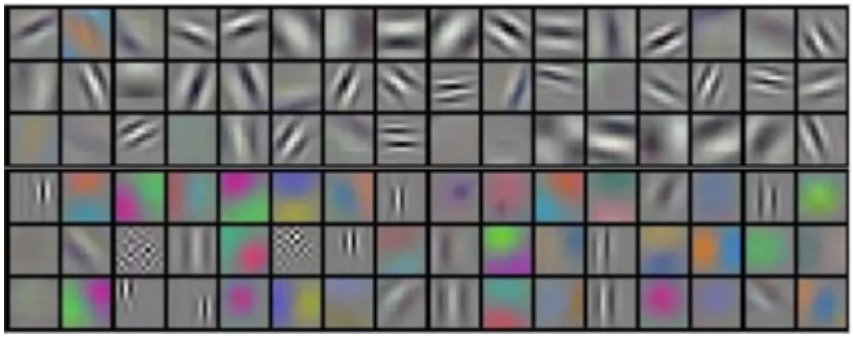

Figure 70 The lower convolutional layer example

In practice, we usually don't initialize and train deep learning convolutional neural networks from scratch. Which always requires a dataset of sufficient size to meet the needs of deep networks. Instead, it is common to pre-train a deep learning model on a large data set and then use this pre-trained model as an initial setting or as a fixed feature extractor for related tasks (For example, ImageNet, which contains 1.2 million images with 1000 categories). Then it will be used as an initialized or fixed feature extractor to complete the task of interest.

Figure 71 shows the workflow of creating the dataset and training models when using this API.

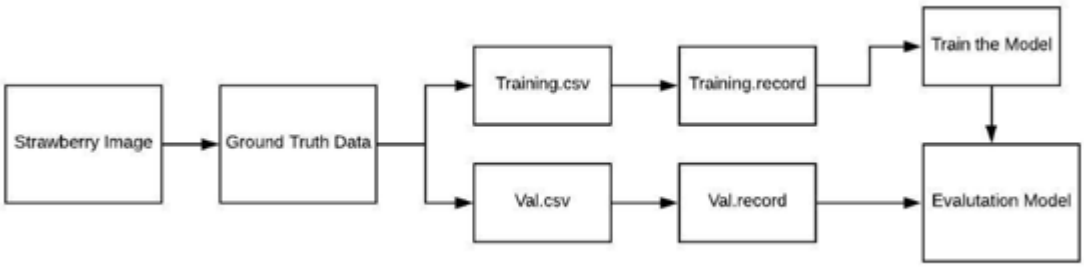

Figure 71 TensorFlow Object Detection API workflow chart

Provided below is an overview of how the database is established, process of which pre-training model is selected, the way remote server is used, as well as the criteria for test image selection.

Dataset

Unlike the previous chapter, the training sample used in this chapter is the ground-truth data. 200 Ground-truth images have been used for hard negative mining before. However, the number is far from enough to train an object detector. Therefore, another 300 ground-truth images are created to enrich the training dataset. In a total of 500 training images, 50 images are evaluation samples, and the rest images are the training samples.

The positive samples used before are single cropped green strawberry, while the negative samples are fixed-size non-strawberry images. These two kinds of samples were manually put into two sets. However, there is no special processing except marking ground-truth data in this chapter. During the training process later, the model will randomly select the proposal region in the input training image. Whether this region will be treated as a positive sample or not depends on the IOU mentioned before, the threshold value here is 0.8.

Remote Server

In order to save training time furthermore, a remote server from Amazon Web Services (AWS) was hired. The instance type is p2.xlarge GPU instance, which has a single NVIDIA K80. As shown in the workflow (Figure71), the ground truth data will be first transferred to the CSV file and then to tf.record format. After uploading two record files of training samples and validation samples into the remote server, the object detection model will be trained in the remote server by running TensorFlow API scripts there. Once the training is completed, the trained models will be saved into the local server and used for future testing.

Pre-trained Model

TensorFlow API provides a collection of detection models pre-trained on the COCO dataset [17]. Which contains photos of 91 objects types with a total of 2.5 million labeled instances in 328k images [18].

Figure72 shows part of them.

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes
faster_rcnn_nas	slow	43	Boxes

Figure 72 TensorFlow Model Zoo Samples

As Pirkko Mustamo mentioned in his object detection in sports: TensorFlow Object Detection API case study [20], there is a speed-accuracy tradeoff among different pre-trained model. Methods such as YOLO or Single-Shot Detector (SSD) are capable of that speed, but this tends to come with a decrease in accuracy of predictions, whereas models such as Faster R-CNN achieve high accuracy but are more expensive to run [20]. Thus, 'ssd_inception_v2_coco' and 'faster_rcnn_resnet101_coco' have been chosen as pre-trained models. In the following section, they will be used to train detectors separately and then to compare the two pre-trained models by comparing the performance of the trained detector.

Training Configuration

TensorFlow object detection API has offered the pipeline.config file for setting the training hyperparameters, which is also shown in the Appendix.

The following are the main parameters that are adjusted in this thesis for training purposes.

Training Step: 3000

Input Image Resizer: [300 300]

Batch Size: 24

Data Augmentation Option: random_horizontal_flip

Testing Image

In the testing part, two types of strawberry pictures are used.

One is a photo of a complete strawberry bush, as shown in Figure 73. There are 34 such testing images, containing a total of 387 green strawberries.

Figure 73 Strawberry Bush

The other one is a single strawberry plant images cropped from the above image, as shown in Figure 74.

Figure 74 Single Strawberry Plant

There are 100 such images, containing 342 green strawberries. These two pictures will help to test the detection capability of the trained detectors.

4.2 Training and Testing

Model 1:

The model 1 is trained based on 'ssd_inception_v2_coco' pre-trained model. Set training steps to 3000. Model1 takes one hour to train and 20 minutes for evaluation.

Take Figure 74 as an example, Figure 75 shows the sample testing result.

Figure 75 Model1 testing result on single strawberry plant

Table4 shows the results of detection 100 images

Table 4 Model 1 testing results on single strawberry plant

Total number	Truth Positive	False Positive	Truth positive rate
342	305	20	0.892

Figure 76 shows the false negative samples and false positive sample. Green strawberries were not detected and the leaf has been mistaken as strawberry.

Figure 76 Model1 Testing Result: False Negative Samples and False Positive Sample

Then, apply the detector on the image of a full strawberry bush. Figure77 shows the result. To my disappointment, the detector's performance on the strawberry bush is worse than on the individual plant.

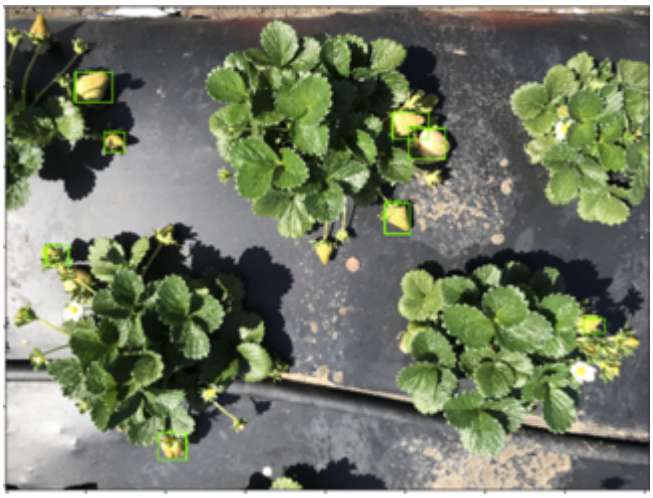

Figure 77 Model1 testing result on strawberry bush

Table5 shows the results of detecting 34 such images.

Table 5 Model1 testing results on strawberry bush

Total number	Truth Positive	False Positive	Truth positive rate
387	303	16	0.782

The truth-positive rate has dropped by 12.7%. This may be due to the detector's lack of adaptation to environmental complexity.

Model2

The model 2 is trained based on 'faster_rcnn_resnet101_coco' pre-trained model. Setting the number of training steps to 3000, model2 took one and a half hours to train and 30 minutes to evaluate. The overall time consumed is 40 minutes longer than model1.

Figure78 shows the sample testing result by model2 on the single strawberry plant.

Figure 78 Model2 testing result on single strawberry plant

Table6 shows testing results of model2 on 100 images.

Table 6 Model2 testing results on single strawberry plant

Total number	Truth Positive	False Positive	Truth positive rate
342	316	16	0.923

Based on the test results, model2 performs better than model1 no matter its truth-positive rate or false-positive number. Then apply detection on the strawberry bush. Figure79 shows the sample testing result. Table7 shows the testing result on 34 strawberry bush images.

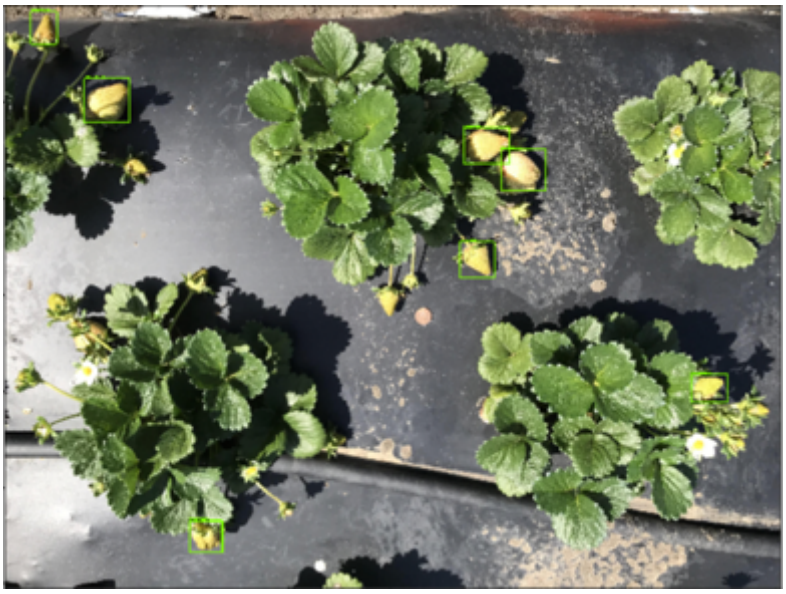

Figure 79 Model2 testing result on strawberry bush

Table 7 Model2 testing results on strawberry bush

Total number	Truth Positive	False Positive	Truth positive rate
387	334	18	0.863

Model2 may also lack tolerance in a realistic environment, but whether it is on a single plant or in the entire strawberry cluster, Model2 performs better than model1. In the following section, improvements in the two models' detection capabilities will be attempted by preprocessing the test images.

4.3 Promotion

4.3.1 Pre-processing methods

In view of the fact that the two models perform well in detecting green strawberries in individual plants, attempts to improve the model will mainly focus on the detection in strawberry clusters.

Image segmentation

First, try to reduce the environmental complexity of strawberry bushes. Applying image segmentation on the 34 testing images, the gray plastic sheet in the image has been blackout while all the bushes left (Figure80).

Figure 80 Strawberry bush without background

Then using two models to do detection on all 34 images respectively.

Figure 81 Pure Strawberry Bush (Model)

Figure 82 Pure Strawberry Bush (Model2)

Table8 shows the results for two models. However, there is no significant improvement. The background does not play an important role in undermining detectors.

Table 8 Testing results for pure strawberry bush

	Total number	Truth Positive	False Positive	Truth positive rate
Model1	387	305	20	0.788
Model2	387	330	18	0.852

Resolution

As the background doesn't help to improve the detector, will the image's resolution be useful? Changing the resolution of all 34 images to 0.2, 0.4, 0.6 and 0.8 of the original image respectively, as shown in Figure 83, Figure84, Figure 85 and Figure 86.

Figure 83 image with resolution 0.2

Figure 84 image with resolution 0.4

Figure 85 image with resolution 0.6

Figure 86 image with resolution 0.8

Table 9 and Table 10 shows the testing results of model1 and model2. However, even if the resolution drops to 0.2 of the original image, the fluctuation of the true positive rate and the false positive number is not large. Thus, changing the resolution will not promote the detector.

Table 9 Model1 Testing Results on Different Resolution

	Total Number	Truth Positive	False Positive	Truth positive rate
0.2	387	300	16	0.775
0.4	387	299	18	0.772
0.6	387	302	13	0.780
0.8	387	302	15	0.780

Table 10 Model2 Testing Results on Different Resolution

	Total Number	Truth Positive	False Positive	Truth positive rate
0.2	387	329	16	0.850
0.4	387	330	19	0.852
0.6	387	330	18	0.852
0.8	387	334	18	0.811

Now, consider utilizing other image pre-processing methods. Such as enhance the contrast, add brightness, tune the saturation, and adjust the Gamma.

Enhance the contrast

Take Figure77 as an example, set the value of enhanced contrast to 1.1, 1.2, and 1.3, respectively. Use two models to detect strawberries in images with different contrasts. Figure 87 and Figure88 show the result. As far as this image is concerned, when the contrast is set to 1.1, both models give the best result. Thus, choose 1.1 to enhance the contrast of all 34 images.

Figure 87 Model1 testing result in contrast of 1.1(left),12(middle),1.3(right)

Figure 88 Model2 testing result in contrast of 1.1(left),12(middle),1.3(right)

Then count the test results of the two models (Table11). However, enhanced contrast won't help to promote the detector based on this testing results.

Table 11 Testing Results with contrast of 1.1

	Total number	Truth Positive	False Positive	Truth positive rate
Model 1	387	303	17	0.782
Model 2	387	336	19	0.868

Add Brightness

Similar to the previous steps, first explore suitable light intensity through experiments. Set the changed light intensity to 0.1, 0.15 and 0.2. The results of the two detectors are shown in Figure 89 and Figure90. According to the detection result, choose 0.1 as the enhanced brightness intensity.

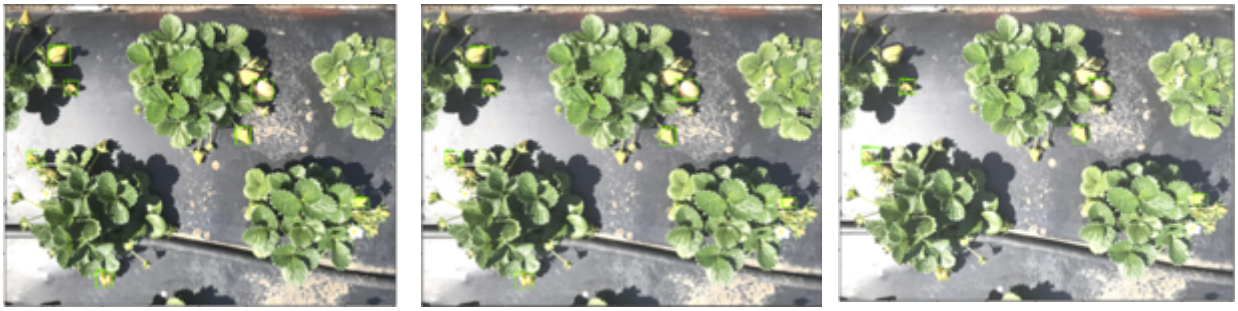

Figure 89 Model1 testing result with enhanced brightness of 0.1(left),0.15(middle),0.2(right)

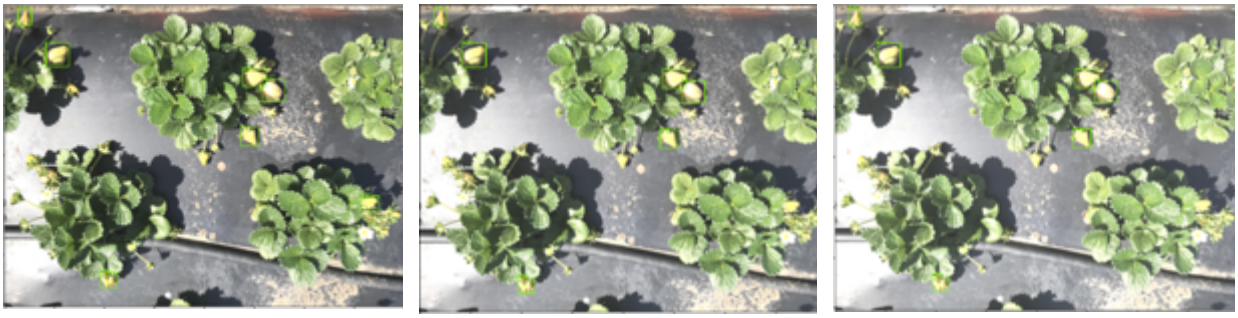

Figure 90 Model2 testing result with enhanced brightness of 0.1(left),0.15(middle),0.2(right)

Two models were used to detect strawberry on 34 pictures, and the Table 12 shows the results.

Table 12 Testing results with enhanced brightness of 0.1

	Total number	Truth Positive	False Positive	Truth positive rate
Model 1	387	299	16	0.772
Model 2	387	334	18	0.863

The detection result shows that adding additional light to the original image will not improve the detector's performance.

Saturation Tuning

First, find out the most suitable saturation adjustment value. Set the adjusted values as 1.5, 2 and 2.5. Figures 91 and Figure 92 show the test results under different saturation values.

From the test results, adjusting the saturation value is of great help to model1. And 1.5 is the most suitable value for two detectors. Then perform detection on 34 images.

Figure 91 Model1 testing result with tuned saturation of 1.5(left),2(middle),2.5(right)

Figure 92 Model2 testing result with tuned saturation of 1.5(left),2(middle),2.5(right)

Then perform detection on 34 images. Table13 shows the testing results. Comparing with the image without any pre-processing, the model1's true positive rate increased from 0.782 to 0.855 after adjusting the saturation. At the same time, the performance of model2 has not changed significantly.

Table 13 Testing results with tuned saturation of 1.5

	Total number	Truth Positive	False Positive	Truth positive rate
Model 1	387	331	21	0.855
Model 2	387	337	19	0.870

Gamma Adjustment

Finally, try to promote detectors by performing gamma correction. Set the gamma values to 0.6, 0.7 and 0.8. From the detection results in Figure93 and Figure 94, different gamma values have little effect on the detector. Therefore, set a Gamma value of 0.6 to process 34 pictures.

Figure 93 Model1 testing result with gamma correction of 0.6(left),0.7(middle),0.8(right)

Figure 94 Model2 testing result with gamma correction of 0.6(left),0.7(middle),0.8(right)

Table14 shows the final test results. However, gamma correction does not help improve performance.

Table 14 Testing results with gamma correction of 0.6

	Total number	Truth Positive	False Positive	Truth positive rate
Model 1	387	312	16	0.806
Model 2	387	330	19	0.852

4.3.2 Retrain Model2

As shown in Figure95, the false-positive samples in testing results come from two types: the leaves and the red strawberries which are mistaken for green strawberries.

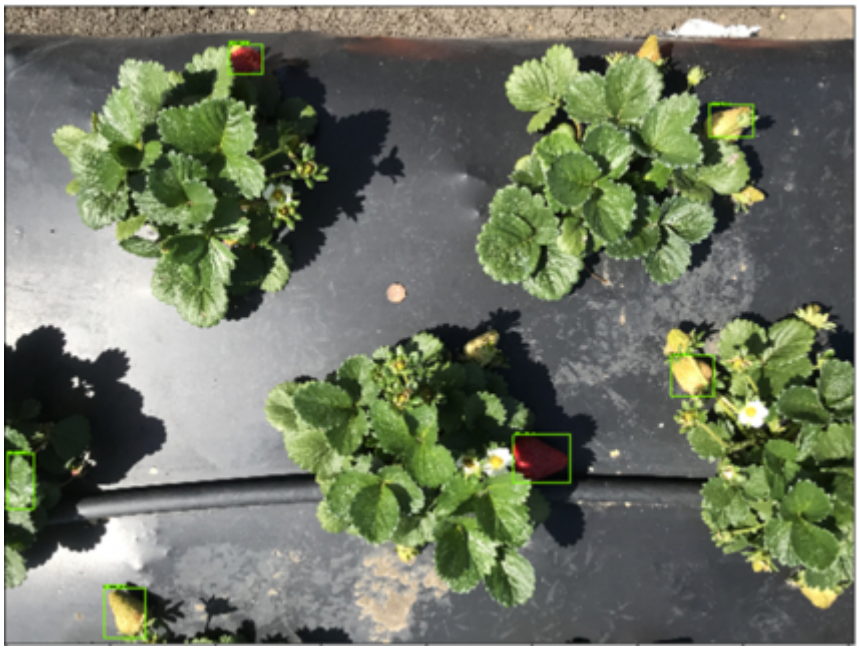

Figure 95 False Positive Samples

For the green leaves type false positive samples, the only way to reduce this kind of error is to retrain the model. Based on previous analysis, we found that model2 which based on 'Faster R-cnn' pretrained model has a better performance. Therefore, I decided to retrain the model2: First, to increase the training steps from 3000 to 4000. Then, to add more image augmentation options. When training the model2, the image augmentation method for enriching the dataset is only to flip images horizontally. Thus, the following options have been exploited for retraining: randomly adjust brightness, randomly adjust contrast, randomly adjust saturation, and randomly adjust Gamma.

Figure 96 Failure detection result by retrained model2

The retraining process has taken around 13 hours. However, it brings out a worse detection result (Figure 96). The truth-positive rate drops dramatically, which is even less than 50%.

Besides, for the red strawberries which are mistaken as the green one, we can utilize the color information we explored previously to exclude the red strawberry out of the testing image and then using trained green strawberry detectors to do the detection.

Chapter 5

Summary and Future Work

This project utilizes a series of methods to detect red and green strawberries.

For ripe strawberries, the K-Mean algorithm was based on various feature spaces along with image segmentation. Several color models are first analyzed by their color distribution in the histogram to choose the suitable color spaces for red strawberries segmentation. As a result, the color components such as the normalized r channel, ‘a’ channel both show sensitivity to the red color. Then, three different feature spaces that are based on normalized r, 'a', and a combination of normalized r, 'a' and hue color spaces have been built to support the K-Mean cluster algorithms to do image segmentation. Detectors that are based on different feature spaces all deliver similar results. The final positive rate reaches 91.50%.

For immature strawberries, the SVM classifier based on the HOG feature is first trained and tested on the testing images set. The first step is to compare the different input images' sizes and the cell size that HOG descriptors will operate on to choose a combination that will train a better SVM classifier. To get a more precise comparison result, two classifiers are trained based on different sizes of HOG feature vectors, which both take around 6 hours for training. As a result, HOG features offered by input [128, 64] image with cell size [8, 8] give the best classification result. Then the SVM classifier will detect green strawberry with the support of the sliding window and non-maximum suppression. However, the detection results are undesirable due to the high false-positive rate. Thus, the hard negative mining method has been employed to improve the classifier. There are two types of hard negative samples: one is the false positive sample that overlaps with the object, another one is the false positive sample that has a high possibility that confuses the classifier, the second one gives improved results. The SVM classifier has been retrained separately by cooperating with the hard negative mining that is based on different hard negative samples. As a comparison results, the second type sample gives better detection results, whose final truth positive rate is 81.11%. However, this number is far from satisfactory to achieve our final goal.

Then, this work has moved to focus on exploring deep convolutional neural network model with transfer learning. Two detectors based on pre-trained models, SSD and Faster R-CNN, have been trained respectively by using TensorFlow API. Model 1 based on SSD takes one hour to train and 20 minutes for evaluation, whereas model2 based on Faster R-CNN took one and a half hours to train and 30 minutes to evaluate. When making a comparison between Model1 and Model2, Model2 has not only a higher truth positives rate but also greater accuracy when drawing the bounding box around green strawberries. Both models 1 and 2 performed well on the single strawberry plant images. Model1 gets 89.2% truth-positive rate, and model2 achieves 92.3%. However, they did not give desirable results when working on strawberry clusters. Thus, several image preprocessing methods such as contrast enhancement and saturation tunning have been hired to improve both detectors.

As the result, the detection ability of Model2 has not improved through various image preprocessing methods, so the true positive rate of Model2 on original strawberry bushes remains at 86.3%. Conversely, by tuning saturation of Model1, an 85.5% truth-positive rate was achieved on the strawberry bushes.

A third model with larger training steps and more image augmentation options has also been trained, and the training process has taken around 13 hours. But this detector did not bring out acceptable results. In fact, its performance was worse than model1 and model2.

In further attempts to get better detection results on green strawberries, the first approach to be taken is to expand the image dataset furthermore. There are two types of datasets that have been used in this thesis. One dataset contains 2000 of the single cropped green strawberry images combined with 8000 the non-strawberry images. Another dataset includes 500 ground-truth images. The first one that was used for training of SVM classifier is not sufficient. Thus, the number of images for single cropped green strawberry should be increased to 4000 at least.

In addition, training a deep learning model from scratch could be considered for future research. Correspondingly, the scale of ground truth should also be expanded, which should be 2000 at the minimum as each ground truth image contains several green strawberries. Then, another search model, such as YOLO, can be examined in future research.

For the red strawberry detection, the primary issue that resulted in false-positive samples were the detector's inability to distinguish objects in close proximity. Several post-processing methods can be considered. Options include to train a texture feature-based model or to train a deep learning model on the image segmentation results.

REFERENCES/WORKS CITED

[1] Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*. Upper Saddle River, N.J: Prentice Hall.

[2] S. Smith, C. Garcia, "*Worker Shortage Hurts California's Agriculture Industry*", Planet Money, 2018.

[3] University of California, Division of Agriculture and Natural Resources. (October, 1999). "*Crop Profile for Strawberries in California*" Available: http://ucanr.edu/datastoreFiles/391-501.pdf

[4] Viola, P.; Jones, M., "*Robust real-time face detection*" Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol.2, no., pp.747, 747, 2001.

[5] Huaizu Jiang, Erik Learned-Miller "*Face Detection with the Faster R-CNN*" 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)

[6] MacQueen, J. *Some methods for classification and analysis of multivariate observations*. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967. https://projecteuclid.org/euclid.bsmsp/1200512992

[7] N. Dalal, B. Triggs "*Histograms of oriented gradients for human detection*" CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01June 2005 Pages 886–893https://doi.org/10.1109/CVPR.2005.177

[8] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *"Rich feature hierarchies for accurate object detection and semantic segmentation".* In IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014.

[9] 2. K.-K. Sung and T. Poggio. *Learning and example se-lection for object and pattern detection*. MIT A.I. Memo, 1521, 1994.

[10] Rothe, Rasmus & Guillaumin, Matthieu & Van Gool, Luc. (2015). *Non-Maximum Suppression for Object Detection by Passing Messages between Windows*. LNCS. 9003. 10.1007/978-3-319-16865-4_19.

[11] Yin, J. (2008). Segmentation Methods of Fruit Image and Comparative Experiments. [online] Ieeexplore.ieee.org. Available at: http://ieeexplore.ieee.org.ezproxy.lib.calpoly.edu/stamp/stamp.jsp?arnumber=472 1944 [Accessed 8 Jan. 2017].

[12] Mathworks "*Train Stop Sign Detector*" Mathworks, 2014. Available at: https://www.mathworks.com/help/vision/ug/train-a-stop-sign-detector.html

[13] Mathworks "*Detect objects using the Viola-Jones algorithm*" Mathworks, 2014,from http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector- class.html

[14] Maria Vanrell; Felipe Lumbreras; Albert Pujol; Ramon Baldrich; Josep Llados; J.J. Villanueva (7 October 2001). *Colour normalisation based on background information*. *Image Processing, 2001*. **1**. Thessaloniki, Greece. pp. 874–877. doi:10.1109/icip.2001.959185. ISBN 978-0-7803-6725-8. INSPEC 7210999.

[15] TensorFlow Object Detection API, from https://github.com/tensorflow/models/tree/master/research/object_detection

[16] Pan, S. J., & Yang, Q. (2009, October 1). '*A Survey on Transfer Learning' - IEEE Journals & Magazine*.

[17] TensorFlow object detection API model zoo, from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

[18] Tsung-Yi Lin, Michael Maire,Serge Belongie,Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. (2018)"*Microsoft COCO: Common Objects in Context*". arXiv:1405.0312 [cs.CV].

[19] Khush Patel.（2019, May 28). "*Custom Object Detection using TensorFlow from Scratch*," from https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087

[20] Pirkko Mustamo. (2018). "*Object detection in sports: TensorFlow Object Detection API case study*"

[21] Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson. (2014). "*How transferable are features in deep neural networks?*". arXiv:1411.1792

[22] Al-Shalfan, K. (2015). "*Auto Vehicle Driving Assistance. International Journal Of Machine Learning And Computing"*, 5(5), 392-398. doi: 10.7763/ijmlc.2015.v5.540

[23] Rho, S., Rahayu, W., & Nguyen, U. (2015). "*Intelligent video surveillance in crowded scenes. Information Fusion*", 24, 1-2. doi: 10.1016/j.inffus.2014.11.002

[24] Kulvatunyou, B., Ivezic, N., Morris, K., & Frechette, S. (2016). "*Drilling down on Smart Manufacturing – enabling composable apps. Manufacturing Letters*", 10, 14-17. doi: 10.1016/j.mfglet.2016.08.004

[25] Nayak, M., R, M., & Dhanusha, M. (2020). "*Fruit Recognition using Image Processing*". Retrieved 31 August 2020, from https://www.ijert.org/fruit-recognition-using-image-processing

[26] S. Ghosal and K. Sarkar, "*Rice Leaf Diseases Classification Using CNN With Transfer Learning*," 2020 IEEE Calcutta Conference (CALCON), Kolkata, India, 2020, pp. 230-236, doi: 10.1109/CALCON49167.2020.9106423.

[27] Liu, T., Chen, W., Wu, W., Sun, C., Guo, W., & Zhu, X. (2016). " *Detection of aphids in wheat fields using a computer vision technique. Biosystems Engineering*, "141, 82-93. doi: 10.1016/j.biosystemseng.2015.11.005

[28] N. Lamb and M. C. Chuah, "*A Strawberry Detection System Using Convolutional Neural Networks*," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 2515-2520, doi: 10.1109/BigData.2018.8622466.

APPENDIX

A. Pipline.config File:

```
model {
 ssd {
  num_classes: 1
  box_coder {
   faster_rcnn_box_coder {
    y_scale: 10.0
    x_scale: 10.0
    height_scale: 5.0
    width_scale: 5.0
   }
  }
  matcher {
   argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
   }
  }
  similarity_calculator {
   iou_similarity {
   }
  }
  anchor_generator {
   ssd_anchor_generator {
    num_layers: 6
    min_scale: 0.2
    max_scale: 0.95
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    aspect_ratios: 3.0
    aspect_ratios: 0.3333
```

```
  }
}
image_resizer {
  fixed_shape_resizer {
    height: 300
    width: 300
  }
}
box_predictor {
  convolutional_box_predictor {
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    use_dropout: false
    dropout_keep_probability: 0.8
    kernel_size: 1
    box_code_size: 4
    apply_sigmoid_to_scores: false
    conv_hyperparams {
      activation: RELU_6,
      regularizer {
        l2_regularizer {
          weight: 0.00004
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
      batch_norm {
        train: true,
        scale: true,
        center: true,
        decay: 0.9997,
        epsilon: 0.001,
      }
```

```
    }
  }
}
feature_extractor {
  type: 'ssd_mobilenet_v2'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
    initializer {
      truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
      }
    }
    batch_norm {
      train: true,
      scale: true,
      center: true,
      decay: 0.9997,
      epsilon: 0.001,
    }
  }
}
loss {
  classification_loss {
    weighted_sigmoid {
    }
  }
  localization_loss {
    weighted_smooth_l1 {
    }
  }
```

```
    hard_example_miner {
      num_hard_examples: 3000
      iou_threshold: 0.99
      loss_type: CLASSIFICATION
      max_negatives_per_positive: 3
      min_negatives_per_image: 3
    }
    classification_weight: 1.0
    localization_weight: 1.0
  }
  normalize_loss_by_num_matches: true
  post_processing {
    batch_non_max_suppression {
      score_threshold: 1e-8
      iou_threshold: 0.6
      max_detections_per_class: 100
      max_total_detections: 100
    }
    score_converter: SIGMOID
  }
 }
}
train_config: {
 batch_size: 24
 optimizer {
  rms_prop_optimizer: {
    learning_rate: {
      exponential_decay_learning_rate {
        initial_learning_rate: 0.004
        decay_steps: 800720
        decay_factor: 0.95
      }
    }
    momentum_optimizer_value: 0.9
    decay: 0.9
    epsilon: 1.0
  }
 }
```

```
  fine_tune_checkpoint: "checkpoints/model.ckpt"
  fine_tune_checkpoint_type:  "detection"
  num_steps: 3000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
train_input_reader: {
  tf_record_input_reader {
    input_path: "tf_record/training.record"
  }
  label_map_path: "tf_record/label_map.pbtxt"
}
eval_config: {
  num_examples: 49
  max_evals: 5
}
eval_input_reader: {
  tf_record_input_reader {
    input_path: "tf_record/val.record"
  }
  label_map_path: "tf_record/label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```