STRAWBERRY DETECTION UNDER VARIOUS HARVESTATION STAGES

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
Yavisht Fitter
March 2019

COMMITTEE MEMBERSHIP

TITLE:	Strawberry Detection Under Various Harvestation Stages
AUTHOR:	Yavisht Fitter
DATE SUBMITTED:	March 2019
COMMITTEE CHAIR:	Jane Zhang, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Helen Yu, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	John Saghri, Ph.D. Professor of Electrical Engineering

ABSTRACT

Strawberry Detection Under Various Harvestation Stages

Yavisht Fitter

This paper analyzes three techniques attempting to detect strawberries at various stages in its growth cycle. Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP) and Convolutional Neural Networks (CNN) were implemented on a limited custom-built dataset. The methodologies were compared in terms of accuracy and computational efficiency. Computational efficiency is defined in terms of image resolution as testing on a smaller dimensional image is much quicker than larger dimensions. The CNN based implementation obtained the best results with an 88% accuracy at the highest level of efficiency as well (600x800). LBP generated moderate results with a 74% detection accuracy at an inefficient rate (5000x4000). Finally, HOG's results were inconclusive as it performed poorly early on, generating too many misclassifications.

Keywords: Histogram of Oriented Gradients, Local Binary Patterns, Convolutional Neural Networks

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Object detection is the ability for a machine to automatically extract and distinguish data from the environment around it. It is a crucial aspect of Computer Vision as it gives the system a sense of perception. Object detection is the second step in the image analysis category as it builds on top of the first step, image classification. Unlike image classification which simply classifies a given image, object detection requires the system to be able to localize the object(s) within the environment while classifying it. Object detection has vastly progressed over the last decade and thus has resulted in multiple techniques being established. Each technique has its unique advantages therefore some tend to perform better under certain environmental conditions over the others. In this paper, three popular techniques will be analyzed; a gradient/angular based feature extractor in Histogram of Oriented Gradients, a texture-based method in Local Binary Patterns, and finally, a neural network-based approach employing a variation of a Convolutional Neural Network. The three techniques will be applied in the agricultural setting, specifically strawberry detection and compared amongst each other in terms of performance.

1.1 Statement of Problem

Computer vision has had an exponential resurgence over the last decade and hence it's being applied into a variety of industries for the primary purpose of improving productivity and safety [1]. The agriculture industry, primarily within the state of California depends heavily on farm workers to constantly look after their crops throughout the year, as California accounts for over 13% of the nation's agricultural value [2]. A shift in the demographic landscape overtime has begun taking its toll on farmers as there has been a significant labor shortage in California [1]. This labor shortage is not cost efficient either and has thus provoked the farmers to double their workers' pay within the last 10 years [1]. Workers are also expected to harvest throughout the year which usually includes in-humane conditions during certain periods of the year [1]. It is key for farmers to have consistent labor crew especially during these extreme weather periods of the year otherwise significant crop loss can occur. Strawberries are the second most popular fruit grown in California however almost 10% of strawberry crops can go unharvested in a single year resulting in massive strawberry wastage [3]. This is a large percentile given the fruit is grown at an astounding rate of over 3 billion per year [2].

To prevent the farming industry from being crippled any further, the primary issue this paper proposes to tackle is detecting the strawberries in its natural bed setting at various stages which can be invaluable for fruit counting and predicting optimal harvest times. The technical challenge this task deals with is small object recognition. In complex environments as this, smaller objects can easily blend in with the background therefore making it difficult to isolate it.

1.2 Data Collection

1.2.1 Plot and Bed Locations

The images were taken at Verticillium Field 25 at the Cal Poly strawberry fields. The specific section of the field where the images were taken was named 90 cv; trial planted on October 23, 2017. The specific region can be seen on the bottom left of the map layout in Figure 1.

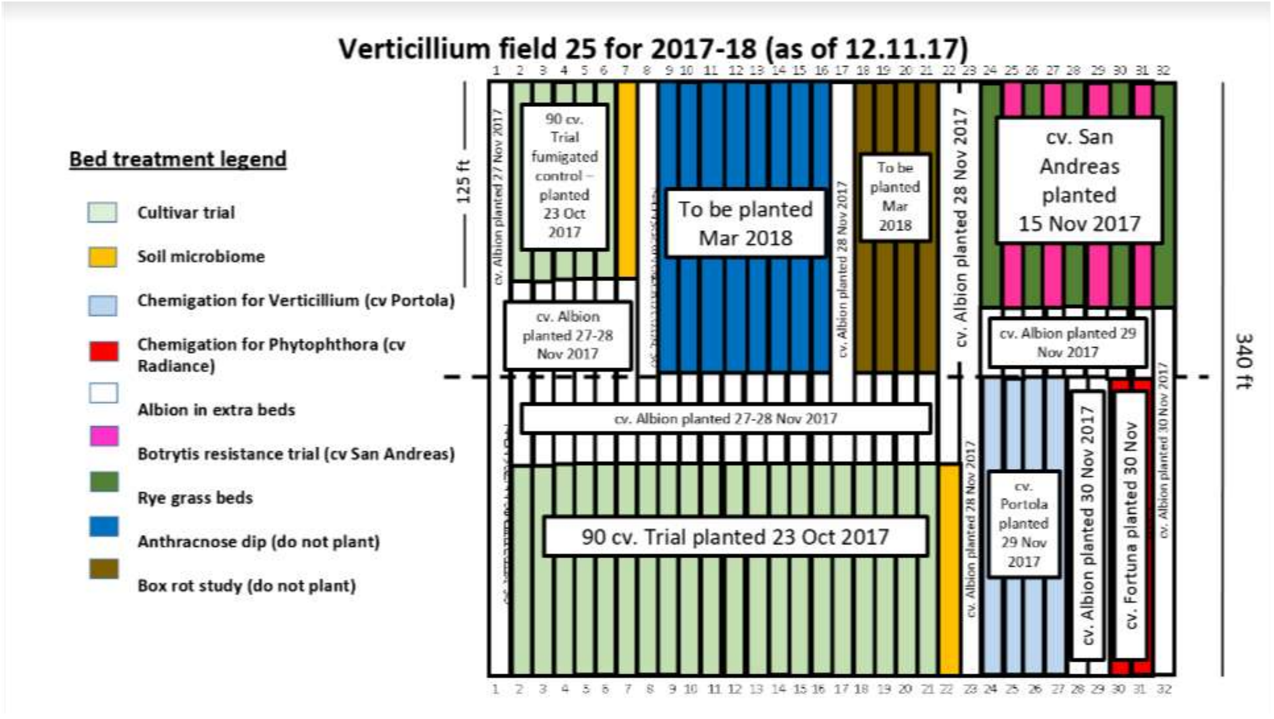

Verticillium field 25 for 2017-18 (as of 12.11.17)
Bed treatment legend
Cultivar trial
Soil microbiome
Chemigation for Verticillium (cv Portola)
Chemigation for Phytophthora (cv Radiance)
Albion in extra beds
Botrytis resistance trial (cv San Andreas)
Rye grass beds
Anthracnose dip (do not plant)
Box rot study (do not plant)
125 ft
340 ft
90 cv. Trial fumigated control – planted 23 Oct 2017
To be planted Mar 2018
To be planted Mar 2018
cv. San Andreas planted 15 Nov 2017
cv. Albion planted 27-28 Nov 2017
cv. Albion planted 29 Nov 2017
cv. Albion planted 27-28 Nov 2017
90 cv. Trial planted 23 Oct 2017
cv. Portola planted 29 Nov 2017
cv. Albion planted 27 Nov 2017
cv. Albion planted 28 Nov 2017
cv. Albion planted 28 Nov 2017
cv. Albion planted 28 Nov 2017
cv. Albion planted 30 Nov 2017
cv. Fortuna planted 30 Nov
cv. Albion planted 30 Nov 2017

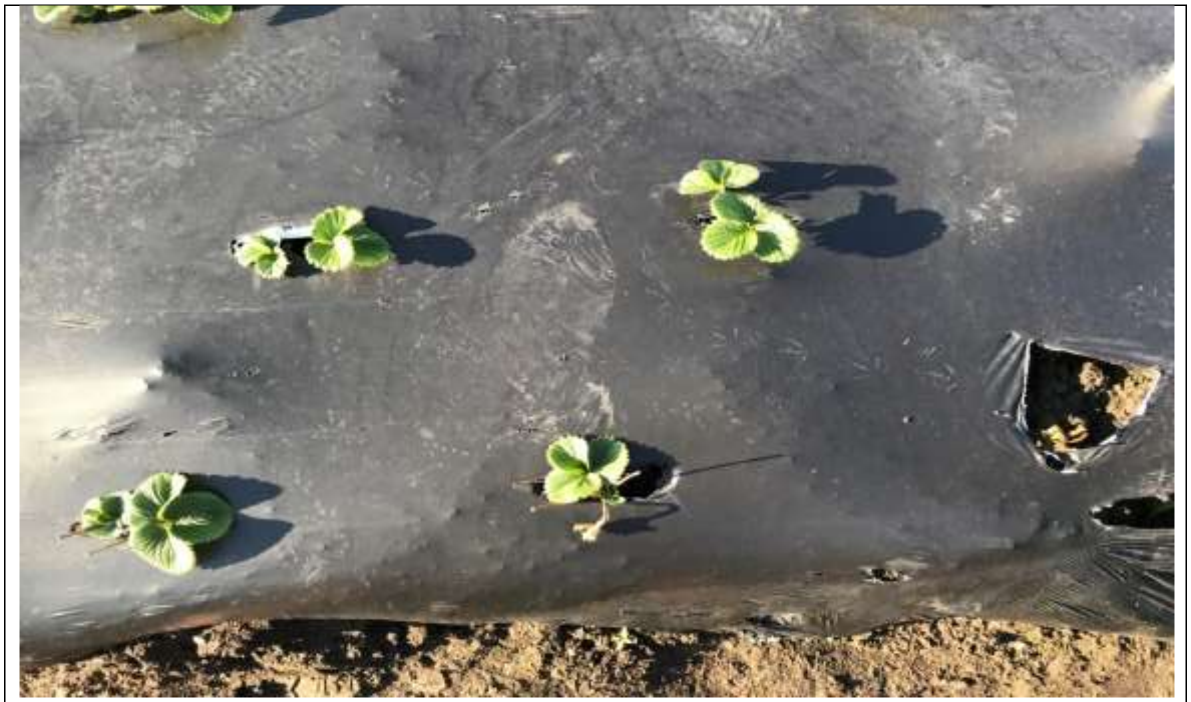
Figure 1: Entire Layout of the Strawberry Beds Part of Field 25 at Cal Poly, SLO

Five beds were chosen in total for data collection; each bed had a total of 20 plants. The beds chosen were the following:

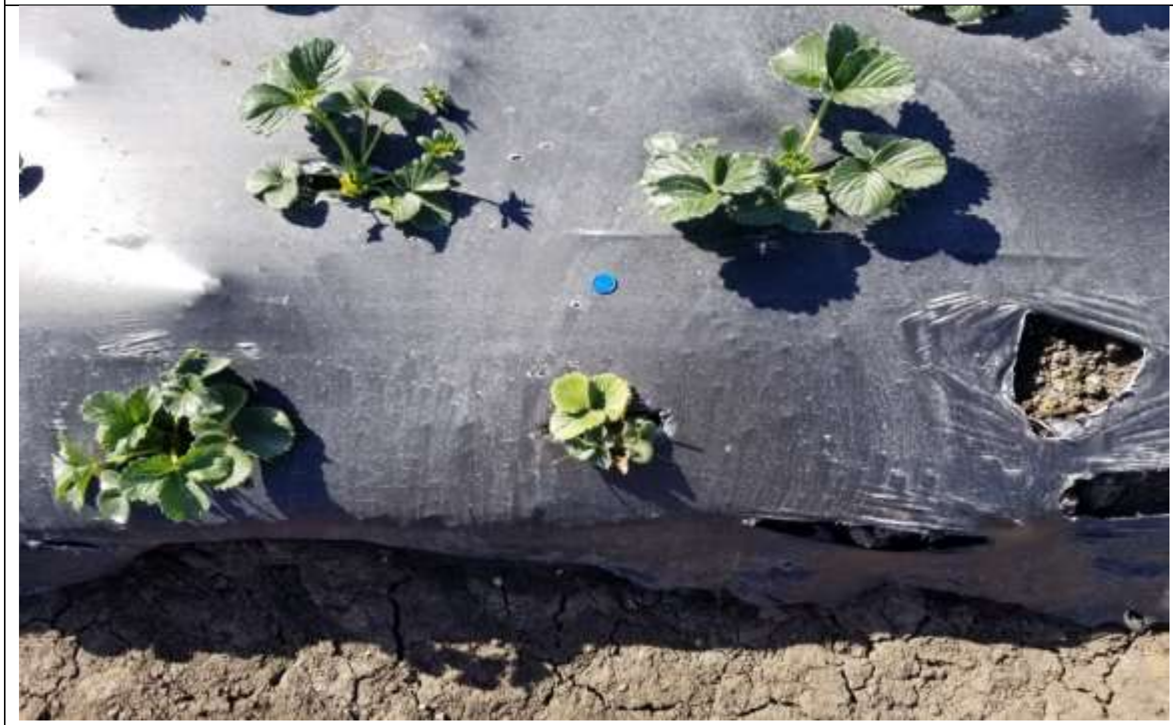
- Bed: Festival (Row 4)
- Bed: Enchante (Row 5)
- Bed: Osceola (Row 5)
- Bed: LC-2(77) (Row 8)
- Bed: BG(4.367) (Row 11)

1.2.2 Images Captured

The data collection process started from November 2017 and ended in June 2018. Images were taken twice a week to witness a consistent growth cycle, primarily Tuesdays and Fridays. Figure 2 displays a time-lapse of a sample bed set taken on the first data collection day of each month from January to June. A standard smartphone (Samsung galaxy s8) was employed to capture all the images at an average height between 4 to 5 ft. The images were taken at a resolution of 3042x4032, where out of the total 600 images collected, 452 were employed for training and 148 for testing, hence resulting in a 75:25 data split. The images were captured in a specific order each time to maintain consistency. On average each bed within this sector of field 25 contained 20 strawberry plants. The images were captured in sets of 5 for each bed instead of taking the picture of the entire bed at once. However, before any of the images were taken, a picture of the bed name tag was taken as well to prevent any confusion towards labeling each picture. The bright red strawberries are considered the ripe strawberries while the yellow/green strawberries are unripe strawberries. A local folder containing two classes of labeled images was employed to train the model. The labels were “strawberry” and “non-strawberry” thus representing a binary data set. The training data was created by manually cropping out each strawberry from the original training data. There wasn’t enough strawberry images collected to split the strawberries into separate folders of ripe and unripe strawberries, so all the strawberries were stored in a single folder. There were 924 strawberry and 1259 not images. A small blue penny was placed in the image for two primary reasons; gauge the distance from the camera to the plants and given the distance determine the size of the strawberries given the penny’s relative size. The blue penny can be seen in most of Figure 2’s images. The goal of this project is to detect ripe/unripe strawberries and the white flowers (white flowers can be seen in Figure 2e).



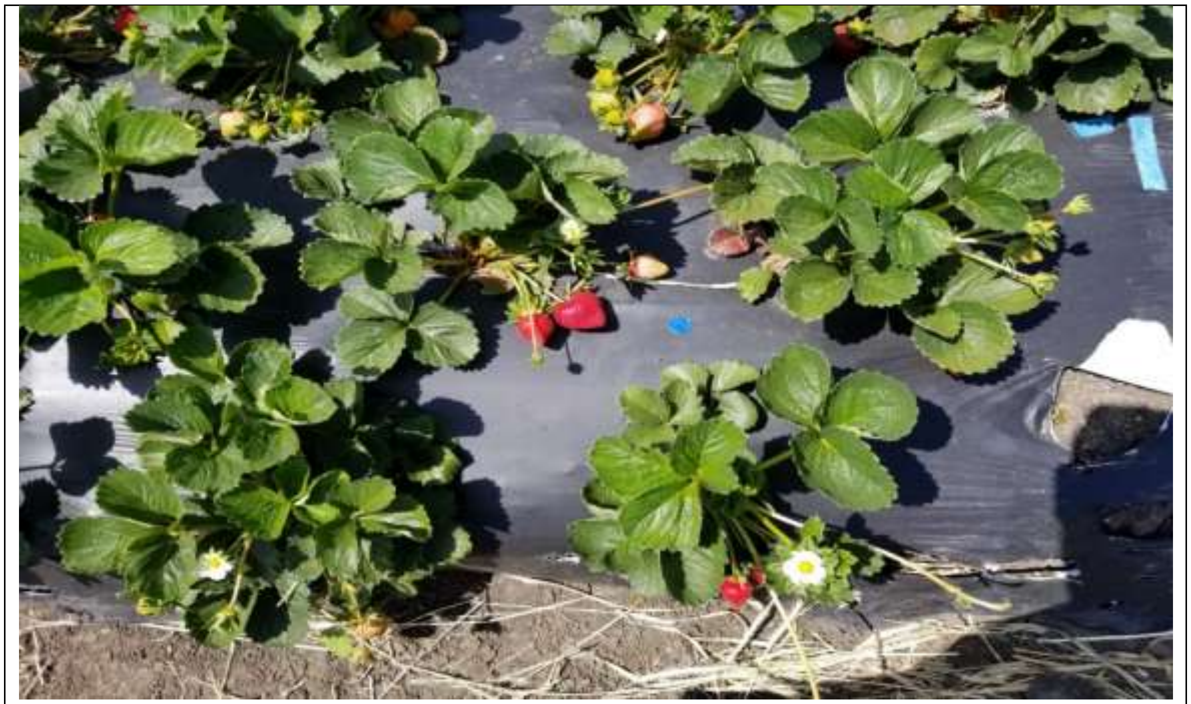
a) January stage



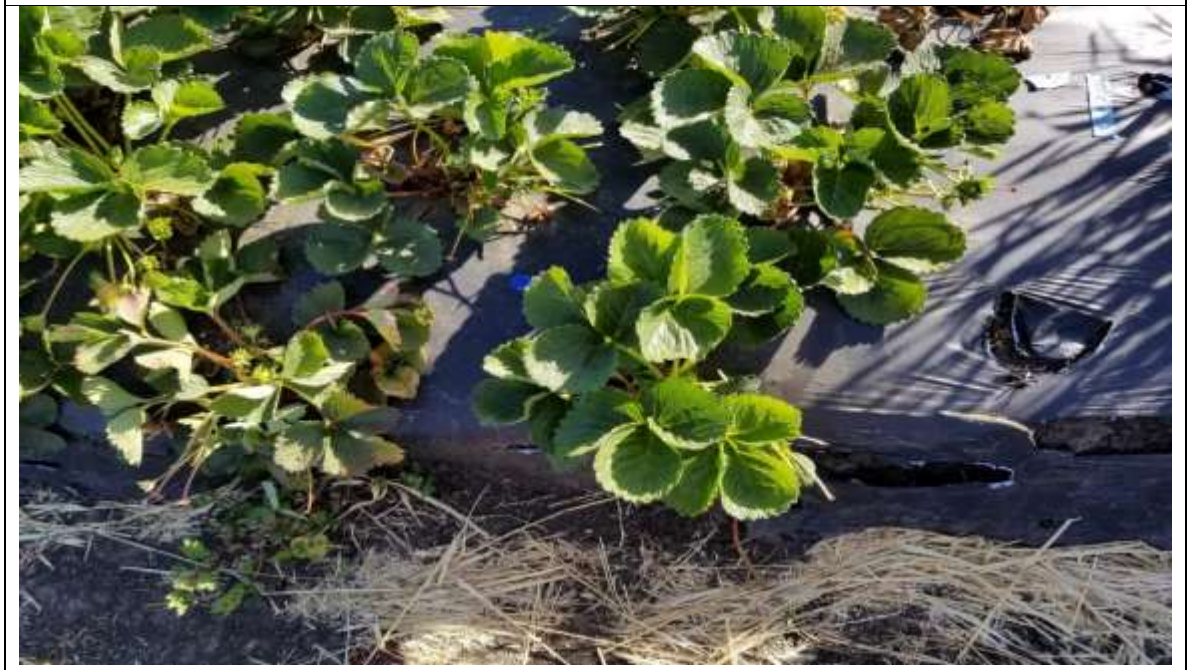
b) February stage

c) March stage

d) April stage



e) May stage



f) June stage

Figure 2: Time Lapse of Enchante Bed 1 from January to June

1.3 Challenges

The challenges faced during the data collection period were two-fold:

Heavy amounts of rain would fall during the months of January till March in San Luis Obispo and therefore this would at times destroy some of the strawberries due to high levels of flooding. The plants would then take another few weeks to grow the strawberries again and hence adding inconsistencies to the data set as this would result in more immature strawberries than mature for those months. The second issue involved the ripe strawberries being picked off the beds without notice. This would prevent us from seeing the entire life cycle of the strawberry, including its decay stage. This is vital information for one attempting to implement a robust harvest prediction model.

Chapter 2

LITERATURE REVIEW

2.1 Automated Visual Fruit Detection for Harvestation Estimation and Robotic Harvesting (S. Puttermans et al., 2016)

This paper tackles automated fruit harvestation by applying machine learning and image processing techniques. The fruits employed in this paper were strawberries and apples as they both have similar color characteristics. Due to my work focusing primarily on strawberry detection I will only focus on the strawberry implementation and results discussed within this paper [4].

2.1.2 Procedure

In the paper, the authors generated their own strawberry dataset as there were none available for their application. The strawberry bed images were captured at a resolution of 1292x964. 205 positive and 200 negative images were gathered for training each at a constant dimension of 35x38. To distinguish between strawberries from the background, strawberry and background leaf images were taken as part of the dataset. To detect the strawberries, Local Binary Patterns (LBP) [5] were used as the primary feature descriptor and a cascade classifier using adaboost was employed to deal with the classification task. Before the training step, a preprocessing technique in histogram equalization was applied to deal with the lighting. The training images were then converted to grayscale as LBP's only take in 2D data thus making color (RGB) images invalid inputs. Since LBP's are unable to distinguish objects with color both unripe and ripe strawberries features were extracted simultaneously. The LBP feature points were then passed into the adaboost classifier [6]. To isolate the ripe strawberry detections, a post image processing technique in channel subtraction was applied where the following equation was employed.

$$I_{RG} \begin{cases} 0 & if\, I_R - I_G < 0 \\ I_R - I_G & if\, I_R - I_G > 0 \end{cases}$$

The equation states to keep the pixel values greater than zero if the red channel is the dominant channel when subtracting the green channel from it. This ensures a "red object" is present within the proposed region. Once the green channel pixels have been subtracted, global binary thresholding was applied upon the proposed region. Upon thresholding, the total pixels were calculated and if more than 50% of the resulting pixels were white, this detection was kept as a ripe strawberry. This post processing method allowed them to get rid of any unripe strawberries or false positives detected and box only the ripe strawberries. Some of the detected strawberries were at times boxed as a single detection due to their close proximity to each other. To deal with this the paper applied watershed segmentation on the detected region.

2.1.3 Results and Analysis

Unfortunately, the paper does not seem to provide concrete results for their strawberry detection techniques stating that the accuracy can be objective due to the lack of ground truths determining what can and cannot be considered a ripe strawberry as the strawberry growth process is not binary. However, the techniques used in this paper seem very useful and have great potential in strawberry detection. Hence, I will be employing several techniques discussed in this paper as part of my project as well. Their technique does have one limiting factor, as their primary goal did not include unripe strawberry detection as well and thus this technique does not guarantee success for immature strawberries. While attempting to employ their paradigm for unripe strawberry recognition, several misclassifications in the form of background leaves is possible.

2.2 A Deep Learning Method for Recognizing Elevated Mature Strawberries (X. Li et al., 2018)

Two approaches are taken in this paper to detect ripe strawberries, first HOG+SVM was employed as a gradient based feature descriptor and thereafter a convolutional neural network-based approach was implemented to classify mature strawberries [7].

2.2.1 Procedure

A total of 2000 training images were utilized, where 1000 represented positive ripe strawberries and the remaining 1000 were background/non-strawberry images. For the HOG technique [8] [9], the training images were then converted to grayscale and passed into the HOG descriptor which returns a multi-dimensional vector representing the shape of the object. However, due to HOG not being able to extract color features from the object, the paper discusses an alternative method to deal with this issue. The image was then separately converted to the HSV color space. Thereafter, the variance of the H channel was taken which gave it information about the strawberry's color. This additional vector was then concatenated onto the multi-dimensional shape vector returned by the HOG descriptor. This led to a final shape and color feature vector being generated which was then passed into the classifier for training. The classifier leveraged in this case was an SVM [10]. The sliding window technique was applied on the test images to extract the ROI and perform ripe strawberry classification. The ROI's HOG + variance vector would get passed into the SVM classifier and if the feature points resided in the strawberry class a bounding box would then get placed on that respective ROI.

The CNN procedure was implemented thereafter, the network architecture employed in their paper was CaffeNet [11]. The network consists of 8 layers, where the first 5 layers are convolutional layers and the remaining 3 are fully connected layers. The images were trained in batches of 64 and the model ran for 20,000 iterations. Figure 3 displays the network structure employed as per their paper.

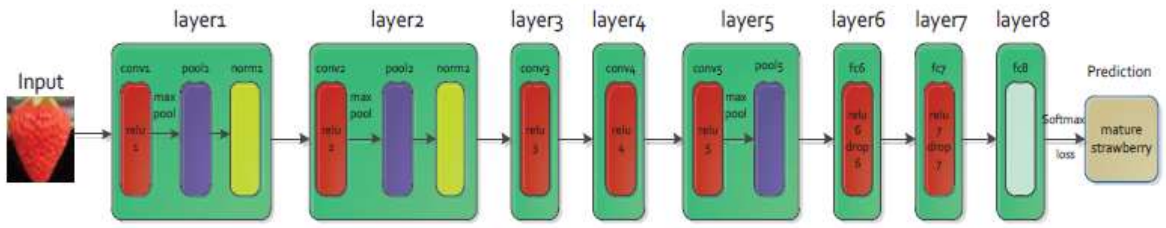


Figure 3: CaffeNet Network Architecture Used for Mature Strawberry Classification

2.2.2 Results and Analysis

The detection criteria were broken down into 3 types for both models; single strawberry, leaf shelter and fruit overlap. Under the HOG+H+SVM model the best results came when employing a Radial Basis Function kernel for the SVM classifier. Single strawberries had an accuracy rate of 99.05%, leaf sheltered was approximately 83.84% while fruit overlap was at 74.26%. When employing CaffeNet the results improved significantly as under the same constraints the accuracies were 99.05%, 92.9% and 95.05% respectively.

The results were relatively positive and specific techniques will certainly be used as part of my methodology as well. However, there are still three major drawbacks from their testing data; their detection process occurred at elevated farms and hence all the strawberries were hanging off the branches. This meant that the background in their testing image set primarily consisted of the tarp with few leaves. Even under the “leaf overlap” constrain, only couple leaves were involved in front or behind the strawberry which is much fewer than in my dataset. Figure 4 displays the scene under which the image was taken. The second issue was in the distance at which the strawberries were detected at. The strawberry images were taken at a relatively close proximity, no more than a foot away from each strawberry at most as seen in Figure 5. Finally, their goal was to only detect ripe/red strawberries and hence if any false positives were detected on the leaves, they could simply segment it out as the green strawberries were not of their concern in this project. These problems will prove to be challenging when dealing with my dataset and hence additional image processing techniques will have to be added into the detection algorithms to prevent these issues from occurring.

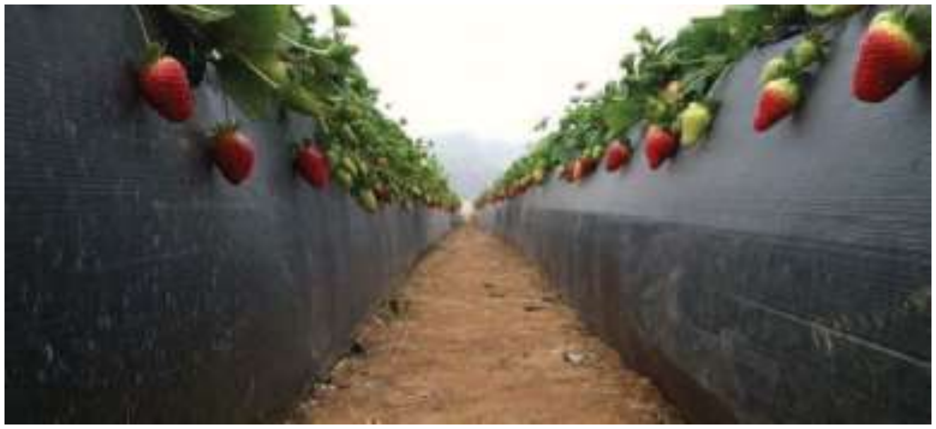


Figure 4: Elevated Strawberry Scene Provided in the Paper

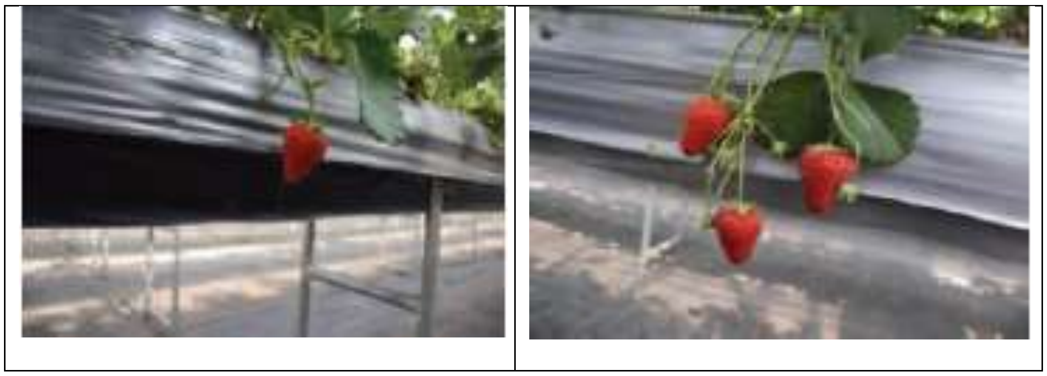


Figure 5: Strawberry Images Taken at Close Distances

2.3 DeepFruits: A Fruit Detection System Using Deep Neural Networks (I. Sa et al., 2016)

This paper tackled fruit detection using only deep convolutional neural networks. Three different classes were used for detection, which included sweet pepper, rock melon and background [12]. While strawberries were not specifically part of their fruit dataset, much of the techniques should still be applicable for my implementation. Their training data was produced in RGB and infrared channels and since RGB images cannot simply be converted to the infrared spectrum, the focus in this section will be on the techniques involving the RGB data as my dataset consists of RGB images as well. To learn more about the infrared based training procedure, refer to the paper [12]. The R-CNN architecture was employed as the detection model; the architecture involves 2 networks, first a regular convolutional network followed by a region-based network. The general steps taken to detect objects using a R-CNN is as follows:

(1) Pass an input image into a CNN which will output a feature map from the last layer.

(2) Run a sliding window across the feature maps, each siding window is generated at multiple aspect ratios.

(3) The sliding window regions are then passed into the region-based network which produces two outputs; a bounding box prediction and a class probability prediction.

4) The classification output returns the probability of the object detected while the regression output returns the bounding box of that respective object.

The specifics of the network architecture is beyond the scope of this section, refer to the original R-CNN paper for more details [13].

2.3.1 Procedure

The authors of the paper only had 100 training data images to work with and therefore training a network from scratch was not a viable option. A pre-built R-CNN network was employed which was trained on the PASCAL VOC dataset which consisted of 20 different objects. However, to detect their specific fruits, fine-tuning was performed where the 2 fruits were trained using the features generated from the original network. The initial generated features from the pre-trained network helped deal with the lack of training data. Off the 2 networks discussed earlier for the R-

CNN architecture, the VGG16 model was used as the first network to generate the feature maps. The VGG model was chosen as it did well in generalizing to the dataset provided.

2.3.2 Results and Analysis

Only 22 images were supplied as part of the testing dataset. However, with such a limited training dataset the network still achieved an average accuracy of 81.6%. This was the best results even over the infrared input images. Fine tuning played a major role in the process. This network is definitely an option to consider as well along with the previous techniques reviewed so far. R-CNN’s are unique as they run the sliding window on the CNN produced feature map instead of just the raw image as done in the past papers. There are two limitations to keep in mind prior to using this method. Fine-tuned models are usually trained on much larger image dimensions. The strawberries in my dataset are originally cropped to a much smaller size, 60x60 pixels on average. Hence, resizing the strawberries to forcefully fit the models larger input size will cause major discrepancies as this is too significant of a change. The fruits they used to detect are much larger than strawberries and therefore their original image sizes while not provided seem to be close to the pre-trained models input size. The second issue pertains to my dataset having a limited number of training data as well and thus not being able to take advantage of fine-tuning will hurt my case even more. R-CNN usually consist of deep networks and not having enough data can cause overfitting or false positive detections. Their results had quite a few false positives even after fine tuning due to their limited dataset and thus my custom CNN will likely detect a few false positives of its own. Unfortunately, they did not provide a specific metric to determine the number of their false positives. Figure 6 displays the false positives shown in their paper.

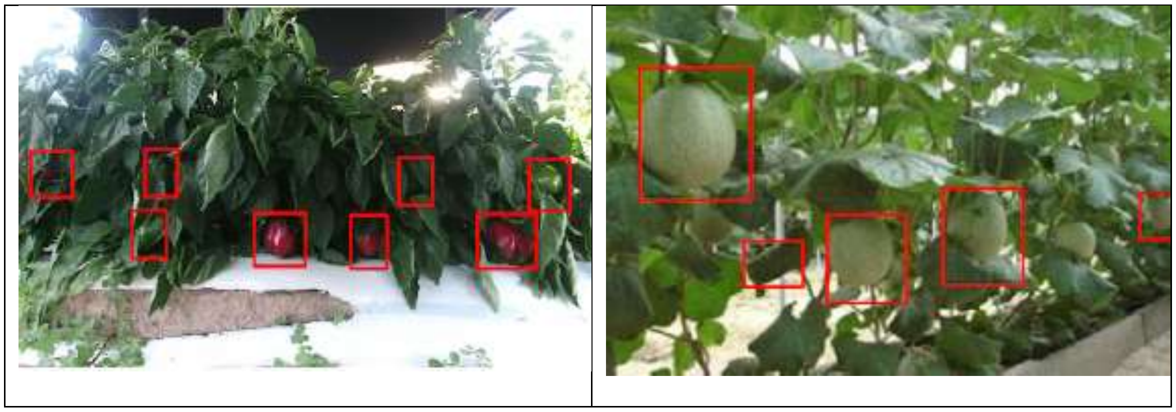


Figure 6: False Positive Detections as a Result from the R-CNN Network

2.4 Deep Convolutional Neural Network for Automatic Discrimination between Fragaria x Ananassa Flowers and Other Similar White Wild Flowers in Fields (P. Lin et al., 2018)

This paper focuses on flower detection for four different white petal species. Their database composed of distinct white flower species in Androsace umbellata Merr, Bidens Pilosa L., Trifolium repens L. and Fragaria x ananassa. This publication was selected for review as strawberry flowers are part of my dataset as well. Before the strawberry growth cycle occurs, they are represented as small white flowers similar to the ones described in this paper [14]. Three different techniques were applied to classify the flowers; Pyramid Histogram of Oriented Gradients (PHOG) , Scale Invariant Feature Transform descriptor (SIFT) and Convolutional Neural Network (CNN).

PHOG is a derivation of HOG with the major difference being once the gradient image is split into cells, the histogram of those cells are taken at several pyramid levels. Applying spatial pyramids enables the descriptor to consider features at different scales which was not possible before with general HOG. Refer to [15] for more details pertaining to PHOG.

SIFT is a key point descriptor which employs the Difference of Gaussian filter to obtain these interest points. Details on SIFT out of this sections scope, refer to [16] for more information.

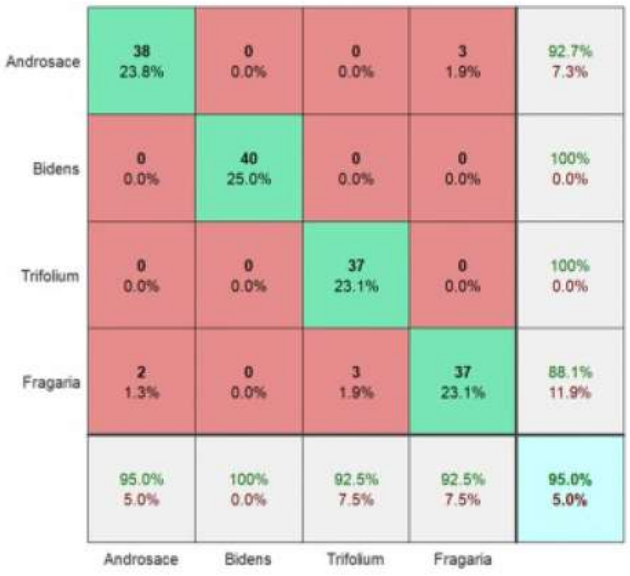
2.4.1 Procedure

Unfortunately, the paper did not discuss the implementation procedures for the PHOG or SIFT techniques in much detail and therefore only a quick overview of the CNN implementation will be focused upon. A total of 400 images were part of their dataset, where each flower specie was allocated 100 images. Out of the 100 images provided for each specie, a 60:40 ratio was used for the training and testing set. The CNN composed of 5 convolutional layers followed by 3 fully connected layers where the networks input size was 227x227. The activation function used throughout the network was “ReLU”.

2.4.2 Results and Analysis

The testing dataset was limited to a total of 160 images, 40 for each specie. Despite this the CNN performed well with a total accuracy of 95%. Due to CNN’s possessing deep feature extraction layers, they can still perform well at times even with limited datasets. The confusion matrix provided by the paper for the CNN results can be seen in Figure 7. However, the PHOG and SIFT descriptors clearly required more training data as they were only able to achieve an accuracy of 63.1% and 55.6% respectively.

Limitations: I will not have the luxury to resize my images to a 227x277 input size as the strawberries in my testing images are much smaller. Furthermore, when applying object detection over classification, one tends to run into more challenges, as the model is being exposed to variety of background environments through the windows which were potentially not part of the initial training dataset.



Androsace	38 23.8%	0 0.0%	0 0.0%	3 1.9%	92.7% 7.3%
Bidens	0 0.0%	40 25.0%	0 0.0%	0 0.0%	100% 0.0%
Trifolium	0 0.0%	0 0.0%	37 23.1%	0 0.0%	100% 0.0%
Fragaria	2 1.3%	0 0.0%	3 1.9%	37 23.1%	88.1% 11.9%
	95.0% 5.0%	100% 0.0%	92.5% 7.5%	92.5% 7.5%	95.0% 5.0%
	Androsace	Bidens	Trifolium	Fragaria	

Figure 7: Confusion Matrix Representing the CNN Results

2.5 Deep Fruit Detection in Orchards (S. Bargoti & J. Underwood, 2016)

Three types of fruit detection were attempted in this paper; mangoes, almonds and apples [17]. The primary technique employed was an R-CNN. The general procedure for R-CNN's can be reviewed again in an earlier discussed paper.

2.5.1 Procedure

The CNN employed as the feature extraction network in the R-CNN was the VGG16 network. While VGG networks are known for their accuracy and ability to generalize to the dataset well, two issues prevented them from training the network from scratch.

(1) Very limited training data was available to them; 729 apples, 1154 mangoes and 385 almonds.

(2) VGG Networks are notoriously slow for training due to their architectural design.

This issue was dealt by applying transfer learning/fine-tuning to the training process. The original image dimensions for the apple, mangoes and almond were 1616x1232, 3296x2472 and 3456x5184 respectively.

2.5.2 Results and Analysis

480 images in total were used for testing. 90.4%, 90.8% and 77.5% of apple, mangoes and almonds were detected respectively. However, even after fine-tuning their results had a large amount of false positive and false negatives as seen in Figure 8. Once again, one must take note of the simple fact, if a large enough dataset is not provided, the probability of false positives being detected are very likely.

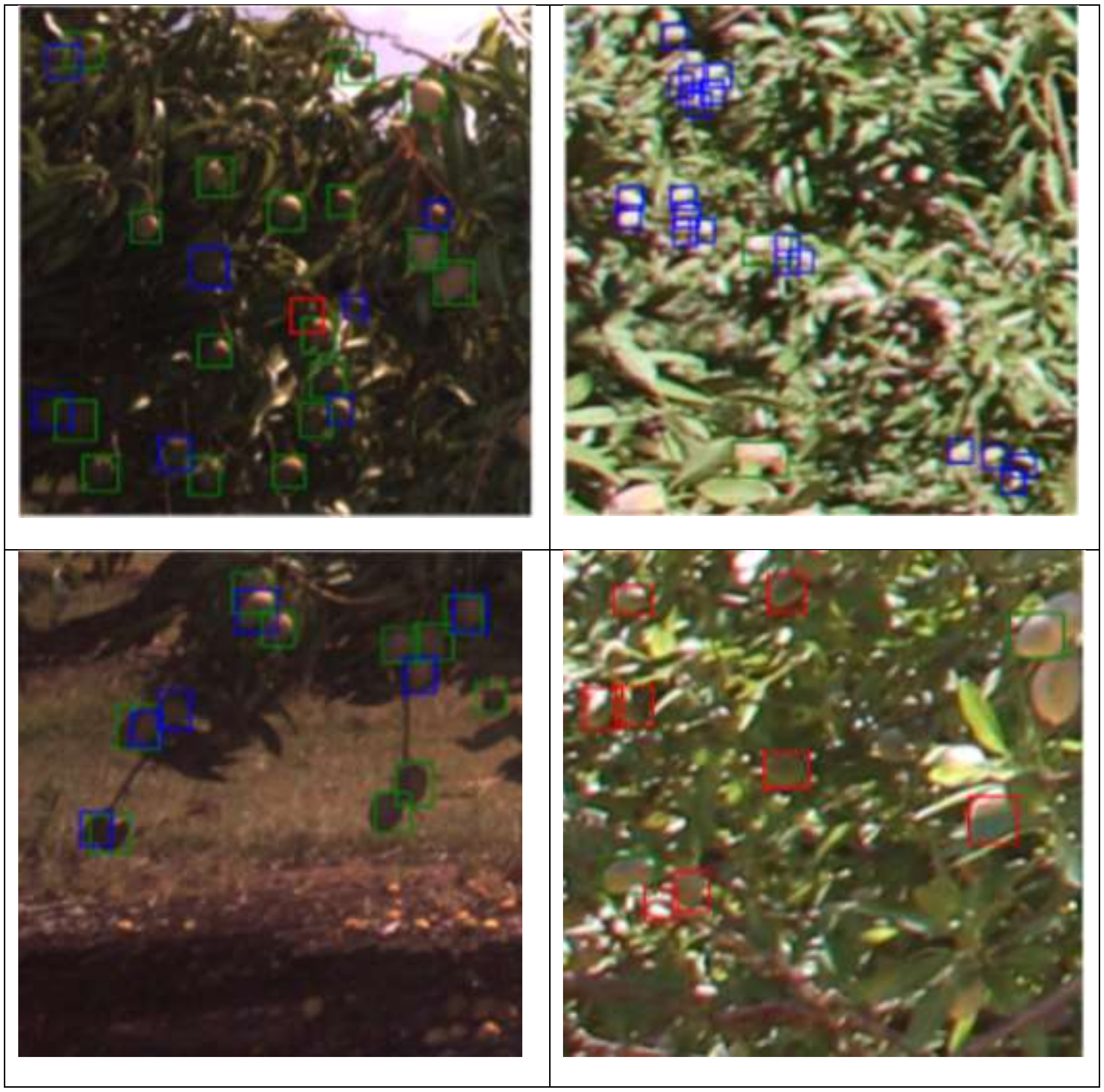


Figure 8: False Positives Represented in Red and False Negatives in Blue Bounding Boxes

Unfortunately, the paper did not provide exact numbers as to how many false positives or negatives were detected in total or on average per image. This proved once again if one wants to employ state of the art detection networks such as R-CNNs, lots of data will have to be available to them.

2.6 Detection and Counting of Immature Green Citrus Fruit Based on the Local Binary Patterns (LBP) Feature Using Illumination-Normalized Images (C. Wang et al., 2018)

This paper discusses three methodologies for detecting unripe citrus fruits; the first method involves applying k-means clustering [18], and circular Hough Transform (CHT) [19] to the pre-processed illuminated images. The second method was using LBP as a feature extractor followed by an adaboost classifier on just the RGB images. Finally, the third method employs the same procedure as method 2 but after applying an illumination enhancement technique to the training images.

2.6.1 Procedure

200 citrus and non-citrus samples were employed for training. For the first method, prior to extracting any of the citrus texture or applying k means segmentation [18] to the images a pre-processing step was implemented which essentially dealt with increasing the brightness of the image. The general steps to enhance image illumination were as follows:

(1) Resize RGB image to 800x600.

(2) Apply fast bilateral filtering-based retinex to enhance the images illumination.

(3) Thereafter, apply 2D discrete wavelet transform to fragment each R, G, B channel into low. and high frequency components.

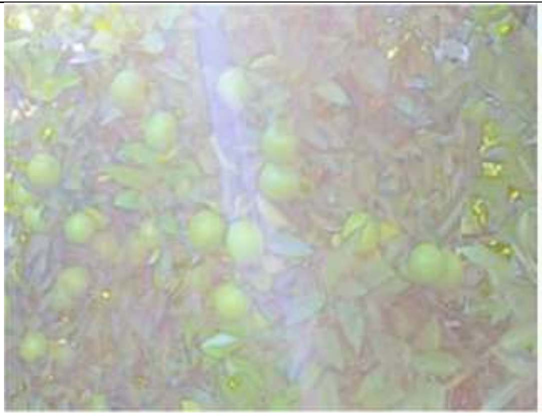
(4) Apply Histogram Equalization and Contrast Enhancement on all frequency components.

(5) Convert 2D frequency components back to RGB space and recombine the channels.

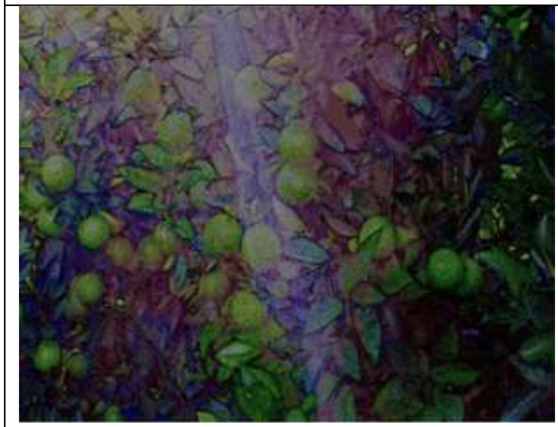
(6) Normalize the illuminated image obtained.

Reviewing each step, in complete detail is out of the scope of this section, however one can refer to the paper for more details [20]. The pre and post illumination results on a citrus orchard can be seen in Figure 9.

a) Original Image



b) After Illumination enhancement



c) After normalizing image b

Figure 9: Images Results after Pre-Processing

Thereafter, two training stages took place; the initial involved extracting the texture features from the raw RGB cropped samples while the second stage involved extracting the texture features from the cropped samples of the illuminated version. The LBP extraction technique was discussed briefly when reviewing an earlier paper in a previous section.

2.6.2 Results and Analysis

Table 1: Citrus Detection Results from all Three Methods

Methods	Total Fruit	True Positives	False Positives	Missed
K-Means + CHT	458	329 (71.8%)	113 (24.8%)	129 (28.2%)
LBP (RGB)	458	351 (76.6%)	121 (26.4)	107 (23.4%)
LBP (Illuminated)	458	392 (85.6%)	54 (11.8)	66 (14.4%)

This paper showed the importance of applying pre-processing to the training data as it can have a significant improvement on the results. The pre-processing technique will definitely be used as inspiration on my training samples through data normalization and segmentation. However, the k-means and CHT techniques will not be employed as one of the techniques on my project as this and previous papers reviewed have proven better results are achievable using other state of the art techniques specific to object detection. LBP will be one of the techniques employed, but a common theme when applying LBP seems to be the high false positive rates, this will prove to be a challenging issue to tackle when implementing it on my testing dataset especially with the strawberries being much smaller and the background being more complicated.

The publications reviewed in this section provided valuable information in understanding a variety of techniques implemented in academia so far for fruits detection. HOG and LBP were commonly used feature extraction algorithms which performed relatively well and hence I will start by implementing those techniques. CNN’s have been the go-to architecture in recent years. The results from CNN based architectures were robust even under limited datasets. Therefore, after HOG and LBP, I will proceed to employ a CNN based strawberry detector.

Chapter 3

TECHNICAL BACKGROUND

Before going into the implementation, a quick technical overview will be covered on the techniques which will be employed. This should give the reader a much better intuition on the specifics when analyzing the implementation section.

3.1 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) are feature descriptors which are known for their ability to understand objects by calculating the gradient magnitude and direction [8] [9]. The general steps taken to obtain the gradient based histogram are as follows:

(1) Extract the region of interest.

(2) Apply Sobel operator on the ROI to obtain gradients and angles.

Magnitude and angle are determined using the equations seen in Figure 10b.

$$\begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

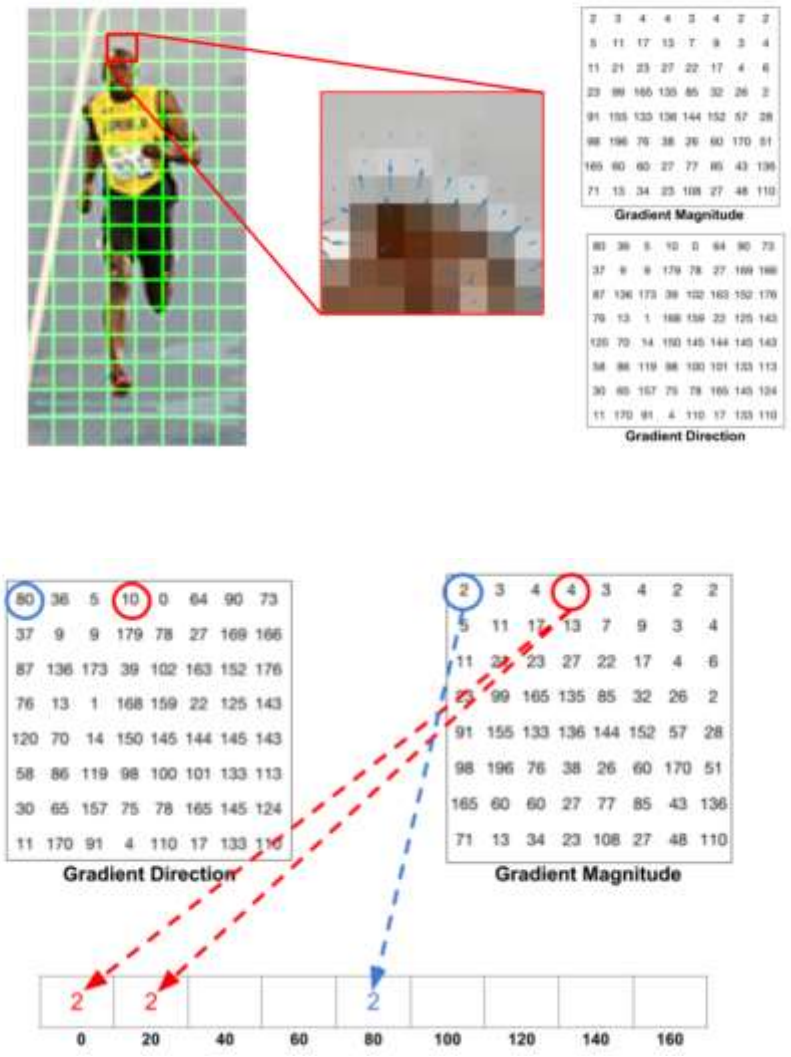
a) Sobel filters to extract the x and y gradients

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

b) Magnitude and angular equations

Figure 10: Filter and Equations Employed to Derive the Magnitude and Angle

(3) Split image into multiple cells and calculate the histogram of each cell.



80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

0	20	40	60	80	100	120	140	160
2	2			2				

Figure 11: Gradients Being Extracted from Each Cell for a Sample Image

(4) Group a block of cells and normalize the histograms respectively.

(5) Generate the feature vector by concatenating the histograms.

The feature vectors can then be employed to learn the shape of the object. HOG is simply a feature extractor and therefore the features must be past into a classifier for it to be trained upon and eventually create a model.

3.2 Scale and Rotation Invariant Local Binary Patterns

Local Binary Patterns (LBP) are another type of feature extractor which have the ability to extract texture from the image. LBP's are processed by iterating a window operator across the image. The general overview of the original LBP algorithm is as follows:

```
for each image pixel in 3x3 filter window
    if a surrounding pixel is greater than the central pixel
        threshold that respective pixel to a 1
    else
        threshold to 0
        Array = Mat2Array(thresholded_window)
        Decimal_val = Array2Dec(Array)
        Output_Mat(i,j) = Decimal_val
```

The texture-based features extracted from the original algorithm had one major flaw; it lacked invariance. A simple 3x3 window can only extract 2^8 bits worth of features and cannot be adjusted to deal with the original images size.

This issue was dealt by T. Ojala et. al [5] by introducing circular windows which could be expanded to any scale based on the number of points and radius parameters. This small change proved to be instrumental as LBP's originally were not scale and rotation invariant. Beyond the initial parameter change the remainder of the algorithm is generally the same. The circular windows at different scales can be seen in Figure 12.

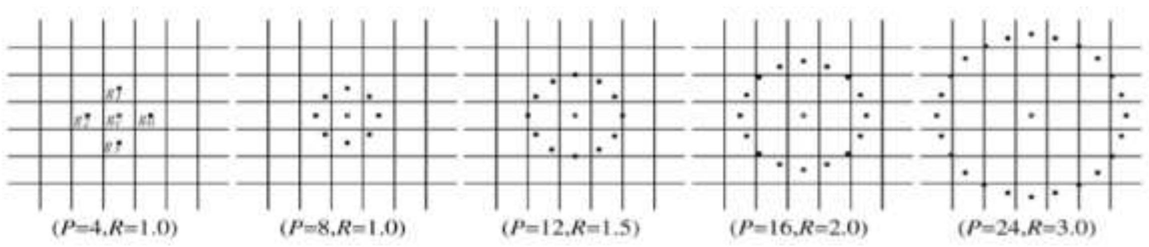


Figure 12: Scale and Rotation Invariant LBP Windows

One must keep in mind that similar to HOG, LBP's are just feature extractors and therefore the features must be passed into classifier for a training model to be built from it.

3.3 Neural Networks

3.3.1 Artificial Neural Networks

Neural Networks are algorithms loosely modeled around the human brain, designed to recognize patterns. However, before getting into the details pertaining to how the entire network functions, one must understand how a single neuron works first. A single neuron is labeled a perceptron. A perceptron takes in multiple inputs and produces an output [26]. The inputs are connected to a neuron through a synaptic link. The synapses are represented as weights. Weights are simply factors expressed as real numbers which determine the importance of the input in respect to the output. A linear transformation between the inputs and their respective weights are performed upon arrival at the neuron. The linear transformation can be expressed as $w \cdot x + b$. The variable b represents bias, which simply enables the points to fit around the model better by shifting it in the vertical direction, preventing it from fitting around the origin at all times. A single perceptron can be seen in Figure 13.

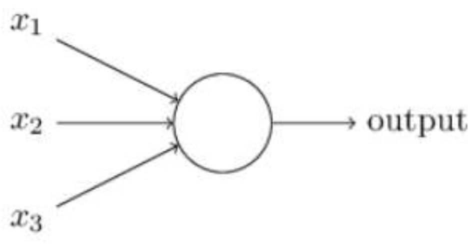


Figure 13: A General Three Input Perceptron

Then the linear equation is passed into an activation function. The activation function determines if the neuron should "fire" its output. Many mathematical functions are employed to represent the activation function, most commonly used functions are Sigmoid and Relu which are expressed as $1/1 + e^{-x}$ and

$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$ respectively.

A single perceptron isn't strong enough to recognize patterns from complicated datasets and thus an entire network of them is employed to serve this purpose. Figure 14 displays the general layout of a multi-layered perceptron.

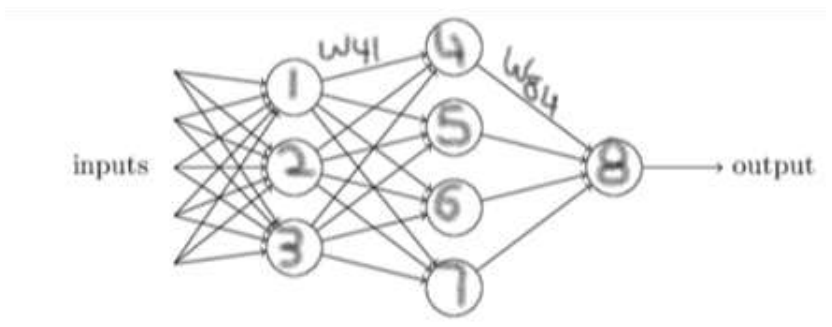


Figure 14: A General Two Layered Artificial Neural Network

Each perceptron works together to make complex decisions; in the first layer raw input values are passed in and hence simple decisions are made and into the hidden layers more abstract decisions are made by utilizing the information giving by the previous layer [26]. However, simply feeding data forward through the network doesn't train the network. The weights must be adjusted from its errors for the network to truly learn. The back-propagation methodology performs weight training and thus decreasing the error rate.

3.3.2 Back Propagation

The weight training process begins after the input has gone through the all the layers in the forward direction and the error is calculated at the end of the output layer [27]. The weights are trained during the backward pass using gradient descent which attempts to decrease the loss as much as possible by improving the weights. Gradient descent involves taking the derivative of the error is respect to the weights $\frac{dE}{dW}$. However, due to the network involving multiple layers and thus having multiple weights, chain rule derivation must be applied to calculate the new weights δW. A quick mathematical explanation of the back-propagation procedure on neurons 8,4,1 from Figure 14 will be demonstrated for additional clarity.

$$\frac{dE}{d\mathrm{W}84} = \frac{dE}{dO8} \cdot \frac{dO8}{d\varphi 8} \cdot \frac{d\varphi 8}{dW84} \quad \text{[Layer 3]}$$

$$\frac{dE}{d\mathrm{W}41} = \frac{dE}{dO8} \cdot \frac{dO8}{d\varphi 8} \cdot \frac{d\varphi 8}{dO4} \cdot \frac{dO4}{d\varphi 4} \cdot \frac{d\varphi 8}{dW41} \quad \text{[Layer 2]}$$

The two equations above represent the chain rule procedure required to update the weights, where φ represents the neurons respective activation function. The same format would follow for all the weights when iterating backwards to update them individually. The training algorithm occurs for n number of epochs where n is supplied by the user. An epoch simply represents the forward and backwards pass for all samples in the training dataset. Multiple epochs cycles are performed to generate strong results.

3.3.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specific type of neural network. It is designed to train on image type data [28]. It involves iterating a convolutional filter over each input image data and thus generating feature/activation maps. Figure 15 displays a filter in respect to its input image data.

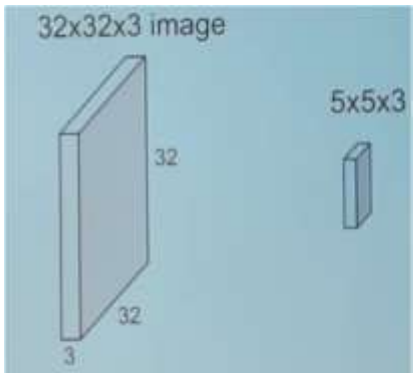


Figure 15: Convolutional Kernel in Respect to Image

Each layer consists of a variety of kernels which represent different types of feature extractors. Iterating a new filter over an image each time results in a stack of feature maps for each respective filter. Each filter can represent different feature extractors. Due to the first layer receiving raw pixel data, earlier layers tend to detect low level features such as edges and texture while deeper hidden layers generate much more abstract features. The feature map build up for 6 different filters after a single convolutional layer can be seen in Figure 16.

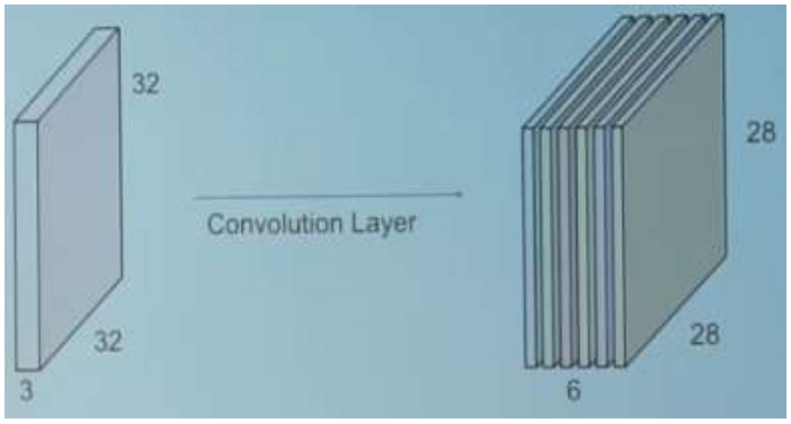


Figure 16: 6 Unique Feature Maps Generated from 6 Different Kernels on the Input Image

A pooling layer usually follows the convolutional layer to reduce the spatial size, however, the layer ordering is dependent on the network architecture. Reducing the feature map dimensions significantly decreases the number of parameters and thus increases computational efficiency as well. Dimension reduction through pooling can be seen in Figure 17.

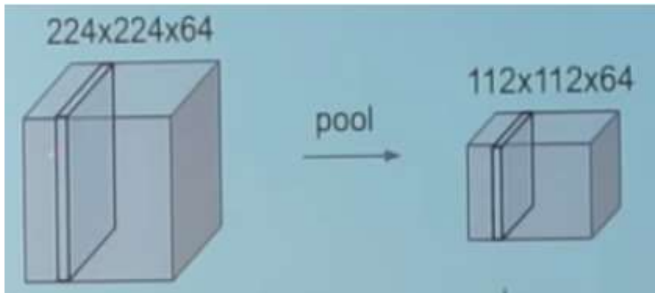


Figure 17: Result of Applying Pooling Layer

3.3.4 CNN vs MLP

In terms of multi-layered perceptron networks, CNN's function in a very similar manner where $w \cdot x$ +b is followed as well, x is the input image or feature map (depending on the layer) and w is the filter. Upon performing the linear transformation, the feature maps are passed through an activation function as well where the function determines which pixel values stay and which get subdued. The back-propagation algorithm is implemented in a similar manner as well, where the filter values/weights are updated to train the network.

Despite much similarities, there are two key difference between CNN's and MLP's.

- All the weight connections are independent in MLP's while weights are shared for each feature map in CNN's due to the same filter being employed for the entire image.
- Due to the convolutional process, CNN's are spatially invariant and hence each neuron can only contain a single filters information not the entire images. This is unlike MLP's where each neurons in the next layer receive all the information from the previous layer.

In summary, Neural Networks are extremely powerful techniques over previously discussed feature extractors in this section for three major reasons. First, they can function as end to end optimizers and thus extract features while being able to classify the data as well. Second, multiple features can be extracted from each input data unlike single feature extractors such as LBP or HOG. In CNN's multiple filters can be employed to extract a variety of low- and high-level features to better train the network. Finally, neural networks have the ability to represents its data in much higher non-linear dimensions and thus is able to group the sample points easily for accurate classifications.

Chapter 4

IMPLEMENTATION & RESULTS

4.1 Ripe Strawberry Detection

4.1.1 HSV Color Segmentation

The HSV color space was selected as the color channel to isolate the ripe strawberries. Attempting to isolate the red strawberries under the RGB color space can cause misclassifications since RGB images tend to generate strong glares over the objects. The HSV color space was tested upon, where HSV represents Hue, Saturation and Value. The Hue channel defines the color of the image, where each color is represented in terms of angles between 0-360. All the general colors and their respective ranges can be seen in Figure 18.

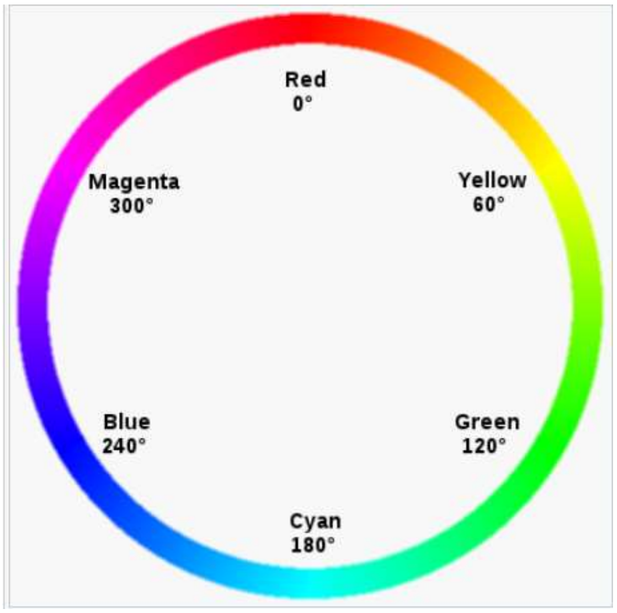


Figure 18: Angle Ranges of the H Channel

The S channel represents the “purity” of the color, lowering the number gives the image a much more faded effect.

Finally, the Value channel represents the intensity or brightness of the image, where 0 is essentially black and 255 is white.

Employing the HSV color space, proved to be very effective for the following reasons:

(1) dealt with light better than the original RGB color space and thus reduces the reflectance and shadow effects casted upon the objects, which was caused by the sun in this dataset.
(2) The robust angular range representing the colors reduced the challenge of isolating the strawberries based on their color.

This can be very effective for isolating ripe strawberries as the images in the dataset don't contain much red in it besides the ripe strawberries themselves.

To prevent repetition, the figures displaying the HSV images and the strawberry isolation can be seen directly in the Ripe Strawberry Implementation section below.

4.1.2 Manual Training

The strategy taken to localize the ripe strawberries within the test image revolved around HSV color space segmentation. The HSV color space was selected because Hue and Saturation To determine the appropriate range which works well with the test images a manual training methodology was applied. This methodology consisted of multiple trial and error experimentations where various HSV segmentation values were tested on a smaller training set. The experimentation was straightforward as the ripe strawberries were the only red colored objects in the entire image, making it quite simple to distinguish the ripe strawberries form the background.

The segmentation steps taken were as follows:

First, convert the original image from RGB to HSV color space. Figure 19 displays the original image compared to the converted HSV image.

```
#convert image color channels
hsv = RGB2HSV(im)
```

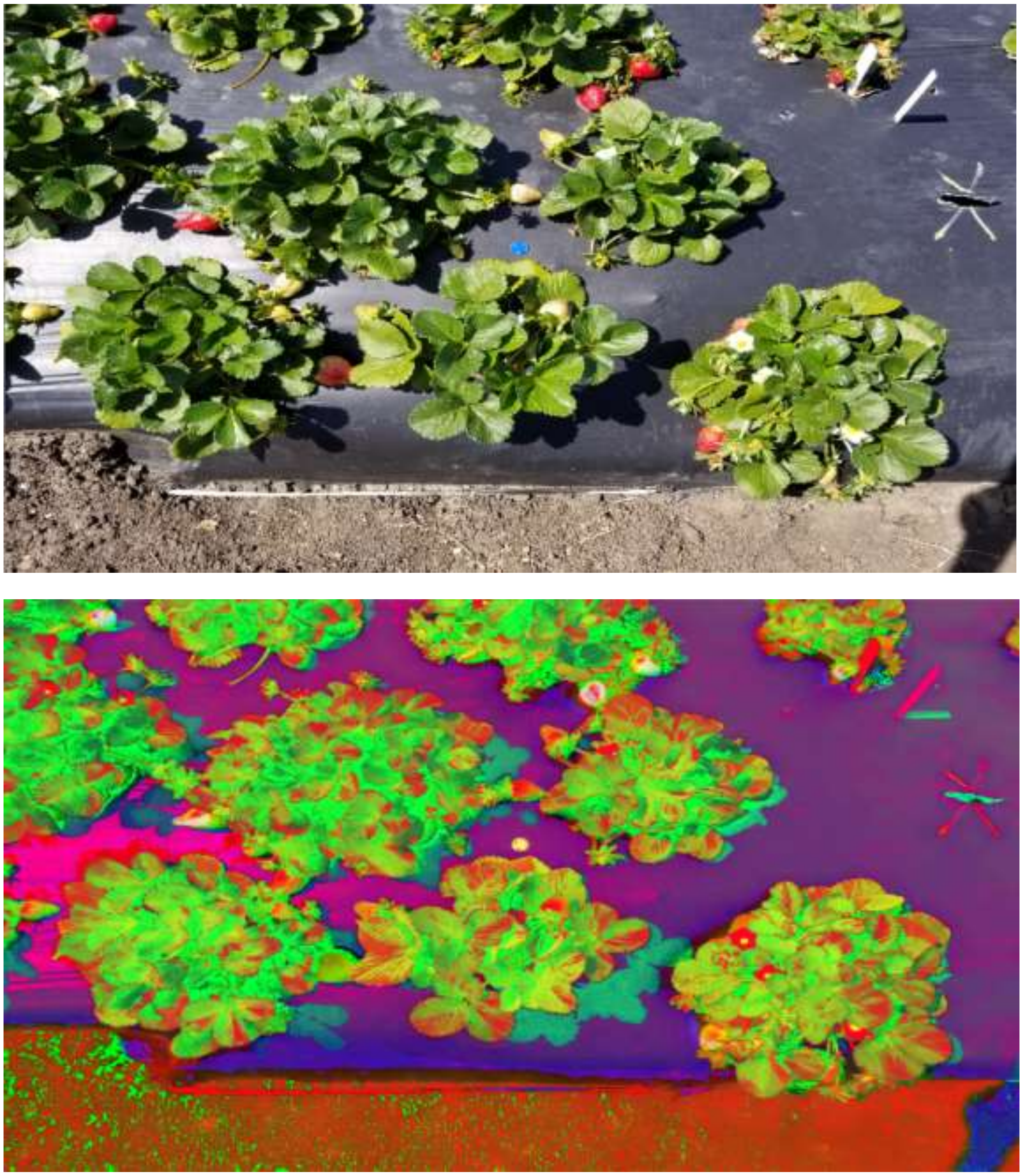


Figure 19: Original RGB Image vs HSV Image

Then, isolate the red color space within the HSV image. After multiple trail and errors, the range of values were determined and then stored into an array. Two range sets were chosen due to red being represented twice, the beginning and end of the hue color spectrum as seen earlier in Figure 18.

```
#range1
lower = array([0,90,85])
upper = array([5,255,255])
#range2
lower = array([165,90,85])
upper = array([180,255,255])
```

Thereafter, threshold the pixel values representing only the red colored objects in binary form. Given the specific range of values, the pixels on the HSV image were isolated.

```
#determine pixel ranges to threshold upon
mask1 = Range(hsv, lower, upper)
mask2 = Range(hsv, lower, upper)
mask = mask1+mask2
```

Besides simply displaying the binary version of the isolated red pixels, the   colored version was displayed as well by applying a bitwise AND between the binary mask and the original image to only display the red colored pixels (Figure 20).

```
#apply AND operation to threshold the color space
out = bitwise_and(im, mask)
```

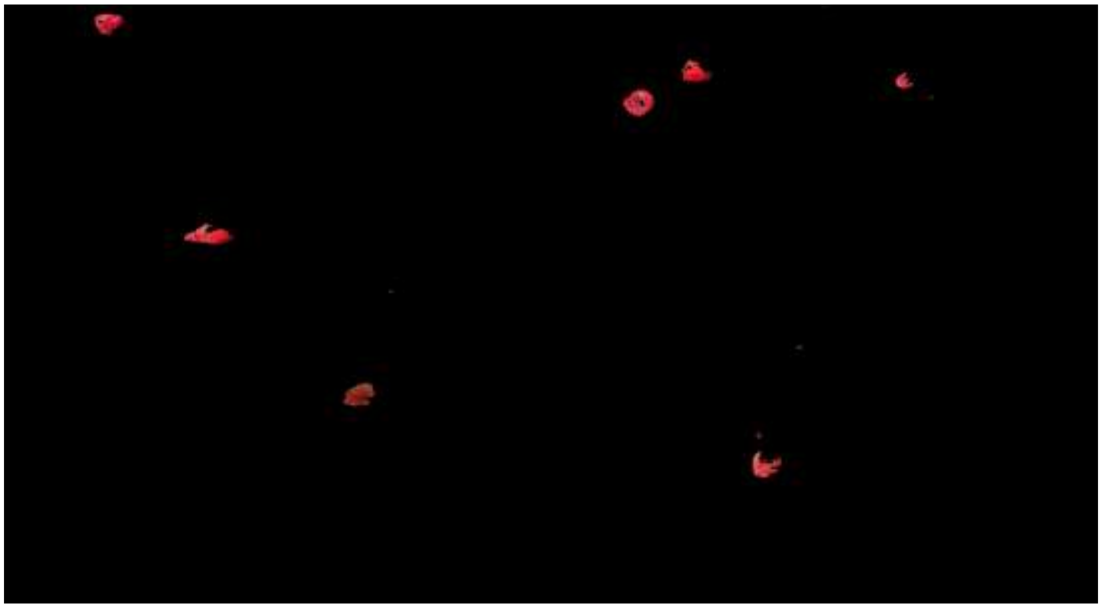
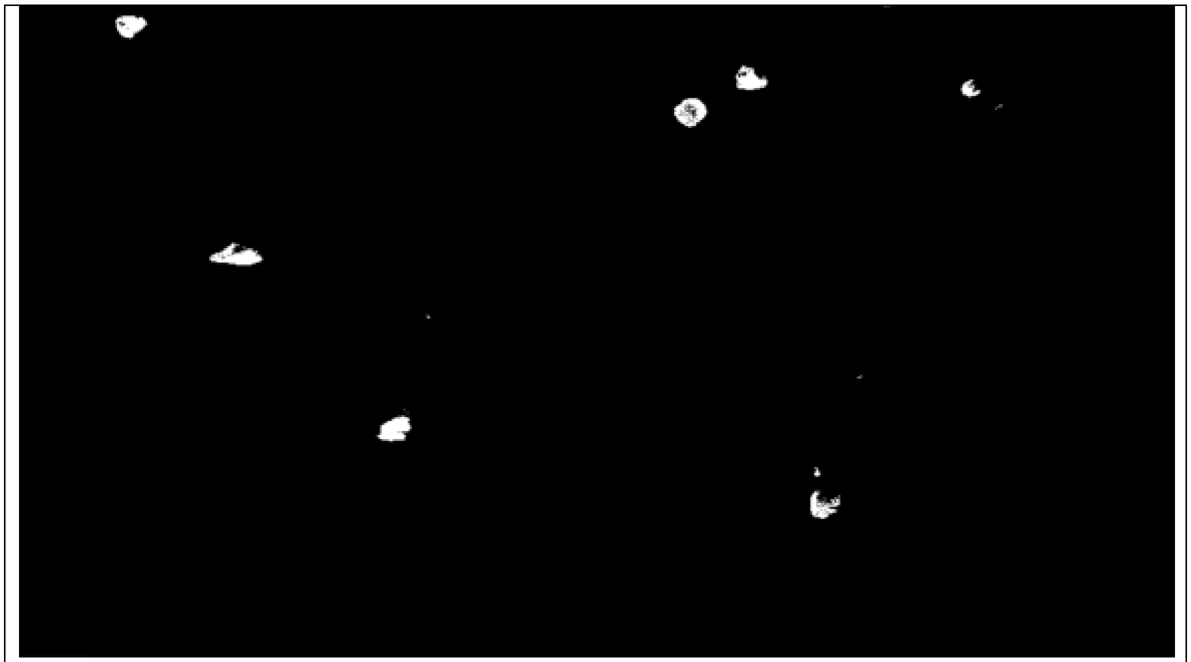
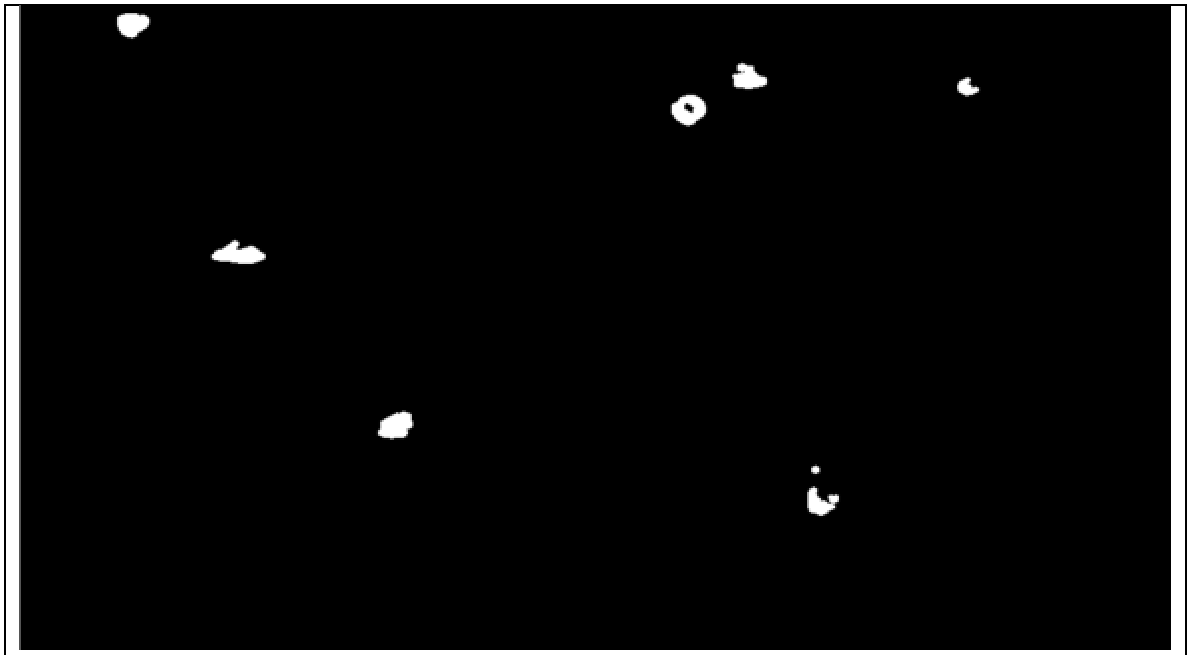


Figure 20: Ripe Strawberries Isolated Based on Red Color Segmentation

As a precautionary step, morphological operations were applied onto the images to reduce minor noise grains. The operations applied were erosion followed by dilation. While erosion got rid of any small noisy pixels, dilation restored the objects that might have gotten damaged due to the erosion. The structuring elements applied onto the binary images were 3x3 and 5x5 elliptical shapes, for erosion and dilation respectively. Figure 21 displays the before and after results of applying morphological operations respectively.



a) Before image consisting of small pixels along with the larger ones



b) After image primarily including the larger pixels

Figure 21: Binarized Images Before and After Applying Morphological Operations

```
#structuring elements
se1 = StructuringElement(ellipse,(3,3))
se2 = StructuringElement(ellipse,(5,5))
#morphological operations applied
clean_im = erode(mask, se1)
clean_im = dilate(clean_im, se2)
```

The contours of the binarized objects (similar concept as connected components) were then determined to localize the objects within the image.

```
#determine the contours of the isolated “objects”
contours = findContours(clean_im)
```

Once the contours were found, an area threshold limit was set on the objects detected to prevent detecting extremely small objects. Extremely small detections are unlikely to be strawberries.

```
#find pixel area of contour object
area = contourArea(contours)
```

Given the contour coordinates from the binarized objects, place bounding boxes on those same locations in the original image for a cleaner visualization.

```
#get box coordinates around the contours and draw them onto the respective objects
(x, y, w, h) = boundingRect(contours)
rectangle(im, (x,y),(x+w,y+h), (255, 255, 0), 2)
```

4.1.3 Results

Table 2: Detection Performance of Ripe Strawberries

Test data	True Positives	False Positives	Clutter boxes	Total Berries	Accuracy (%)
1-21	48	2	2	56	86
22-43	97	5	1	101	96
44-65	99	6	3	109	91
66-87	40	1	0	42	95
88-109	91	6	0	95	96
110-131	37	4	1	39	95
132-148	108	5	0	117	92

Implementing simple color segmentation resulted in being an extremely effective technique as it achieved the goal of not only maximizing most of the true positives but also minimizing much of the false positives. This was primarily due to the ripe strawberries being the only red objects within the test images. Due to its primitive procedure the computation was also extremely efficient displaying each test result instantaneously. The average accuracy of the entire testing data set was at 93% as inferred from Table 2. A sample result can be seen in Figure 22.

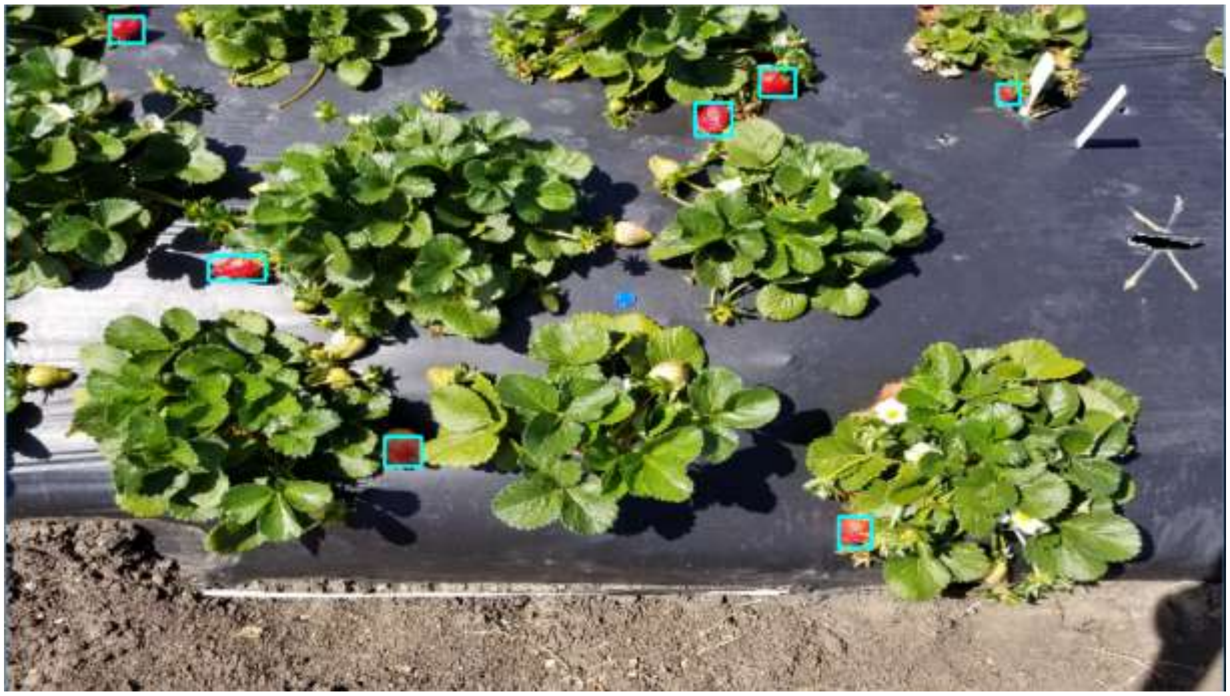
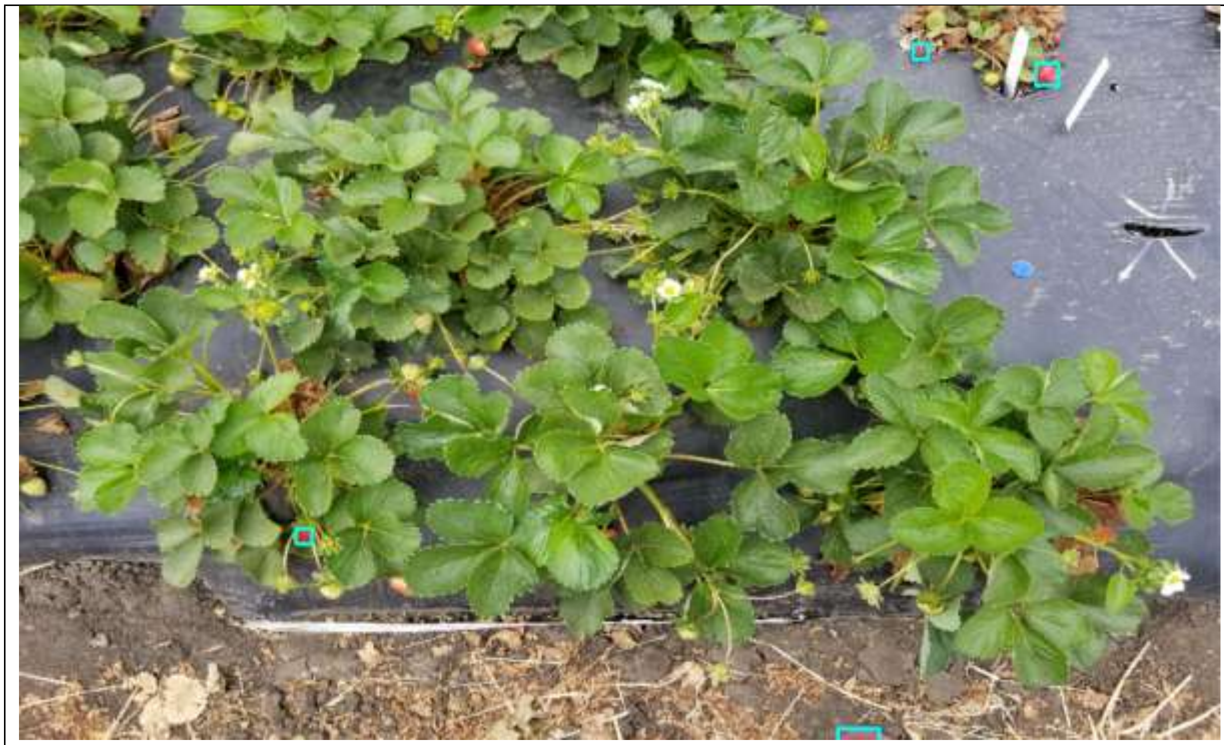


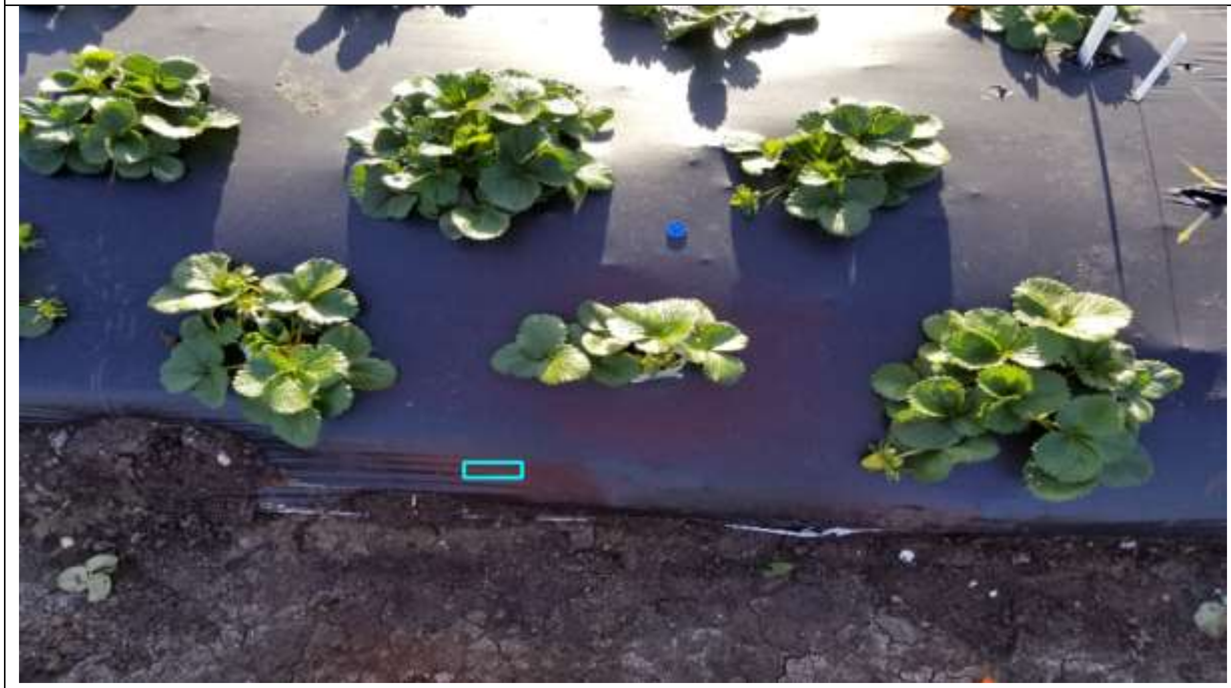
Figure 22: Bounding Boxes Applied on the Original Image (Final Result)

4.1.4 Flaws

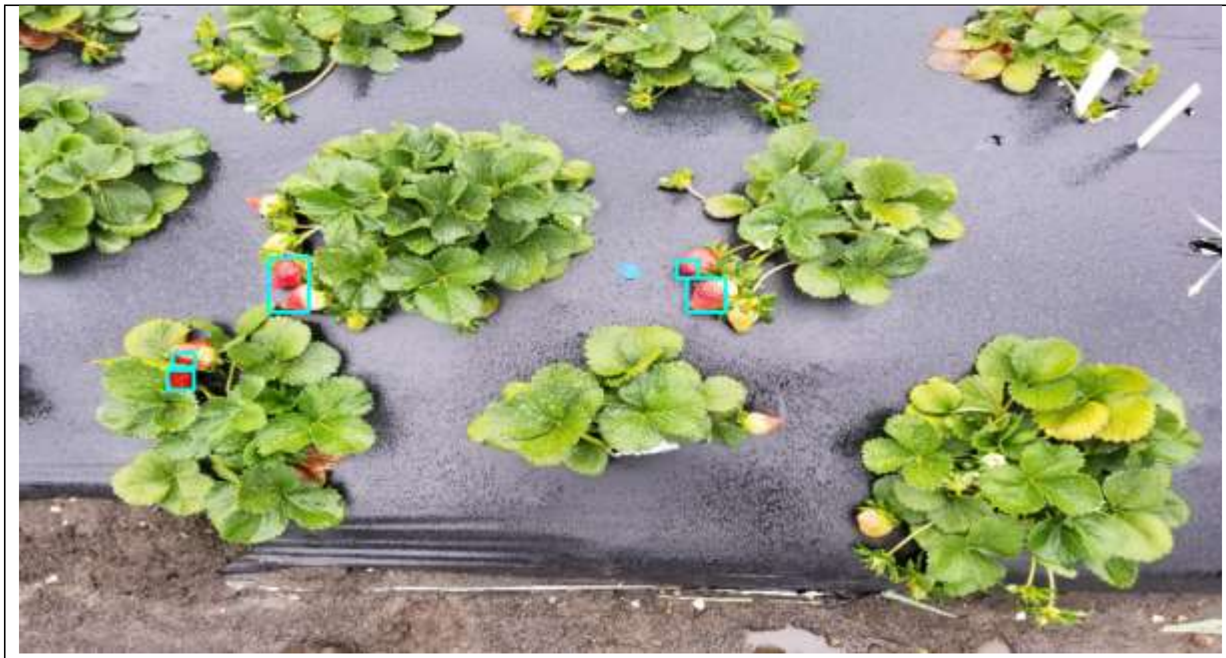
While this implementation was primarily successful, there were still three minor flaws that arose. First, some of the test images had a few false positives at the dirt/soil region as brown gravitates closely to red on the hue color spectrum and thus overlapping between the two colors can occur (Figure 23a). Second, the edge of the tarp on a few test images were misclassified as the sun would cause a strong reflection off the red strawberries onto the tarp, giving that specific region a red tint and thus confusing the system (Figure 23b). As displayed from Table 2 only 29 false positives were detected out of 148 test images; well below even 1 per image. The third and final flaw occurred where two red strawberries grew alongside each other thus classifying them as one strawberry. Figure 23c displays samples of the cluttering bounding boxes. However, this was not a major issue either as this occurred only a total of 7 times within the entire dataset.



a) Brown dirt and soil at the bottom being misclassified as a red strawberry



b) Red reflection off the tarp being misclassified as a red strawberry



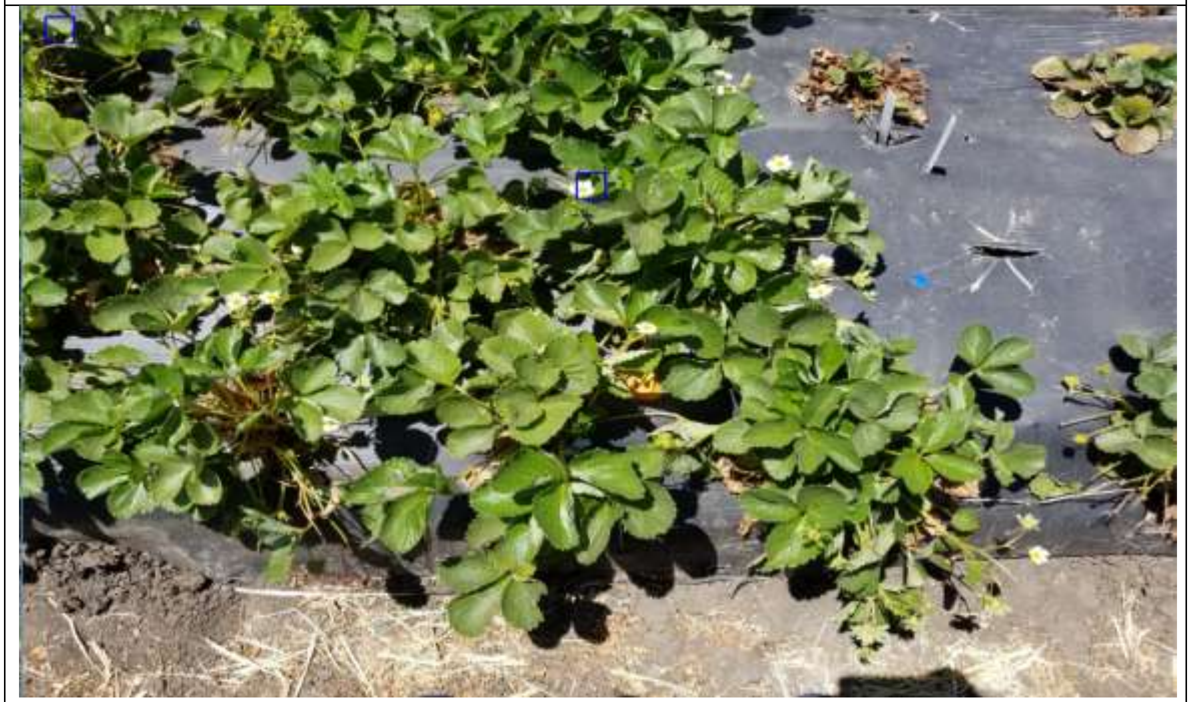
c) Two red strawberries (top-left) beside each other being classified as one

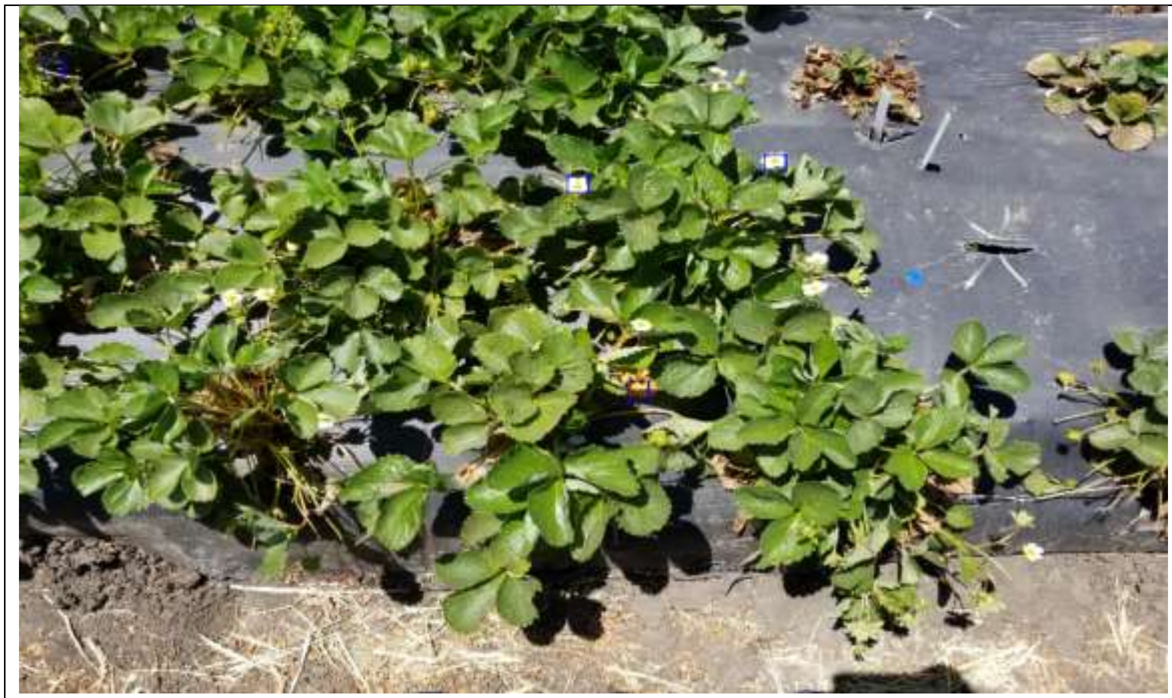
Figure 23: Sample Test Images with False Positives

4.2 Resolution

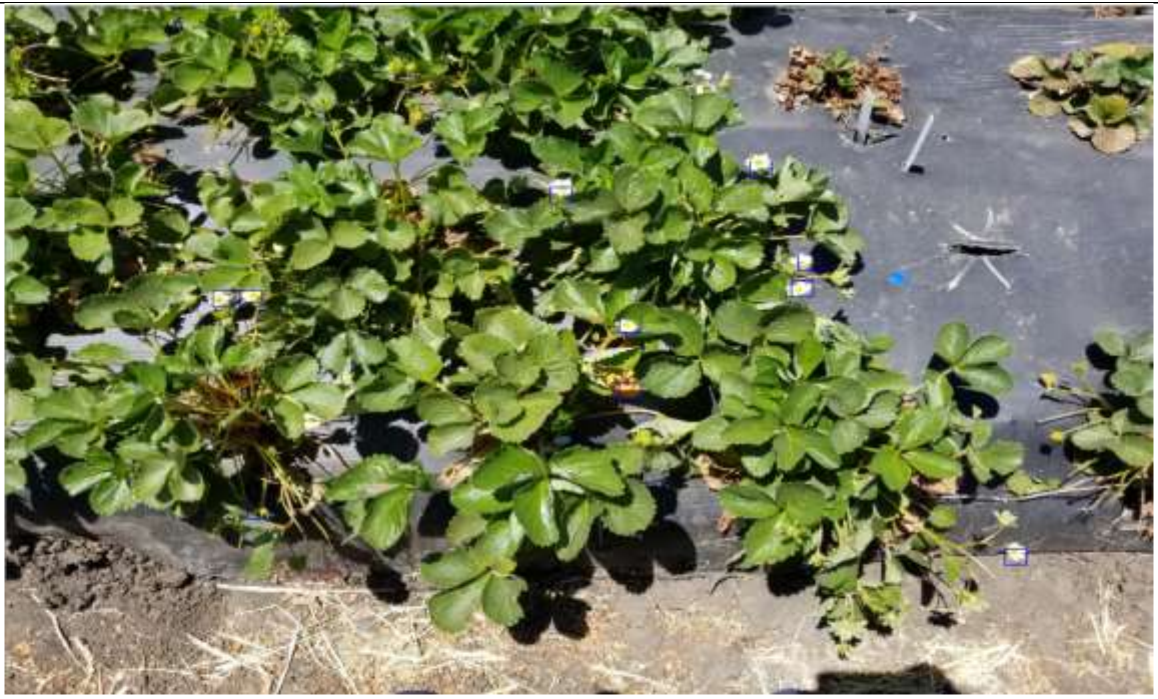
Key modifications in terms of accuracy and efficiency can be done under the appropriate image resolution. Flower and strawberry detections were attempted under three different resolution sizes to determine the lowest resolution at which maximum accuracy can be achieved. This also allows one to calculate the maximum height at which images can be captured for data collection purposes. To determine the best resolution, the conspicuous criteria would have to be met of detecting the most flowers or strawberries within the image. These experiments were attempted by applying the appropriate window dimensions for the respective image size to prevent disrupting the accuracy. Figure 24 displays the detection results under various resolution dimensions for flower detection. The lowest resolution size performed the worst as it missed most of the flowers and strawberries. The augmented image dimension performed the best as it detected most of the flowers. An important observation to note; even though 4000x5000 had better results, a resolution of 2000x3000 might be a suitable option as well as the computation would be more efficient but at the expense of fewer detections. This decision would depend on the user and their application. In conclusion, in terms of pure accuracy increasing the image resolution will improve the accuracy percentile.

a) Image size: (1000,2000)


b) Image size: (2000,3000)



c) Original Image size: (3024,4032)



d) Augmented Image size: (4000,5000)

Figure 24: Flower Detection Under Various Resolutions

4.3 Unripe Strawberry Detection

4.3.1 Local Binary Patterns

4.3.1.1 Training Procedure

Before testing the performance of the LBP, training must occur on the strawberry dataset using the feature descriptor and then fitted into a classifier to generate a model. Each previously cropped image would get extracted from the training folder and then get resized to a 180x180 dimension. After multiple experimentations, this dimension was determined as the optimal size for extracting the strawberry’s texture. A significant decrease in dimension size would be too small for the descriptor to extract good features from, thus decreasing accuracy and increasing the false positive rate. The image was then converted into a single channel (grayscale) as LBP only works on 2D images.

```
#convert each training data to grayscale
im = RGB2GRAY(im)
```

Thereafter, the image would pass into the LBP function. LBP’s consist of three primary parameters to take note off, numPoints, radius, method. NumPoints represents the size of the window which will traverse over the image matrix to perform texture analysis at that respective region. A radius parameter is also present due to rotation-invariant LBP functions utilizing a circular shaped window instead of square per usual. Finally, method determines the type of LBP pattern applied. There are various types of patterns, but the most commonly used type is “uniform”. This was also the type that produces the best results.

```
#image and respective parameters passed into LBP function
lbp = local_binary_pattern(image, numPoints, radius, method="uniform")
```

The LBP function returns a matrix representing the specific texture pattern of the respective training image. The pattern representation must then be converted into a histogram, so it can be passed into the classifier. This was done by converting the matrix into a 1D array. Thereafter, the array was then represented as a histogram. The histograms were also normalized for increased accuracy and computational efficiency.

Once a descriptor is created for each image, the images' descriptor and its respective label are fitted into the Support Vector Machine classifier. The transformational kernel for this SVM classifier was linear and its primary parameter usually represents the weighted margin separating both classes.

```
#define SVM model and then fit data and its respective labels into the model
model = LinearSVM(C=5)
model.fit(features, labels)
```

4.3.1.2 Testing Procedure

This task deals with object detection not simply image classification and thus a sliding window approach will be applied on the testing images. The testing data will consist of images representing the entire plant bed with multiple strawberries within it, as displayed earlier in Figure 2. To deal with this issue, the sliding window method creates a MxN window that iterates across the image horizontally given a step size.

```
#sliding window looping over entire image
for win_coordinates in slidingWindow(im, stride, windowSize=(W, H))
```

Each window iteration is extracted and passed into a pre-processing function which checks if the object within that window is bright yellow-greenish under the HSV color space which represents the first feature extracted to distinguish unripe strawberries. This preprocessing stage was implemented to increase computation efficiency. Most of the windows will initially be green due to the leaves in the background and thus, instead of running those irrelevant objects through the model, they can be filtered out initially. A sample yellow segmented window can be seen in figure 25.

```
hsv = RGB2HSV(window)
#apply yellow segmentation
lower = array([20,195,100])
upper = array([30,255,255])
mask = Range(hsv, lower, upper)
```

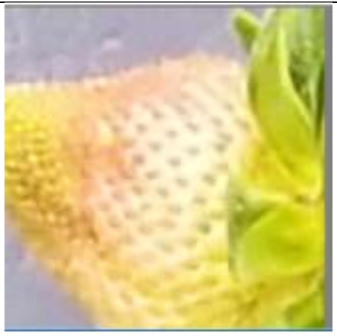
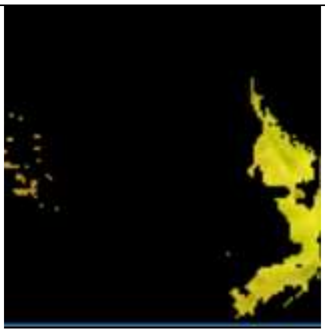
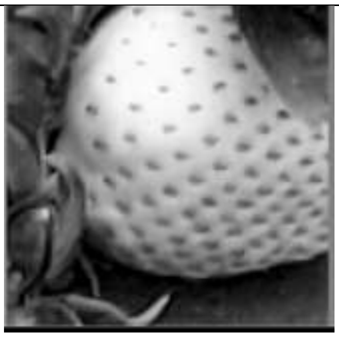
 a) Original window	 b) Yellow segmentation	 c) Blue channel filtering

Figure 25: Individual Sample Window

Connected components was then used to determine if an object existed within the window once segmentation was applied.

```
#connected components applied on binary image
blob = connectedComponents(mask, 4)
```

Then the object was binarized under the LAB color space. The LAB color space was employed as lighter colored objects tend to have high valued pixels under the Luminance channel of LAB and thus ensuring a bright unripe strawberry would get passed on once thresholding was applied, not just a leaf with light reflection over it. Figure 26 displays an unripe strawberry under the L channel of the LAB color space.

```
#threshold applied on LAB's luminance channel
lab = RGB2LAB(window)
l,a,b = SplitChannel(lab)
thresh = BinaryThreshold(l, 180, 255) #pixel value 180 or larger becomes 255
```

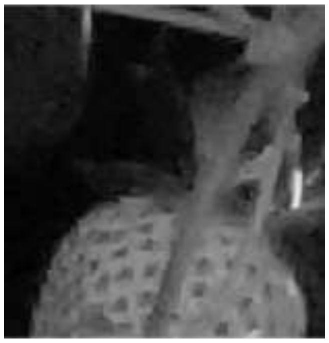


Figure 26: Unripe Strawberry Under L Color Channel

After thresholding the pixels, the area of the luminance channel and the variance of the HSV binary image was taken. Segmenting the window within such a specific range and checking additional statistical features early on saves time and computational complexities by simply moving on to the next window iteration. If the requirement is met then the LBP descriptor of that window is then passed into the SVM model to predict which class it belongs too. Due to LBP's employing histograms to describe the feature patterns, the testing windows dimension size can be different from the training if both histograms consist of the same range (0-255 or 0-1). The testing windows were set to a smaller dimension than the training windows due to the size of the testing image. The testing images were (4000,5000) and consequently if a testing window of (180,180) was applied on the testing images it would cover more than simply the strawberries. This would prevent quality feature extractions and thus disrupt the prediction process and result in misclassifications. To have equal training and testing window dimensions, the testing image would have to be significantly larger which is extremely computationally inefficient. Hence, the testing windows are set at (107,107) for the strawberries.

Finally, if the prediction is "strawberry" then another post-processing step is applied to that window. This post-processing step involves taking the average of the yellow segmented pixels within the window and the average of the pixels under the blue channel of the original RGB window image. An unripe strawberry under the blue channel can be seen in Figure 27.

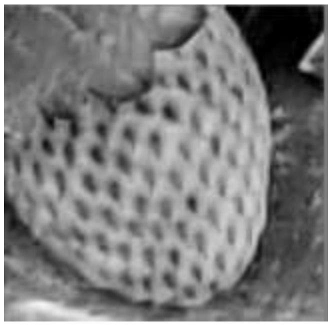
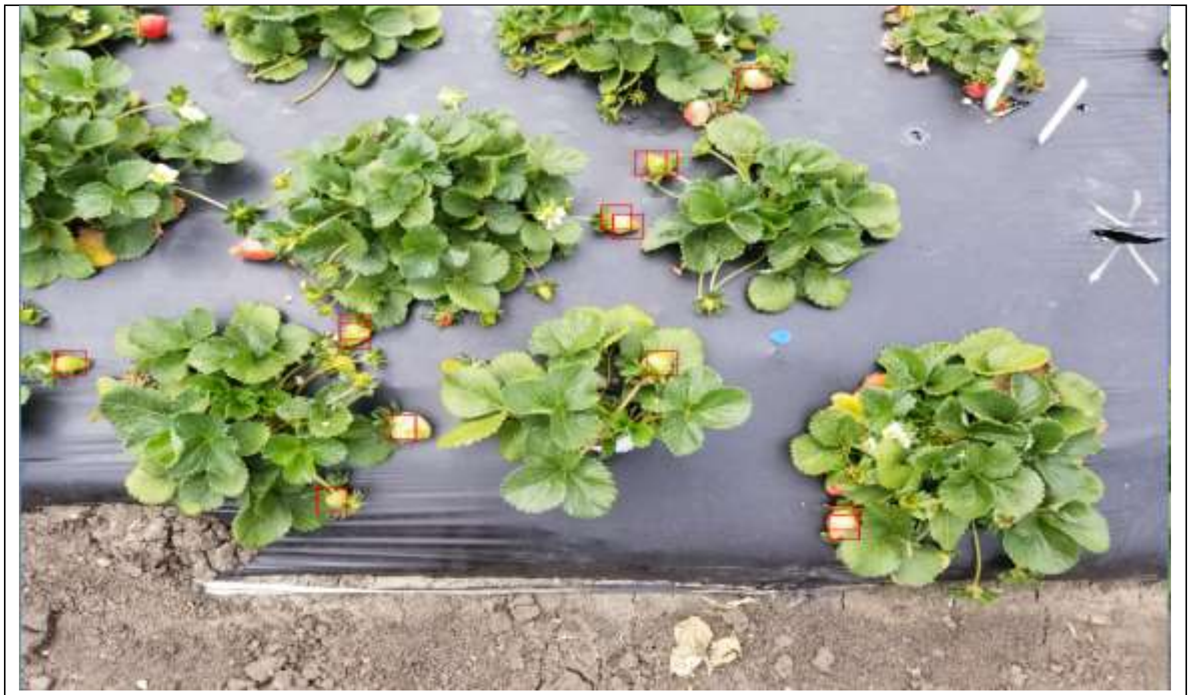


Figure 27: Blue Channel Representation

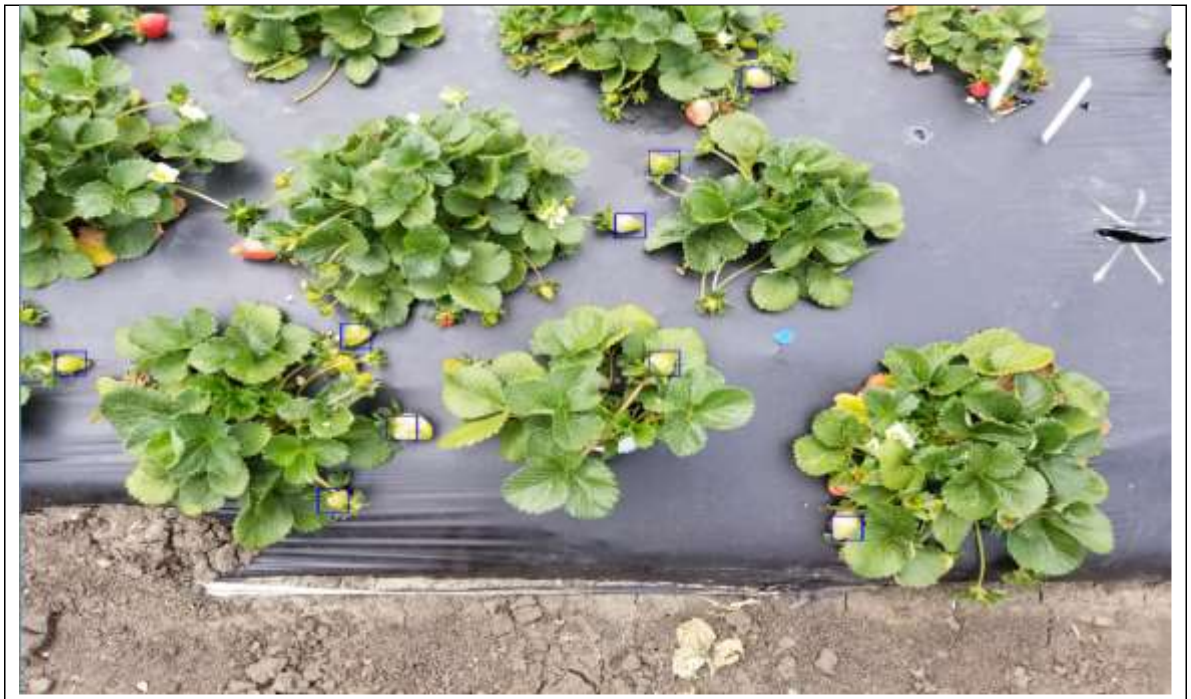
This final processing step gets rid of even more misclassifications. If the object falls within the specified range for both averages, then that window coordinates can be saved, and it is officially classified as a strawberry. Before boxes can be drawn given those saved coordinates on the original image, non-maximum suppression must be applied onto those coordinates as a single detected object can have multiple boxes drawn over it due to the object being detected multiple times as the sliding window slides over it in small step sizes.

```
#applying non-maximum suppression on initial detections
detections = NonMaxSuppression(detections, 0.01)
```

Figure 18a displays the detection result before applying non-maximum suppression. The detected coordinates are usually passed along with a box overlap threshold parameter into the non-maximum function. The overlap threshold value represents the ratio at which every overlapping bounding box should be removed for that region. The post image can be seen in Figure 28b.



a) Multiple bounding boxes around few strawberries



b) Only a single box around each detected strawberry

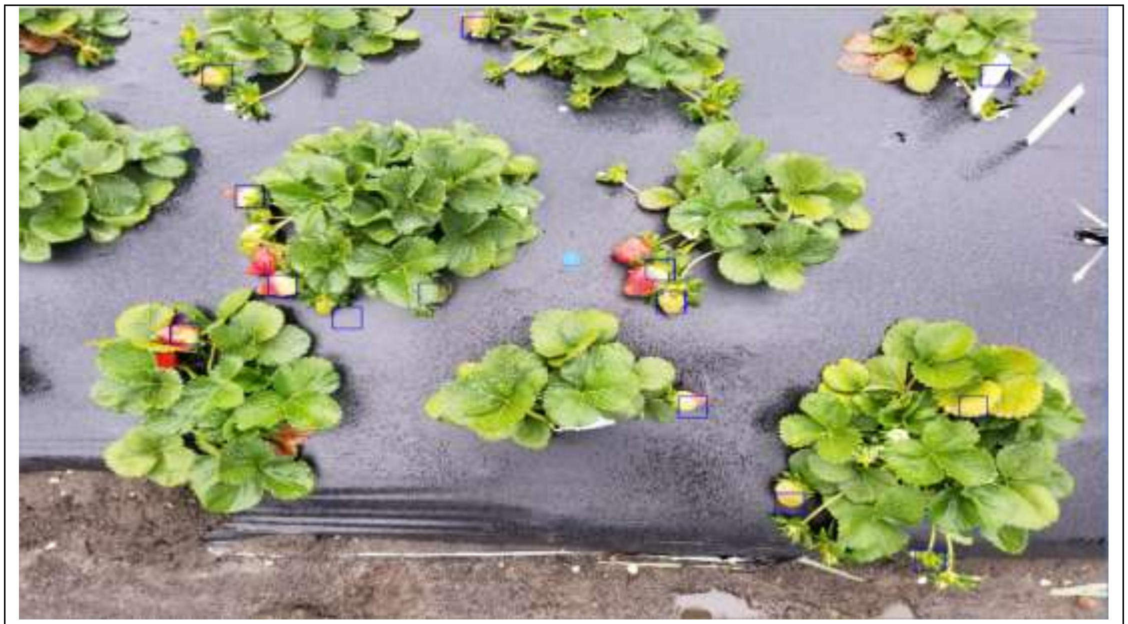
Figure 28: Detection Before and After Applying Non-Maximum Suppression

4.3.1.3 Results

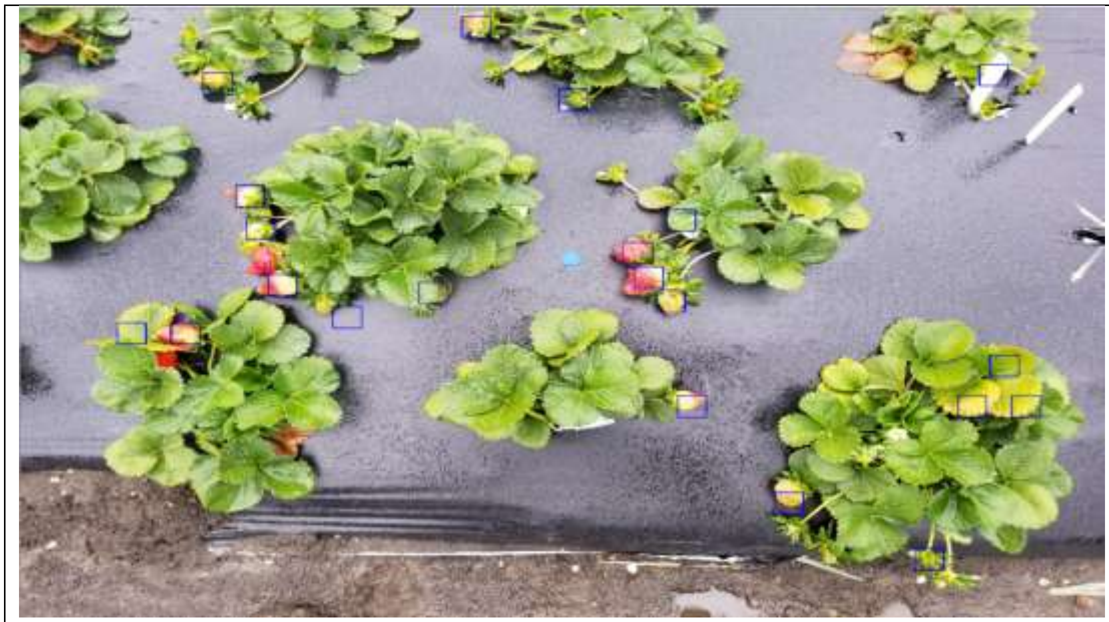
Table 3: Unripe Strawberry Detection Accuracy Under Dimension Size of (4000,5000)

Test data	True Positives	False Positives	Total berries	Accuracy (%)
1-21	73	37	95	77
22-43	75	103	100	75
44-65	100	73	124	81
66-87	45	89	72	62
88-109	95	120	113	84
110-131	79	43	97	81
132-148	84	90	140	60

First the experiments with successful results will be analyzed thereafter, the failures will be described. A detection criterion was set initially to validate the accuracy metric. The criteria simply stated, extremely small strawberries just past their flower stage should not be counted as true positives if not detected as those immature strawberries will grow into larger unripe strawberries eventually. Under the LBP implementation, the results were adequate especially given the limited data set. Approximately, 74.3% unripe strawberries were detected under a dataset of 928 strawberries to 1259 non-strawberry images. Table 3 displays the accuracy values for test data in sets of 21 excluding the last set which included the remaining samples. As stated earlier, the two primary parameters for the LBP descriptor are the number of points and radius to describe the texture window size. After multiple experimentations 64 points with a radius of 32 and an SVM marginal value of 5 were determined to be the optimal parameter values. Setting the margin (C) to 5 compared to higher values such as 100 tightens the margin between both classes and thus preventing the respective data points from overlapping between the two class. However, if C was lowered too much then true positives would get eliminated as well therefore lowering the algorithms accuracy. This is not worth the loss in this application as it would be beneficial to detect many unripe strawberries at the expense of a few false positive leaves detected in the background, in compared to less leaves being detected but less strawberries being detected as well. The true positive comparison between SVM parameters C=5 and C=3 can be seen in Figure 29.



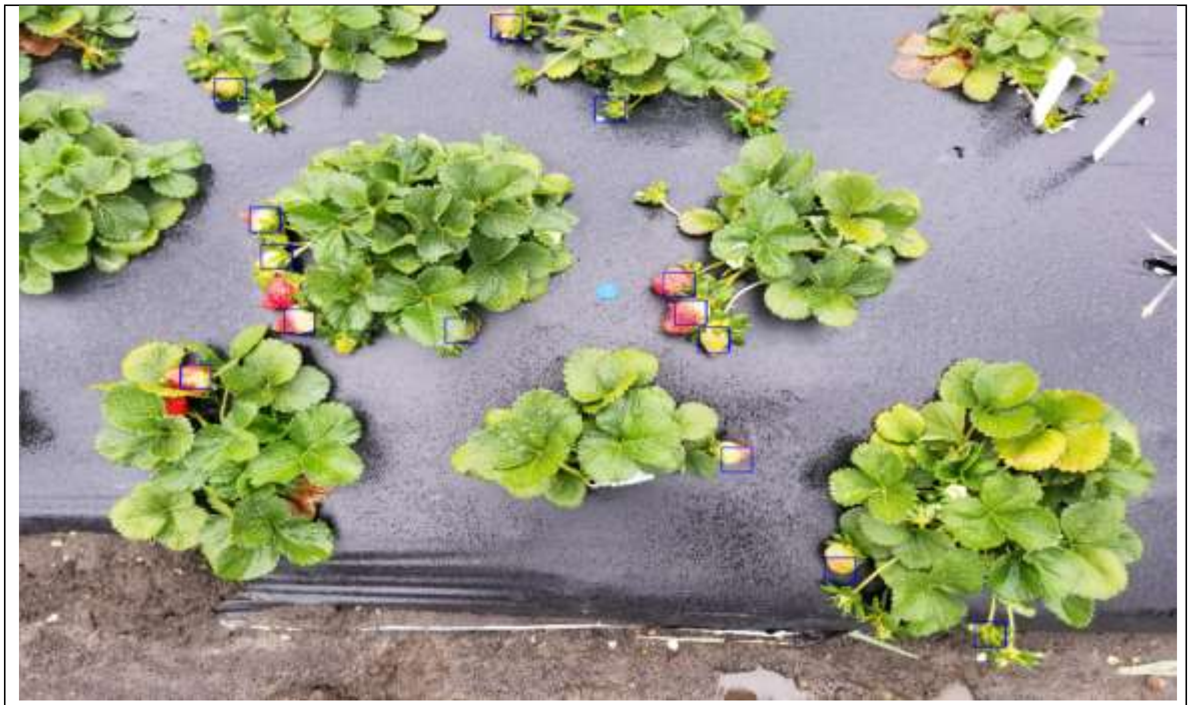
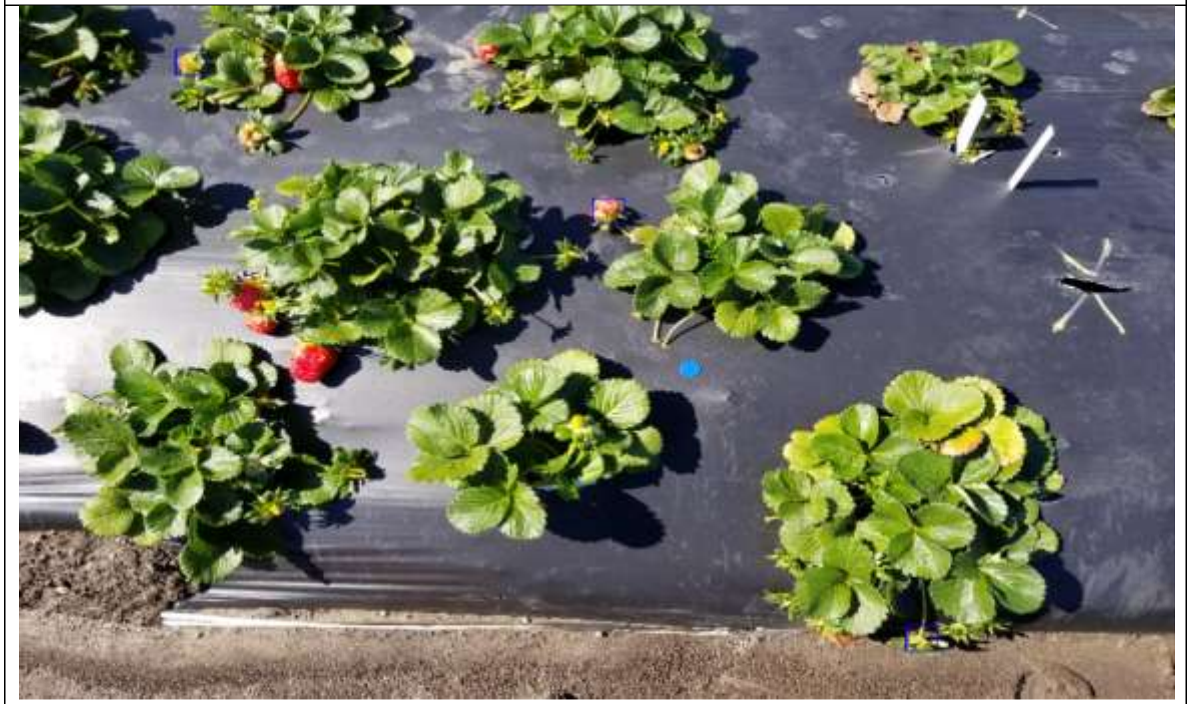
a) Results with C=3

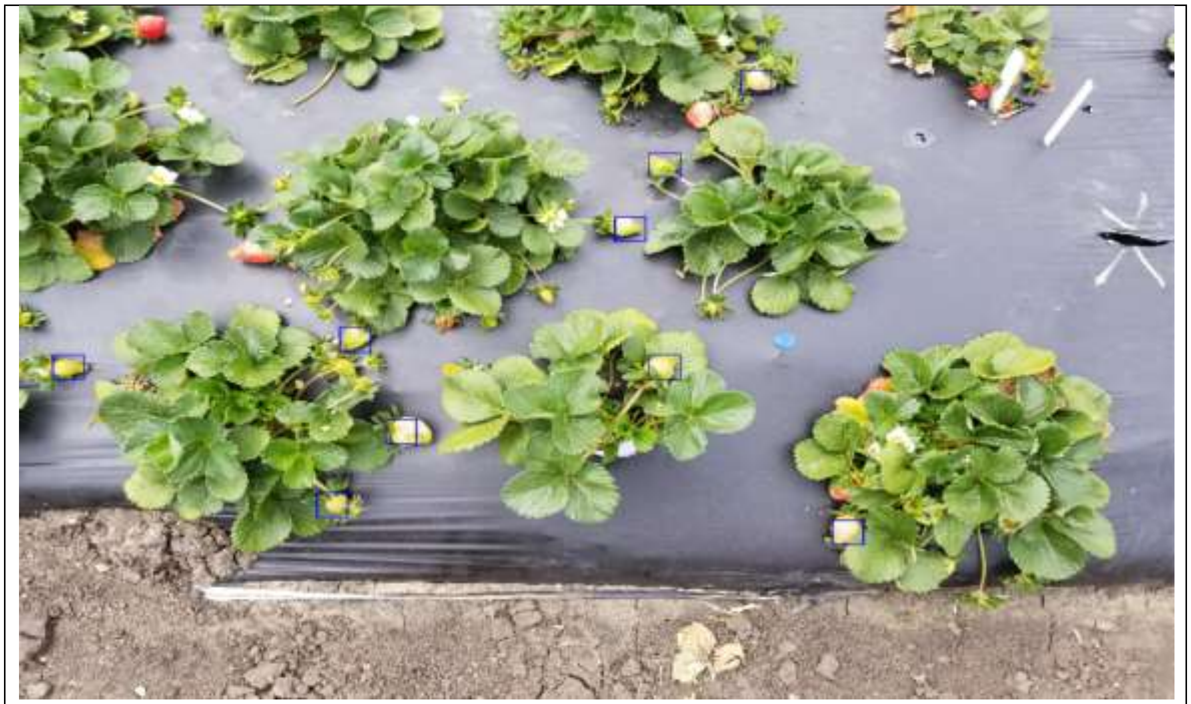


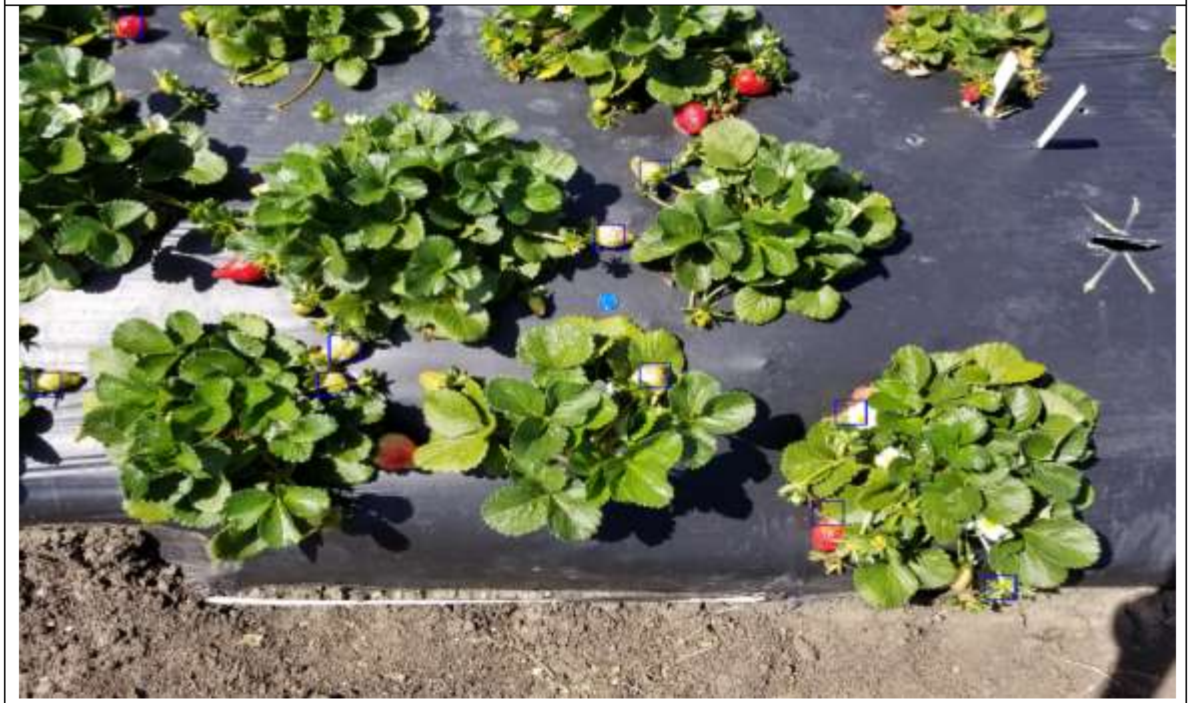
b) Results with C=5

Figure 29: SVM Marginal Boundary Comparison on the Same Test Data

While the results predicted by the LBP texture analyzer under the C=5 parameters were reasonable, there were still some obvious false positives that could be eliminated with some additional post processing by dealing with the pixels through various statistical properties under different color channels as explained above in the implementation section. Despite, all the additional pre and post processing applied, few false positives remained as their features were deemed similar in respect to the strawberries. As inferred from Table 3, each image had an average of 4 false positives. The test images during the early stages of the strawberry growth cycle had much fewer false positives compared to the images much further along the growth cycle where the images were much more complex as the plants had completely grown all over the bed and hence covering few of the strawberries as well. Another key factor to recognize is the images during the latter stages were also taken at a higher viewpoint to cover all 5 beds within a single image. This can hurt the accuracy of the detection algorithm as the resolution of the strawberries are lower. Figure 30 displays the optimal results under these circumstances after all the fine tuning and feature boosting.





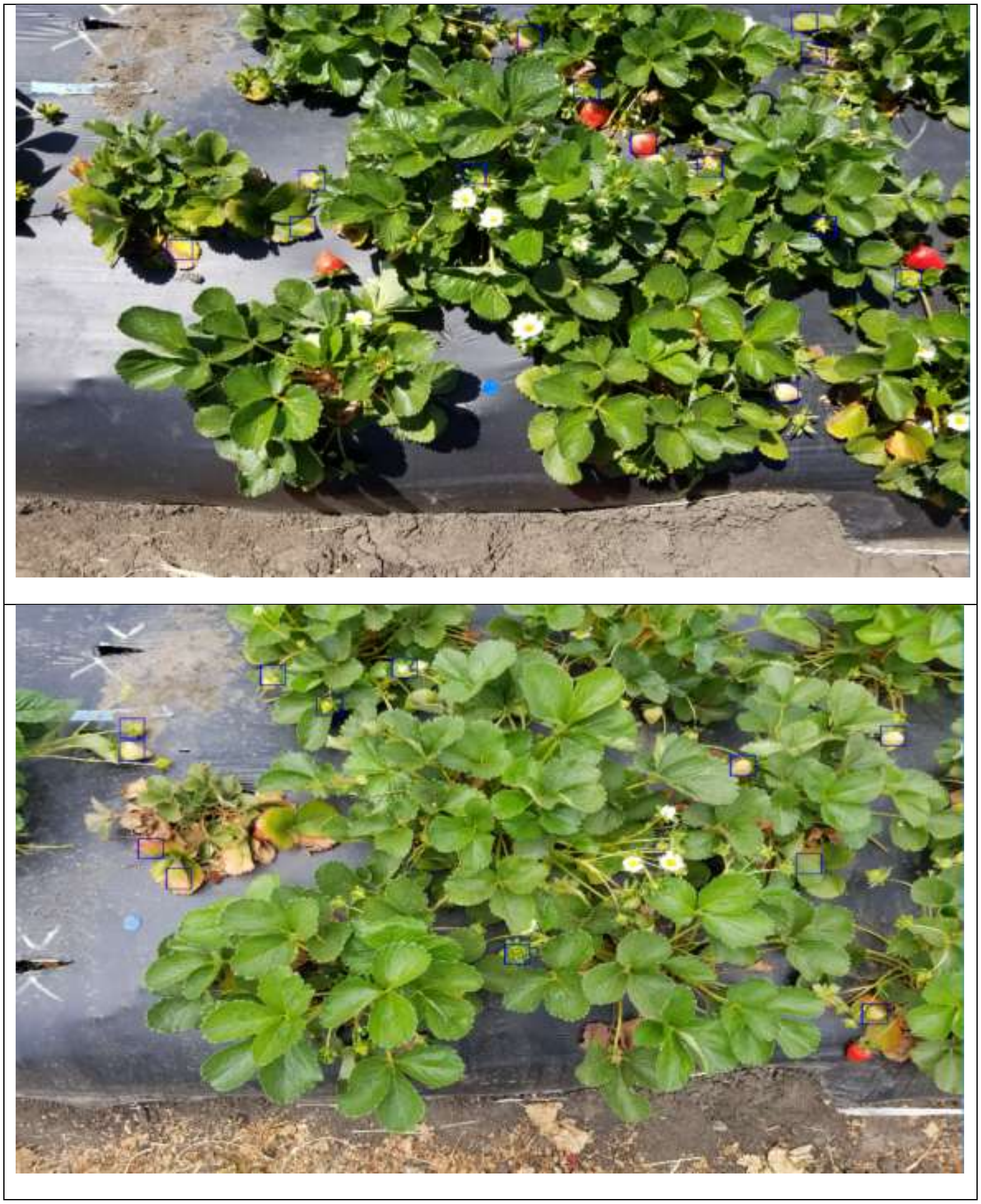


Figure 30: Optimal Results After Final Post-Processing

The LBP technique did not perform as well under certain circumstances. 64 points along with a radius of 24 and a C parameter of 100 was selected initially. This resulted in many more false positives compared to (64,32) as this texture window couldn't accurately distinguish the strawberries from the background object even in the earlier stages under less complex environments. Figure 31 displays this prediction failure.

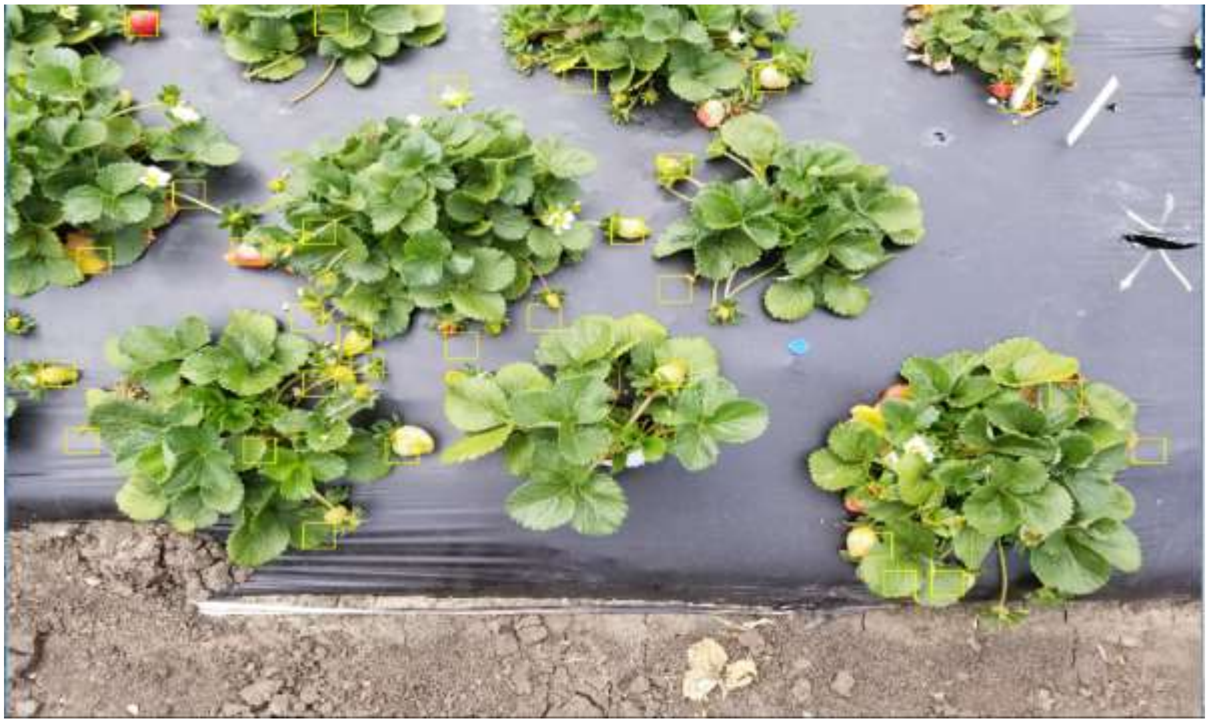


Figure 31: LBP Analysis with Window Size of (64,24), C=100 trained on 924 Strawberries Images

Due to the limited number of original strawberry images in the dataset, the training image set was increased by generating artificial data. Data augmentation was applied to the strawberry training data set in terms of rotating various image at an angled step size of 30 degrees for 360 degrees, therefore generating multiple additional images for each image. Sample rotations can be seen in Figure 32. This significantly increased the data set to 2252 strawberries, adding over 1000 additional strawberry training images. The supplementary background class also had to be increased to prevent any bias predications from occurring for one class over the other. The background non-strawberry class was increased to 2270.

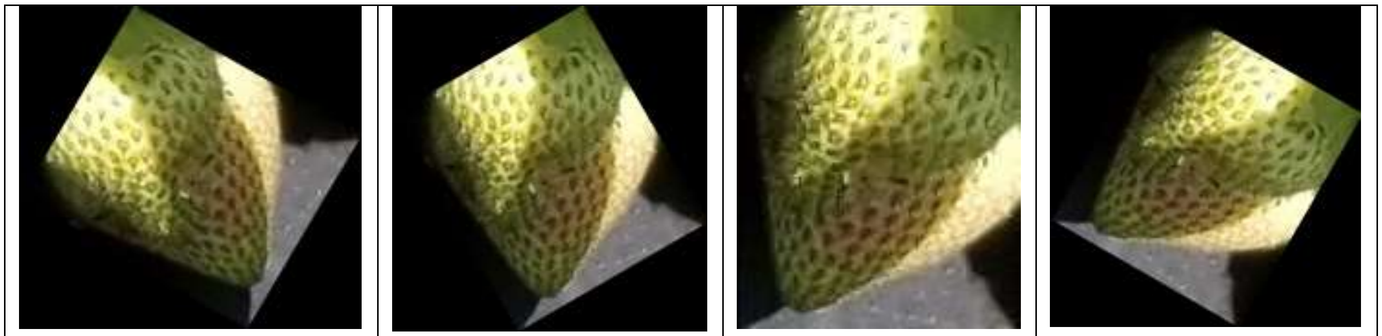
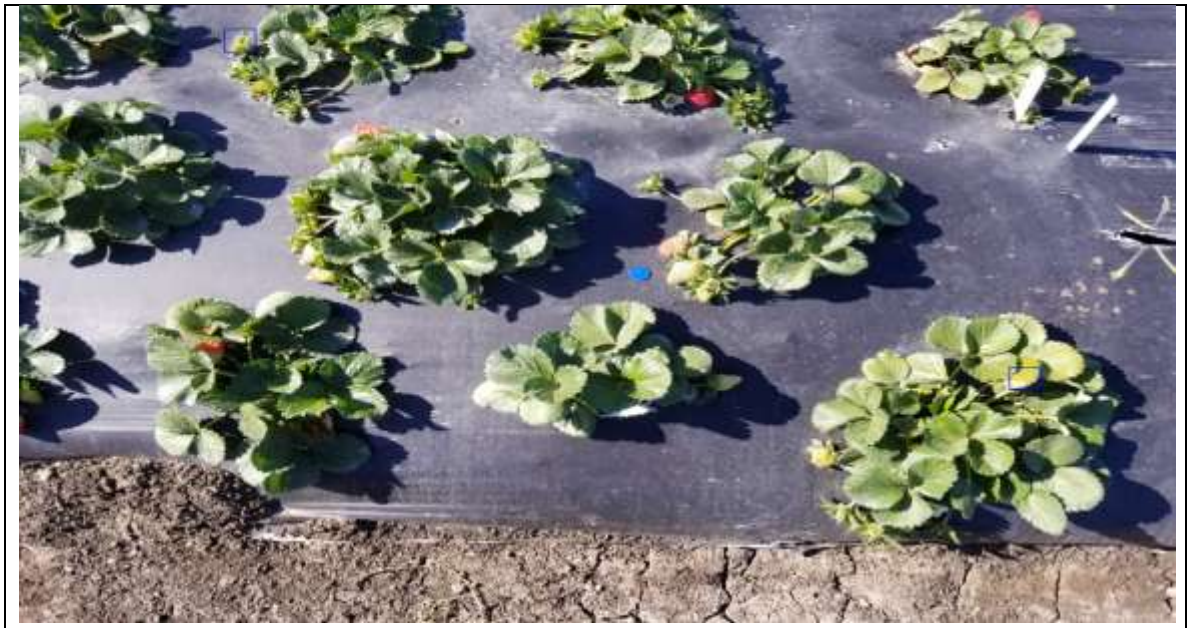


Figure 32: Sample training images rotated

This resulted in having a negative effect on the testing data by dramatically decreasing the total number of true positives. There are two primary reasons as to why data augmentation did not work; first the rotated images generated created a basic black background where the original image was placed which disrupted the texture analysis. Second, rotating the data is not very useful because in this situation as rotational-invariant LBP is already being employed, therefore rotating the data even further simply causes over-fitting within the data set. The negative effect data augmentation had on the test images can be seen in Figure 33.



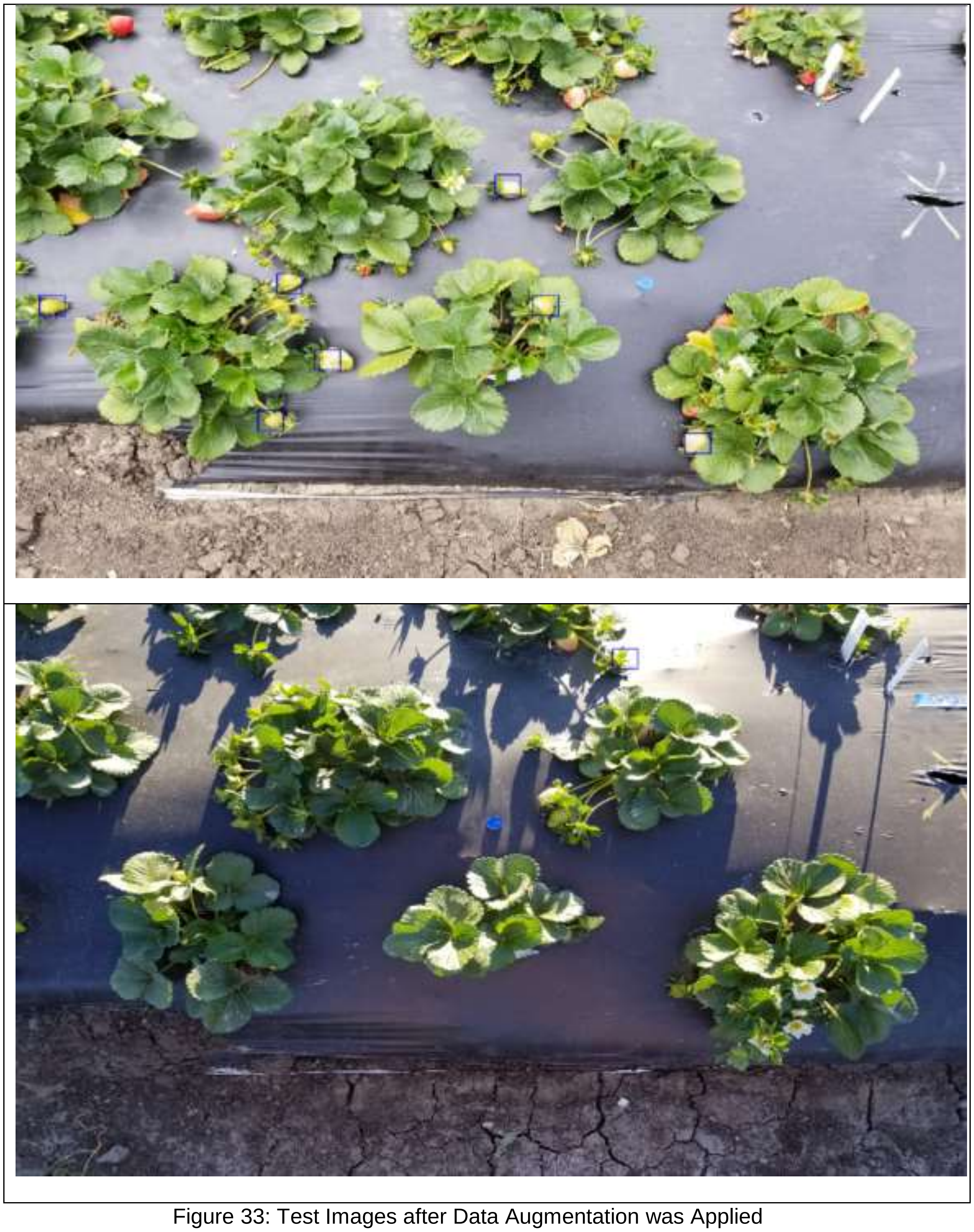
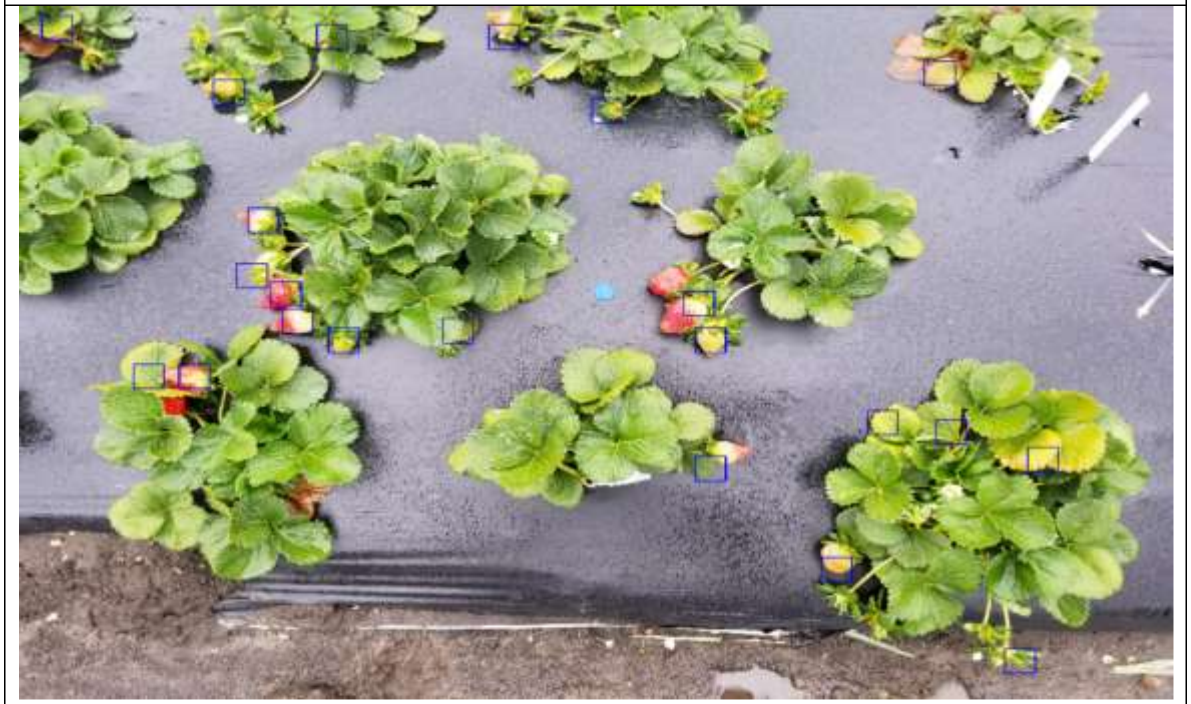
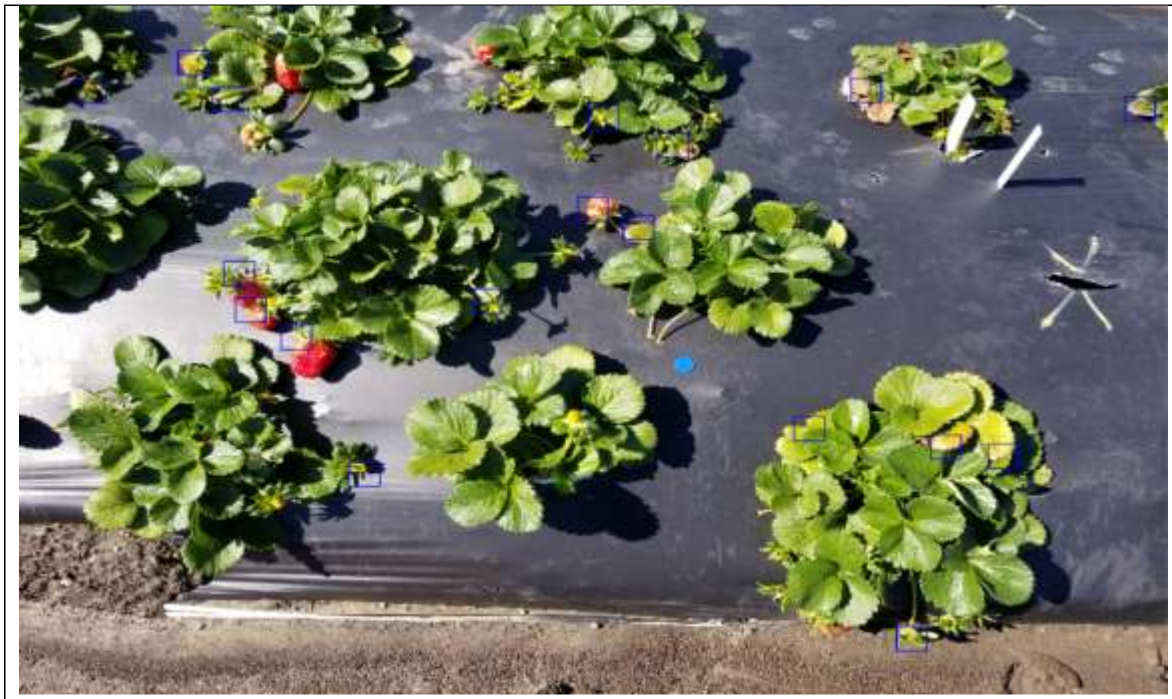
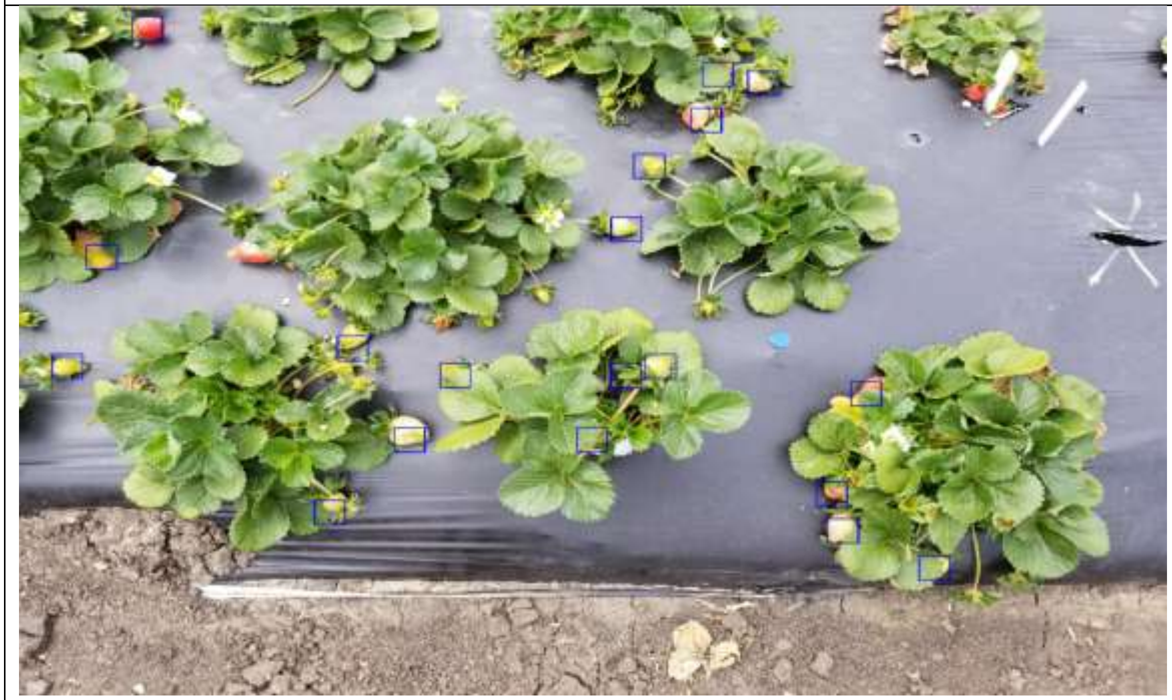


Figure 33: Test Images after Data Augmentation was Applied

4.3.2 K-NN Classifier

K-nearest neighbor was chosen as another model chosen to classify the LBP features. The K-NN classifier replaced the SVM classifier in the algorithm to determine if the true positives would be maximized or decreasing the false positives. After multiple experimentations, k=7 produced the best results for this classifier but as seen in Figure 34, K-NN did not out-perform the SVM classifier. This entails that in the multi-dimensional feature space many of the background features were within proximity to the strawberry features, resulting in more false positives than SVM. Thus, distance is not a good metric for separating features.



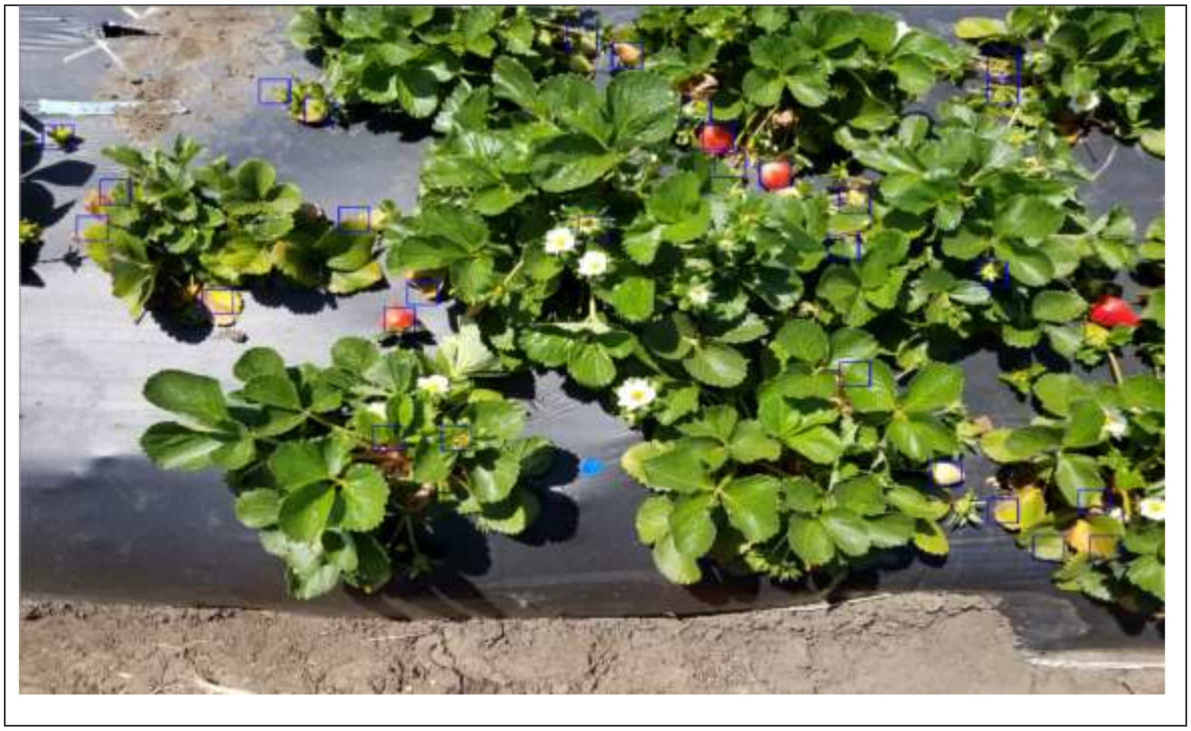


Figure 34: Sample K-NN Results Where K=7

4.4 Flower Detection

4.4.1 Local Binary Patterns

Flowers were also detected as part of the strawberry detection system. Flowers within the image represents the earliest stage of the strawberry growth cycle. Before the strawberry grows into a fruit, it begins as a plant with white petals and a yellow core as shown in Figure 35.

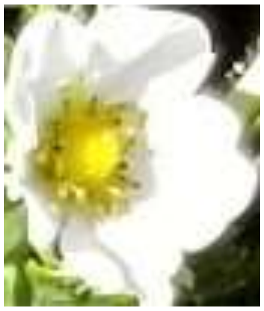


Figure 35: Flower Stage

4.4.2 Training & Testing Procedure

The major difference between the unripe strawberry and flower detection algorithms are four-fold:

First, the training data did not consist of binary classes as seen before in the unripe strawberry algorithm. The flower training dataset consisted of three classes; “flower”, “non-flower” and “strawberry”. Three classes were used over two as employed by the unripe strawberry algorithm as this prevented any strawberries from being classified as flowers, therefore improving the accuracy. The classifier employed was still the SVM classifier. Even though the SVM classifier functions as a binary classifier, it can be modified to employ a “one vs all” strategy in terms of classes. When distinguishing for flowers, it treats the other classes as a single opposing class.

Second, the LBP parameters and window sizing had to be adjusted to learn the flower patterns accurately. The window size used on the test data was (95,95). The same LBP class format was employed from the unripe strawberry detection algorithm to distinguish the flowers. After multiple experimentations, 10 points with a radius of 4 was concluded as the optimal points to extract the flower’s texture efficiently.

```
#parameters being passed into the LBP class
desc = LocalBinaryPatterns(10,4)
```

Third, the post processing function was simplified to only isolate the flower taking advantage of its yellow core.

```
function pre_check(window):
hsv = RGB2HSV(window)
#isolate yellow pixels
lower = array([23,200,100])
upper = array([30,255,255])
mask = Range(hsv, lower, upper)
```

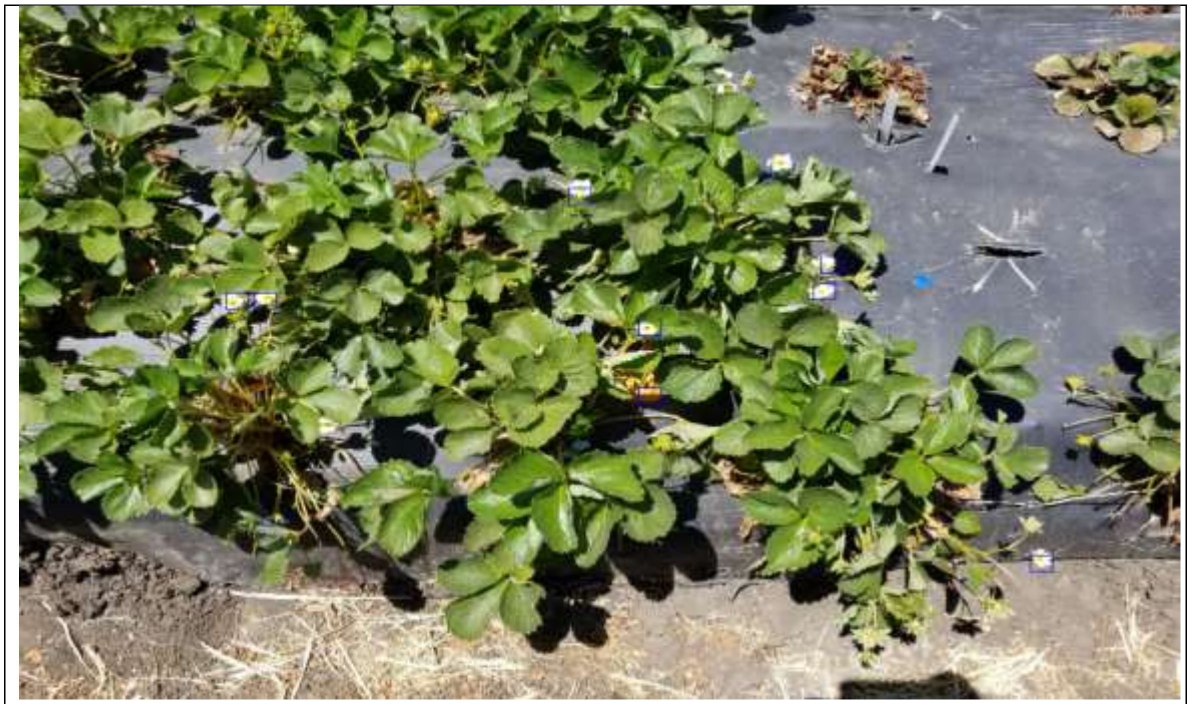
Finally, image pyramids were introduced, to maximize the detection accuracy. Unlike normal convention of pyramids which involve down sampling the image, up sampling was applied instead. If the flowers are too small, this will limit the model's ability to extract the information accurately thus increasing the small flower sizes were necessary. The images resolution was increased from its original size of (3042, 4032) to (4000,5000). Apart from those four changes everything else was precisely the same as explained in the strawberry detection algorithm above.

4.4.3 Results

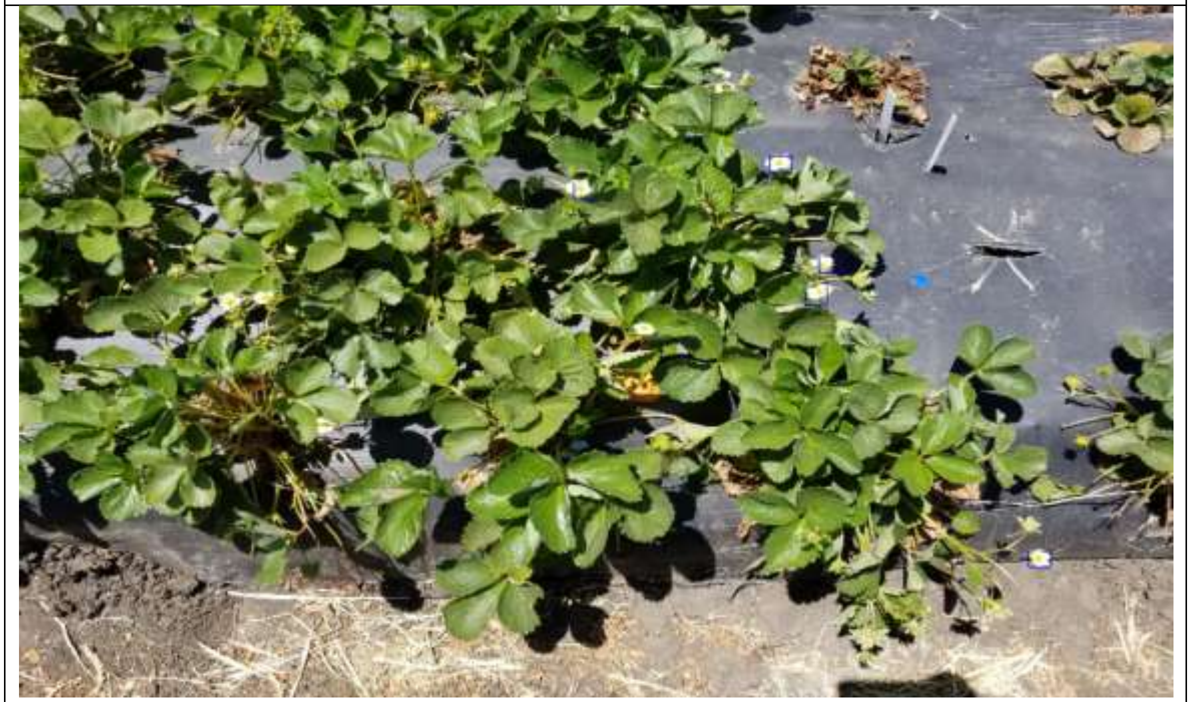
Table 4: Metrics Representing the Performance of Flower Detection Under (4000, 5000) Dimension Size

Test Data	True Positives	False Positives	Total flowers	Accuracy (%)
1-21	40	45	51	78
22-43	115	83	130	88
44-65	89	66	105	84
66-87	58	70	68	85
88-109	98	76	118	83
110-131	68	53	86	79
132-148	84	59	109	77

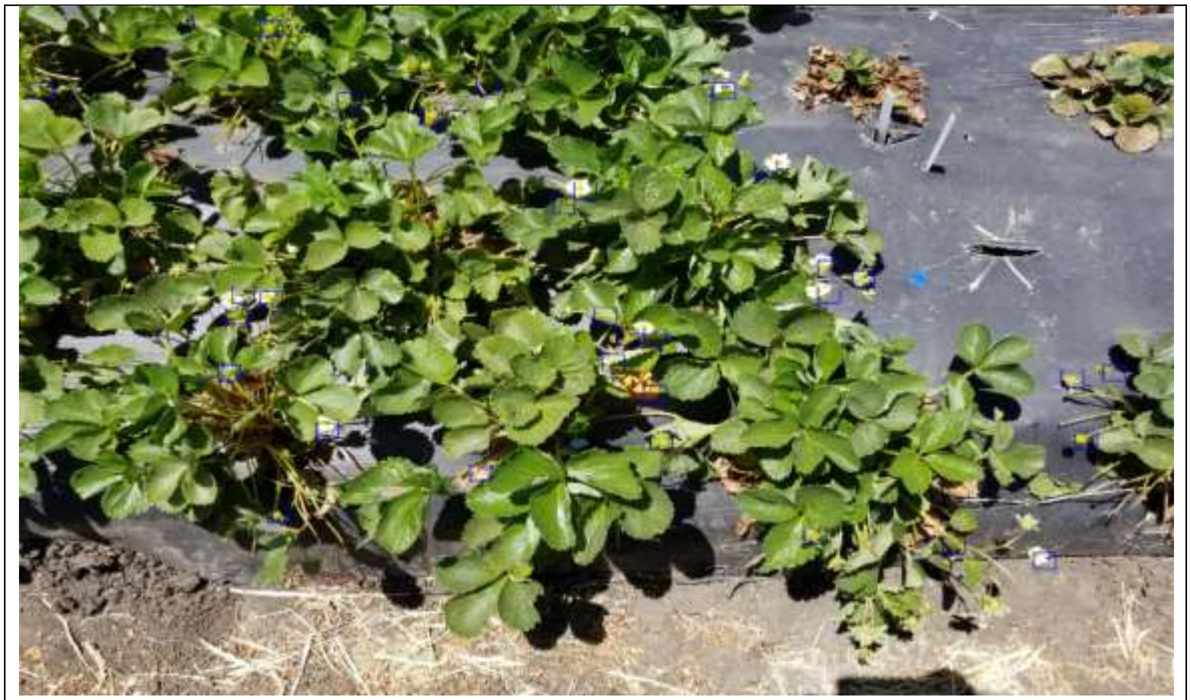
The flower training data was gathered in similar fashion as the strawberry training data. However, naturally there will always be less flowers than strawberries in most images due to strawberries having a much longer growth cycle than flowers. This led to a very limited data set of 788 flowers. No initial criteria were established for their sizes as done earlier for the strawberries due to flowers being the earliest stage within the strawberry cycle. All these complications made the flower detection technique more complicated than the earlier strawberry detection techniques. Multiple experimentations were attempted to determine the best possible parameters at which texture could be extracted from the flower data. 10 points with a radius of 4 proved to be the best LBP values to train and test upon. Figure 36 displays the effect various LBP parameters had.



a) LBP parameters of (10,4)



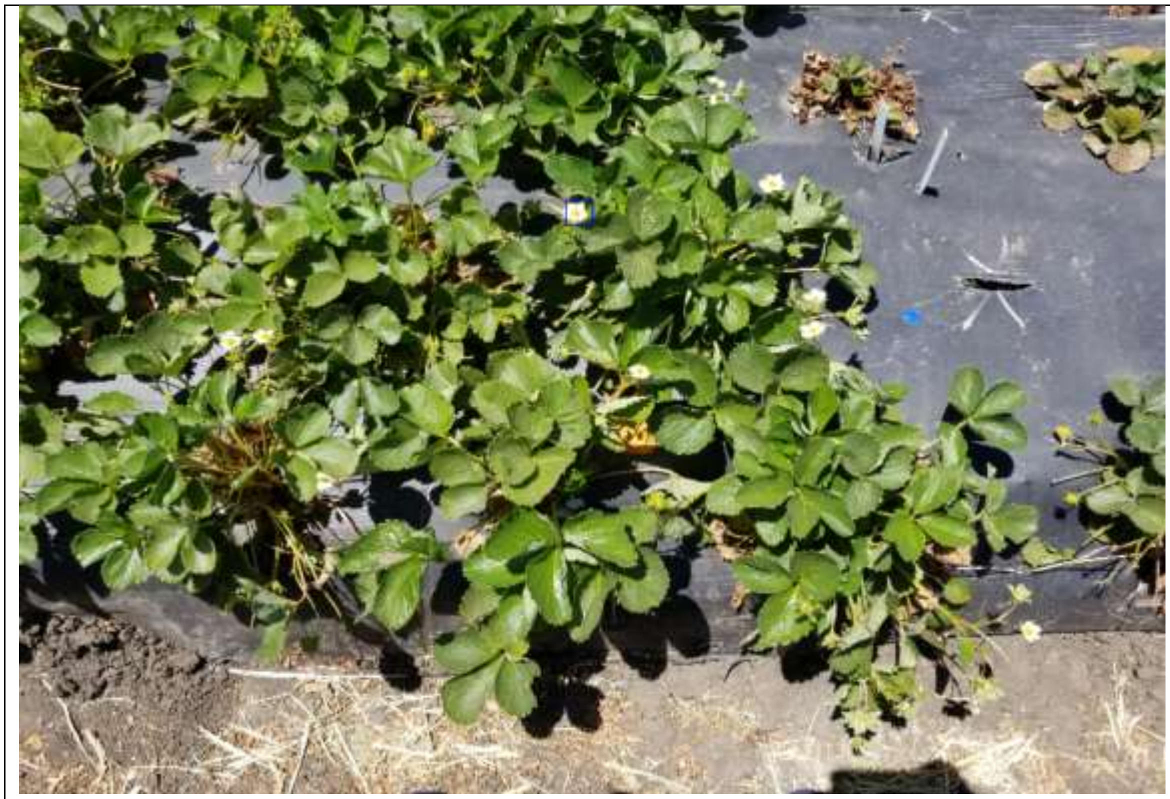
b) LBP parameters of (24,8)



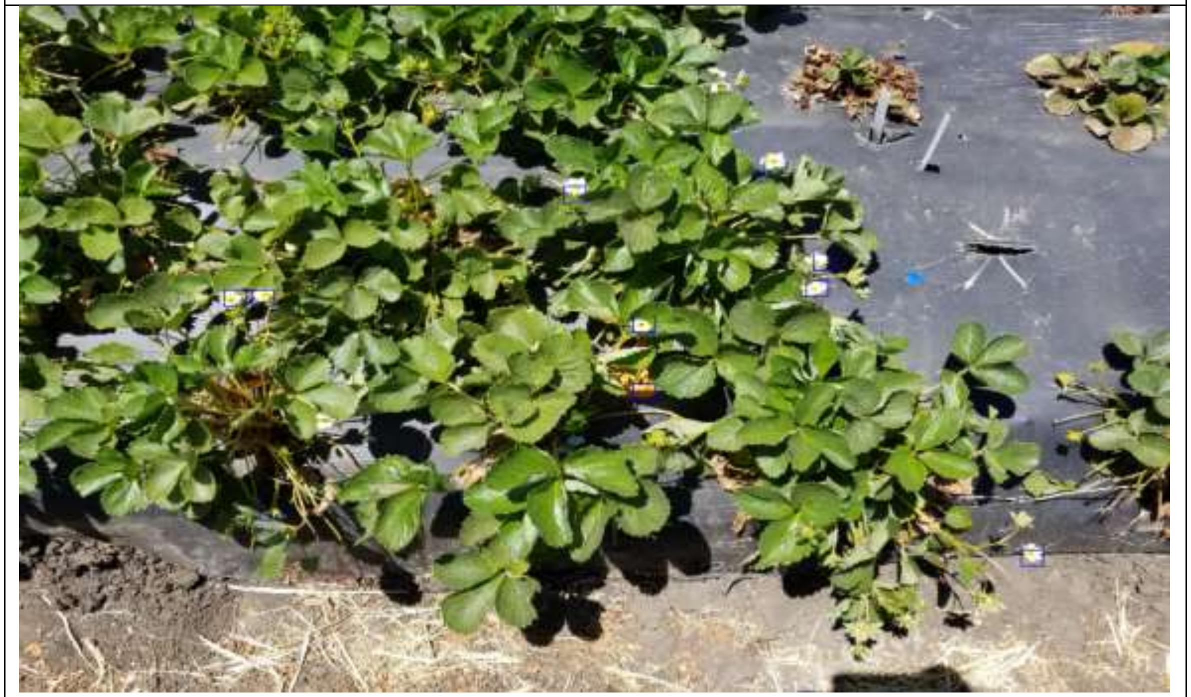
c) LBP parameters of (64,32)

Figure 36: Results Under Various LBP Parameters

Once the appropriate parameters were determined, attempting to detect the flowers under the original image's dimension proved to be insufficient as many of the flowers were too small to extract its texture. Up sampling the image dealt with this issue quite well. After employing up sampling the average accuracy for detecting flowers regardless of size was at 82% as seen in Table 4. Figure 37 displays the detection accuracy for much smaller flowers under the original image resolution vs the up sampled image resolution.



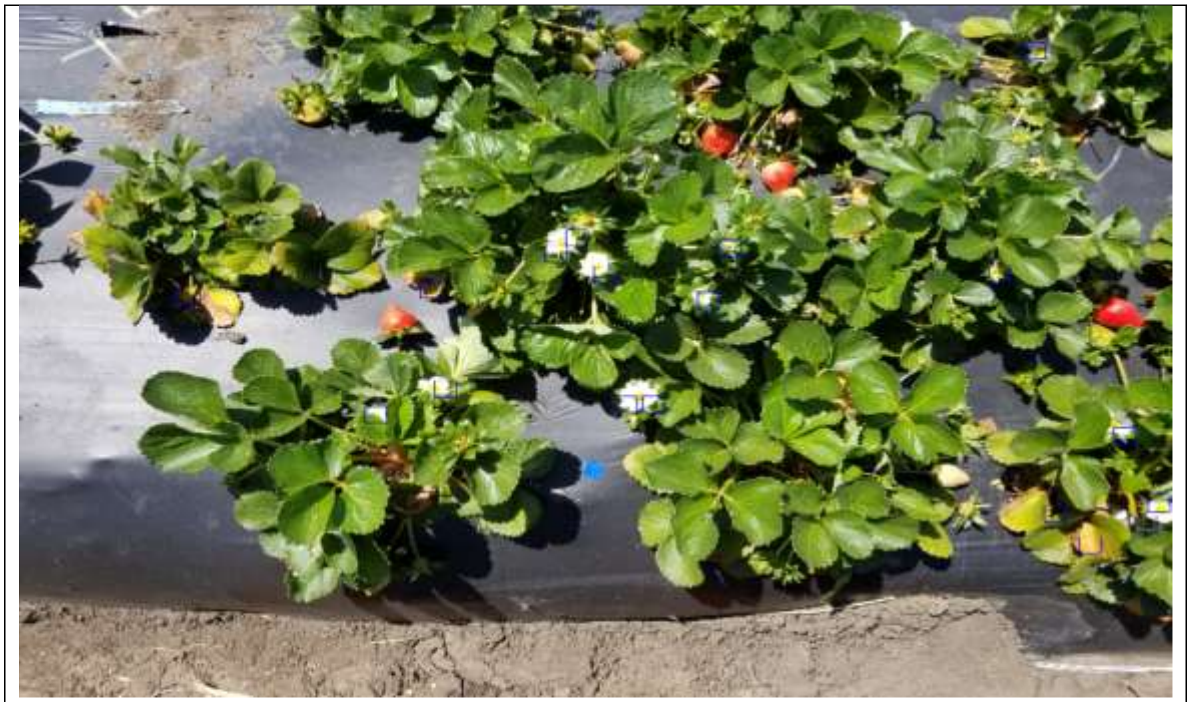
a) Result under image original resolution of (3042, 4032)



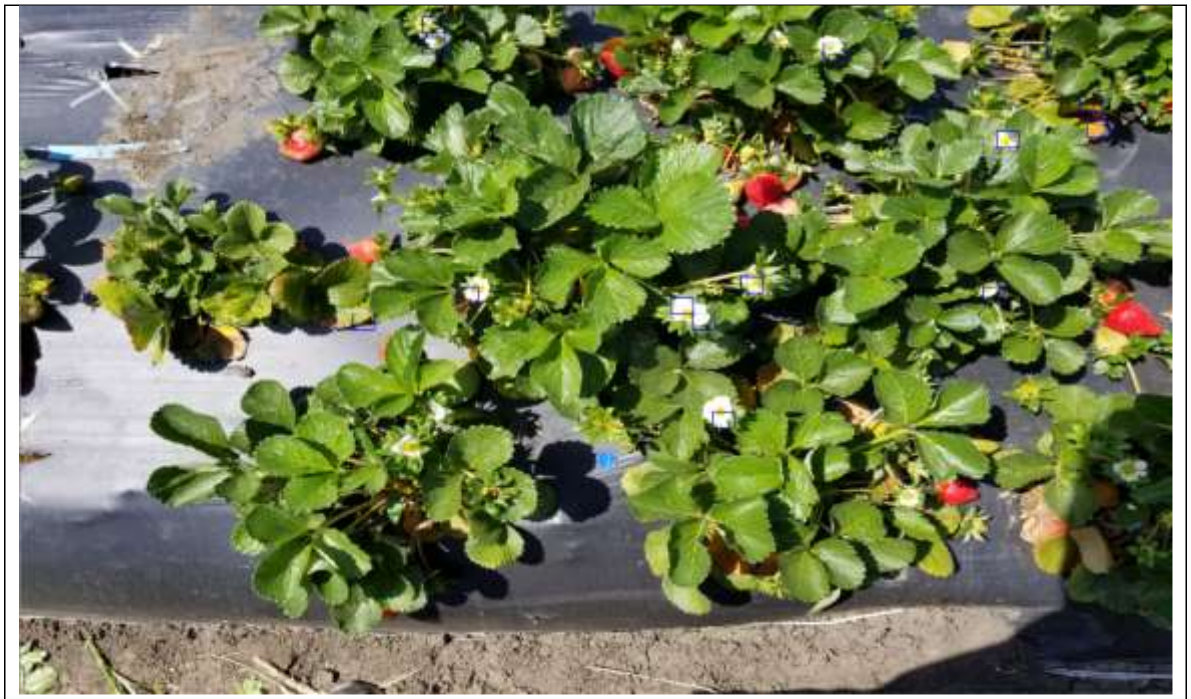
b) Result under up sampled image resolution of (4000,5000)

Figure 37: Comparison of Small Flower Detection Under Original vs Up Sampled Resolution

While the accuracy improvement after up sampling the images were remarkable, it did come with two drawbacks. First, the computation speed was slower as well as the prediction window had more pixels to cover now. Two, due to the window size remaining the same even after the image was up sampled it detected the larger flowers' multiple times thus bounding the large flowers with 2 boxes instead of one. This issue can be seen in Figures 38a and 38b. Given the limited amount of training data texture analysis performed very well. Figure 39 exhibits successful sample test images for flower detection.



a) Lower large flower detected twice



b) Central large flower detected twice

Figure 38: Large Single Flower Bounded with 2 Boxes Under Up Sampled Resolution

Figure 39: Sample Test Images Displaying Successful Flower Detection Under Up Sampled Resolution

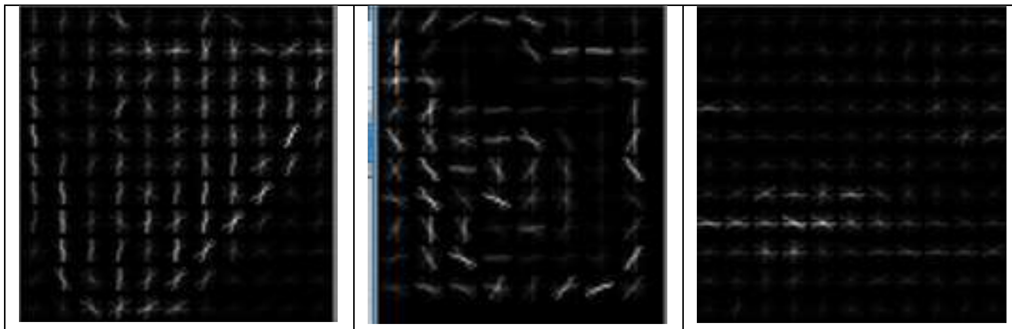
4.5 Histogram of Oriented Gradients

4.5.1 Training & Testing Procedure

The algorithm for the HOG technique followed a very similar architecture as the LBP methodology. The differences occurred in one key area; the image and window dimensions for training and testing must be the same as HOG returns a feature vector and thus both vectors in the classifier must be the same size when being compared. It's not possible to compare feature vectors of different lengths.

```
#image passed into the HOG descriptor function
hog_features = hog(im, Orientations=9, PixelsPerCell=(n,n), CellsPerBlock=(m,m))
```

After that each image extracted from the training folder would need to be converted into a single channel (grayscale) due to HOG only working on 2D images as well. Sample HOG features from the training images are shown in Figure 40.



a. Strawberry   b. Flower   c. Not

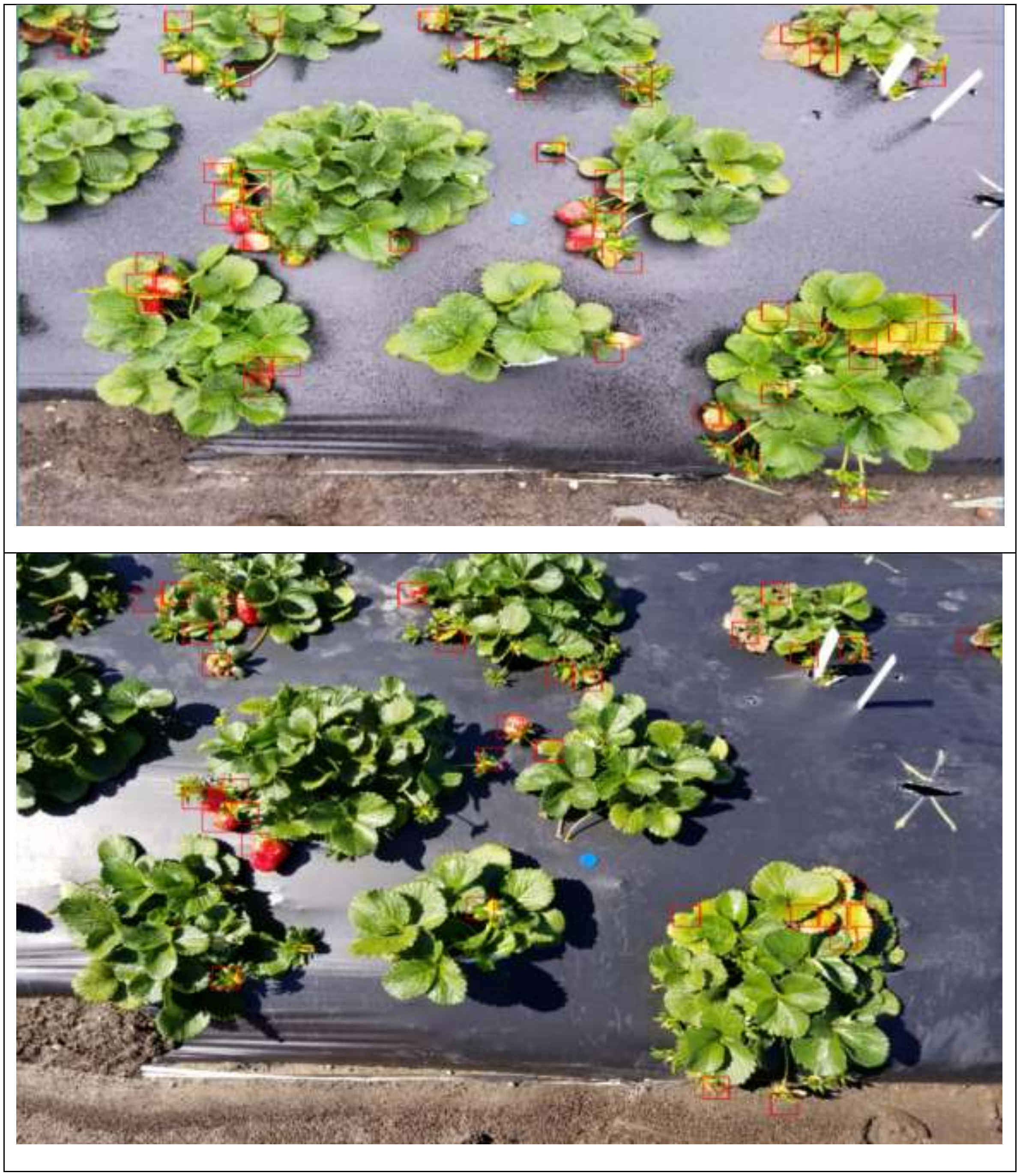
Figure 40: HOG Gradient Extraction on Training Data

The function has multiple parameters, however the primary ones to take note off are orientations, ‘PixelsPerCell’ and ‘CellsPerBlock’. ‘Orientations’ are the number of the bins in the histogram to insert the magnitude values, ‘PixelsPerCell’ states the pixel dimension required for each cell as HOG will be performed within those cells instead of the entire image at once. The ‘CellPerBlock’ parameter represents how many cells should be covered by this additional overlaying window. This window is where the normalization step occurs.

The remainder of the steps taken for training and testing were very similar to the LBP procedure for training and testing.

4.5.2 Results

The HOG feature descriptor function has multiple parameters associated within it and thus various combination and changes were applied to them to witness if any significant changes occurred in its accuracy. Despite all the experimentation this approach ended up performing poorly as it resulted in high false positives. Many false positives were eliminated through post processing however, a large portion of the misclassifications remained and that is not acceptable as there should be a limit. Many of the HOG’s false positives consistently fell upon the yellow stems unlike LBP’s detections and those are difficult to eliminate even with post processing due to color similarities. HOG would simply classify a large variety of objects within the image as strawberries or flowers respectively making it not a robust technique. Additionally, HOG did not end up performing well because strawberries and flowers do not have a definitive shape as they all morph into unique sizes and forms. Thus, the descriptor was unable to determine a concrete shape to describe either and HOG heavily relies on those features. The unsuccessful HOG results can be seen in Figure 41.



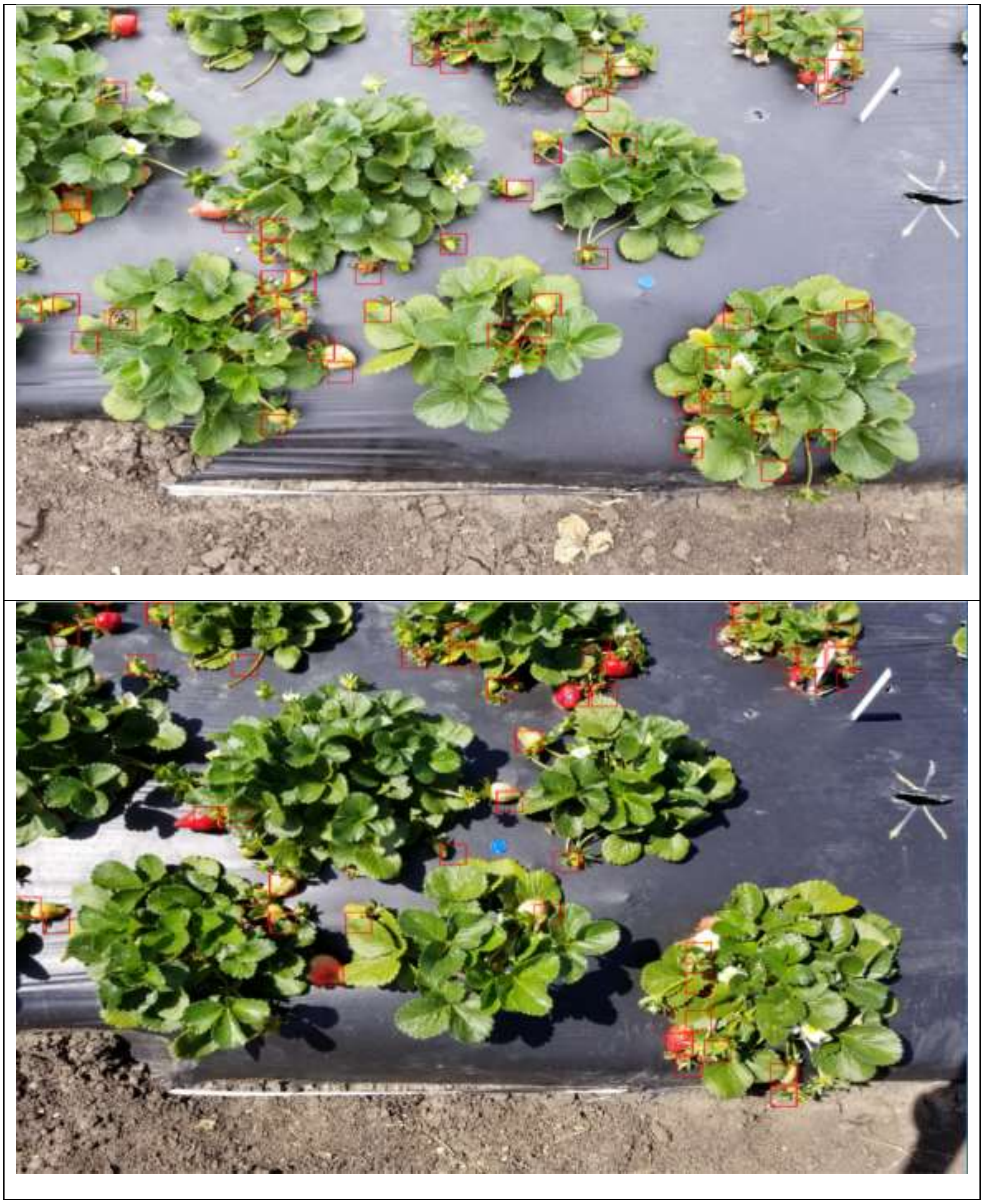


Figure 41: Sample Results of Applying HOG Descriptor

4.6 CNN based Strawberry Detection

CNN’s require a relatively large amount of data to build a robust feature extractor and due to only a limited number of strawberries being available, both unripe and ripe strawberry images were combined together to maximize the dataset. Hence, detecting strawberries in general were the primary goal of the CNN based feature extractor.

4.6.1 Training Procedure

The same training data set employed from the LBP and HOG techniques, 928 “strawberry” and 1024 “non-strawberry” images. The data was trained on a custom CNN model which was derived by the VGG Net architecture [21]. The VGG network was chosen as inspiration due its simple yet effective design. VGGNet’s are known for their ability to generalize to datasets since the architecture follows two primary patterns:

1. Stacking multiple sets of 3 convolutional layers on top of each other followed by a max pooling layer
2. The number of filters double for each new convolutional set

Stacking smaller filters on top of each other over having one large filter enables the network to better learn complex features as it increases the depth of network. The number of feature parameters increase deeper into the network and thus by doubling the number of filters, the layers ahead can appropriately encode the previous layer features past into it. The specifics on the VGG network architecture can be found in its original paper [21]. The actual network was too large (19 layers) for the size of the data set on hand and hence a smaller version was designed. If only a small proportion of training samples are provided into a large network, then significant overfitting can occur. First the images had to be resized into a constant dimension as CNN’s require all its data follow the same dimensional format. Next, the data labels are converted to a one-hot encoded format. This helps keep the labels binary and increases efficiency within the network.

```
#one-hot encoding
training_labels = one_hot(training_labels)
```

Thereafter, the network is constructed. The VGG inspired architecture consists of two blocks; block one covers the convolutional and pooling layers while the second includes the final fully connected layers. The Keras framework was employed to create the neural network architecture to train the data upon.

```
#BLOCK-1
Conv2D(filters=32, kernel=(3,3), input=(w,h))
Activation('Relu')
Conv2D(filters=32, kernel=(3,3))
Activation('Relu')
MaxPooling(size=(2,2))
Dropout(0.25)
Conv2D(filters=64, kernel=(3,3), input=(w,h))
Activation('Relu')
Conv2D(filters=64, kernel=(3,3))
Activation('Relu')
MaxPooling(size=(2,2))
Dropout(0.25)
#BLOCK-2
Flatten(previous layer)
FullyConnected(channels=512)
Activation('Relu')
Dropout(0.50)
FullyConnected(outputs=num_classes)
Activation('Sigmoid Function')
```

Due to the limited number of training data in the set, data augmentation is applied to artificially increase the training data. Hence, the parameters to augment the data were defined, which would eventually be passed into the network once the model was defined.

```
#Data Augmentation set up
```

data = DataGenerator(rotation_range, width_shift, height_shift, zoom_range, horizontal_flip)

Sample augmentations generated for a single training data is represented in Figure 42.

Note that these augmentations are applied on every training sample and hence this technique can significantly increase the dataset.

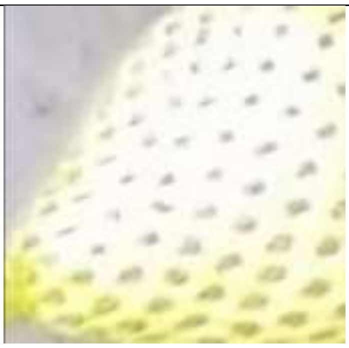
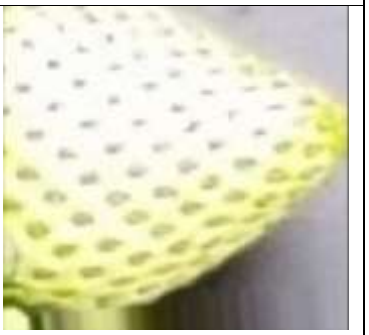
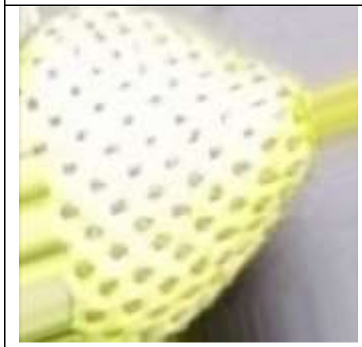
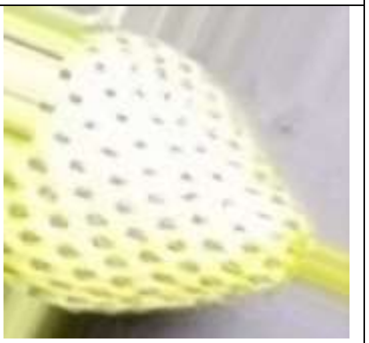
a) original	 b) augment 1
c) augment 2	 d) augment 3
 e) augment 4	 f) augment 5

Figure 42: Various Augmentations of a Single Data Point

Finally, the model is compiled using “binary crossentropy” as only two classes are involved in this training set. Thus, sigmoid is employed as the activation function as it treats both classes as an independent probability distribution.

4.6.2 Testing Procedure

The sliding window method previously mentioned earlier sections was utilized again to localize the strawberries within the test image. The major difference for the neural network approach however was sliding windows retrieved had to be stored as batches before being passed into the network for prediction.

```
#sliding over testing image and passing window batches for prediction
for (x, y, window) in sliding_window(test_image, step_size, window_size)
    predictions = model_prediction(window_region)
  for p in predictions
            if p == ‘strawberry’:
                box coordinates.append(x,y)
          return box coordinates
```

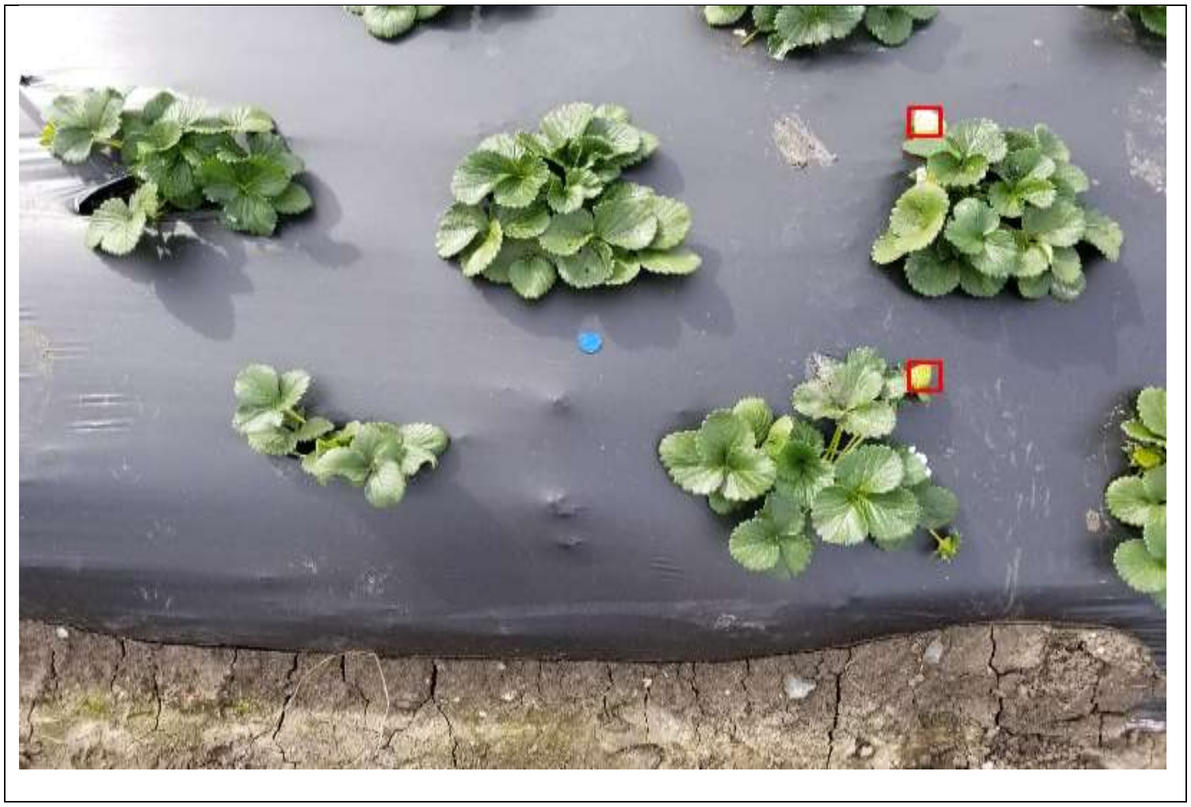
Thereafter, the strawberry detected coordinates could then be applied onto the test image and non-maximum suppression can be applied if needed.

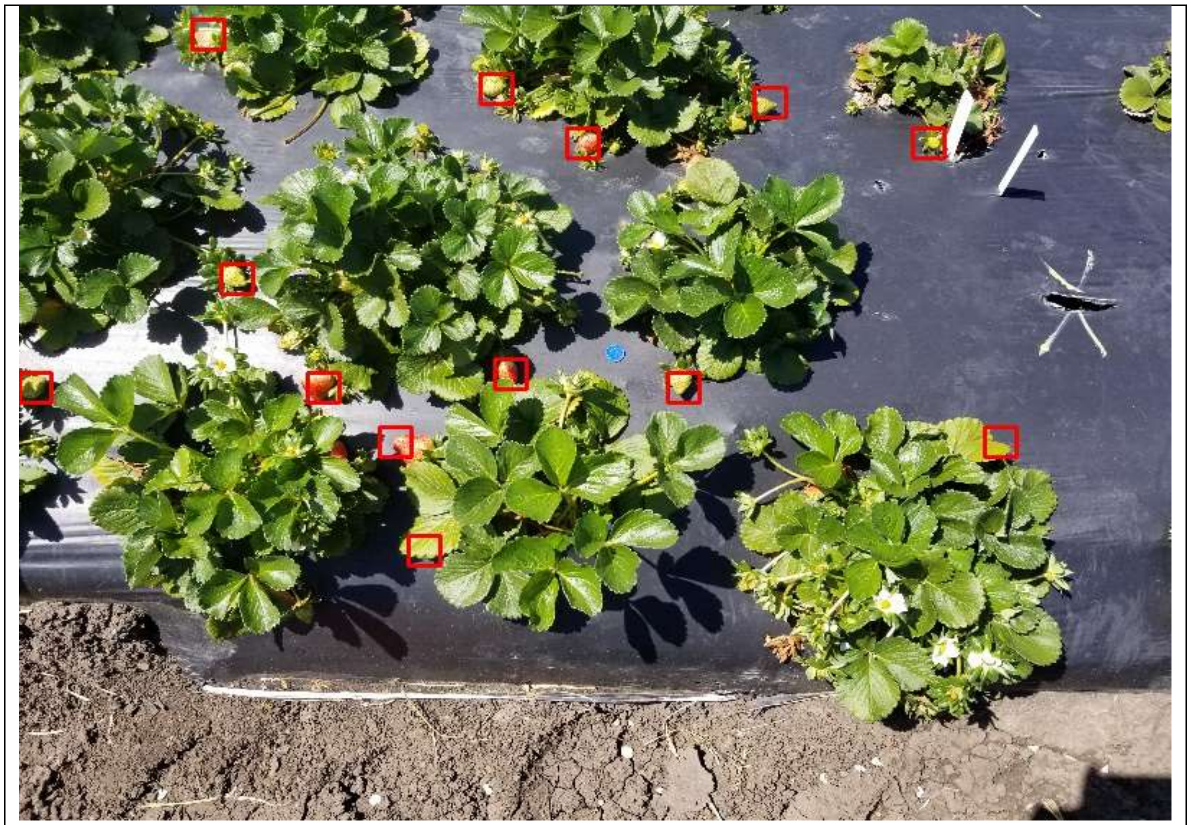
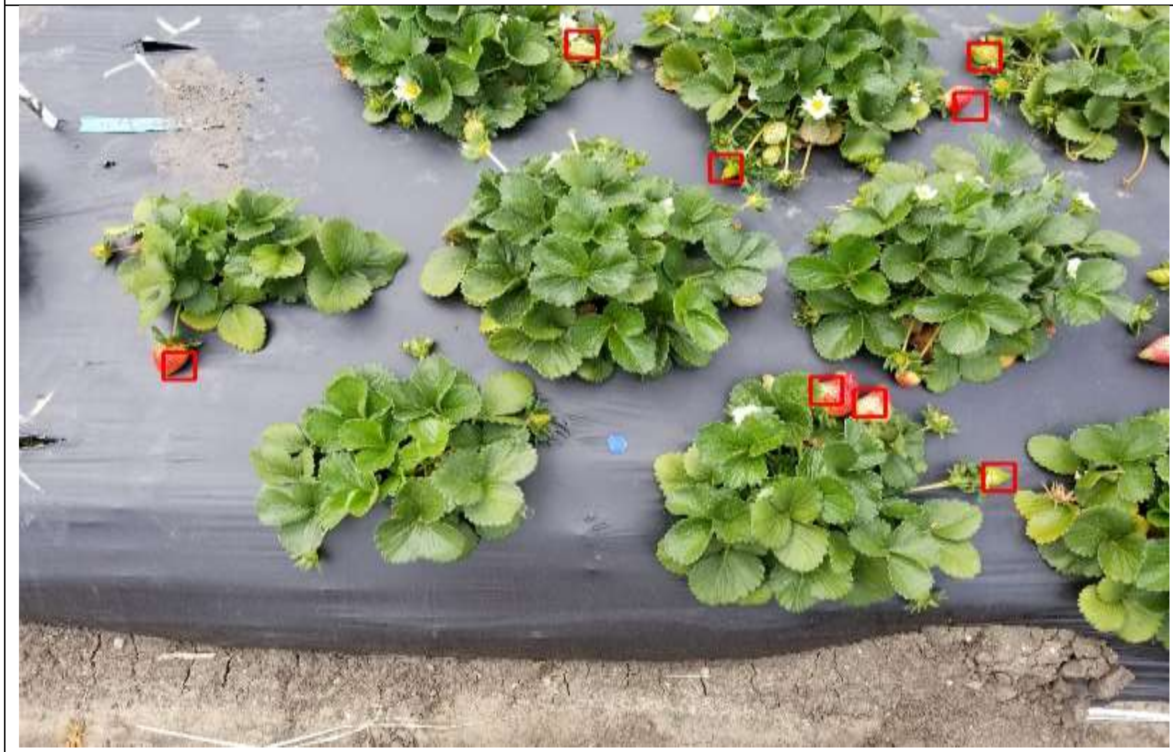
4.6.3 Results

Table 5: CNN Strawberry Detection Results Under Resolution (600,800)

Test data	True Positives	False Positives	Total Berries	Accuracy (%)
1-21	137	70	154	89
22-43	140	73	168	83
44-65	172	152	192	90
66-87	95	114	110	86
88-109	212	115	232	91
110-131	136	78	143	95
132-148	185	79	216	86

Employing a convolutional neural network proved to be extremely effective compared to the previous texture and gradient extraction techniques. It performed better in two primary ways; the first and most noticeable metric it outperformed in was accuracy. The average accuracy when using a CNN was 88.5%. This was a significant increase of 14.2% from the previous benchmark (LBP+SVM). Table 5 displays the accuracy in intervals of 21. Figure 43 displays some sample results when employing a CNN.



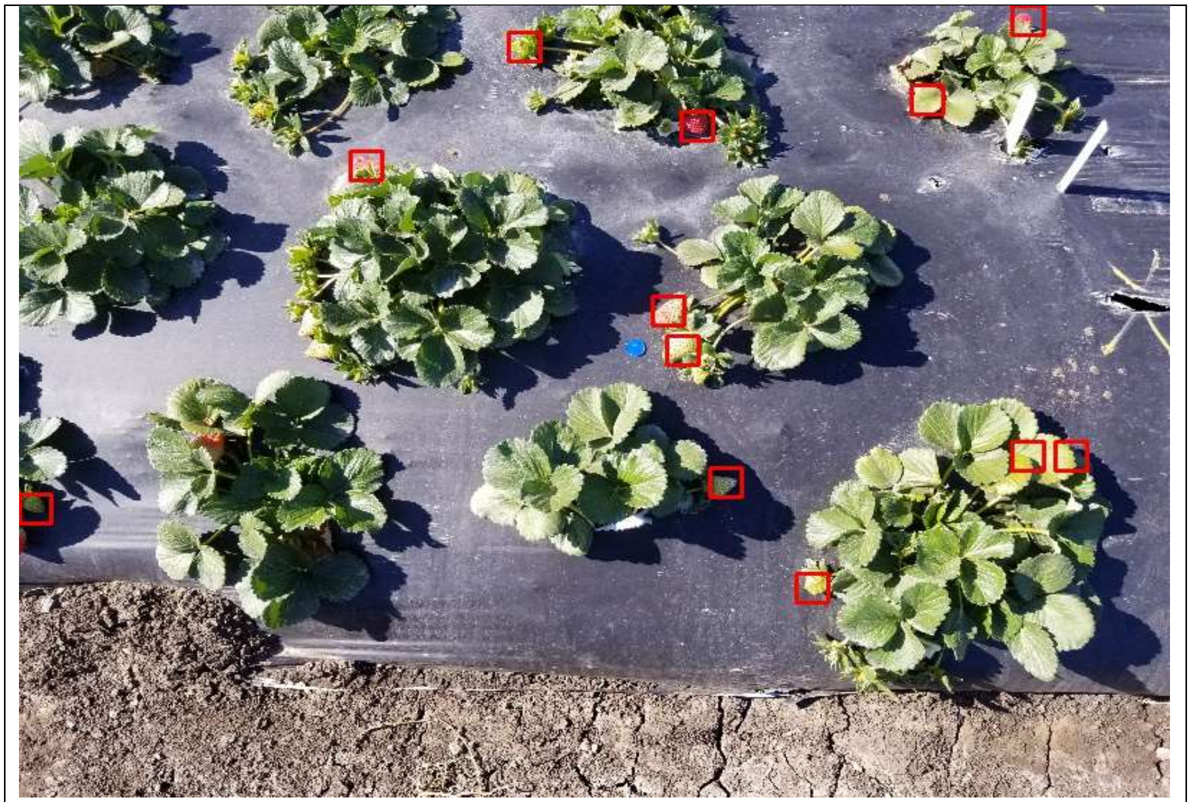


Figure 43: CNN Strawberry Detection Results Under a (600,800) Image Resolution

The second instance under which it excelled in was strong detections regardless of image resolution. This was remarkable as it proves the strength of the networks feature extractors. Due to the neural network being able to extract several features over just one as in the previous techniques, it can generate a stronger "understanding" for what represents a strawberry. To present this, feature maps were extracted at various layers as shown in Figure 44. The initial layers are much more general as it extracts essentially the entire image and uses various forms of edge detectors. As you go deeper into the network, the features become less visually representable. The filters in those layers are in charge of determining the images' class and thus is seeking much more specific features. Few of the results under the (2000,3000) resolution can be seen in Figure 45. When being compared to the images under the (600,800) resolution, the detections are consistent hence establishing the robust nature of the network. The small dimension size is also much more computationally efficient process.

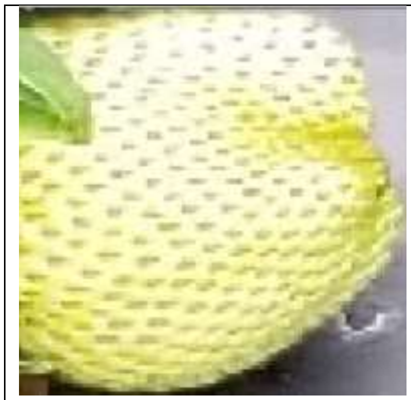
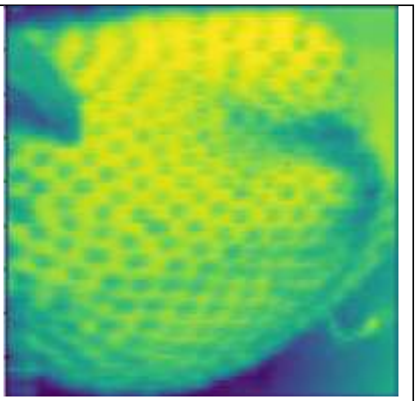
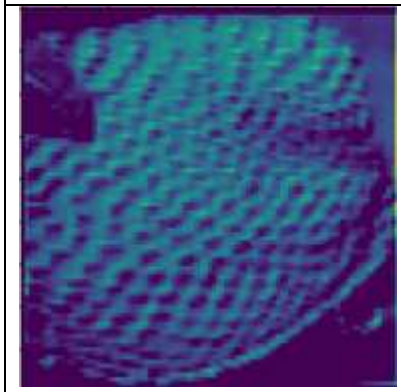
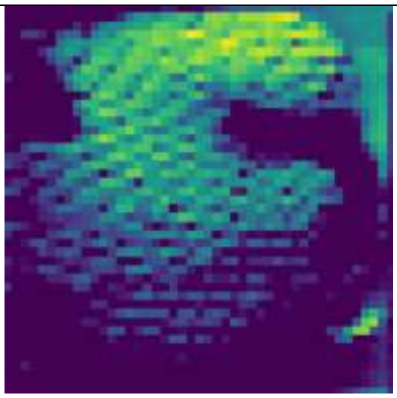
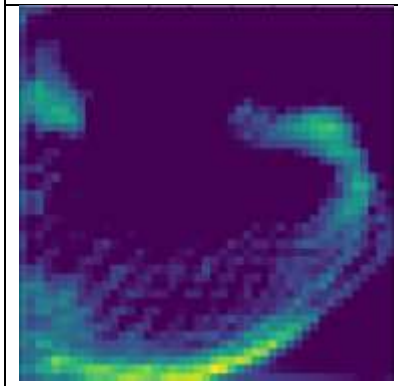
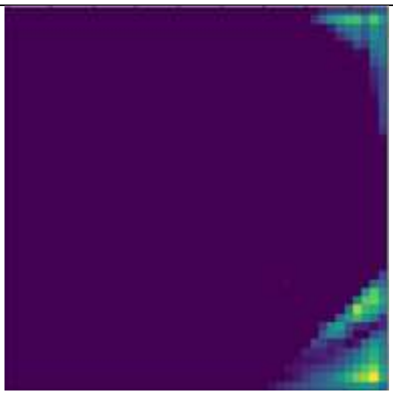
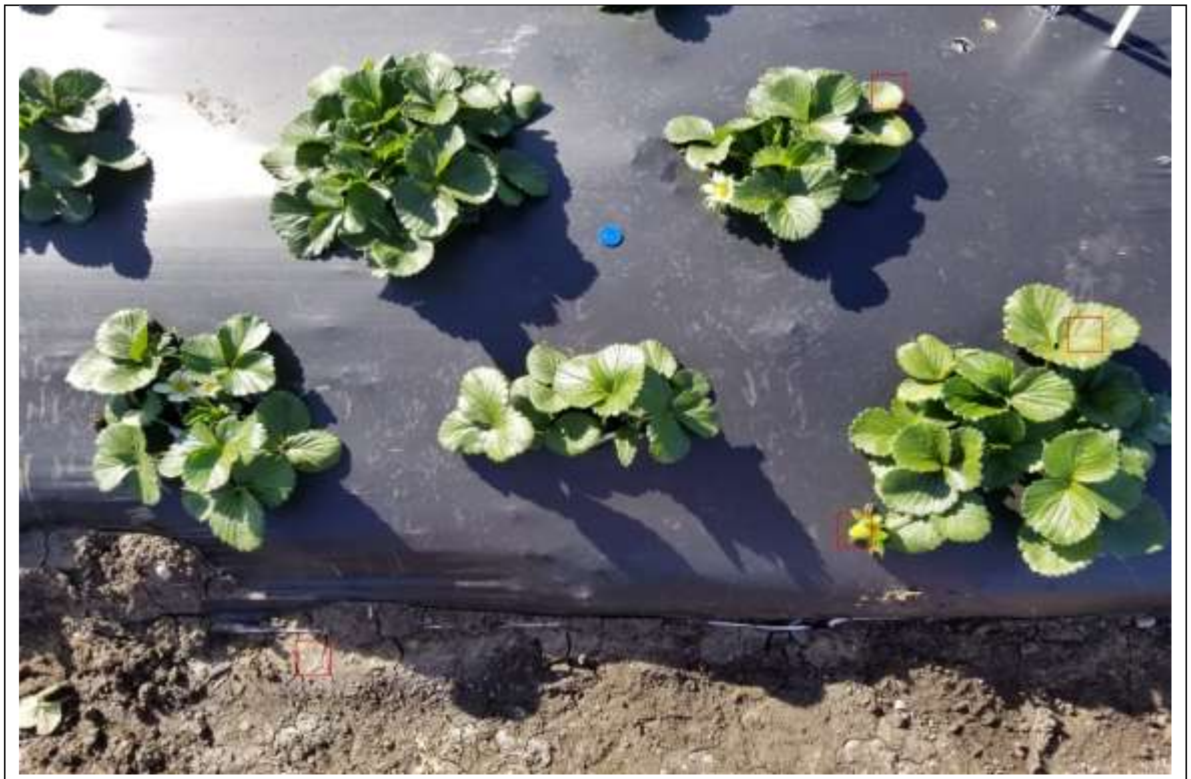
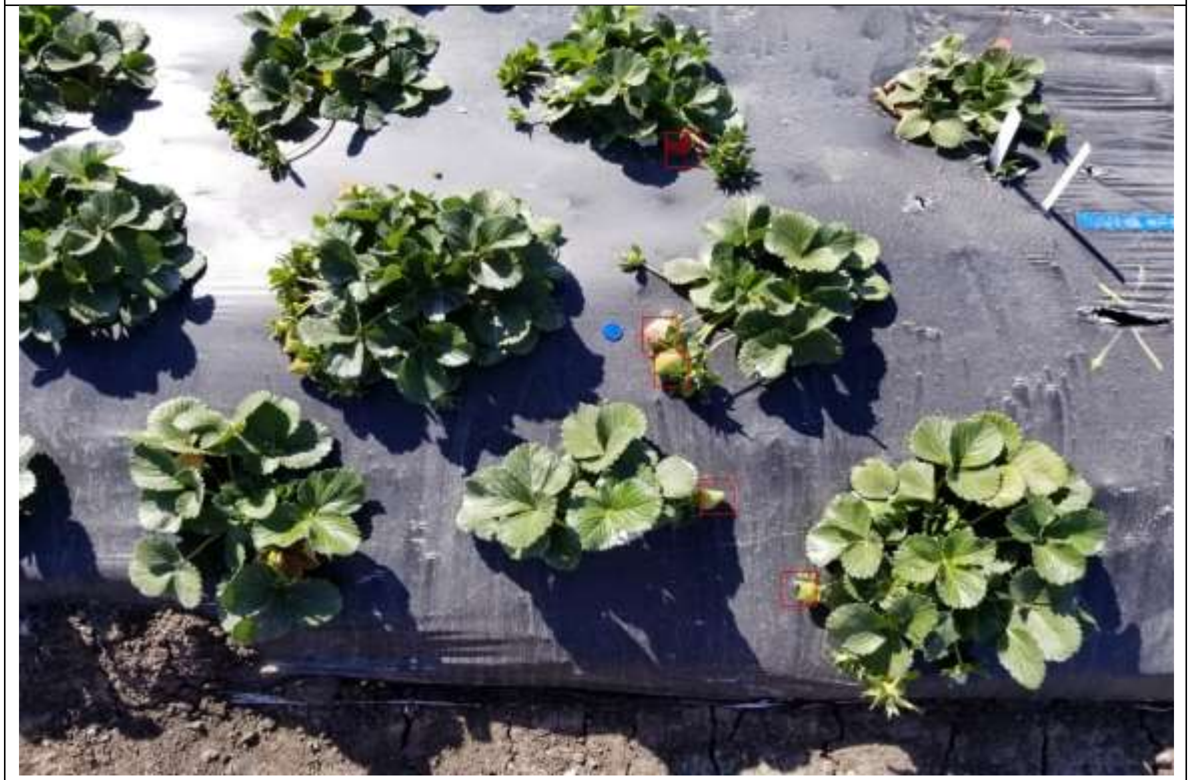
 a) original	 b) layer 1
 c) layer 3	 d) layer 5
 e) layer 7	 f) layer 9

Figure 44: Feature Maps Extracted at Various Layers of the Network

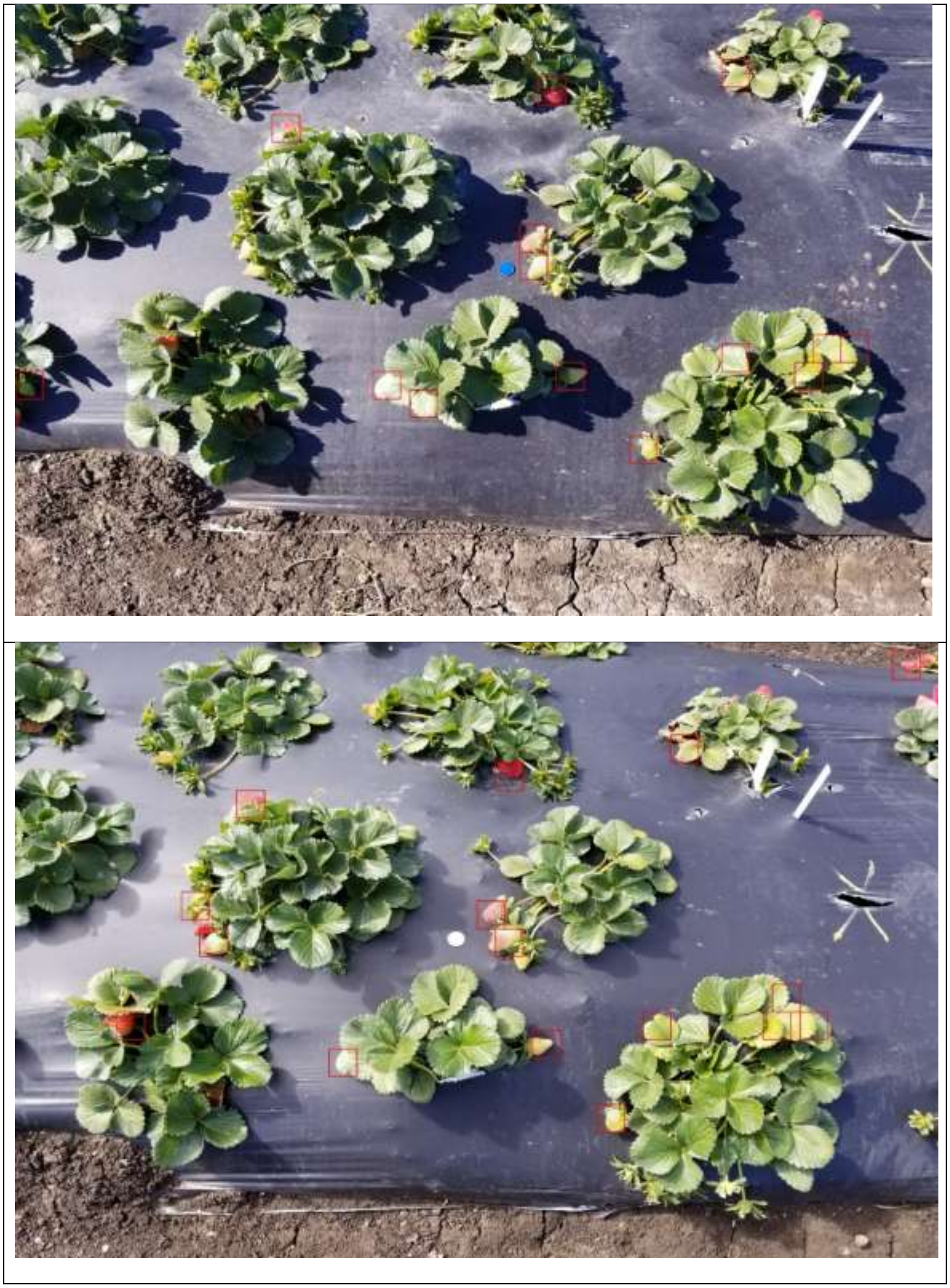


Figure 45: Strawberry Detection Results Under the (2000,3000) Image Resolution

4.6.4 Flaws

The primary flaw that occurred while employing this technique was the false positive detections. An average of 4.6 false positives were detected for each test image as inferred from the Table 5. While this is almost 1 extra false positive per image than the LBP technique, the LBP results were generated under a much larger resolution of (5000,4000) so one must keep in mind the tradeoffs. Due to the limited data set it was not possible to hard mine the false positives. Only adding non-strawberries to the training set severely biased the network towards the “non-strawberry” class and thus in the process decreased the true positive classifications as well. If more data was available at the time, I am confident the false positives would decrease while possibly improving the true positive results as well.

4.7 Coin Distance Measurement

An additional coin detection algorithm was implemented. One of the reasons for placing the coin was to determine the height at which the images were taken by the camera. However, the height had to be displayed using the image information itself not manual measurement. The pinhole technique was applied to measure the height [25]. Before the distance could be measured, the coin was first detected. The coin was detected using a rudimentary color segmentation technique like the one explained earlier for ripe strawberry detection. The primary difference being the color isolation which in this application was blue. Thereafter a binary threshold was taken of the isolated pixels to localize the object.

Finally, the contours of the binary image were taken so the coordinates could be retrieved and used to box the coin for clear visualization. Details of how color segmentation can be employed to leverage to detect objects can be seen under the ripe strawberry section. The purpose of this section is to determine the coin’s distance. The pinhole technique began once the coin’s coordinates were found. Focal Length (F) was the first variable required to be determined. The focal length had to initialized only once for calibration purposes and thereafter the process could be automated. Without initializing the focal length, it’s not possible to employ this methodology to determine distance as the initial focal length value serves as a base position for the pixel changes. Focal length can be determined given the parameters pixel width, object distance and object width.

```
#Focal Length calibration formula
F = (pixW * objDist) / objWidth
```

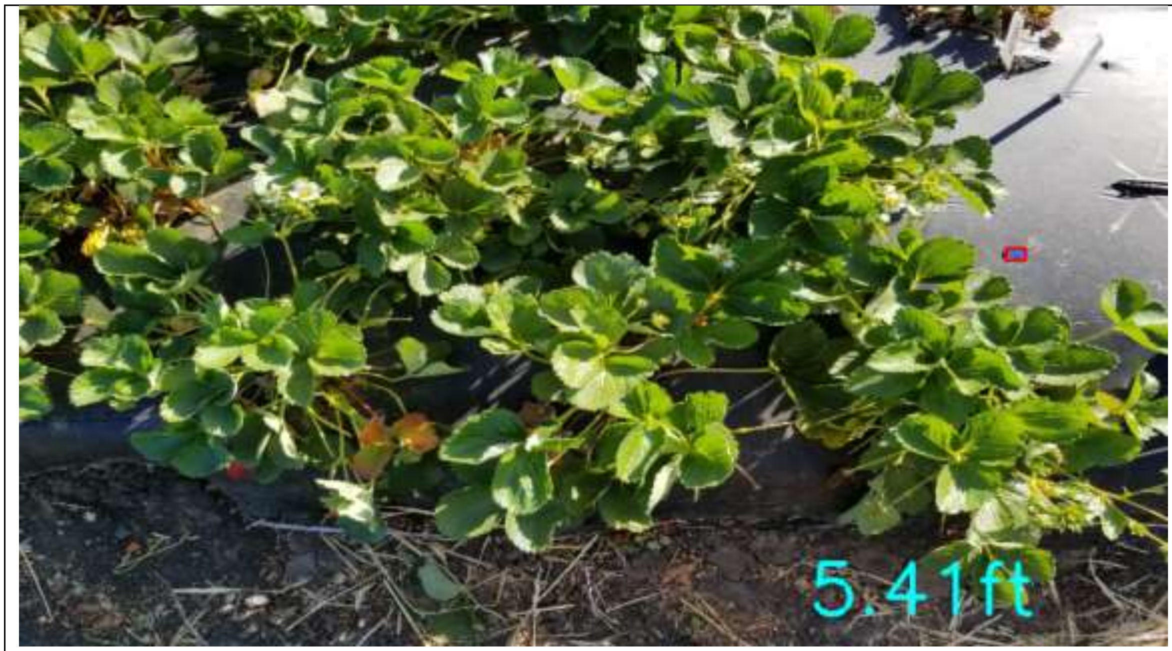
Once the focal length was initialized with a set constant, the distance deltas can be easily measured using

```
#distance per detected coin
Distance = (objWidth * F) / pixW
```

Due to the coin’s width and focal length remaining constant always the distance change is only affected by the increase or decrease in the coin’s pixel width. Determining the image height and size of the coin can be very useful as the coin can then be used as a reference object to determine the approximate size of the strawberries within the image.

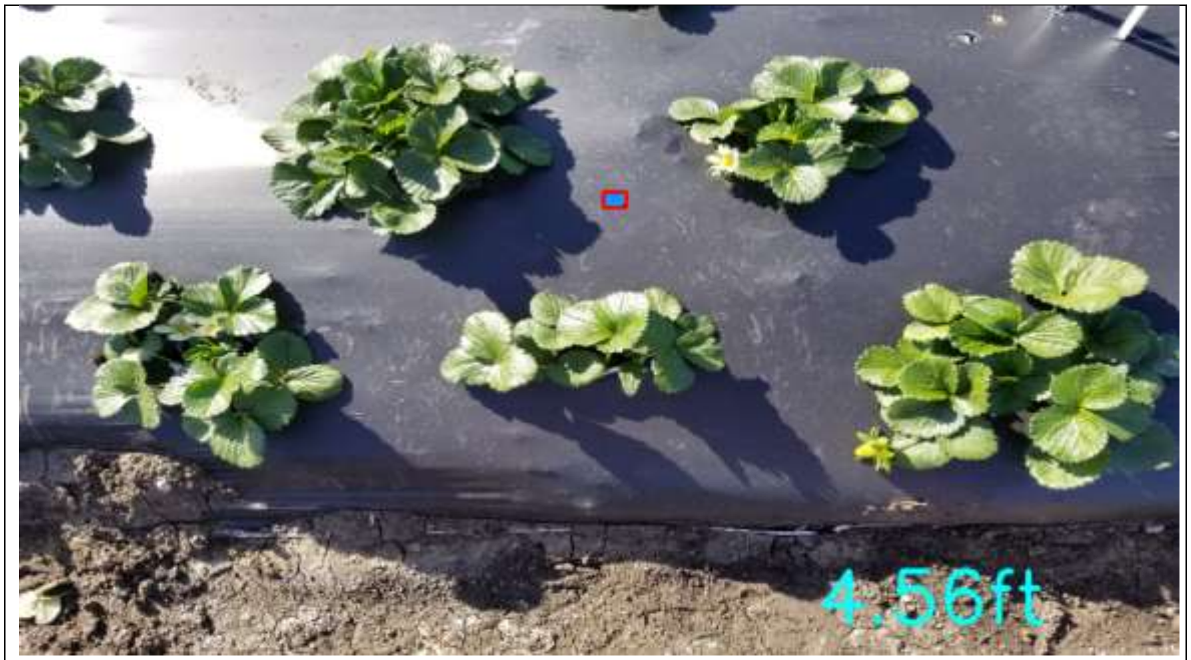
4.7.1 Results

The pinhole technique performed well as the distance measurements were reasonably accurate given the approximate height at which the images were captured. Figure 46 displays samples of the distance measurements. Issues arose more towards the detection aspect of the implementation as only color segmentation was applied and therefore it was not as robust of a technique to isolate the coin in every frame. If the coin was slightly detected and the pixel width was not accurately representing the entire coin, the distance measurement would be much further than its actual measurement. As the pixel width ends up being very small thus fabricating the coin to seem much smaller and at an even greater distance than its original position. Figure 47a and 47b represents how the detection issues disrupted the distance measurements. Another issue occurred under certain conditions due to strong lighting glares. In this scenario, the algorithm did not view the coin under the blue channel in the HSV spectrum, thus preventing it from even having the ability to determine the coin’s distance as seen in Figure 47c and 47d. While this methodology had strong results, one must note this is a relatively simple method towards approaching this problem and hence is sensitive to factors such as lighting, angle and initial pixel calibration.

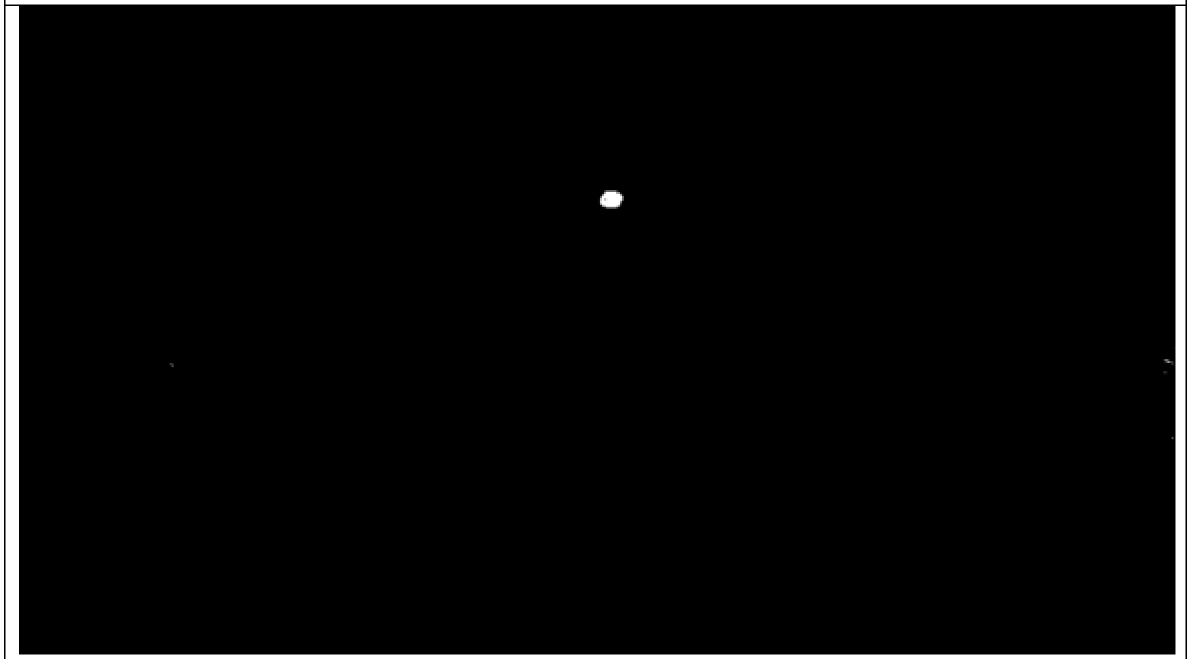


a) Coin measured at a distance at approximately 5 ft

b) Binary coin detected

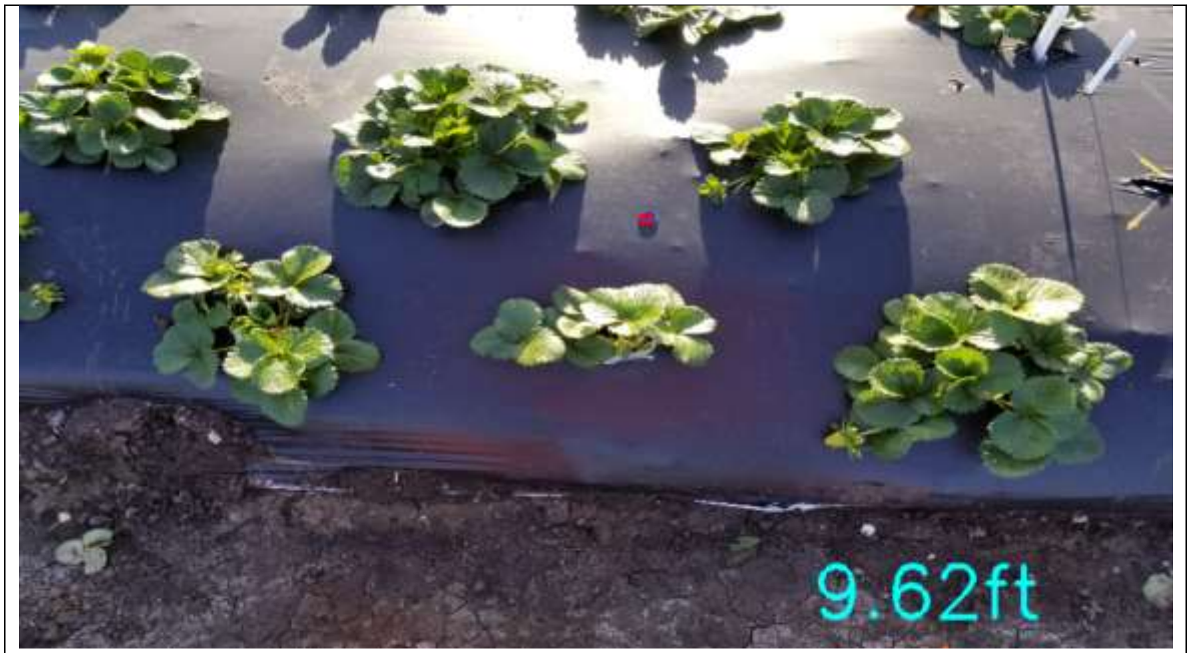


c) Coin measured at approximately 4 ft

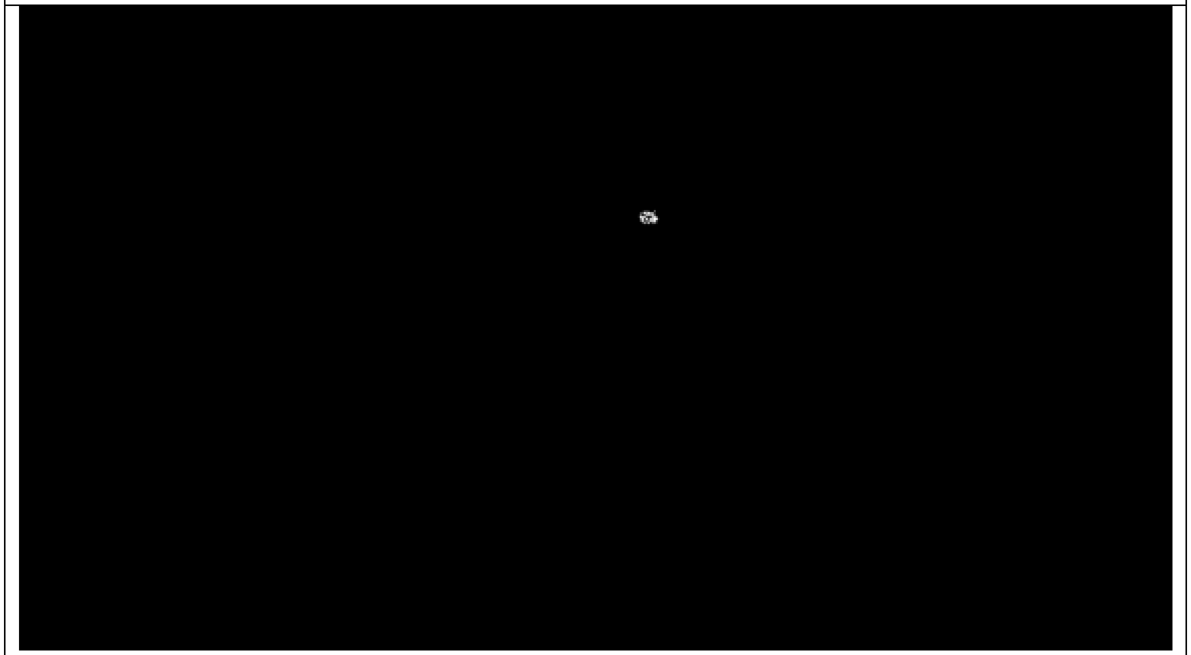


d) Binarized image

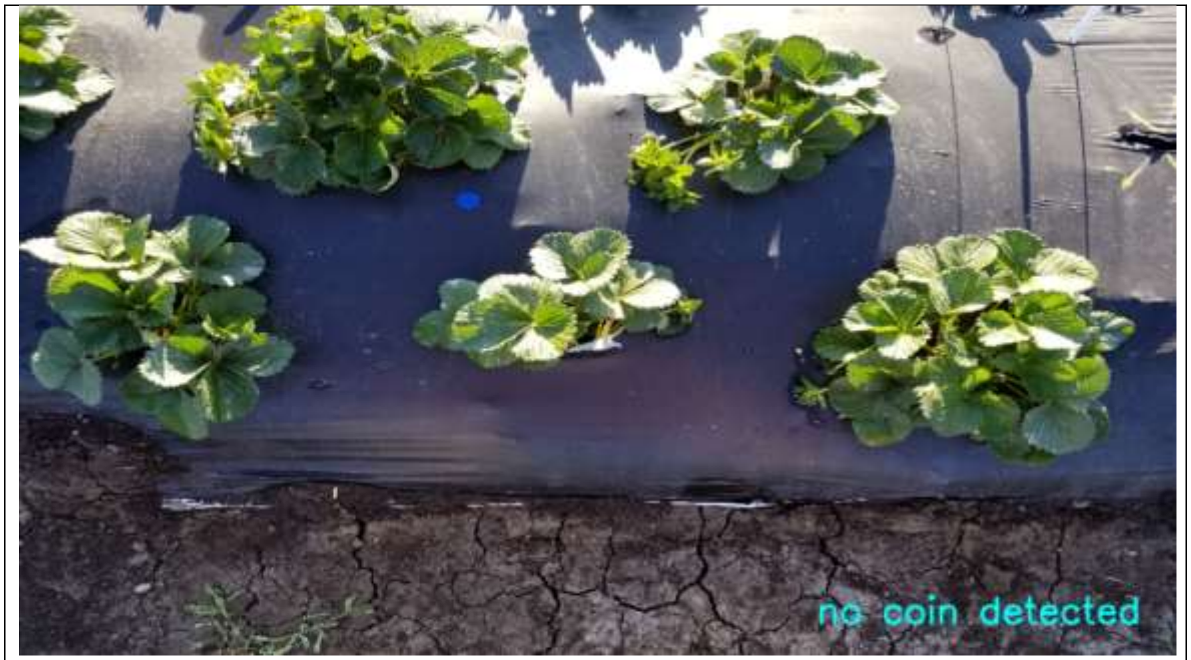
Figure 46: Coins Detected with Appropriate Measurements



a) Miscalculated coin distance



b) Threshold image not well binarized



c) HSV range not able to capture this specific shade of blue

d) No coin detected to determine distance

Figure 47: Issues with the Coin Measurement Algorithm

Chapter 5

CONCLUSION & FUTURE WORKS

5.1 Conclusion

Farmers in the agriculture industry have been struggling within the past decade to find consistent number of workers [1]. This can have a negative impact on crop management. Three well-known computer vision techniques in Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP) and Convolutional Neural Networks (CNN) were applied specifically to detect strawberries at different stages of its growth cycle. All three have performed well in the past for object detection purposes [5] [8] [21]. However, the two major challenges in this situation was the strawberry size, which are very small in relation to most objects detected in the past by these algorithms and the limited data set available. Computer Vision is a very data driven process and thus the larger the data set the higher one's chance of improving their accuracy. HOG performed very poorly under all experimented parameters as the shape of the strawberries were very inconsistent for each image. This prevented HOG from creating a consistent training model given the training data.

The LBP methodology performed quite well under the appropriate parameters especially given the limited data set. However, it required up-sampling the testing images above the original size to determine the objects texture. Increasing the image dimension compromised the computation efficiency, this was the byproduct of improving accuracy. Along with the feature descriptors a classifier was also required to determine which class the test data belong too. Two classifiers were experimented with and Support Vector Machines (SVM) proved to perform better than the K-Nearest Neighbor classifier (K-NN). Finally, a CNN inspired from the VGGNet architecture [21] was employed to classify and detect the strawberries. This proved to be the best methodology as it detected 88% of the strawberries in the testing images. It was able to achieve this accuracy under an extremely low resolution of 600x800 thus being the fastest run-time test algorithm of all the algorithms implemented. The accuracy was similar even under high resolutions as well, therefore being the most consistent detector of the three. The deep network architecture enabled it to learn many more features than the single featured descriptors. An average of 4 false positives were detected per image but it can be reduced with more available training data. Overall, the CNN performed the best between the three techniques as a multi-feature extraction enabled the algorithm to distinguish strawberries from the other objects in the background.

5.2 Future Works

A lot was accomplished as part of this strawberry detection project, three unique methodologies were implemented and compared against each other with the CNN structure being the most effective. However, more models can be implemented to compare with such as the R-CNN technique. Due to lack of time and data this technique was not attempted but given much more data, the R-CNN might prove to be the most effective methodology yet. Ripe and Unripe strawberry detections can also be part of the CNN based object detector given a larger amount of data. Once detection is completed, it would be beneficial to implement a prediction-based algorithm which can potentially determine the amount of time before the strawberry has ripened. This can be very beneficial to farmers as they can prepare, and plan beforehand and thus prevent major crop wastage.

BIBLIOGRAPHY

[1] S. Smith, C. Garcia, “Worker Shortage Hurts California’s Agriculture Industry”, Planet Money, 2018.

[2] CDFA, “California Agricultural Production Statistics”, CA.gov, 2018.

[3] N. Pullman, “Almost 10 per cent of strawberry crops go to waste”, Fresh Produce Journal, 2017.

[4] S. Puttemans et al., “Automated visual fruit detection for harvest estimation and robotic harvesting”, IEEE IPTA, 2016.

[5] T. Ojala, M. Pietikainen, T. Maenpaa, “Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns”, IEEE PAMI, 2002.

[6] R. Schapire, “Explaining AdaBoost”, Springer: Empirical Inference, 2013.

[7] Xin Li et al., “A Deep Learning Method for Recognizing Elevated Mature Strawberries”, IEEE YAC, 2018.

[8] N. Dalal, B. Triggs, “Histogram of Oriented Gradients for Human Detection”, IEEE CVPR, 2005.

[9] S. Mallick, “Histogram of Oriented Gradients”, Learn OpenCV, 2016.

[10] C. Cortes, V. Vapnik, “Support-Vector Networks”, Bell Labs, 1995.

[11] Yangquing Jia et al., “Caffe: Convolutional Architecture for Fast Feature Embedding”, ACM MM, 2014.

[12] Inkyu Sa et al., “DeepFruits: A Fruit Detection System Using Deep Neural Networks”, Sensors, 2016.

[13] Shaoqing Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, IEEE PAMI, 2017.

[14] Ping Lin et al., “Deep convolutional neural network for automatic discrimination between Fragaria × Ananassa flowers and other similar white wild flowers in fields”, BioMed Central: Plant Methods, 2018.

[15] A. Bosch et al., “Representing Shape with A Spatial Pyramid Kernal”, ACM CIVR, 2007.

[16] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, ACM Journal of Computer Vision, 2004.

[17] Suchet Bargoti & James Underwood, “Deep Fruit Detection in Orchards”, IEEE ICRA, 2017.

[18] Tapas Kanungo et al., “The analysis of a simple k-means clustering algorithm”, ACM Symposium on Computational Geometry, 2000.

[19] S. Pedersen, “Circular Hough Transform”, Semantic Scholar, 2007.

[20] C. Wang et al., “Detection and counting of immature green citrus fruit based on Local Binary Patterns (LBP) feature using illumination-normalized images”, Springer: Precision Agriculture, 2018.

[21] K. Simonyan & A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv: CVPR, 2014.

[22] A. Rosebrock, “Histogram of Oriented Gradients for Object Detection”, Pyimagesearch, 2014.

[23] A. Rosebrock, “Sliding Window for Object Detection”, Pyimagesearch, 2015.

[24] T. Malisiewicz, “Blazing Fast NMS”, ComputerVisionBlog, 2011

[25] R.C. Gonzalez, R.E Woods, “Digital Image Processing”, Pearson Prentice hall, 2008.

[26] M. Nielsen, “Using Neural Networks to recognize hand written digits”, Neural Networks and Deep Learning, 2018.

[27] M. Nielsen, “How Back Propagation works”, Neural Networks and Deep Learning, 2018.

[28] H. Pokharna, “The best Convolutional Neural Network explanation on the internet”, Medium, 2016.

[29] “Color Models: RGB, HSV, HSL”, WikiBooks, 2018.

[30] T. Malisiewicz, A. Gupta, A. Efros, “Ensemble of Exemplar-SVM’s for Object Detection and Beyond”, ICCV, 2011.

[31] S. Van Der Walt et al., “Scikit-Image: Image Processing in Python”, Scikit-Image, 2014.