TOWARDS A STRAWBERRY HARVEST PREDICTION SYSTEM USING COMPUTER VISION AND PATTERN RECOGNITION

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
Andreas Apitz
June 2018

COMMITTEE MEMBERSHIP

TITLE:	Strawberry Harvest Prediction Using Computer Vision and Pattern Recognition
AUTHOR:	Andreas Apitz
DATE SUBMITTED:	June 2018
COMMITTEE CHAIR:	Jane Zhang, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Xiao-Hua Yu, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Wayne Pilkington, Ph.D. Associate Professor of Electrical Engineering

ABSTRACT

Towards A Strawberry Harvest Prediction System Using Computer Vision and Pattern Recognition

Andreas Apitz

Farmers require advance notice when a harvest is approaching, so they can allocate resources and hire workers as efficiently as possible. Existing methods are subjective and labor intensive, and require the expertise of a professional forecaster. Cal Poly's EE department has been collaborating with the Cal Poly Strawberry Center to investigate the potential in using digital imaging processing to predict harvests more reliably. This paper shows the progress of that ongoing project, as well as what aspects could still be improved. Three main blocks comprise this system: data acquisition, which obtains and catalogues images of the strawberry plants; computer vision, which extracts information from the images and constructs a time-series model of the field as a whole; and prediction, which uses the field's history to guess when the most likely harvest window will be. The best method of data acquisition is determined through a decision matrix to be a small autonomous rover. Several challenges specific to images captured via drone, such as fisheye distortion and dirt masking, are examined and mitigated. Using thresholding, the nRGB color space is shown to be the most promising for image segmentation of red strawberries. Data from field 25 at the Cal Poly Strawberry Center is tabulated, analyzed, and compared against industry trends across California. Ultimately, this work serves as a strong benchmark towards a full strawberry yield prediction system.

ACKNOWLEDGMENTS

Thank you to Dr. Jane Zhang for suggesting this project as my thesis, and for offering support, advice, feedback, and resources throughout the thesis process. Thanks to the rest of my thesis panel, Dr. Helen Yu and Dr. Wayne Pilkington, for their contributions. Thank you to Dr. Gerald Holmes for sharing your insight into the agriculture industry and experience in the growth patterns of strawberries.

To my parents, thank you for all your support; not only in the financial sense, but also moral support, giving me the time and motivation that I needed to focus on my schoolwork both on and off vacation.

Thank you to Wei-Chih Chern and Jon Sato for laying the groundwork for this project, and thank you to Jeannie Whitmore and Nathan Adelman, who assisted in my Computer Vision final project, in which we examined red strawberry detection from images obtained by drone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. BACKGROUND AND OVERVIEW

1.1 THE PROBLEM

Over the past few years, the Californian agriculture industry has seen widespread labor shortages [1]. In a survey conducted by the California Farm Bureau Federation (CFBF), of 762 farmers who responded, 55% reported employee shortages (69%, for farmers who employ laborers seasonally). Offering higher wages and/or benefits did not solve this problem either.

As a result, farmers are increasingly turning towards mechanized and autonomous solutions to stay competitive [2]. For now, the expectation is not that human labor will be entirely replaced, but this technology does have the potential to create many highly-skilled jobs in the industry, especially for engineers.

Given these trends, it is highly desirable to create an autonomous system which could monitor a field of strawberries through its growth cycle and build an accurate and objective prediction as to when the berries will be ready for harvest, and how many berries will need to be picked. This would allow farmers more flexibility in hiring an appropriate number of laborers when they will be needed. Misestimating labor requirements results in fiscal loss, making this system an attractive alternative for strawberry farmers.

1.2 CURRENT PRACTICES

At present, the only major method available to increase harvest prediction accuracy is hiring a forecaster to visit the field. They will take a random sampling of plants and tally berries in each stage of life, evaluate the field's prior seasons and overall health (noting any diseases or fungi that may be present), and consider the field's climate, among other considerations. From this information, they make a holistic and subjective prediction as to the current season's harvest date and volume. However, such a method is expensive and has higher variance than one based on numerical data and objective analysis.

In the opinion of Dr. Gerald Holmes, director at the Cal Poly Strawberry Center, farmers require a minimum of 10 days' notice in order to rent equipment and hire laborers to pick strawberries. Therefore, this is the target for the system to make an accurate harvest prediction.

1.3 RELATED WORK

For the past year and a half, Cal Poly SLO's Electrical Engineering department has had a partnership with the Cal Poly Strawberry Center, which has kindly set aside a few rows of strawberries for experimentation. This has already resulted in several successful projects, notably including Wei-Chih Chern's thesis, A Comparison of Image Processing Techniques for Strawberry Detection, which was completed in June 2017, and Jon Sato's research project, Computer Vision Strawberry Research, which was completed in December 2017. Wei-Chih's thesis discussed several key computer vision techniques such as thresholding, mean shift, K-mean clustering, template matching, and the Viola-Jones cascaded classifier. Jon Sato's research paper documented his data acquisition process through Summer and Fall 2017, using a cell phone's camera to photograph each plant every other day. Jon also explored image segmentation through thresholding and using a univariate ARIMA model to predict the field's future activity. Both papers served as an excellent starting point for this thesis.

In Wei-Chih Chern's thesis, "A Comparison of Image Processing Techniques for Strawberry Detection", Chern focused primarily on segmenting out strawberries from close-up images under variable lighting. He also worked on recognizing green strawberries and flowers, which is outside the scope of this project (but important for the overall Strawberry Harvest Prediction system).

Chern concluded that the best method for recognizing red strawberries was thresholding in the Lab color space. He reported a 96.70% accuracy rate using this method for “close-up” images, and a 95.29% accuracy rate for “far-view” images. The thesis defines far-view images as follows:

> “The far-view images at most contain two rows of strawberry plants, and were photographed with different angles and days. The farview images were all photographed around 3-4 PM in sunny days.”

Unfortunately, distance and resolution metrics were not supplied, making it difficult to compare results between the thesis and this project. The figures shown in Chern’s thesis are all exceptionally crisp and have no visible fisheye effect, and the strawberries take up a significant amount of the image, so it is likely that our dataset will perform less well than Chern’s, even using the same methods. Figure 1 below shows one example of a far-view image, with bounding boxes around detected red strawberries.

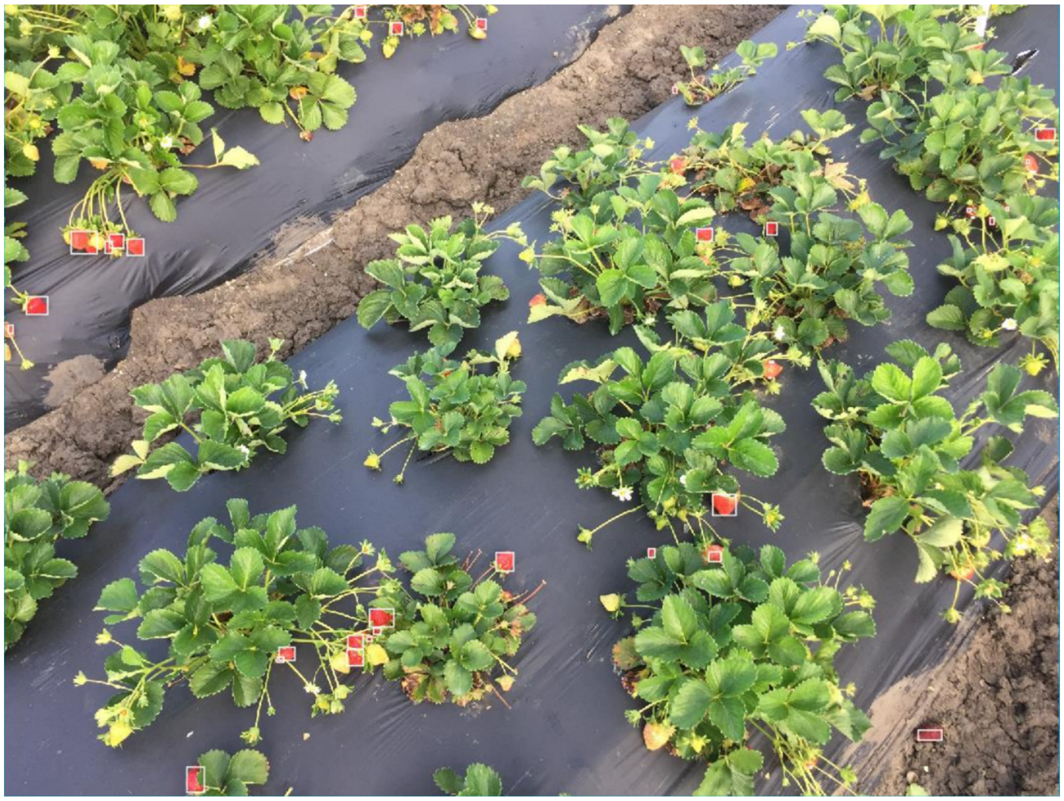


Figure 1: “Far-view” image from Chern’s thesis, with bounding boxes around detected red strawberries

This picture appears to have been taken from roughly standing height. It is unlikely for drones to achieve even this level of proximity to the plants, so the far-view results from Chern’s thesis should not be considered representative of the performance achievable with the same methods in the case of drone-based data acquisition.

Thresholding was also faster than the other methods considered, but more reliant on precise thresholding values, which needed to be picked by hand and could easily vary with lighting and camera model. It should also be noted that a Gaussian filter was applied prior to thresholding, to reduce the effects of “intense sunlight”, and a morphological closing operation was performed prior to thresholding.

This same type of automation is being researched in industry as well. Current projects include robots that can perform seeding around the clock or pick fruits with near-human precision [3]. These advances could ease the labor shortage in the near future, ideally without the cost of human-level accuracy in computer vision problems, and with the added efficacy of 24-hour performance.

1.4 OVERVIEW AND BLOCK DIAGRAM

Based on the needs of strawberry farmers, the proposed system should use digital imaging to reliably predict the date and yield of the field under test, at least 10 days in advance. This system should be relatively inexpensive, easy to install, and require as little user input and expertise as possible to operate. A labeled set of training data is necessary to train the classifier to separate objects of interest such as flowers, green strawberries, and red strawberries from their surroundings. In future, weather data and seasonal trends could provide useful variables for consideration as well.

Figure 2 shows the main three blocks within the proposed system.

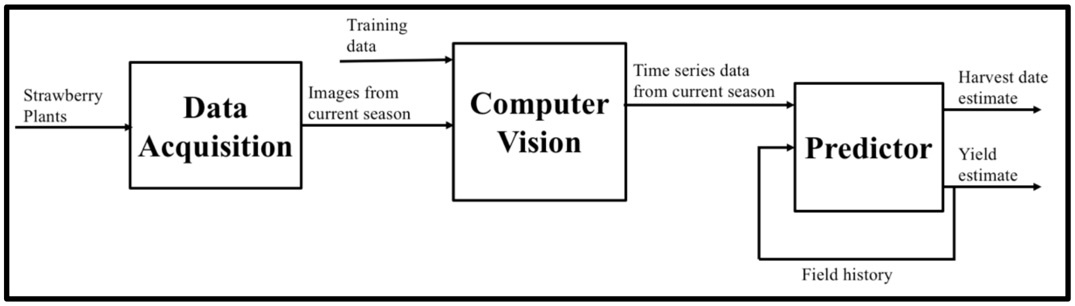


Figure 2: Level 1 Block Diagram

Tables 1-3 discuss each of these blocks in greater detail. Currently, the proposed system can only accommodate visual data; in the future, additional inputs such as temperature data, fruit size, field health, fertilizer types, local climate, etc. could be added to better guide the predictor. However, extensive research would be required to determine the effects of each of these variables on the harvest date and yield.

Table 1: Level 1 Functionality Table, Data Acquisition

Module	**Data Acquisition**
Inputs	• Strawberry plants
Outputs	• Images from current season
Functionality	In data acquisition, the strawberry plants are photographed in such a manner as to minimize occlusion and lighting issues. These images are prepped for analysis by removing distortion and consolidating data into the minimum number of image files. In the future, this stage could also record non-visual data such as temperature and weather conditions, since this information helps to create a more accurate model.

Table 2: Level 1 Functionality Table, Computer Vision

Module	**Computer Vision**
Inputs	• Images from current season • Training data
Outputs	• Time series data from current season (Excel spreadsheet, one column per stage)
Functionality	The computer vision block converts images into data about how many objects are detected per sample, in each stage of development. Labeled training samples are used to train the classifier to recognize objects of interest: flowers, green strawberries, and red strawberries. Then, each image supplied by the data acquisition block is analyzed for these objects. Results are exported as a spreadsheet and sent to the predictor.

Table 3: Level 1 Functionality Table, Predictor

Module	**Predictor**
Inputs	• Time series data from current season (Excel spreadsheet, one column per stage) • Field history
Outputs	• Harvest date estimate (graphical) • Yield estimate (graphical)
Functionality	Previous data from the field under test is used to update a generalized probability model. Using this model and data from the current season,

	this block will predict how many strawberries will be ready for harvest at least ten days in advance.

Chapter 2 explores several methods of data acquisition and uses a decision matrix to select the optimal method. Chapter 3 applies computer vision techniques to problems with existing data captured by a drone and compares different color spaces to find one that best discriminates red strawberries from background pixels. Chapter 4 includes a primitive Gaussian model for a strawberry season; this is compared against actual data obtained from the field and trends observed in California as a whole. Finally, Chapter 5 lists some future directions for this project.

2. DATA ACQUISITION

This project cannot succeed without a large amount of data collected from the beginning to the end of a season. This data must be well suited for object detection and recognition. As the first stage in the system, it is vitally important that the method of data acquisition employed is one which minimizes negative effects such as occlusion and lighting variance, both of which make object detection difficult to impossible to automate. Therefore, this chapter will compare several methods of gathering image data for a full season. These methods are: taking pictures with a handheld camera or smartphone, placing cameras in the field to regularly and automatically record data, and flying a drone over a field while recording a video.

2.1 HANDHELD CAMERA METHOD

In this method, the end user takes pictures of each plant with a smartphone, covering the same section of the field in each sample. This may be the simplest method for the consumer to perform, but it takes far longer to cover even a subset of the field under test. Figure 3 below shows an example of an image taken by a smartphone. A penny is included in frame for size comparison.

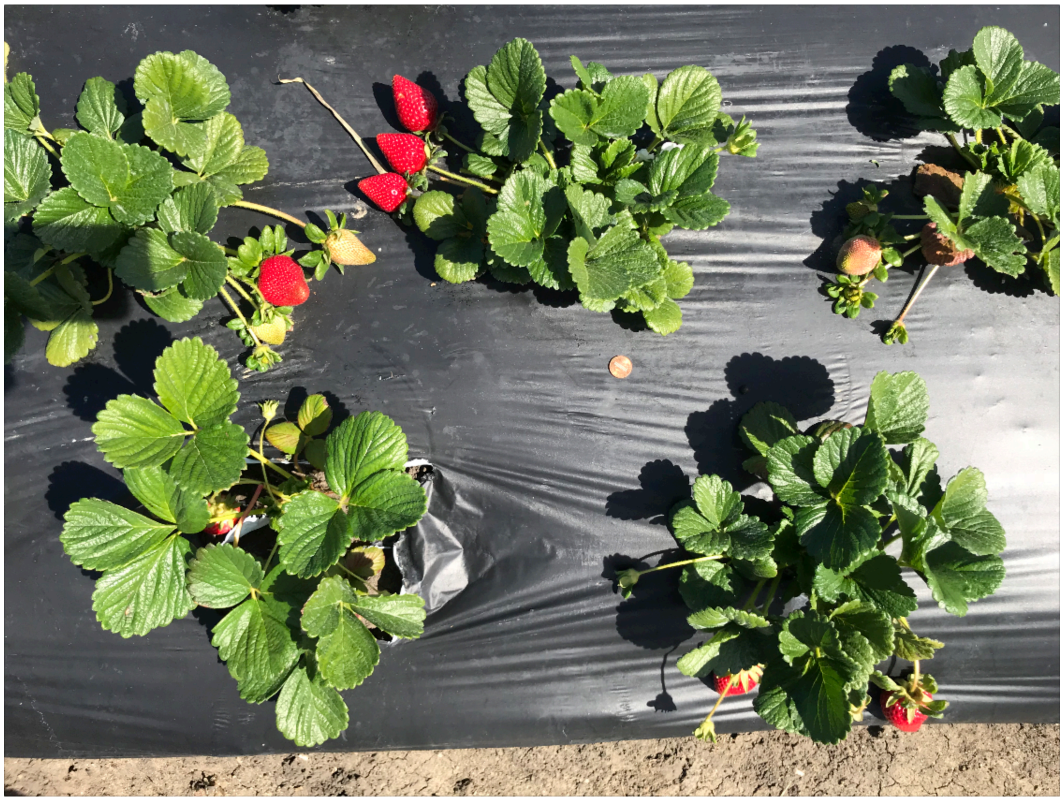


Figure 3: Smartphone image

Pew Research Center reports that 95% of Americans already own a cell phone [5], meaning that this method is also likely the least expensive. In fact, 43% of people polled on pcmag.com said they use their cell phone as their primary camera, rather than purchasing a dedicated camera [6]. This survey was conducted seven years ago; doubtless, even more people use their phone's camera primarily today. The picture quality capable with cell phone cameras has increased drastically in recent years as well [7]. For example, the image shown above was taken at a resolution of 4032 x 3024 by an iPhone 7 Plus, with a focal length of 3.99mm and f-number of 1.8 [26]. This resolution is even higher than most drone cameras. As a result, photos taken with a consumer's cell phone are perfectly suitable for computer vision techniques.

Another key advantage of this method is that it could easily allow the predictor to track individual plants (and individual fruits) through a season, rather than the sum total of the field. This would give a more detailed and useful view of the strawberries' growth.

When Jon Sato used this method in his research paper, he placed a dime in frame in each image. (This dime was painted blue in order to easily extract the dime from its surroundings using thresholding and algorithms which look for circles in images.) He used the dime's constant size to calculate the size of every detected strawberry. This allows the system to track a fruit's growth over time. For one example of an application where this would be useful, if it was observed that the fruit's size was decreasing, it could indicate that the fruit is overripe or diseased.

Unfortunately, taking a picture of a few plants individually presents a large time investment for the person obtaining samples, especially if measurements are to be taken daily. Crouching for images puts strain on the back as well. Therefore, options which are less strenuous for the end user should be considered.

2.2 MOUNTED CAMERA METHOD

Another possible implementation would be mounting a series of cameras throughout the field to automatically and periodically take pictures of the plants and upload them to a server remotely, through Bluetooth or Wi-Fi. Supplying power and wireless connection is a serious issue for this solution, and cost ramps up quickly for larger fields. Wireless sensor networks (WSNs) such as ZigBee [27] may provide a low-power low-cost scalable solution to this problem. For now, commercially available cameras mostly exist within the home security industry, and therefore record real-time continuous video rather than taking periodic pictures, but a product likely exists which meets this need.

One would also need to select an adequate density of cameras per square meter, ensuring that each strawberry plant is close enough that its fruits could be seen by the camera, but not by any others nearby, which would result in them being double-counted. Viewing angle becomes less important if the camera is mounted on a motorized gimbal, allowing it to take a picture in each direction. If cameras are permanently mounted in the field, they become subject to a number of other considerations, such as theft and water resistance.

To extend this idea, consider an autonomous rover with one camera on each side, which drives from its home base down each row of strawberries every day and records images automatically. For a small field, this is much more expensive than static cameras, but would likely be less expensive for large fields. Strawberry fields have been organized in matted rows for decades [8], so programming a small rover to drive between rows is a simple task. In general, strawberry fields usually have rows with a width of 64", although rows are narrower in the north (48-52"). It is unclear whether the spacing between rows is held constant like this, so the width of the rover should be minimized. This implementation solves the problem of pictures being taken too far from the strawberries, or at an angle that would be likely to cause occlusion, and if the rover's home base is closed off, then theft and water resistance become less important as well.

2.3 DRONE METHOD

Unmanned Aerial Vehicles (UAVs) such as quadcopters have proven useful for a variety of applications such as photography, shipping packages, recording sports footage, hobby flight, and law enforcement [9]. In addition, they have recently been applied to research and engineering problems including agricultural applications, as they provide a relatively cheap and unobtrusive way to gather data from overhead and can be equipped with lightweight sensors and cameras.

The main consideration for the drone method is determining at which altitude the drone should fly to generate the most useful results. For manual flight, this becomes a tradeoff between the amount of time it takes to generate a full sample and the resolution of the resulting data.

Using a pinhole camera model, one can derive the relationship between the minimum size of an object of interest and the maximum allowable drone altitude, given the resolution of the drone's camera. In the case of this project, the minimum size of an object of interest is in the range of half an inch. For a red strawberry, it is acceptable for this object to be only 4 pixels long in the resulting image, since simple thresholding is sufficient for segmenting red strawberries from background pixels. However, more advanced techniques such as texture analysis is required for green strawberries and flowers, so these objects must be at least 20 pixels in width in the resulting image. Figure 4 shows this concept graphically.

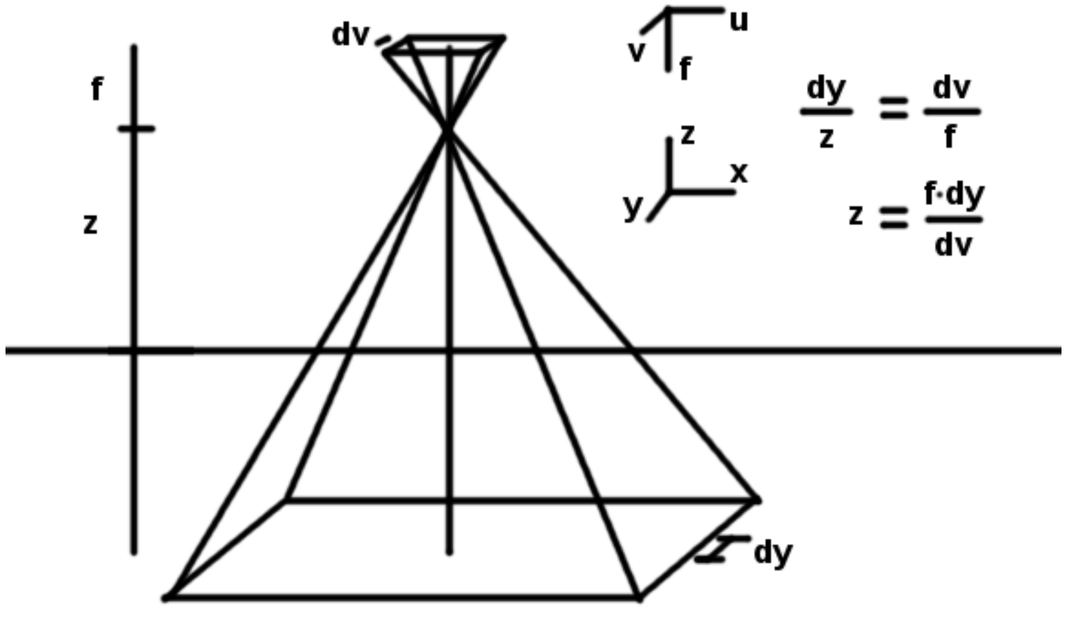


Figure 4: Pinhole camera representation

So, the relationship becomes:

$$\frac{dy}{z} = \frac{dv}{f} = \frac{v_{obj}}{r_{vert}}$$, where

dy = length of object,

z = distance to camera (altitude of drone),

dv = reflected length of object inside camera,

f = focal distance,

v_{obj} = length of object in output image in pixels, and

r_{vert} = vertical resolution

Or, solving for z,

$$z = \frac{dy * r_{vert}}{v_{obj}}$$

Therefore, the maximum required altitude can be calculated independently of the focal length of the camera. Table 4 shows some common values. 3000 pixels is typical vertical resolution for the DJI Phantom 4 drone, and 2160 pixels is typical vertical resolution for the GDU Byrd.

Table 4: Common values for altitude equation

y (in)	r_{vert} (px)	v_{obj} (px)	z (in)	z (m)
0.5	3000	4	375	9.525
0.5	3000	20	75	1.905
0.5	2160	4	270	6.858
0.5	2160	20	54	1.372

Drones are already being used to monitor the strawberry field at the Cal Poly Strawberry Center. Another team used a DJI Phantom 4 [10] to record last year's season, and a GDU Byrd Premium 2.0 [11] is recording the current season. In the former case, DJI offers software which makes photographing an area incredibly simple. After the user selects waypoints on a map of the area, the drone automatically flies over the specified area while photographing the ground. These images are stored with their corresponding GPS coordinates, and then the program is able to stitch the data into a single panoramic image. Unfortunately, the system has the limitation that it can only use the autopilot to travel between waypoints if it is flying 10 meters over the ground. As discussed earlier, this altitude is far too high to spot strawberries in the resulting panorama. Therefore, for the drone to fly lower to the ground, it would need to be controlled manually, and it is unrealistic to assume the end user for this project would have the flight experience necessary to fly the drone on straight paths to cover the entire field.

Furthermore, drones typically only possess half an hour's worth of battery life before they must land and recharge, which poses a problem if the field is large or the drone is flying near the ground (and therefore requires more trips across the field).

One last consideration for drones specifically is the presence of fisheye distortion for most drone cameras. Figure 5 shows the effect of fisheye distortion on an image of a strawberry field, with red lines to show how these straight rows have become curved.

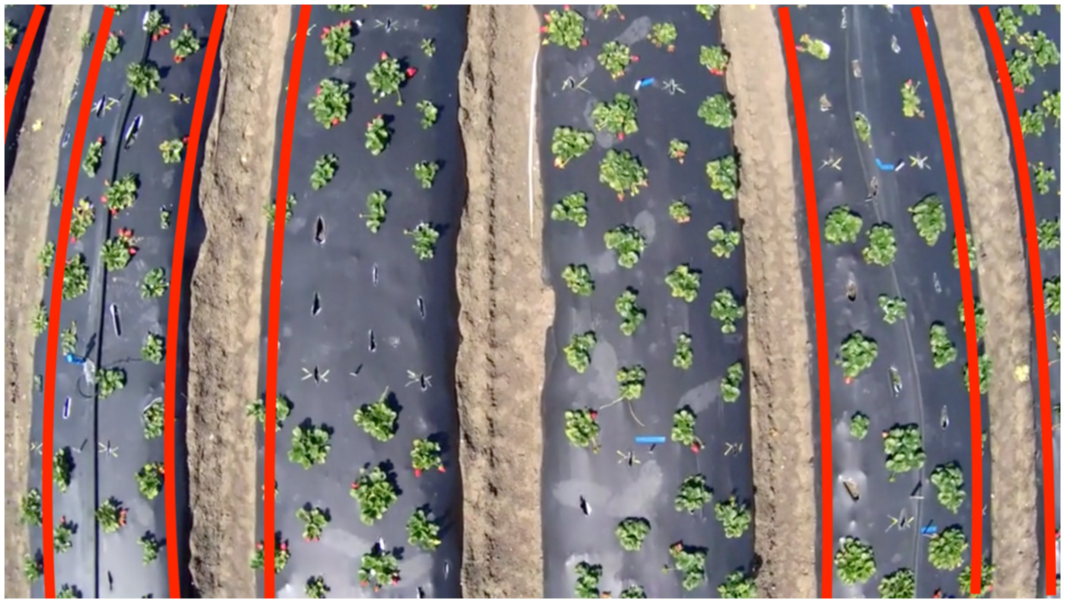


Figure 5: Fisheye distortion in strawberry field image (red lines added afterward)

Several algorithms are capable of removing fisheye distortion by taking an image of a checkerboard and obtaining the fisheye attributes of the camera [21] [28], but some information is still lost from the original image. Still, straightening out the drone images would be a necessary step before merging them to form a panoramic image of the full field.

2.4 DECISION MATRIX

Decision matrices are a common method of choosing between viable alternative implementations for a project [15]. In this case, the four methods discussed earlier in this chapter will be compared by the sum of their weighted scores in several key attributes. These attributes are cost, power, proximity to plants, ease of use, image quality, time required, consistency, and exposure to elements.

Cost: How much would the implementation increase the overall budget for the full project? This is one of the most important aspects to consider. For the handheld camera method, this amount is zero, assuming the consumer already has a cell phone. Mounted cameras cost more to cover a larger area, while the rover and drone are a fixed but large cost. At the time of writing, The DJI Phantom 4 Quadcopter is listed on Amazon for $784, down from its typical price of $1,199 [12], making this the most expensive option. The GDU Byrd Premium 2.0 Quadcopter is listed for $1,094 [13].

Power: How much power does the device consume while recording a full sample? Cell phones are relatively low-power devices, designed to hold charge for a typical day of usage. Ideally, the mounted cameras will only be turned on to record one picture each day, so they will likely consume about as much power as a cell phone. The power requirement for the rover will vary from product to product but will likely be on the same order as the drone, which, as previously mentioned, exhausts a full 5350mAh battery in half an hour.

Proximity to plants: When taking pictures, how close is the camera to the plant? This aspect is much more important for the computer vision block. Handheld cameras and the rover will be closest to the plants, while the mounted cameras vary by camera density. The drone will obviously be furthest from the plants for any reasonable flying altitude.

Ease of use: How simple is this implementation for the end user? The only option which requires any expertise is the drone, since not everyone has quadcopter flying experience. If waypoints could be configured at a lower altitude, this would not be as much of an issue.

Image quality: What is the resolution of the images this implementation produces? Older models of cell phones may have a more archaic camera, but newer ones are at least on par with the kinds found in drones. Unfortunately, drone images often have fisheye distortion as well.

Time required: How long does it take to obtain one full sample? For mounted cameras, this may be a matter of seconds; the rover will take longer but can be fully automated. Flying a drone and taking pictures by hand with a cell phone will take far longer.

Consistency: How similar are images from trial to trial? Consistency is lowest for cell phone pictures and highest for mounted cameras. Flying a drone in a perfectly straight line by hand is also very difficult. Rover performance depends on what kinds of obstacles or rough terrain is present between rows of strawberries.

Exposure to elements: How sensitive is this implementation to poor weather? How frequently will it be exposed? This affects mounted cameras the most, although all other methods will need to be at least water resistant in case of rain.

Table 5 shows the full decision matrix, with a row for each of these considerations listed. Each possible implementation was rated from 1-10 in each aspect, where 1 is the worst score and 10 is the best. The final row shows the weighted sum for each implementation.

Table 5: Data acquisition method decision matrix

Aspect	Weight	Cell phone	Mounted	Rover	Drone
Cost	10	10	3	7	5
Power	3	8	10	6	4
Proximity	9	10	6	9	4
Ease of use	7	7	10	9	2
Image quality	7	6	9	9	9
Time required	8	1	10	8	4
Consistency	6	3	10	8	4

Exposure	5	9	1	7	8
Total (out of 550)		376	392	442	273

By the total scores in Table 5, the rover is the best implementation, since it excels in all of the most important aspects. Close behind is the mounted camera option, then cell phones, then the drone method. Therefore, it would be a good idea to assemble a small rover with cameras on either side and program it to autonomously travel down strawberry rows and take pictures before the next season begins.

Note that these values are all purely subjective, and the true best implementation for the job may not have scored the highest in this evaluation.

2.5 OTHER CONSIDERATIONS

Several problems still exist regardless of implementation. Chief among these is occlusion, when a berry is double counted due to a leaf making it appear to be two objects. Relatedly, several contiguous berries may be counted as only one.

Lighting is another factor. The samples should all be collected during the day to minimize variance between samples. However, poor weather conditions are unavoidable, so the computer vision block should be robust enough to account for lighting variance.

Farmers should take care to minimize red objects on the field prior to testing, such as red tape and faucets. Figure 6 shows an example of red objects seen by the drone which could be mistaken for strawberries by a thresholding algorithm.

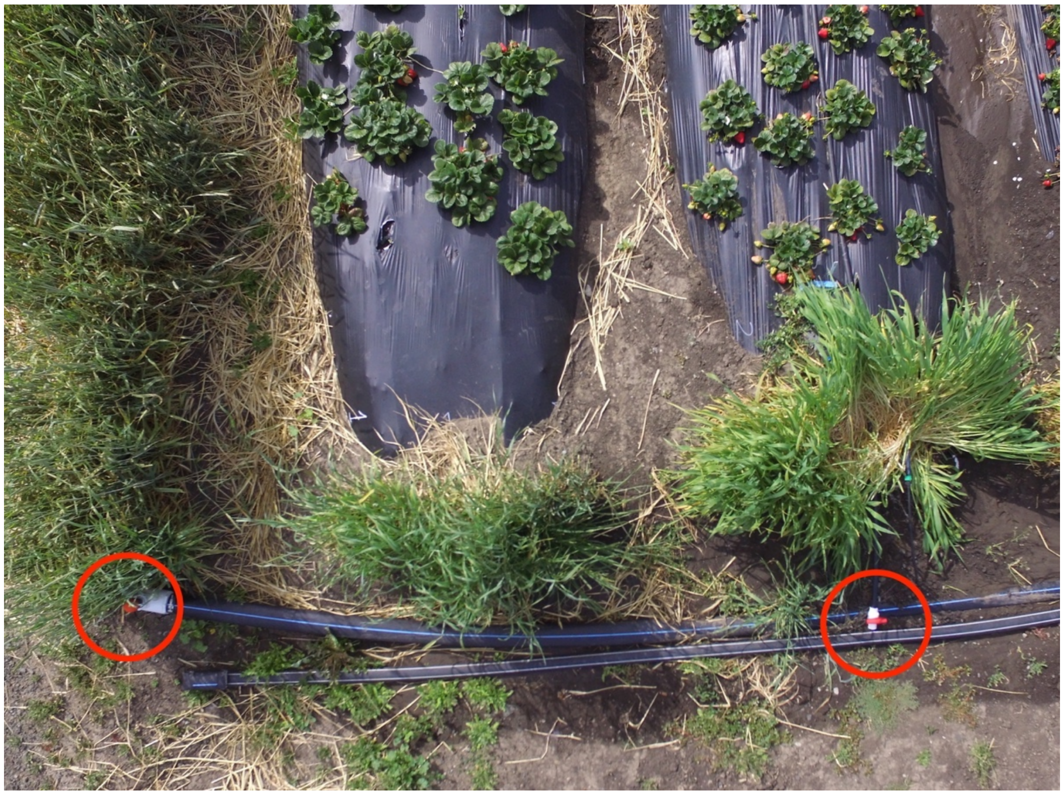


Figure 6: Red objects on the field

Figure 7 shows strawberries which have fallen off the plant and into the dirt between rows. These berries will not be harvested, and therefore should not be counted by the system. If detected, they should be considered false positives, similar to other detected non-strawberry objects such as faucets and tape.

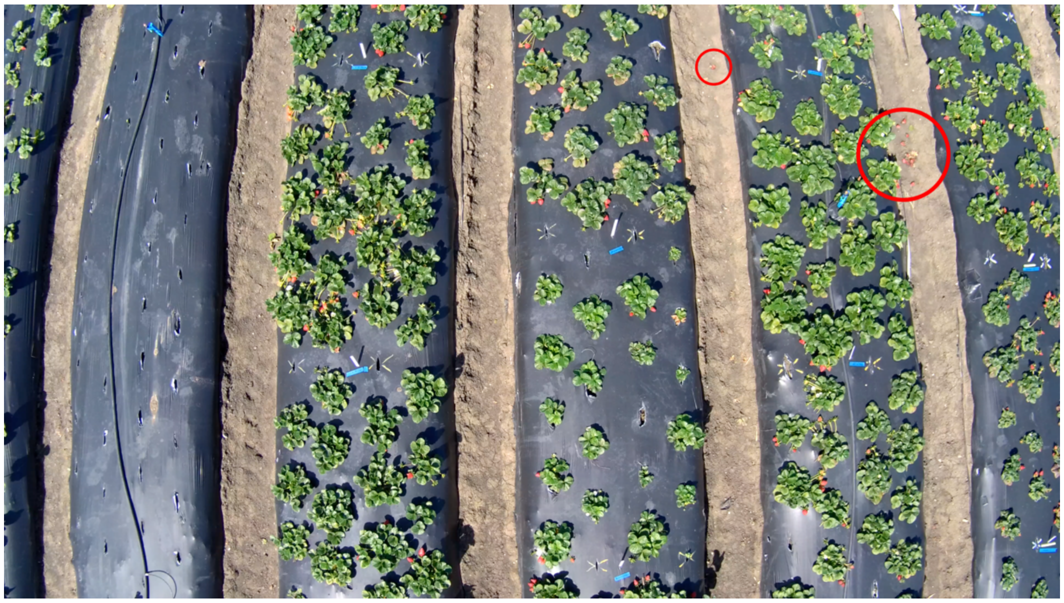


Figure 7: Fallen strawberries on the field

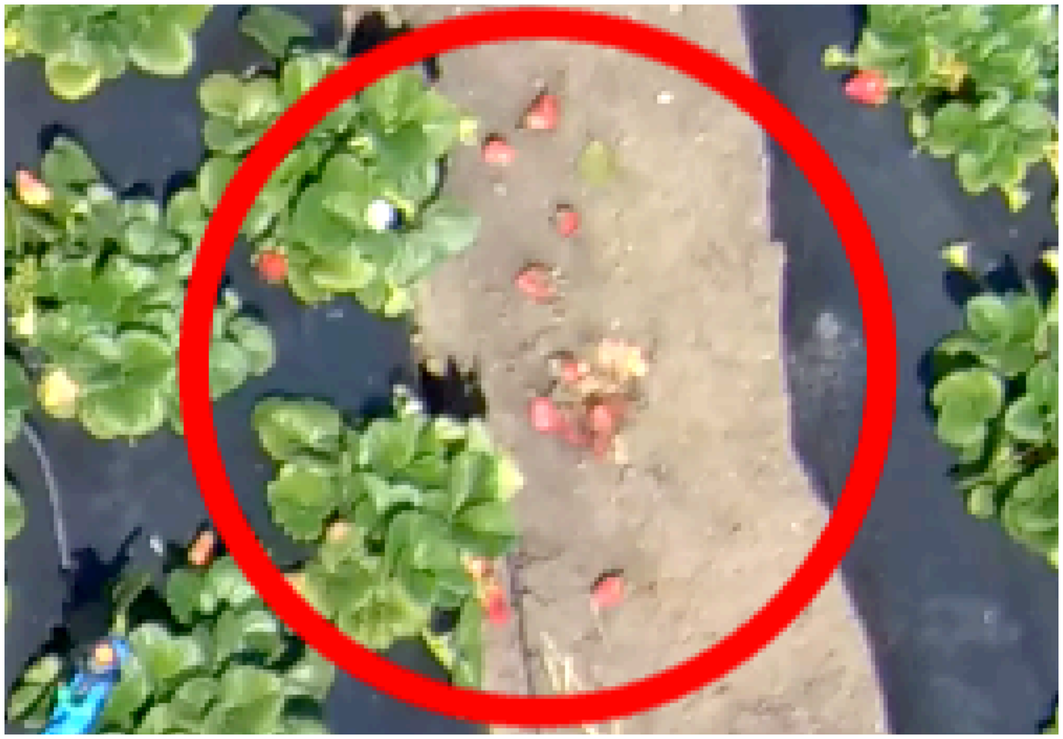


Figure 8: Zoomed version of previous figure

The next chapter will discuss methods of discounting these fallen strawberries using computer vision techniques.

Lastly, it may be useful in the future to track individual fruits through their growth cycle, rather than counting how many fruits are in each stage across the whole field. Selecting an implementation that allows this approach could be helpful in increasing accuracy in future iterations of the system.

3. COMPUTER VISION

The computer vision block must accurately find flowers, green strawberries, and red strawberries in the images supplied by the data acquisition stage. Then, this block tallies how many instances of each stage were seen per day and passes the time series information to the predictor. Detecting red strawberries is a fairly straightforward task, provided there are no other red objects on the field. Occlusion, lighting variance, and contiguous berries all pose challenges as well. Detecting green strawberries and flowers requires more advanced techniques.

3.1 THRESHOLDING

Thresholding is one of the simpler computer vision tools. For a given pixel, if the value is lower than the threshold, it is classified as part of the background; otherwise, it belongs to an object of interest, such as a strawberry. There are numerous methods of determining the best threshold, but the most straightforward is to analyze an image's histogram and find the optimal threshold through trial and error. If possible, looking at separate histograms for background pixels and for strawberry pixels helps to choose a good threshold. Figure 9 shows a training image, and a binary mask used to discriminate the strawberry pixels from background pixels.

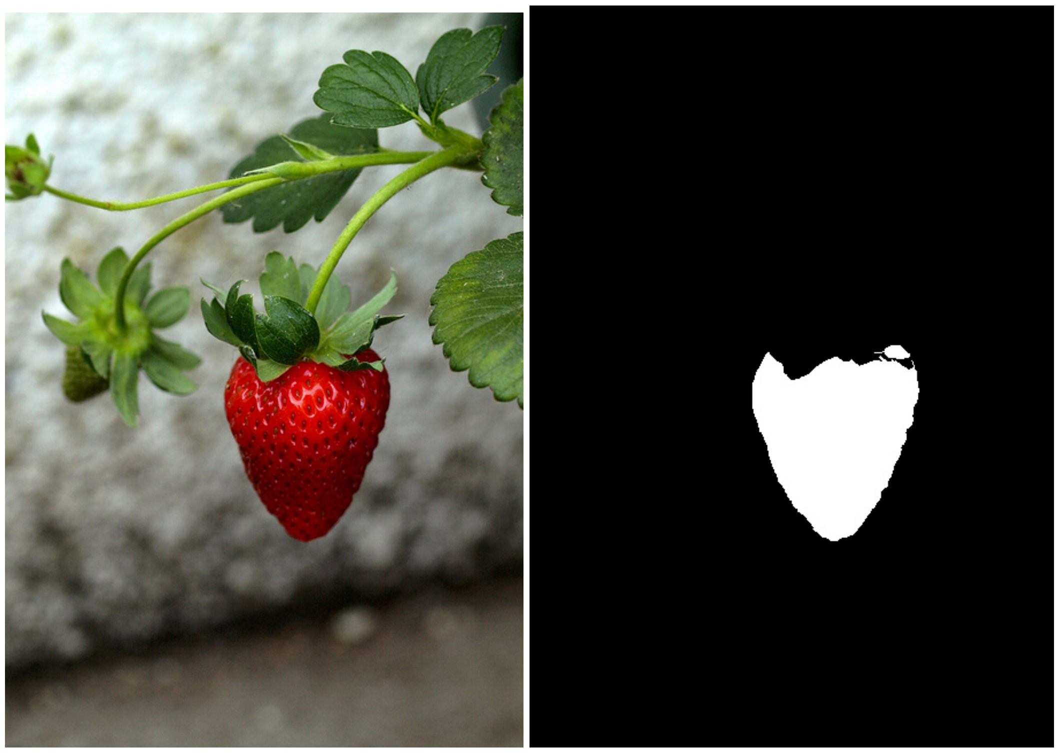


Figure 9: Training image (left) and its binary mask (right). 0 = background, 1 = strawberry.

Masks were also created for 19 other training images. These images and their masks are all available in Appendix B. Next, component layer histograms were analyzed in a number of color spaces. The goal was to find a color space with good discrimination between strawberry and non-strawberry pixels. In Matlab, this was accomplished by iterating through pixels; if the pixel in the mask is white, increment the on-histogram's values corresponding to the input image's pixel data; if not, increment the off-histogram's values. This test was performed for the following color spaces: RGB, nRGB, HSV, YCbCr, and L*a*b*. Matlab code is available in Appendix A.

RGB: Since each pixel on a computer monitor consists of a red, green, and blue LED, most color images store color data by saving the red, green, and blue level at each pixel, as an integer from 0 to 255 (True color, 24-bit). Figures 10-12 show the normalized probability distributions for each layer, for strawberry and background pixels.

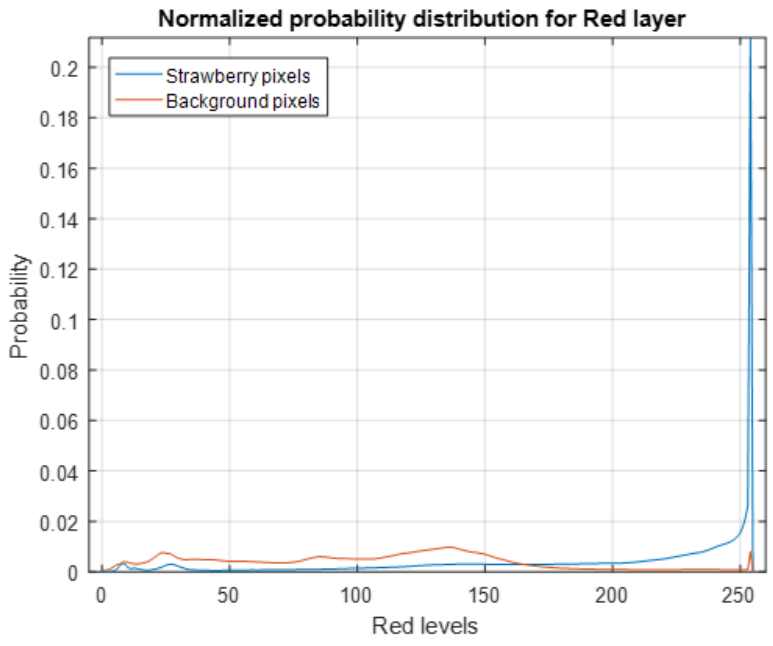


Figure 10: Normalized probability distribution for red layer, 255 bins

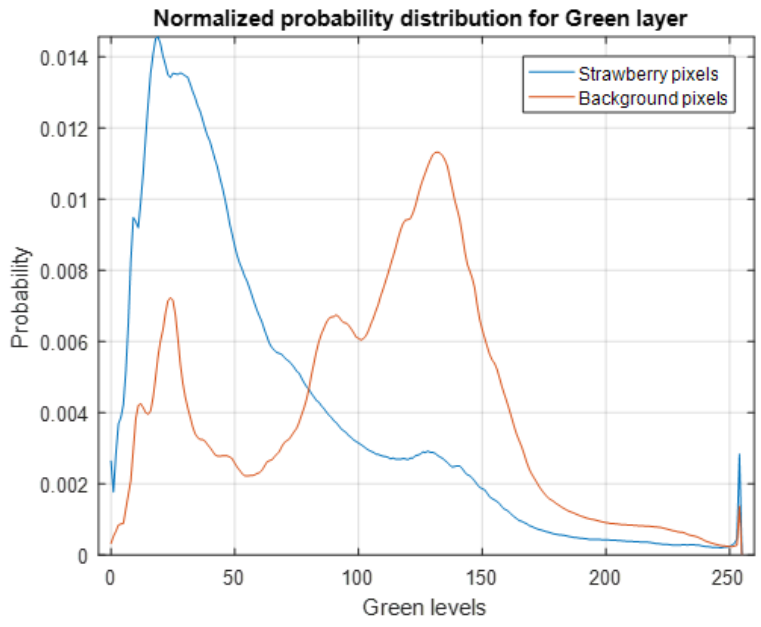


Figure 11: Normalized probability distribution for green layer, 255 bins

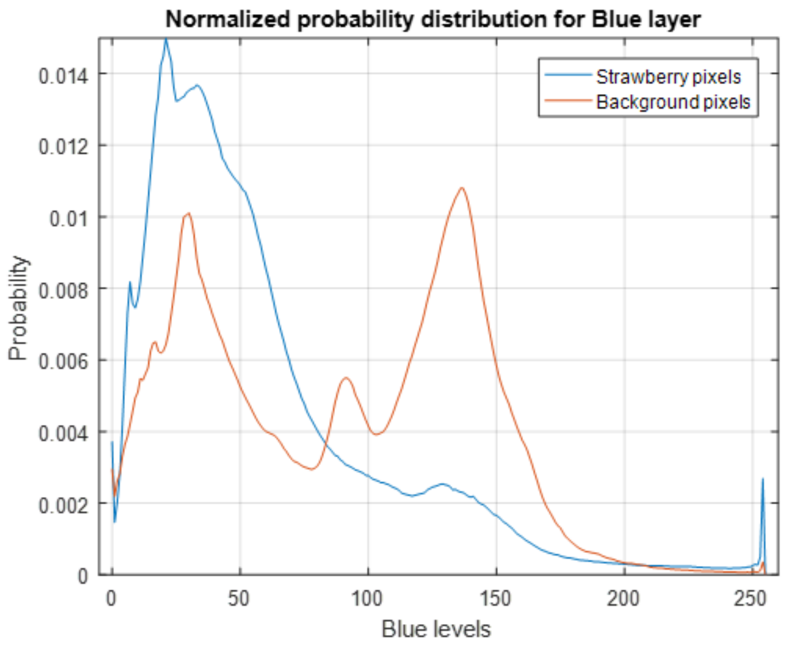


Figure 12: Normalized probability distribution for blue layer, 255 bins

As expected, the strawberry pixels have exceptionally high red values, and much lower blue and green values. The reason for the spike at 255 in each layer is the fact that white pixels are encoded as (255,255,255). This is also why brighter pixels have higher blue and green values, which makes those layers poor choices for discriminating between strawberry and background pixels.

nRGB: This color space seeks to correct the fact that brighter pixels have stronger values in non-dominant component layers in RGB. RGB is converted to nRGB by the following set of equations:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

By normalizing the component layer at each pixel by the sum of red, blue, and green at that pixel, we get the ratio between the three colors, but lose lighting information. Figures 13-15 show the probability distributions for normalized red, green, and blue, again using 255 bins.

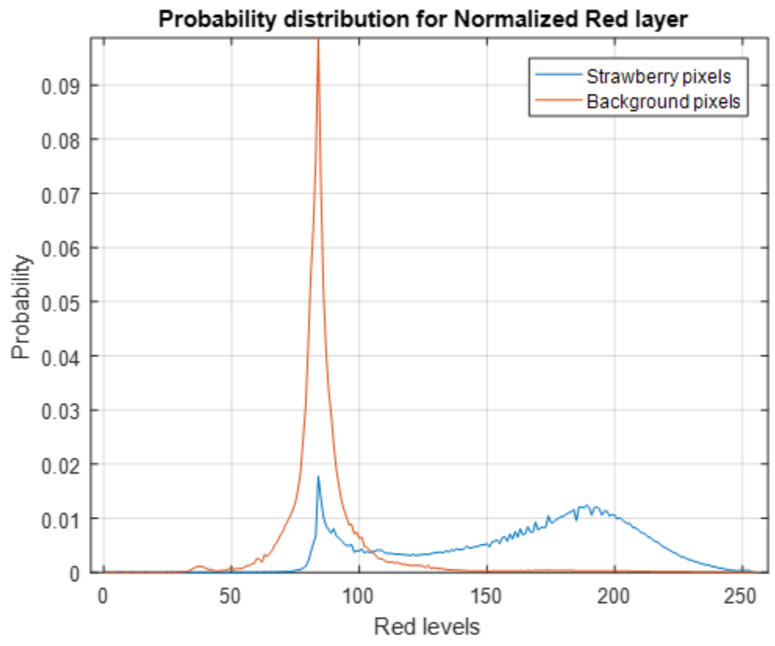


Figure 13: Probability distribution for normalized red layer, 255 bins

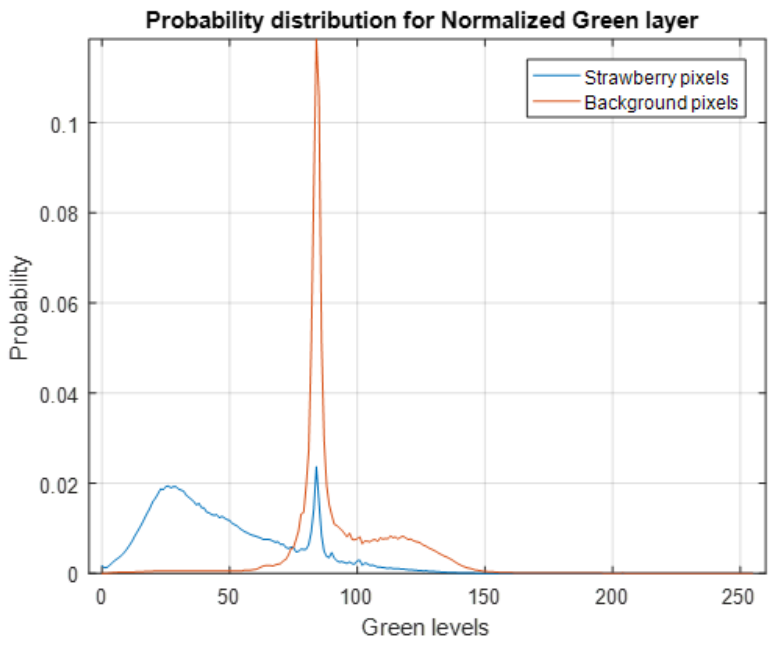


Figure 14: Probability distribution for normalized green layer, 255 bins

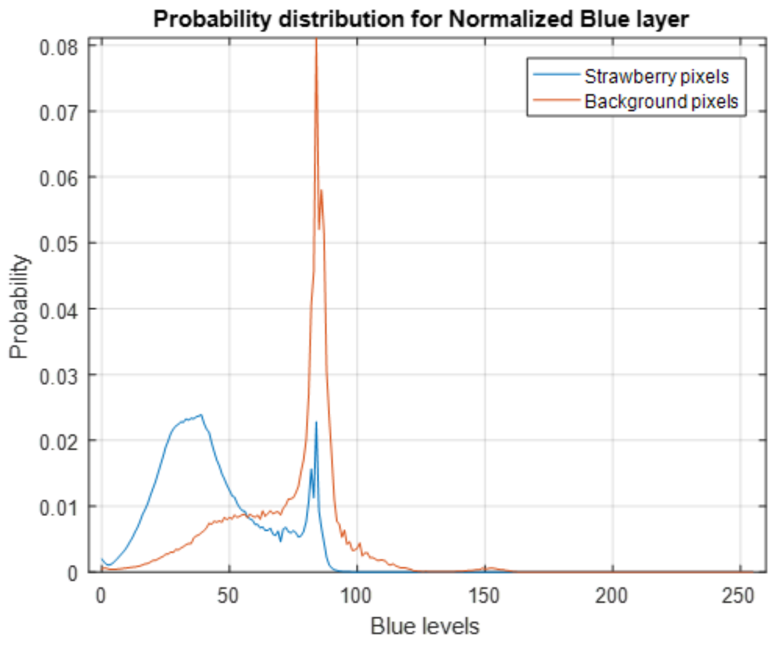


Figure 15: Probability distribution for normalized blue layer, 255 bins

Unsurprisingly, the normalized red layer shows strong discrimination between strawberry and background pixels, and the normalized green layer shows that strawberry pixels often have much lower green values than background pixels. Any grayscale value (black to white spectrum) in the original image will show up at ⅓ * 255 = 85, since in those cases the red, green, and blue values in RGB are all nearly the same. This is why all three layers show significant spikes at 85.

HSV: Since grayscale values clump together in nRGB, it would be useful to abstract the gray level to one dimension. HSV (hue, saturation, value) accomplishes this by placing hue in one polar dimension (on a continuous scale from 0 to 360 degrees, where red = 0, green = 120, and blue = 240), saturation in another dimension (strength of color; low saturation is a grayscale color, high saturation is a vivid hue), and value in a third dimension (black to white value). Figure 16 shows the HSV color space.

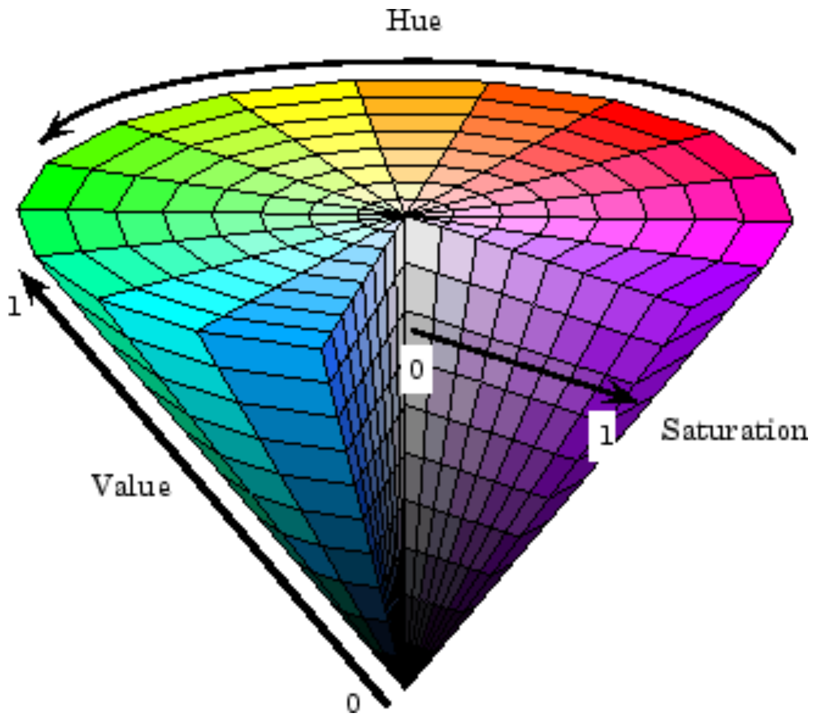


Figure 16: HSV color space [14]

Assuming RGB values are already represented from 0 to 1, the equations for converting RGB values to HSV are:

$$C_{max} = \max(R, G, B),\ C_{min} = \min\,(R, G, B),\ \Delta = C_{max} - C_{min}$$

$$H = \begin{cases} 60^\circ \times \left(\dfrac{G-B}{\Delta} \% 6\right), C_{max} = R \\ 60^\circ \times \left(\dfrac{B-R}{\Delta} + 2\right), C_{max} = G \\ 60^\circ \times \left(\dfrac{R-G}{\Delta} + 4\right), C_{max} = B \end{cases}$$

$$S = \begin{cases} 0, C_{max} = 0 \\ \dfrac{\Delta}{C_{max}}, C_{max} = 0 \end{cases}$$

Note that % in this case represents the modulo operation. Figures 17-19 show the probability distributions for hue, saturation, and value for the twenty test images, using 360 bins for hue, and 255 bins each for value and saturation.

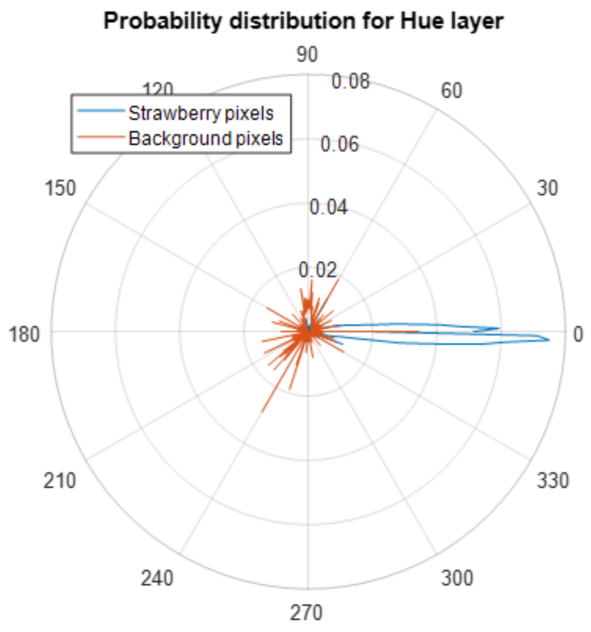


Figure 17: Probability distribution for hue layer, 360 bins. Shown on a polar plot for continuous range.

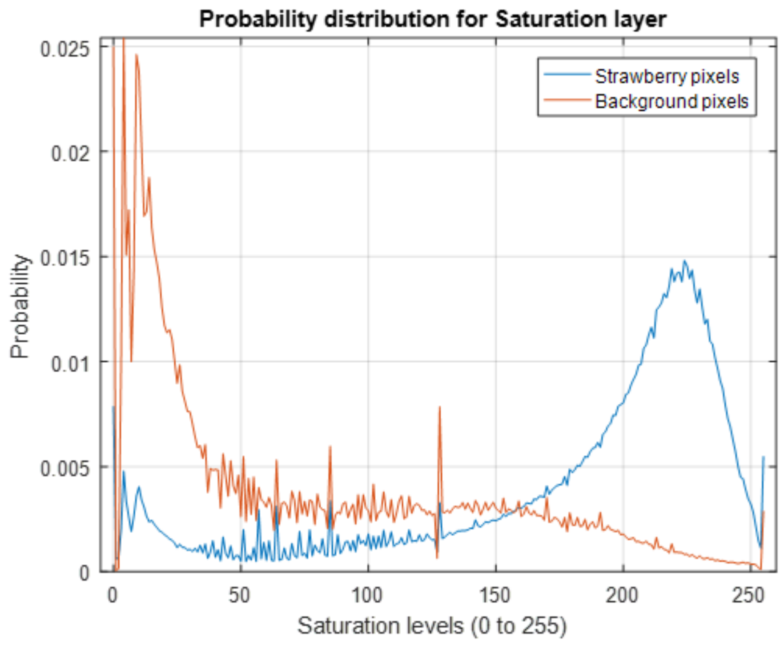


Figure 18: Probability distribution for saturation layer, 255 bins

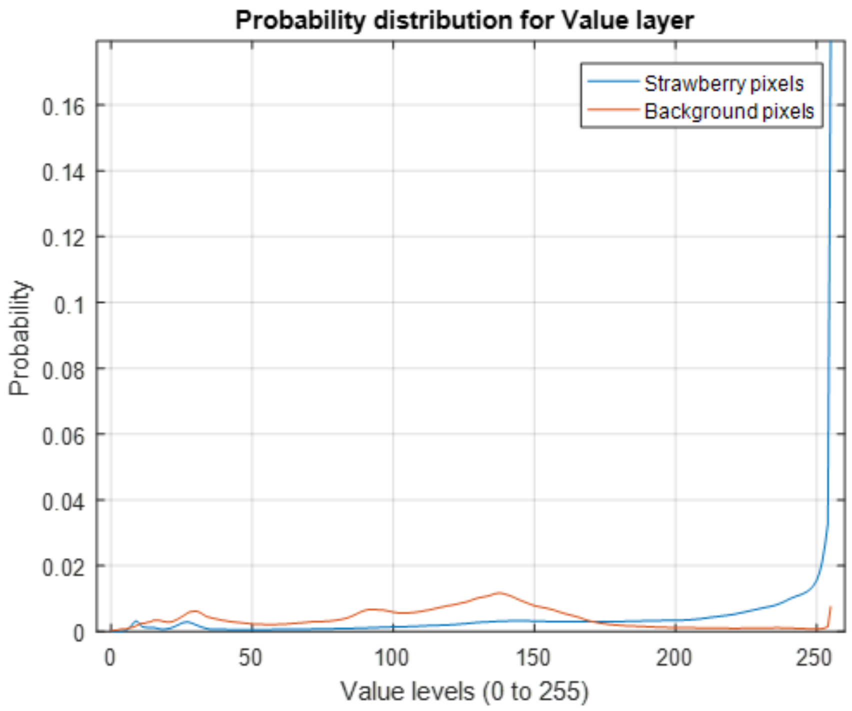


Figure 19: Probability distribution for value layer, 255 bins

Matlab does not allow for as much control of the view with polar plots as with rectangular plots, but the hue plot still shows that many strawberry pixels have a hue about 0. They also have higher saturation and much higher value than background pixels.

YCbCr: In the YCbCr color space, Y corresponds to brightness (luma), Cb is blue (from RGB) minus the luma, and Cr is red minus the luma. Therefore, there is one component for brightness, and two for color. Figures 20-22 show the probability distributions for Y, Cb, and Cr, using 255 bins.

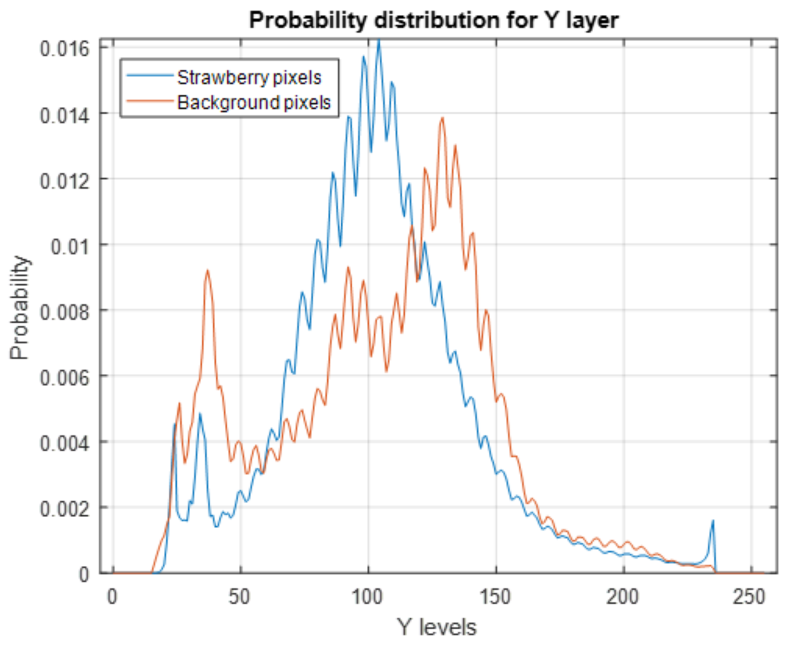


Figure 20: Probability distribution for Y layer, 255 bins

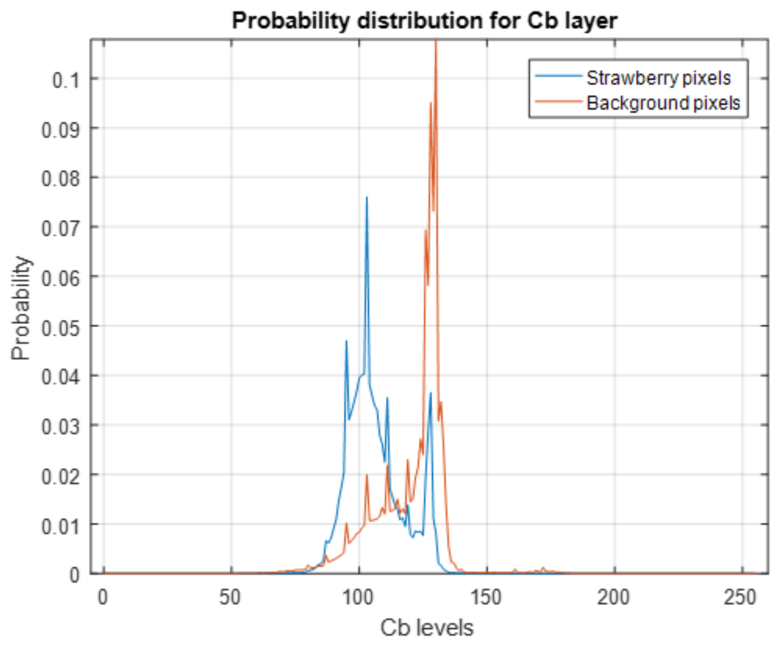


Figure 21: Probability distribution for Cb layer, 255 bins

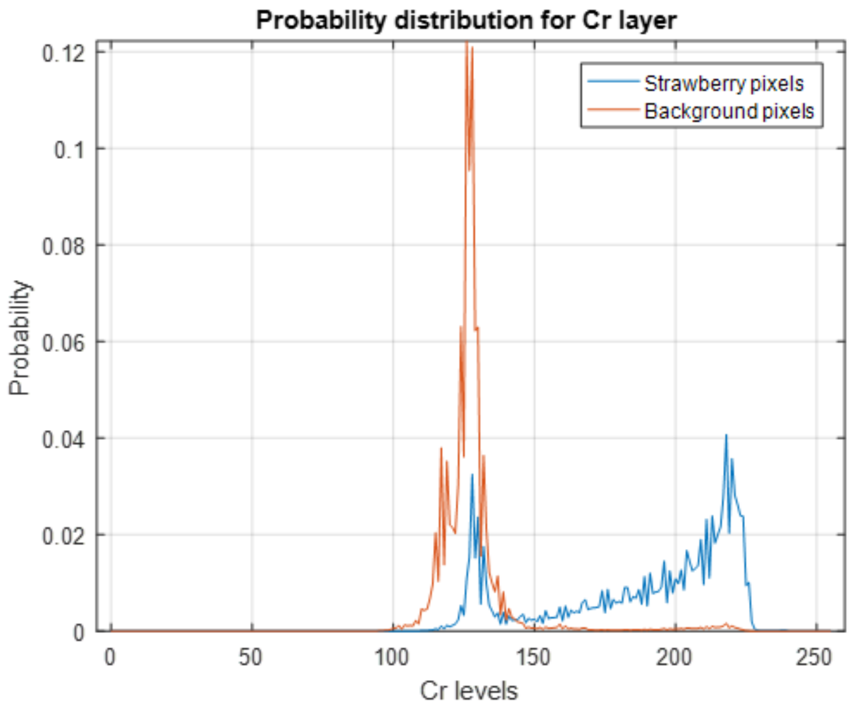


Figure 22: Probability distribution for Cr layer, 255 bins

Once again, Cr shows strong discrimination, but simply thresholding in this dimension would lose out on a lot of strawberry pixels with Cr in the range of 128 ± 10. The other two component layers do not show a strong separation between strawberry and background pixels.

L*a*b*: Finally, the Lab color space is based on all colors that human eyes can perceive. Similar to YCbCr, L is lightness, a is a color range from green to magenta, and b is a color range from blue to yellow. Figure 23 shows the portion of the LAB color space which can be represented within the RGB gamut, and therefore displayed on a computer monitor.

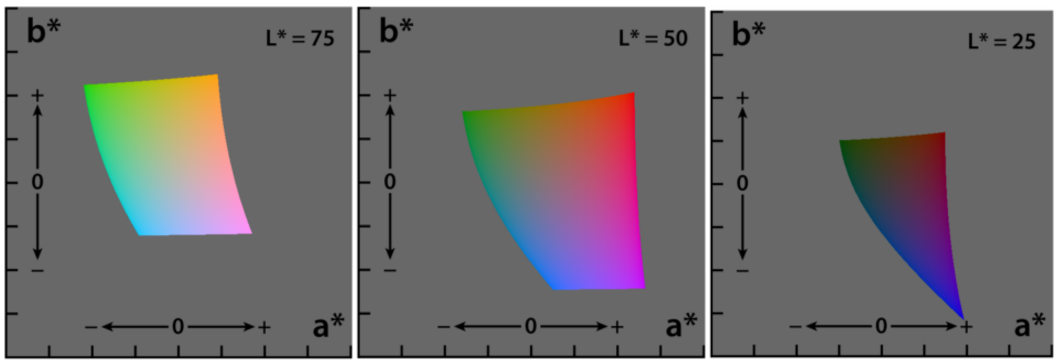


Figure 23: The portion of the Lab color space which can be displayed on computer monitors. The color space is sliced at L = 75 (left), 50 (middle), and 25 (right).

Using the built-in Matlab function for converting rgb to Lab, L has a range from 0 to 100, while a and b range from -100 to 100. Figures 24-26 show the probability distributions for L, a, and b, with 200 bins each.

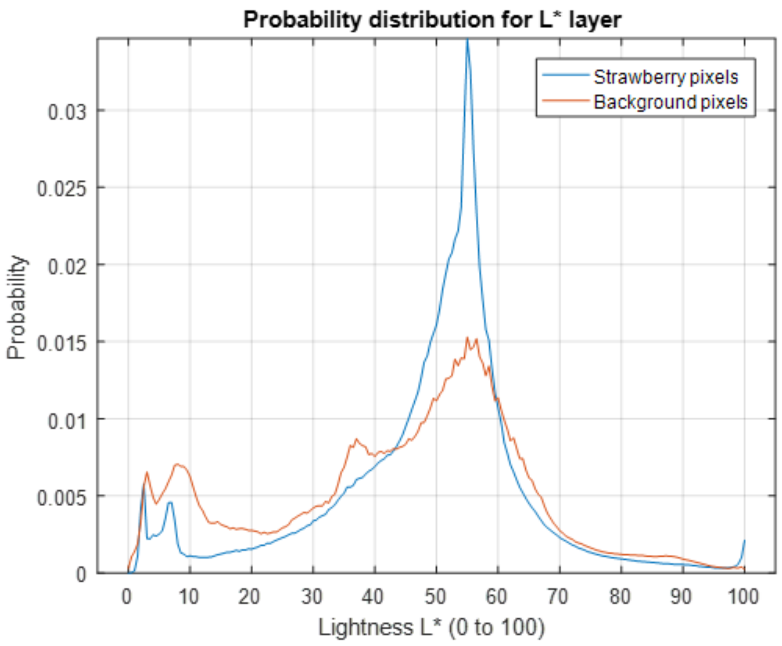


Figure 24: Probability distribution for L layer, 200 bins

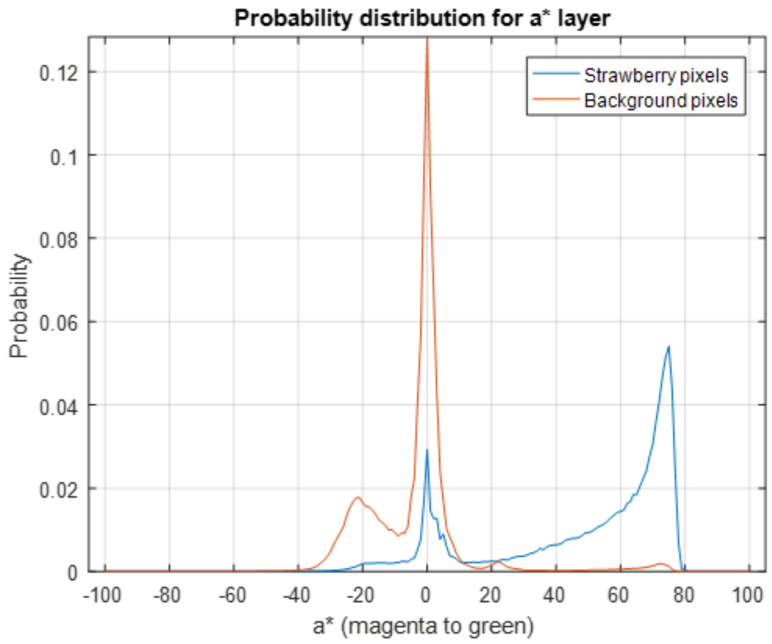


Figure 25: Probability distribution for a layer, 200 bins

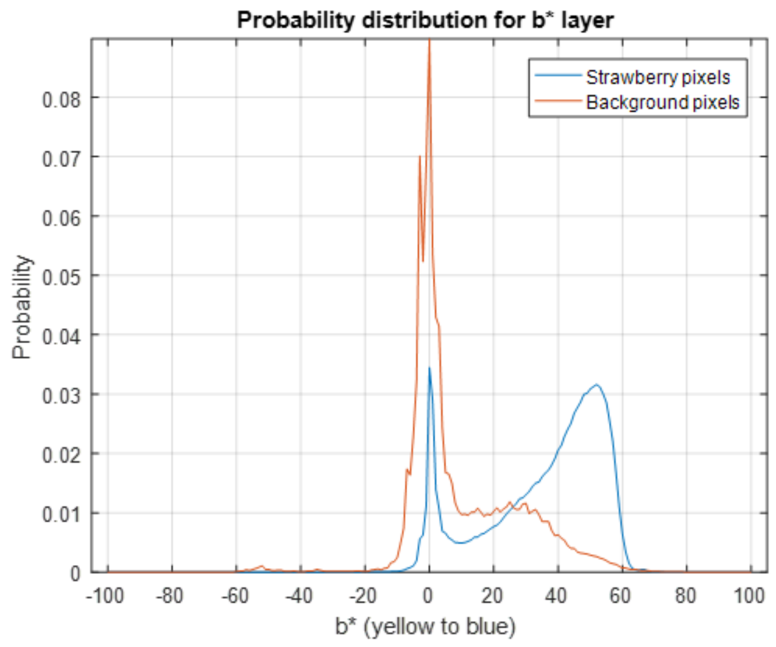


Figure 26: Probability distribution for b layer, 200 bins

Lab shows very good discrimination in the a dimension, with many strawberry pixels located towards the magenta side, and decent discrimination in the b dimension, with many strawberry pixels located towards the yellow side.

Overall, the most promising features for discrimination were found in the red and green layers in nRGB, with the a and b layers in LAB taking second place. In both cases, a majority of the pixels represented in the strawberry pixel probability distribution were located in ranges where background pixels were not as likely. Perfect segmentation using only color data is of course impossible with this dataset, given the number of white pixels in the strawberries and some backgrounds, but one can expect much stronger results by choosing the right color space. It is also possible that a ratio of features, such as R/G, may provide even better segmentation.

These results are not just useful for thresholding, either. Many computer vision techniques rely on segmentation in the "parameter space" rather than in the spatial or image domain, so choosing parameters such as nRGB's Red and Green layers, which already have good separation in the training images, serves as a good start for performing more advanced techniques such as K-means, mean shift, etc.

To test the potential of thresholding in nRGB and Lab color spaces, three images of the field were obtained at different heights: 3m, 5m, and 7m above ground. Figures 28, 29, and 30 show these images.

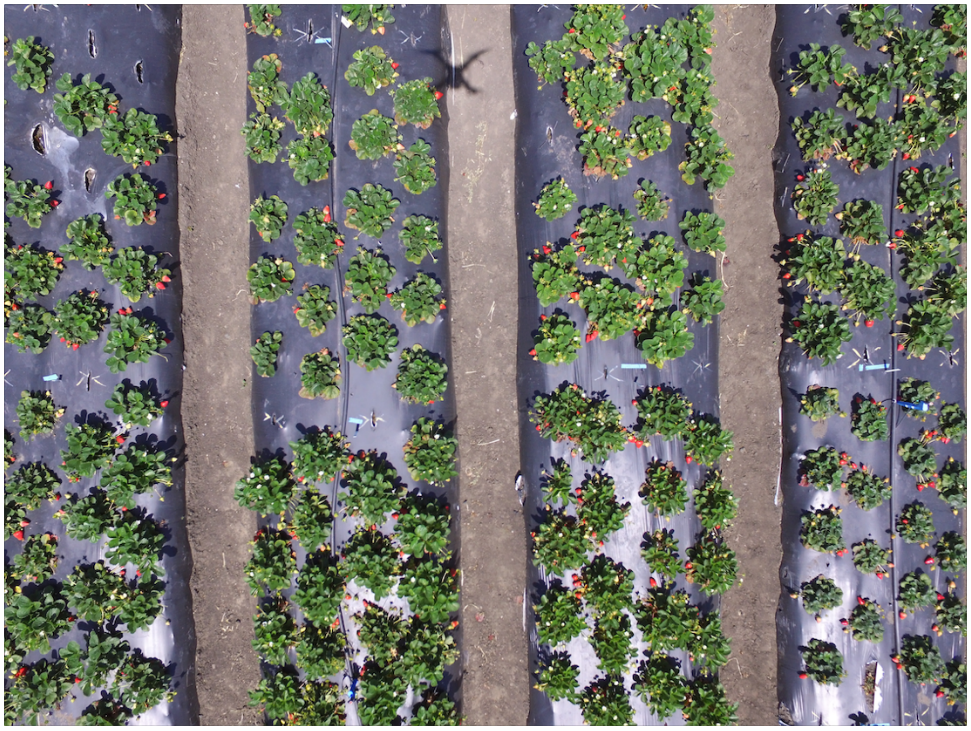


Figure 27: Image taken at 3m

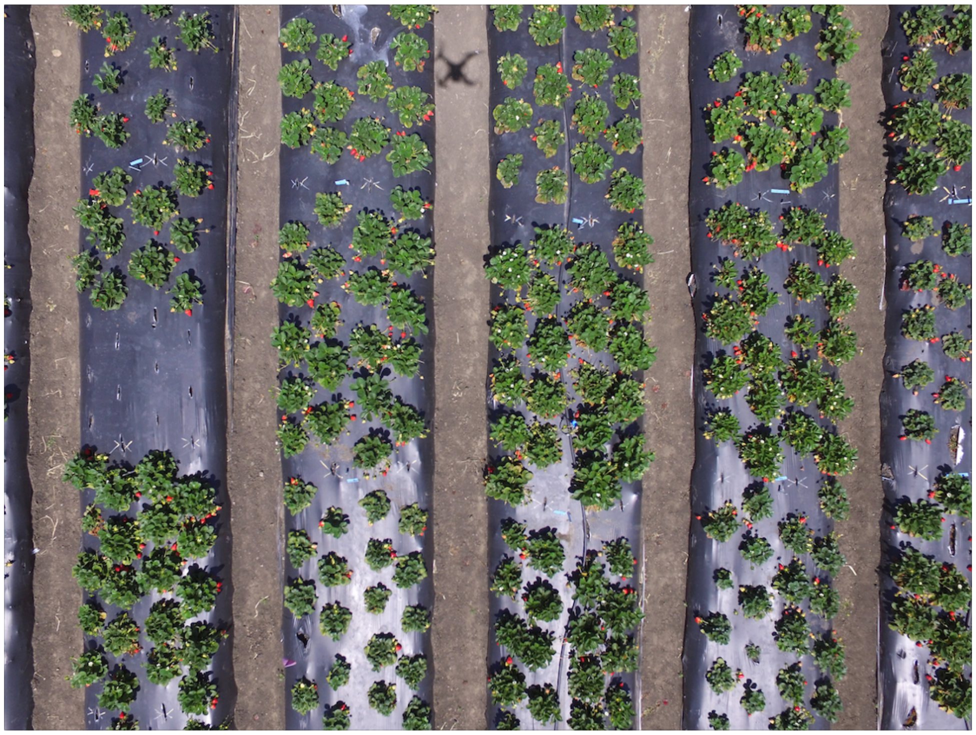


Figure 28: Image taken at 5m

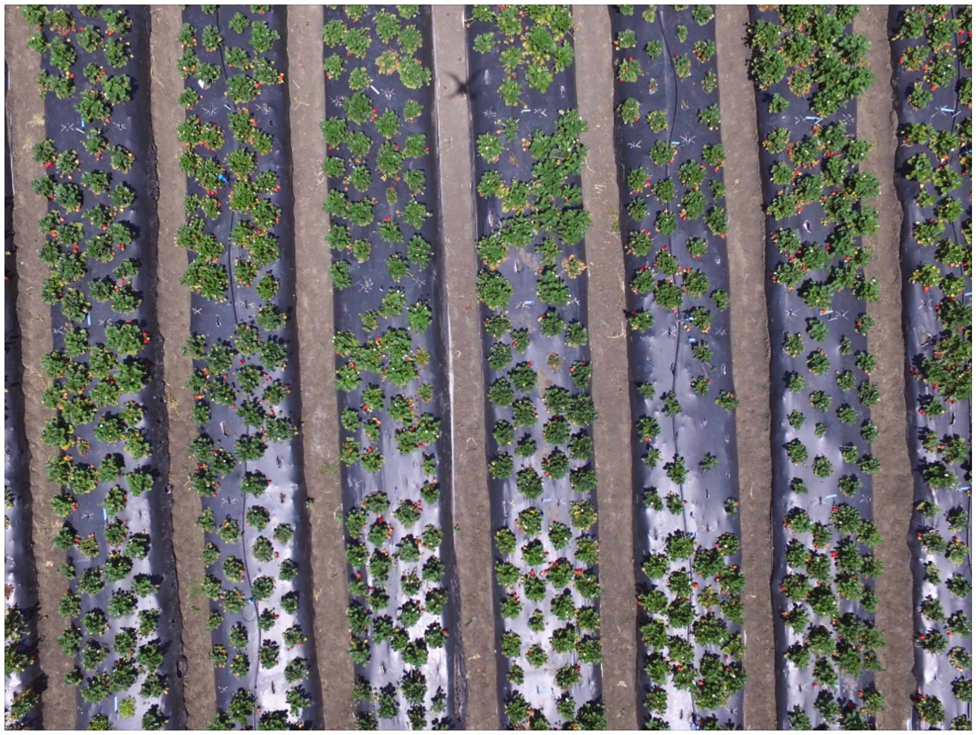


Figure 29: Image taken at 7m

Each image was first converted into nRGB using the conversion method discussed earlier. Then, the image is thresholded by the normalized red and normalized green layers. If the red value (between 0 and 1) is greater than 0.45, and the green value (between 0 and 1) is less than 0.4, then this pixel belongs to a strawberry, and is assigned a 1, or white, value. All others are assigned 0, or black. Figure 31 shows this image for the 3-meter case.

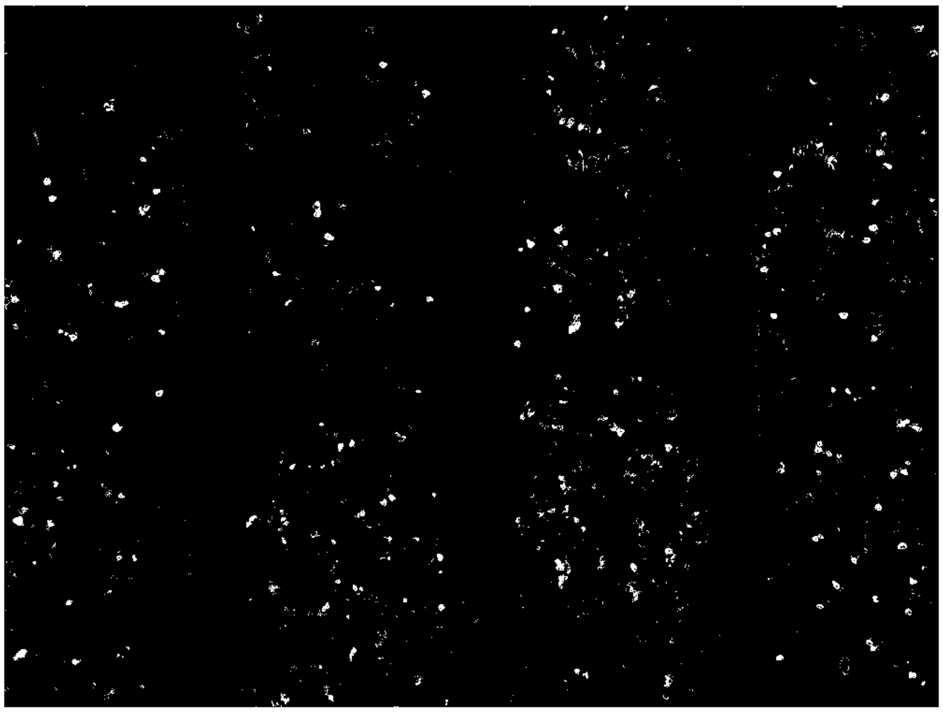


Figure 30: Thresholded 3m image prior to morphological operations

Since there are a lot of stray pixels and small objects in this image, morphological closing was used to join together nearby shapes which may have become separated, and morphological opening was used to remove small objects. A 5-pixel disk-shaped structuring element was used for 3m, and a 3-pixel disk was used for 5m and 7m.

Figures 32, 33, and 34 show each of the thresholded images after morphological operations are performed.

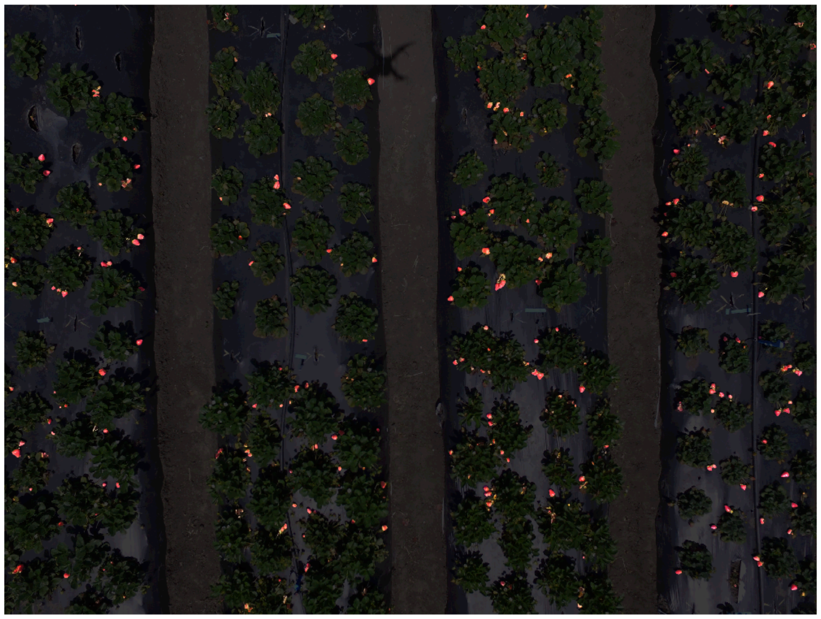


Figure 31: Result of thresholding in nRGB at 3m, overlaid on original image

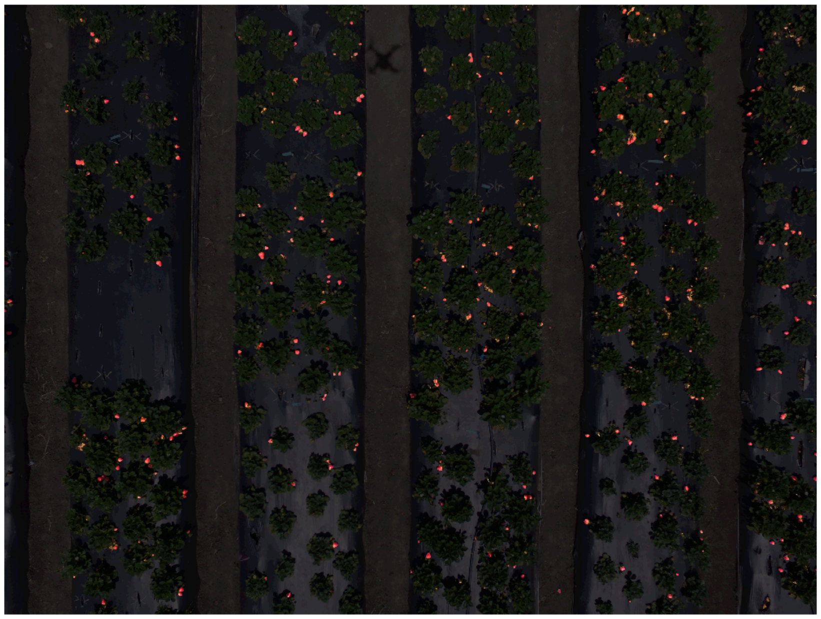


Figure 32: Result of thresholding in nRGB at 5m, overlaid on original image

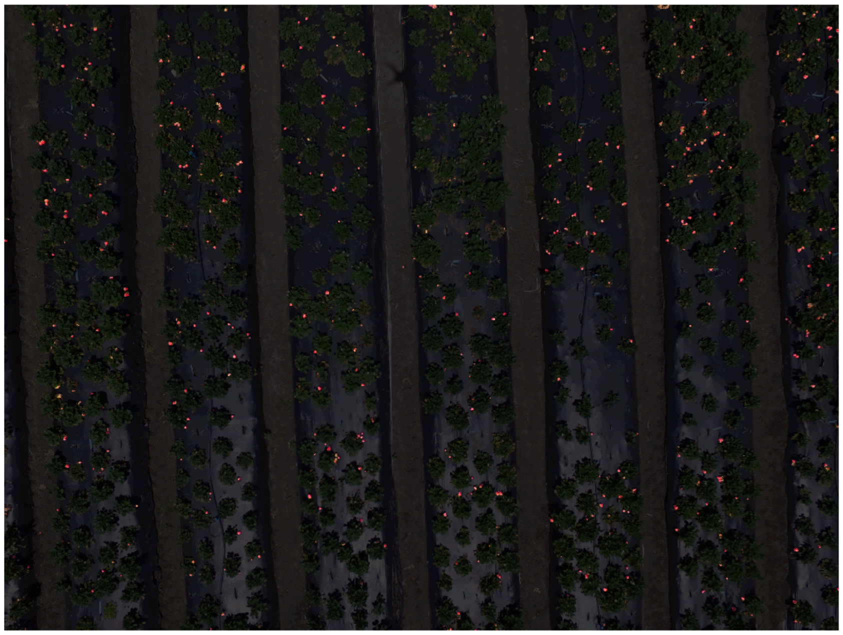


Figure 33: Result of thresholding in nRGB at 7m, overlaid on original image

Lastly, Matlab counted the number of white regions in each image. There were 267 detected objects in the 3-meter case, 431 in the 5-meter case, and 539 in the 7-meter case.

The actual number of strawberries was then counted by hand for comparison. False positive values were counted by comparing objects in the mask against the original image. False negative values were counted by checking whether red strawberries in the original image were represented by an object in the mask. Table 6 summarizes the results of the thresholding test. Error rate was calculated with the following equation:

$$Error\ rate = \frac{Objects\ detected - (True\ Positive + False\ Negative)}{True\ Positive + False\ Negative}$$

Table 6: nRGB thresholding results summary

Altitude	Objects detected from nRGB thresholding	True Positive	False Positive	False Negative	Error Rate (%)
3m	267	189	78	40	16.6%
5m	431	289	142	34	33.4%
7m	539	438	101	49	10.7%

These results seem somewhat counterintuitive, since it is expected that error rate should rise with distance. However, the sample size was somewhat small for this problem, and all values for this script were chosen by hand, so performance could vary greatly at different distances. In the future, an automated process for finding optimal threshold values and structuring elements would likely have better results.

As expected, the nRGB color space performs even better at finding strawberries in the plant's shade than a human observing the image would, due to its lighting invariance. Conversely, glare from the sun can sometimes result in false negatives. Almost all false negatives occurred whenever fruits were touching, as shown in Figure 35 below. In this case, only two objects are detected where there should be three.

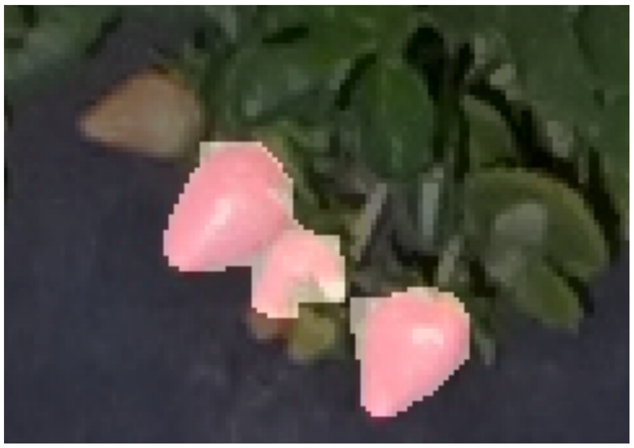


Figure 34: Two strawberries touching are not counted as separate objects in this case.

This is an especially difficult problem to solve with thresholding alone. The human visual system relies on the expected shape of the fruit and small variations in shading in order to tell whether there is one strawberry or more. Small objects, such as berries mostly occluded by leaves, were sometimes undetected by the system, which accounted for several other false negatives. It should be noted that in the 3m and 5m case, however, the only false negatives came from contiguous strawberries; all objects which a human observer would classify as a red strawberry were discovered by the system.

The vast majority of false positives occurred when dead leaves became red-colored, sharing a very similar hue to the strawberries. Figure 36 below shows an example of a red leaf being detected as two objects.

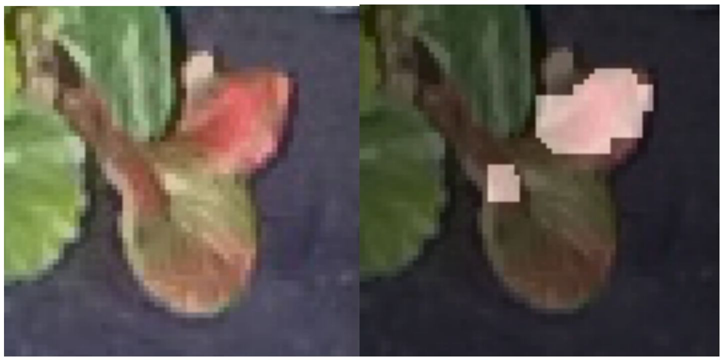


Figure 35: Red leaf causes two false positives

At 7 meters especially, it becomes difficult even for a human to recognize objects of interest with any confidence. Figure 37 below is one such case.

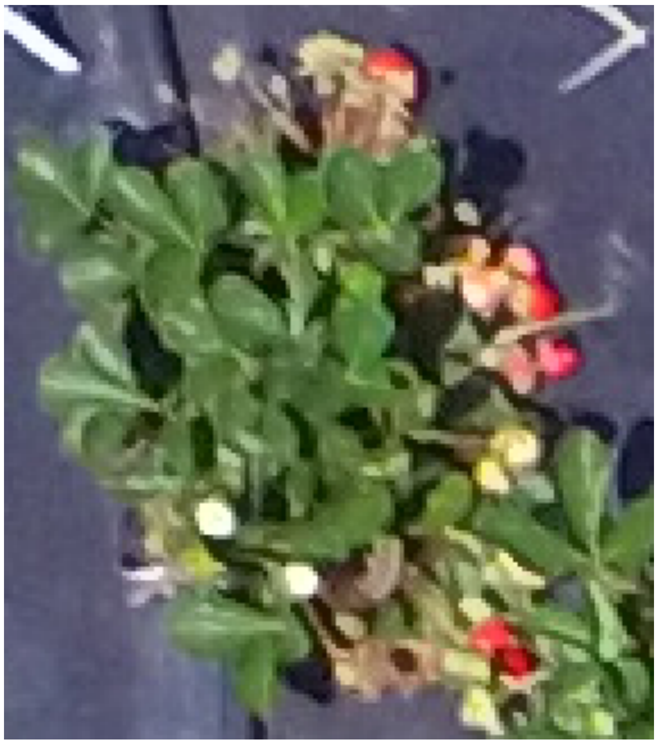


Figure 36: Unclear excerpt due to high altitude

The number of red strawberries in this excerpt is somewhere between 3 and 8, but the pixelation makes it difficult to distinguish more reliably than this estimate. This is proof that 7 meters is too high to reliably classify even red strawberries.

For this test, pink strawberries (fruits which have have not fully transitioned from green to ripe, and may have green patches remaining, or a lighter color) were often detected as red strawberries. This is an unavoidable error for thresholding, and thus was not counted as a false positive.

The thresholding test was also performed in the Lab color space. However, as shown in Figure 38 below, the dirt rows were highlighted along with the strawberries, so this was not as successful for thresholding as the nRGB color space. Both Matlab scripts are available in Appendix A, as well as the image files used (“7m 5-4-18”, “5m 5-4-18”, and “3m 5-4-18”).

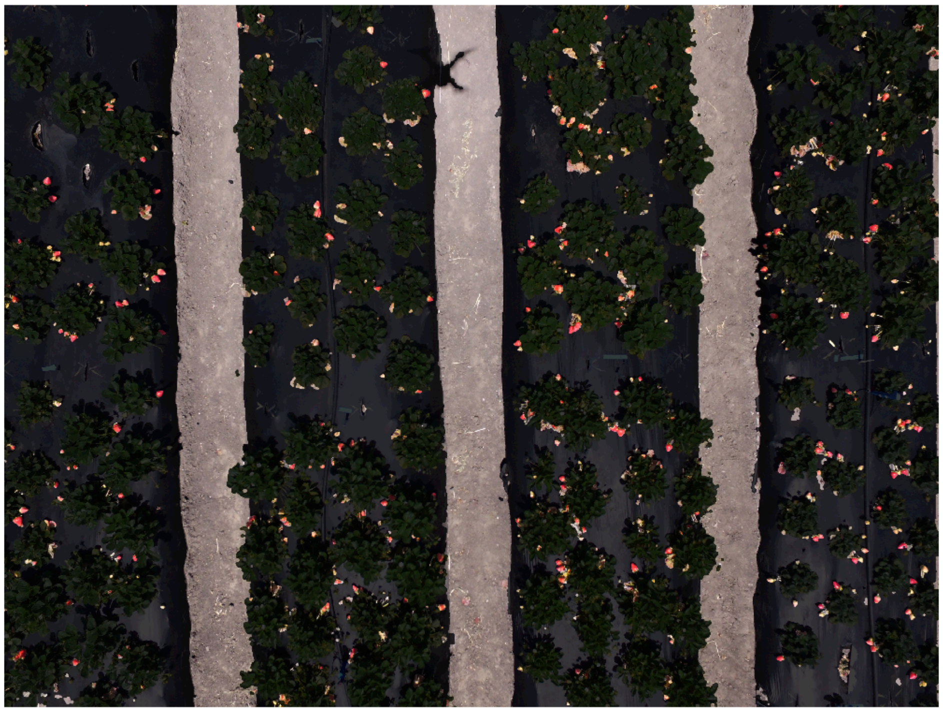


Figure 37: Result of thresholding in Lab at 3m, overlaid on original image. Note that the dirt rows are counted as objects of interest

Ultimately, this test alone cannot determine the maximum allowable altitude for obtaining data. Red strawberries are much simpler to detect than green strawberries and flowers due to their color, which makes simple thresholding a viable solution for these objects. For other objects, Chern concluded in his thesis that more advanced features such as texture and shape perform better, and the camera should probably be much closer than 7 or even 5 meters to reliably obtain this information.

3.2 DRONE IMAGE OBSTACLES

Images taken via drone present a few unique challenges compared to other methods. Drone cameras often come with a significant level of fisheye distortion. After this is removed, images must be merged into one panorama/mosaic that can be analyzed for objects of interest.

Matlab has a tutorial for correcting fisheye distortion using a training set of images of a checkerboard pattern held at various depths and angles. The algorithm looks for corner points in each checkerboard image to model a three-dimensional transform between the camera's perceived points, and the points as they exist in reality [21]. Then, Matlab performs the inverse of the transform to correct a given image. Figure 39 below shows this process graphically.

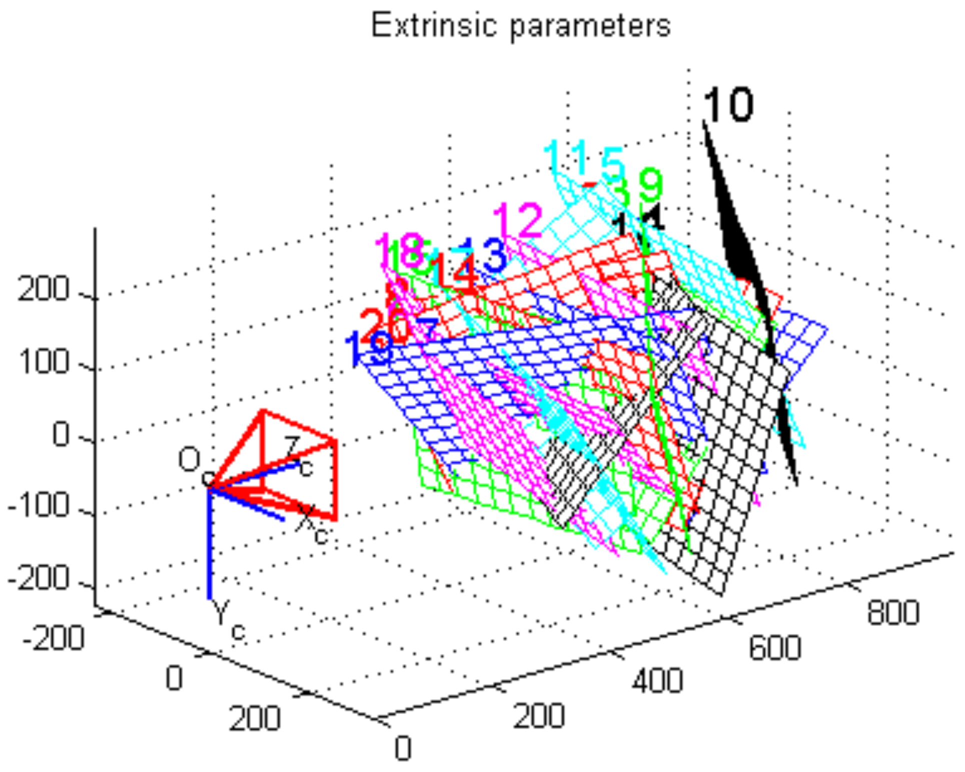


Figure 38: Placement of checkerboard pattern in training images, as determined by finding corner points in each image. [28]

Executing the script project_fisheye.m from Appendix A (largely adapted from Mathworks' fisheye calibration tutorial) in the same directory as the fisheye distortion calibration dataset from Appendix D shows how a checkerboard image with fisheye distortion can be corrected using functions available in Matlab's Computer Vision toolbox. This approach takes a long time to compute, as Matlab looks for corner points in each of the checkerboard images and creates a 3D grid to estimate fisheye distortion parameters, but the output image shows minimal distortion. Figures 40 and 41 show the input image and the undistorted version, and Figure 42 shows the full image without cropping to the largest valid rectangle. Note the unnecessary information located in the corners of the original image.

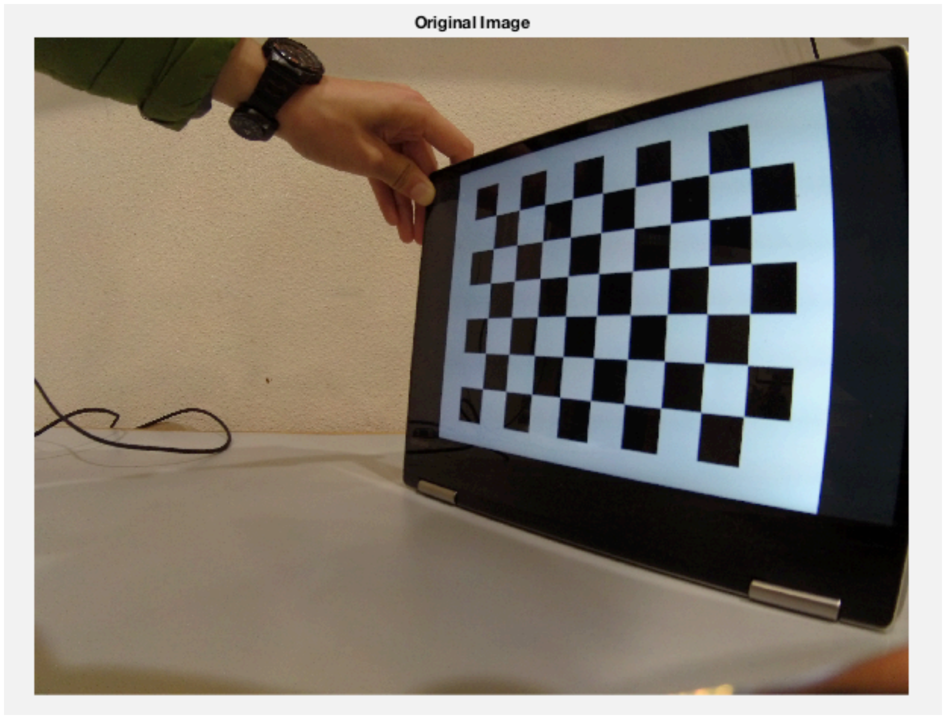


Figure 39: Original checkerboard image with fisheye distortion. Note severe rounding towards the right edge of the image.

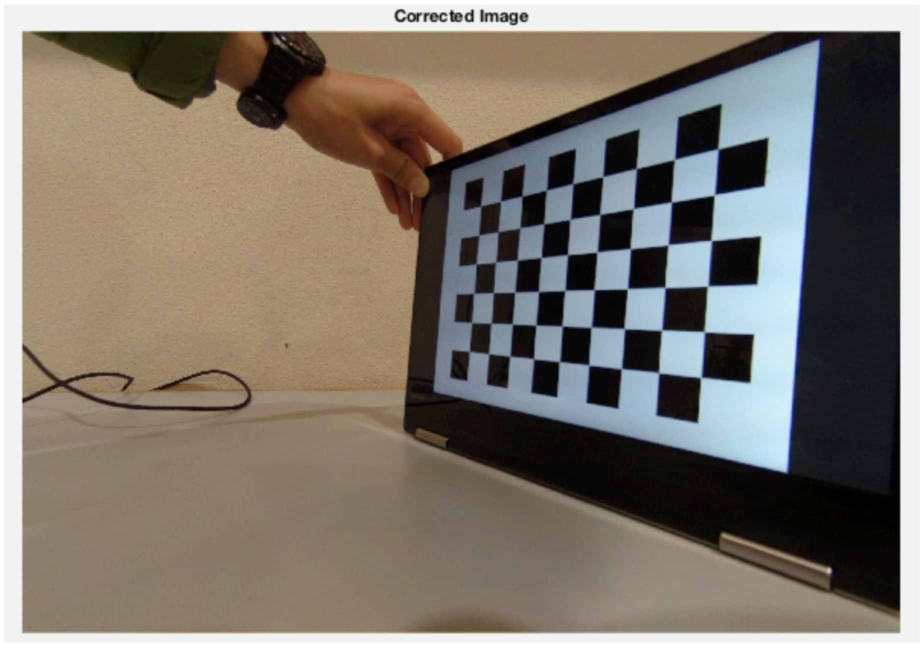


Figure 40: Corrected image with no fisheye distortion. Note that the curves in the previous figure have become straight lines.

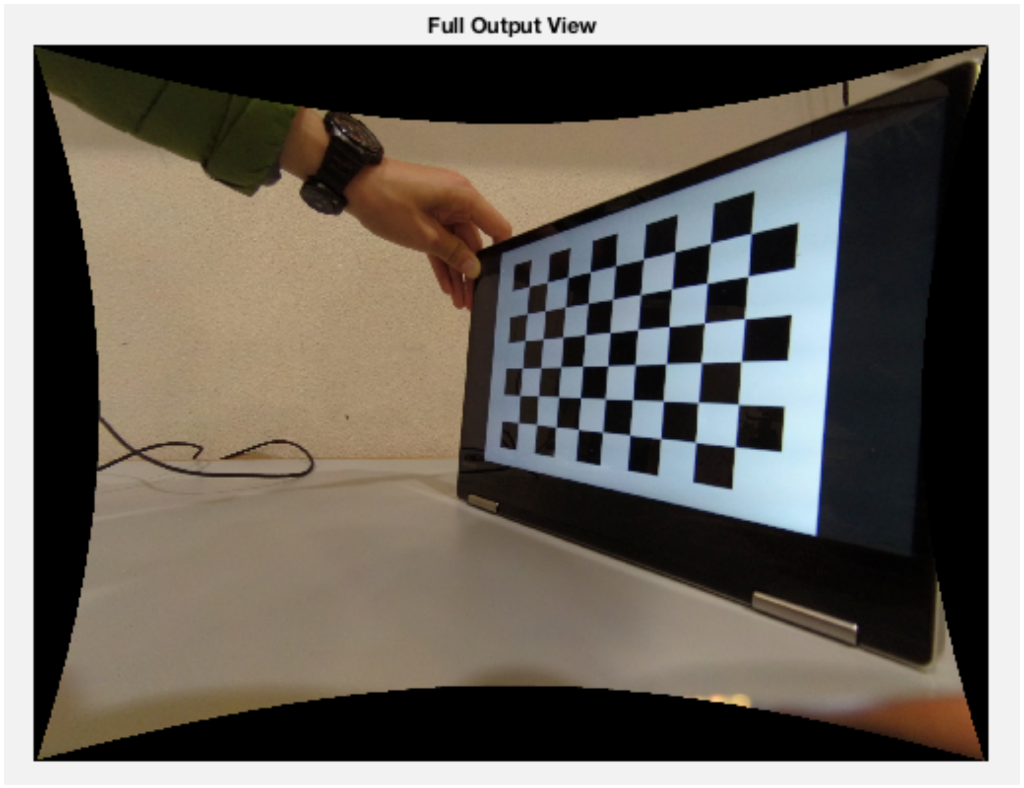


Figure 41: Full corrected image before cropping

Hypothetically, this should fix the issue raised earlier with fisheye distortion in the GDU Byrd Premium 2.0 drone's camera. However, since the exact pixels are nonlinearly adjusted, the output image may become blurred, losing texture information. Thus, it is still better to use a camera without a noticeable fisheye problem in the first place, such as the DJI Phantom 4 drone, which is responsible for the images in the Masking Dirt Rows section below.

The larger problem is consolidating data for each sample into the minimum number of images to be analyzed. Recognizing objects in consecutive frames of a video or images taken a few seconds apart will result in a huge amount of double-counting. Therefore, Matlab should be used for video mosaicking [22] or image stitching [23] as appropriate.

Appendix A contains the script dji_mosaicking.m, which was meant to stitch the images together using functions from Matlab's Computer Vision toolbox. Unfortunately, even with only 5 images (found in Appendix E), and only using the top 300 feature points from each image, the matrix operations were too computationally heavy, and repeatedly froze the 2011 Macbook Pro on which the script was running. Therefore, two online image stitching services were used.

The first, Maps Made Easy [24], uses the drone's GPS signal and altitude to create the map seen in Figure 43. However, small variations in altitude not accounted by the drone's altitude sensor led to large distortions in the image, making this undesirable for analysis. Figure 44 shows an excerpt of the full image to show the distortion.

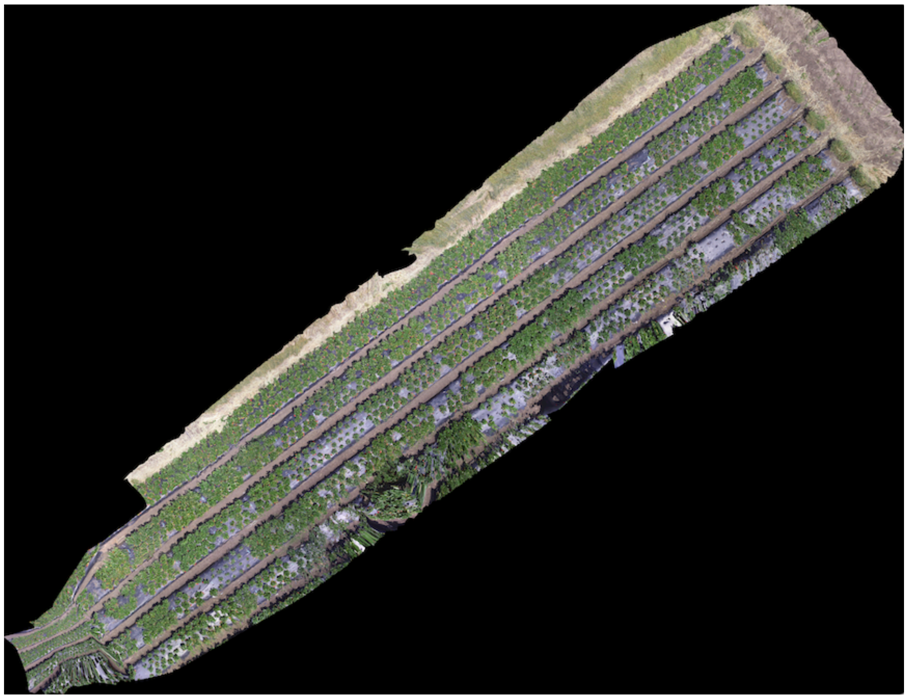


Figure 42: Maps Made Easy output image

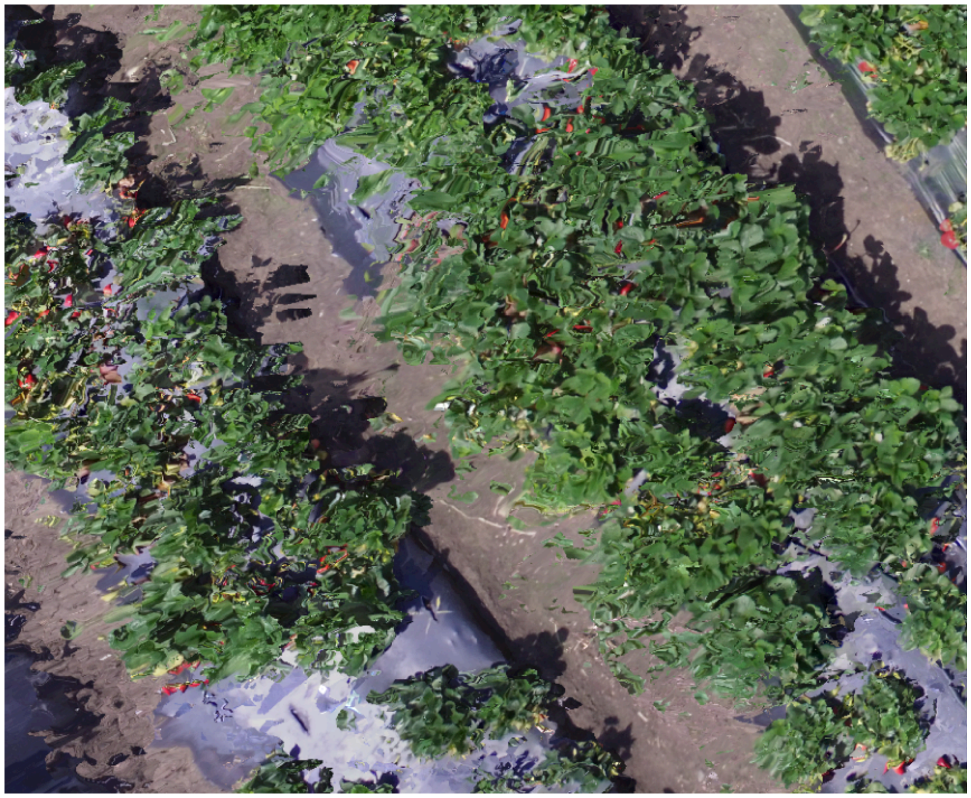


Figure 43: Distorted excerpt from Maps Made Easy image

The second, DroneDeploy [25], took about 12 hours to compile the image in Figure 45. Figure 46 shows an excerpt of this image.

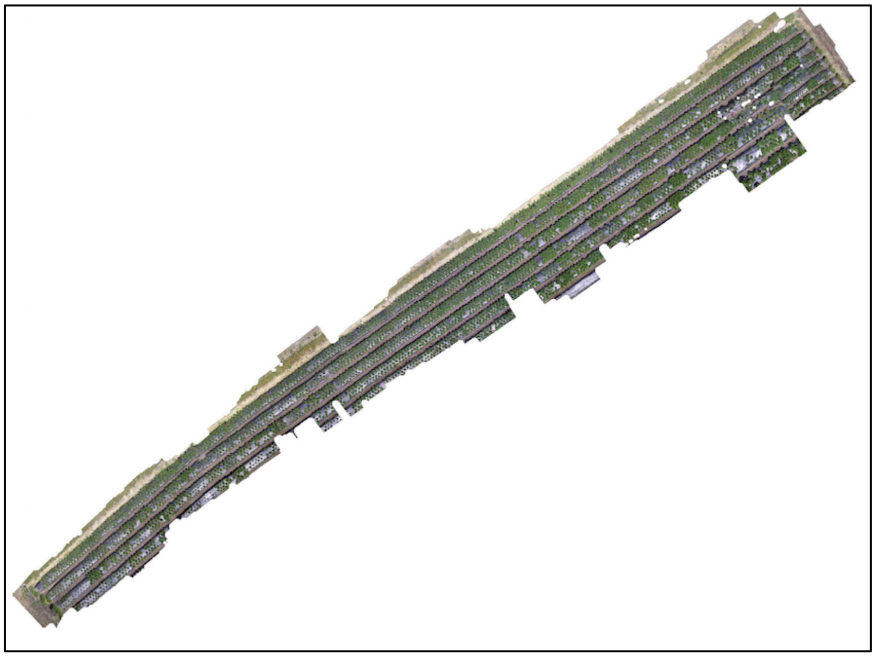


Figure 44: Drone Deploy output image

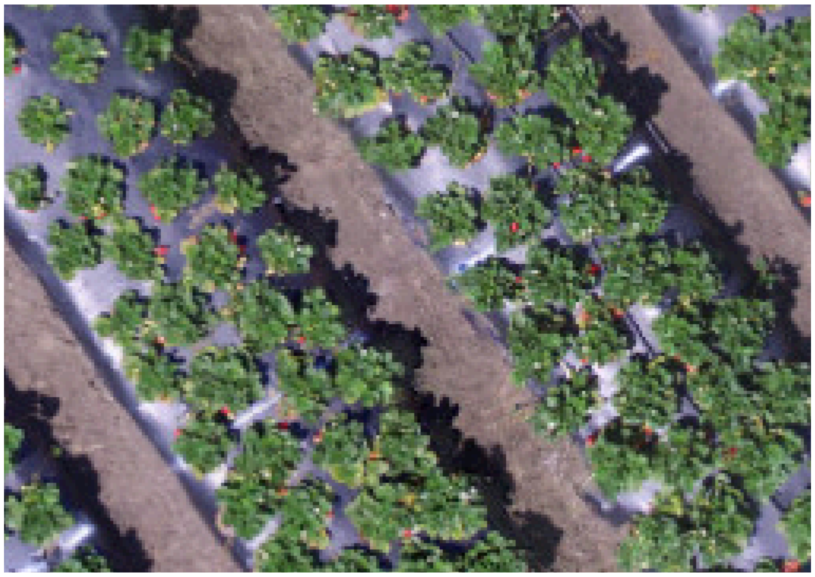


Figure 45: Low-resolution excerpt from Drone Deploy image

Although the distortion is minimal, the resolution in the image has been greatly reduced, making this option a bad fit for this project as well. Video mosaicking remains an open problem for this project going forward, if either the drone or rover implementations are to be used for data acquisition.

3.3 MASKING DIRT ROWS

As mentioned previously, it is not uncommon for overripe strawberries to fall off the plant and into the dirt rows. Therefore, the best course of action is to automatically create a mask to block out the dirt and any unwanted objects therein. The final script which accomplishes this can be found in the Appendix.

After the image is read in, it is resized by a factor of 4, to lessen computational intensity. Strictly speaking, this step is not necessary, but some parameters later in the program are dependent on the image resolution and should be changed if this step is skipped. Figure 47 shows the image under test (after resizing).

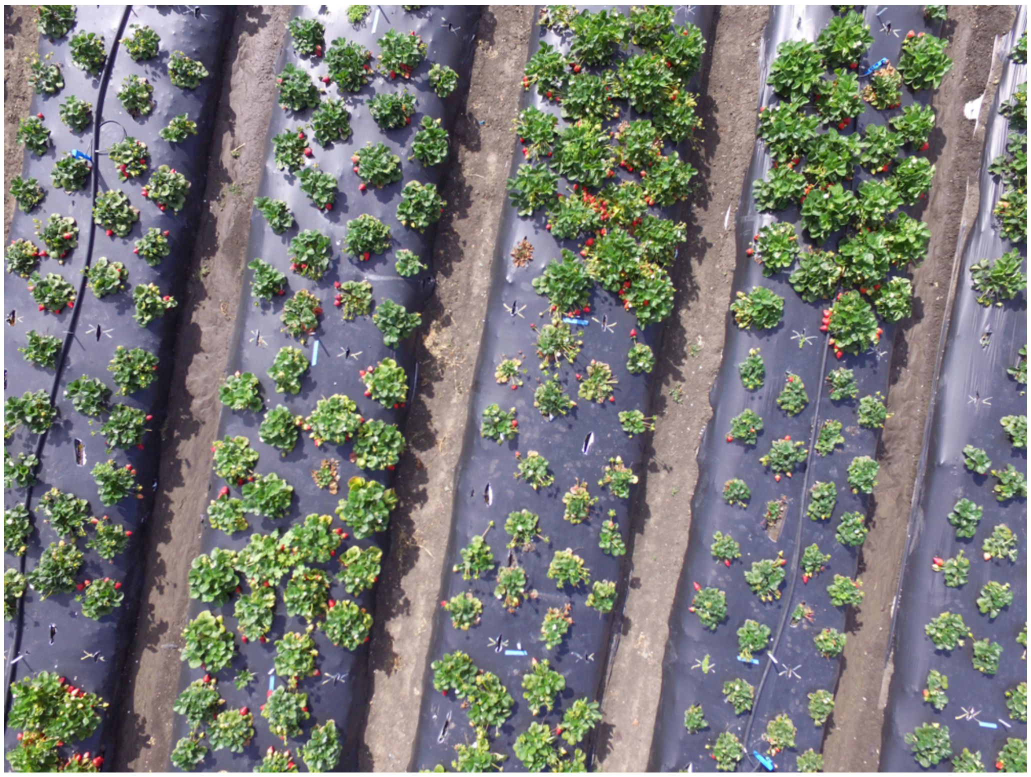


Figure 46: Original image of strawberry rows

The image is converted to HSV and thresholded in the hue layer. As explained previously, the hue layer is a good choice of parameter, since it is lighting-invariant. In this case, the dirt all has roughly similar hue, which makes thresholding simple and effective. The other advantage of this step is that the plants join the same layer as the plastic tarp (aside from discolored leaves or red strawberries, which are part of the dirt layer). This value is likely to vary for different lighting conditions and for different fields. Figure 48 shows the hue layer of the original image, and Figure 49 shows the hue layer after thresholding. The threshold range chosen was -0.2 to +0.2, to separate out pixels whose hue was near 0 (red). Since hue is a continuous range between 0 and 1, the hue range was "rotated" by adding 0.2 and then performing modulo by 1. Then, the desired hue range was 0 to 0.4.

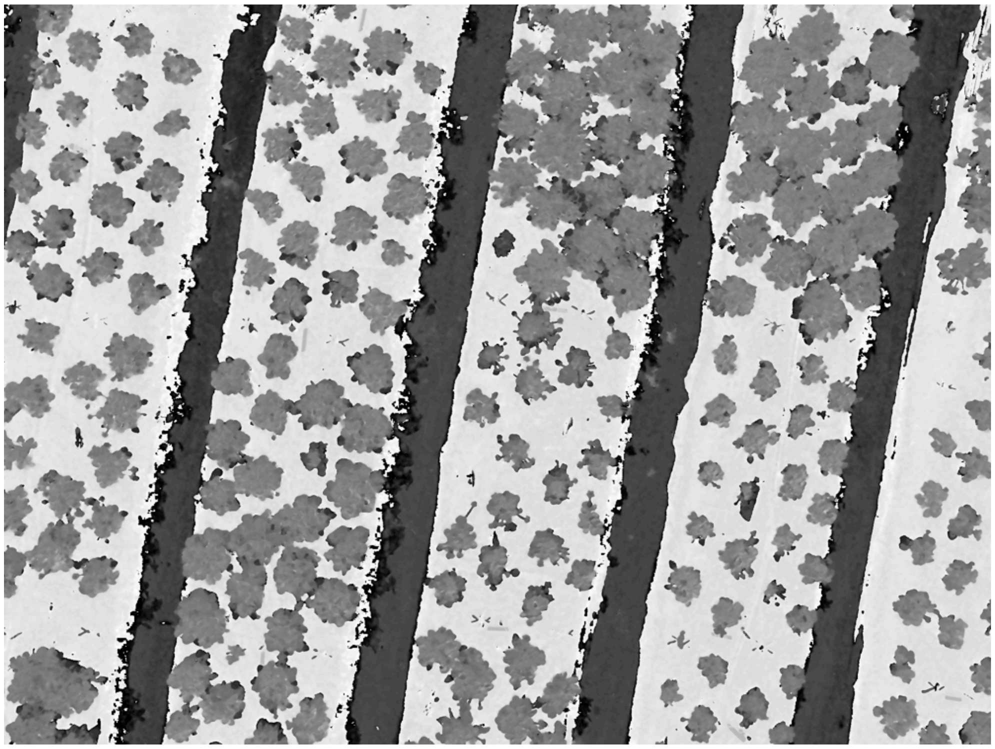


Figure 47: Hue layer of original image

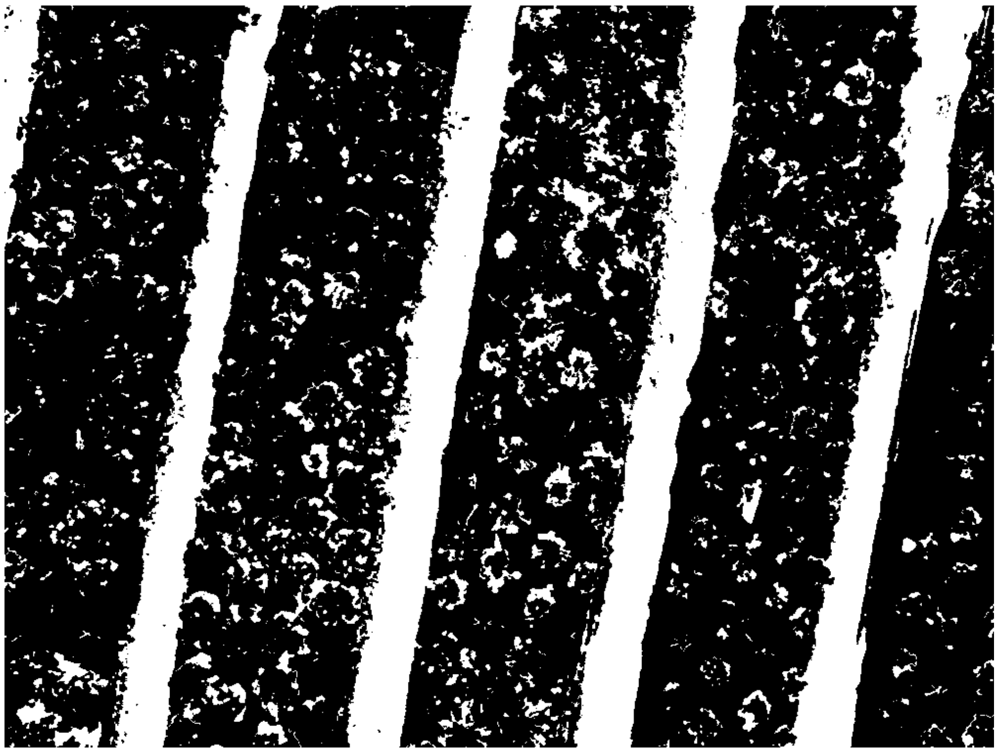


Figure 48: Strawberry rows after HSV conversion and hue-layer thresholding

Next, Canny edge detection is performed on the thresholded image [17]. Matlab's Canny edge detector has a Gaussian blur filter built in; for this downsized image with a resolution of 750x1000 pixels, a Gaussian filter of 25 pixels was chosen. (Note that this value will vary with the resolution of the image under test.) This is large enough to remove the high-frequency edges caused by plants protruding over the edge of the plastic, as well as decrease the intensity of the edges around the small objects which joined the same layer as the dirt after thresholding.

The other input parameters for the Canny edge detector are the low and high threshold values, since Canny edge detection uses non-maxima suppression to create an edgemap where edges have a width of one pixel. The threshold values remove less intense edges, leaving only the edges of the dirt rows. Figure 50 shows the output of the Canny edge detector. For this script, threshold values of 0.3 and 0.4 were chosen.

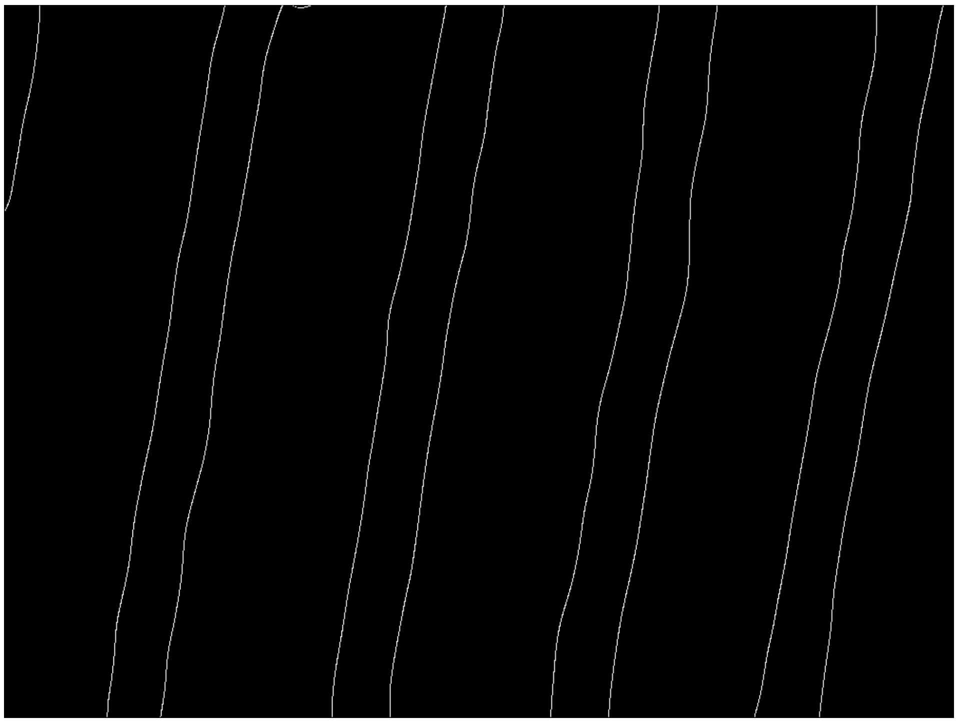


Figure 49: Strawberry rows after Canny edge detection

After edge detection, the Hough transform is applied to look for lines in the edgemap [18]. This process converts all lines into rho-theta pairs in the corresponding parameter space. Figure 51 shows the heatmap of the parameter space. Note the high level of convergence in the middle indicating strong lines present in the edgemap, each with an angle of roughly 10 degrees.

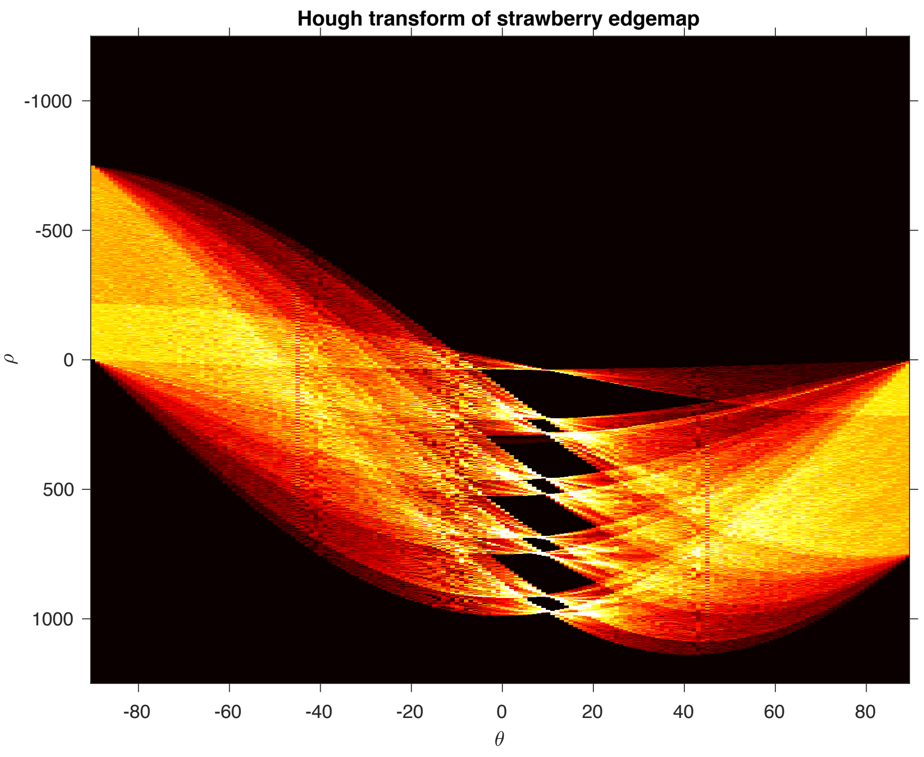


Figure 50: Rho-theta parameter space of Canny edgemap

Matlab quantizes the rho-theta field to find the coordinates with the highest number of lines and returns these rho and theta values. There should only be about 4 dirt rows visible in a typical image, so the 8 strongest lines are kept. For higher-altitude images, the maximum number of lines should increase. Figure 52 shows the lines superimposed on a grayscale version of the original image. These are well aligned with the boundary between the plants and the dirt rows.

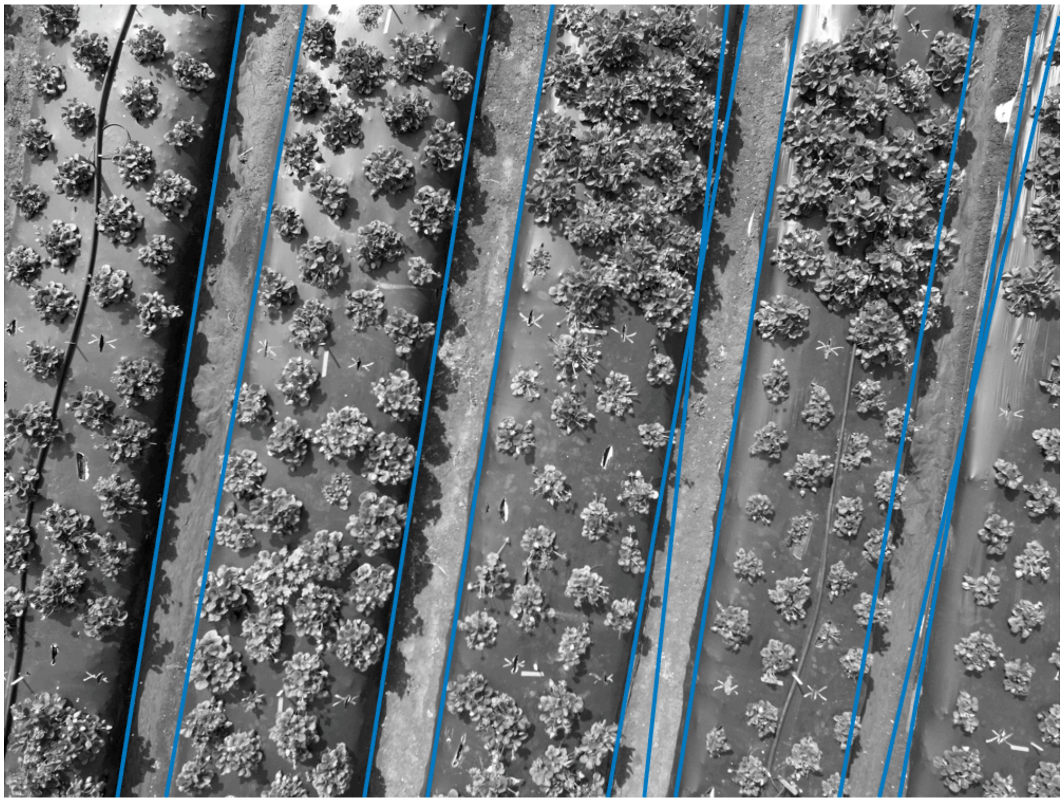


Figure 51: Strawberry rows with lines found through Hough transform

Now that the lines have been found, they are sorted by y-intercept from left to right. A mask is initialized with the same size as the image. For each pixel in the mask, its value is incremented by 2^l if there is a line to its left, where *l* is the line's number in the array. Using this pseudo-binary approach, each polygon is assigned to a different value between 0 and $2^{(total\ number\ of\ lines)}$. In an ideal scenario with no spurious lines detected, this algorithm results in gray stripes of ascending value from left to right. Figure 53 shows this mask.

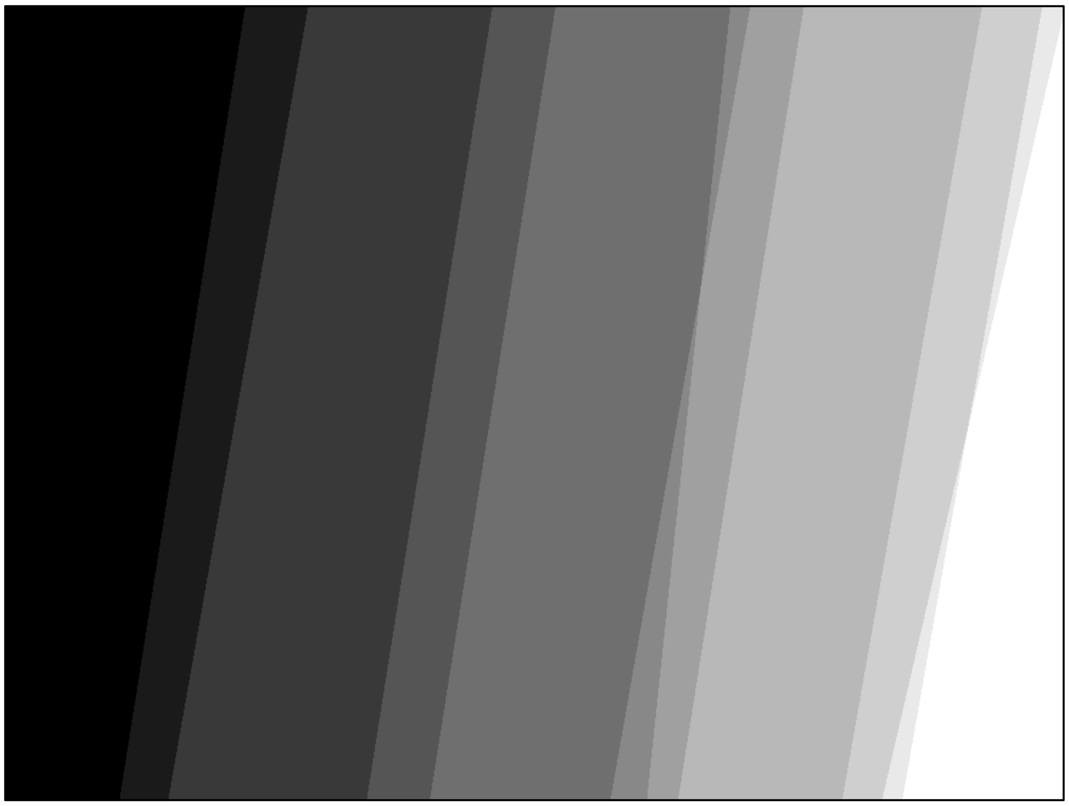


Figure 52: Incomplete mask with gray stripes

The area of each stripe is counted. Then, for each stripe, if its area is lower than the average, it is assumed to be a dirt row, and if it is higher than the average, it is assumed to be a row of plants. By using the area of the stripe as the distinguishing feature, rather than simply alternating between 0 and 1 values, there is less risk of masking out a row of plants if the line on one side of a dirt row is not detected, or if too many lines are detected. Finally, the white areas of the mask are dilated by a 15-pixel square structuring element to avoid cutting off plants protruding over the edge. Figure 54 shows the final mask, and Figure 55 shows the mask applied to the original image through multiplication.

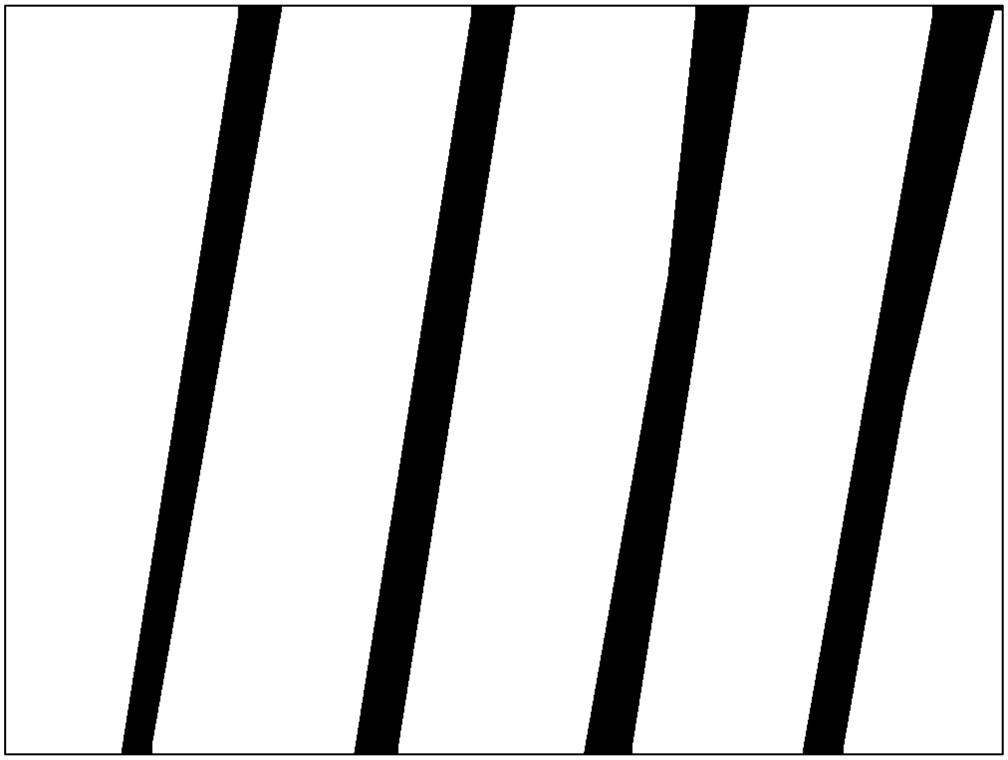


Figure 53: Final black and white mask

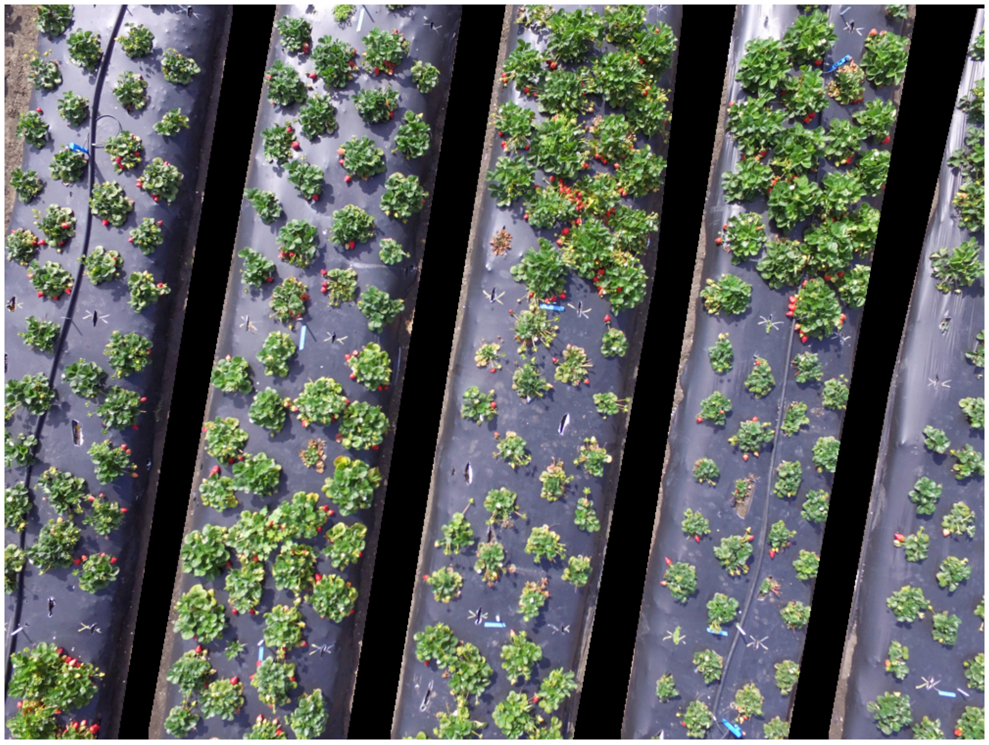


Figure 54: Original image after masking

Note that even with the extra lines found in the Hough transform stage, this method successfully distinguishes between dirt and non-dirt.

To summarize, the parameters that must be tuned when adjusting this script to new images and new fields are:

1. Hue layer threshold value
2. Gaussian filter size
3. Low and high threshold values for Canny edge detector
4. Maximum number of lines to detect
5. Structuring element size for dilation

Of these, parameters 2 and 5 depend on the image's resolution. Parameter 4 depends on the altitude; a higher altitude drone will see more rows simultaneously, and therefore more lines should be expected.

Overall, this process works fairly reliably for images from this dataset with only complete rows in frame. Since the camera used for this trial has negligible fisheye distortion, the rows are straight enough to be defined by only one or two lines. However, the process breaks down at the start or end of the field. Figure 56 shows one such image, Figure 57 shows the Canny edgemap of this image, Figure 58 shows the lines detected by the Hough transform, Figure 59 shows the mask with gray stripes, and Figure 60 shows the final masked image.

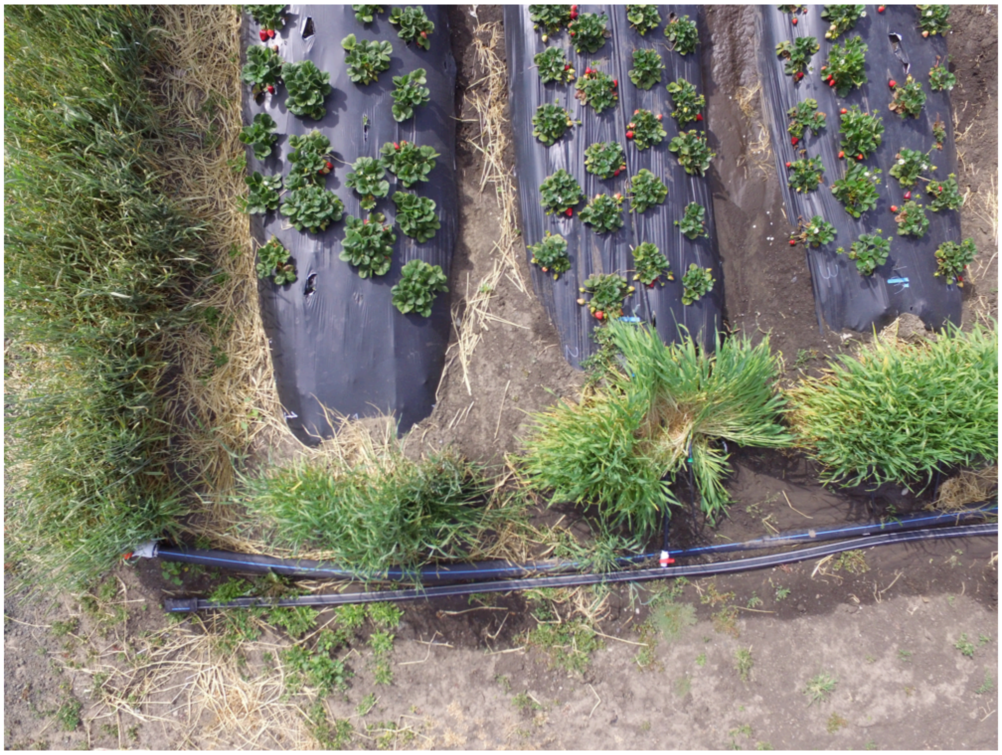


Figure 55: Original image at the edge of the field

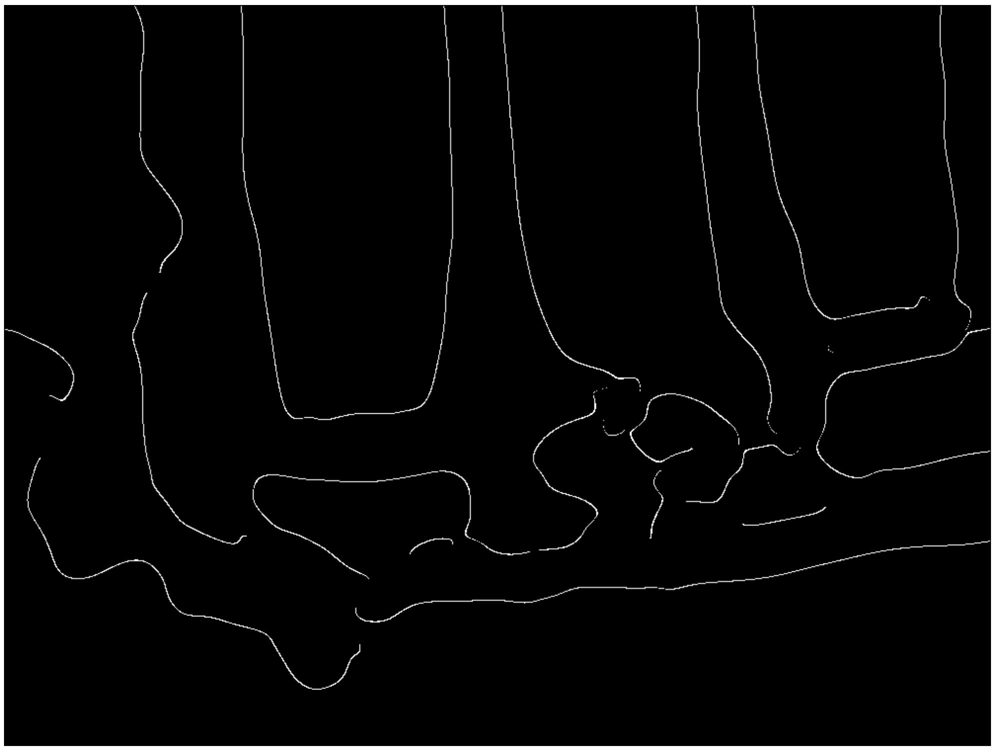


Figure 56: Edgemap of the image at the edge of the field

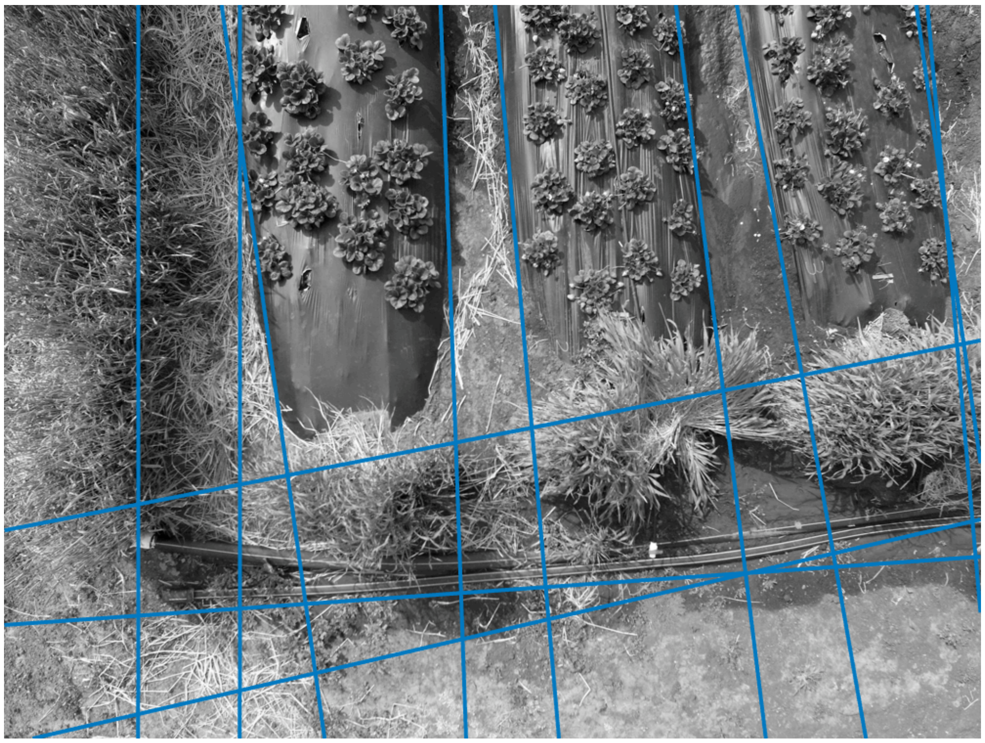


Figure 57: Lines detected in image at the edge of the field

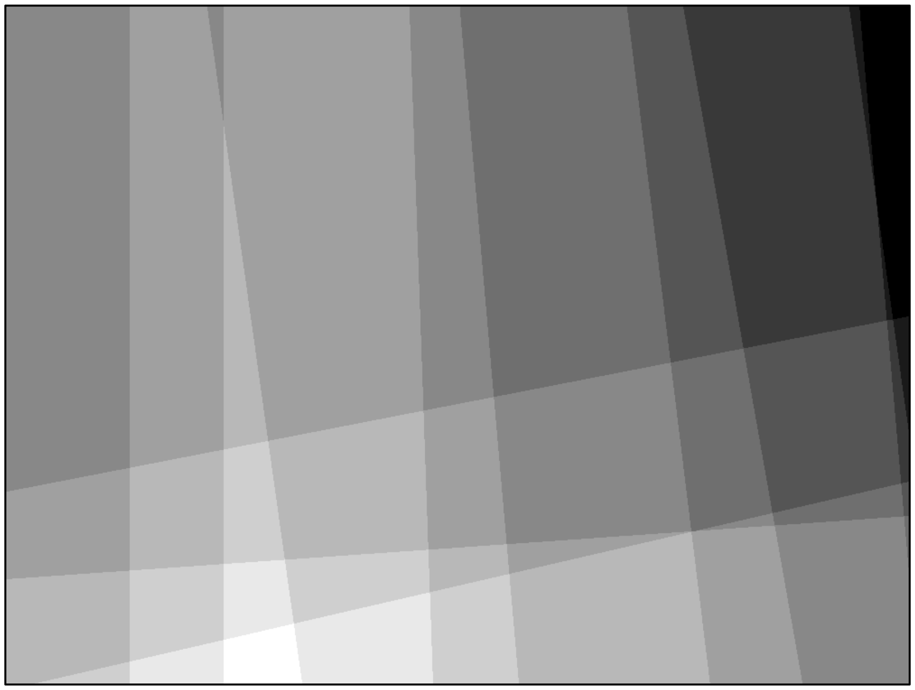


Figure 58: Gray striped mask for image at the edge of the field

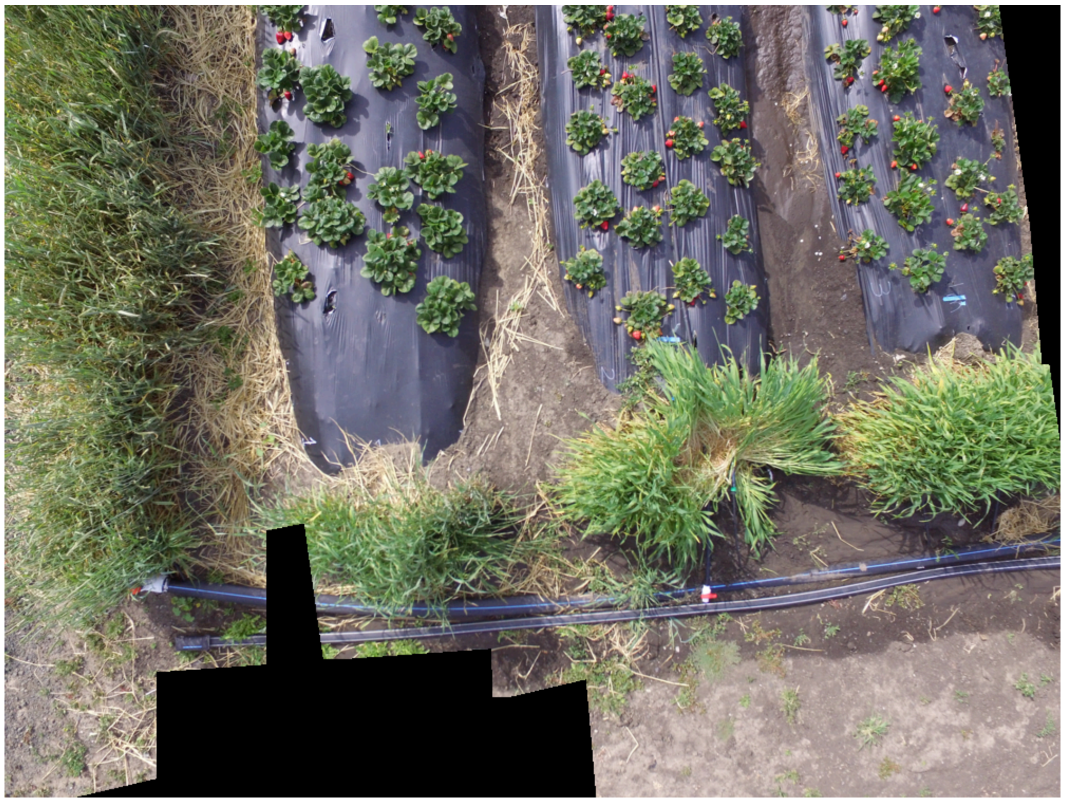


Figure 59: Masked image at the edge of the field

The presence of the hose at the bottom of the image throws off the Hough transform, although it was able to successfully find the edges of the rows in the upper half of the image. In the future, this problem may be mitigated through mosaicking or smart cropping.

To verify the performance of this algorithm, the script was executed for 75 “well-behaved” images from the same flight as the previous figures in this section. In this context, “well-behaved” images are those which do not contain any areas outside the field, as in the previous failed case. For each sample, the number of dirt rows masked out was compared against the number of rows that should have been masked out. The current algorithm masked 72.16% of all dirt rows. However, 41.67% of images had areas of interest masked out as well. Therefore, this method needs additional research for improved reliability.

Several trends became apparent with this test. First, mature plants are much more likely to grow over the edge of the plastic and into the dirt row, where valid fruits could be masked out by this algorithm. Taller plants near the edges of the image face a higher risk, since the drone views them off-angle. In addition, quite frequently, two lines will be identified at the boundary of a dirt row instead of one, intersecting in an X-shape. Figure 61 below shows two examples of this type of error.

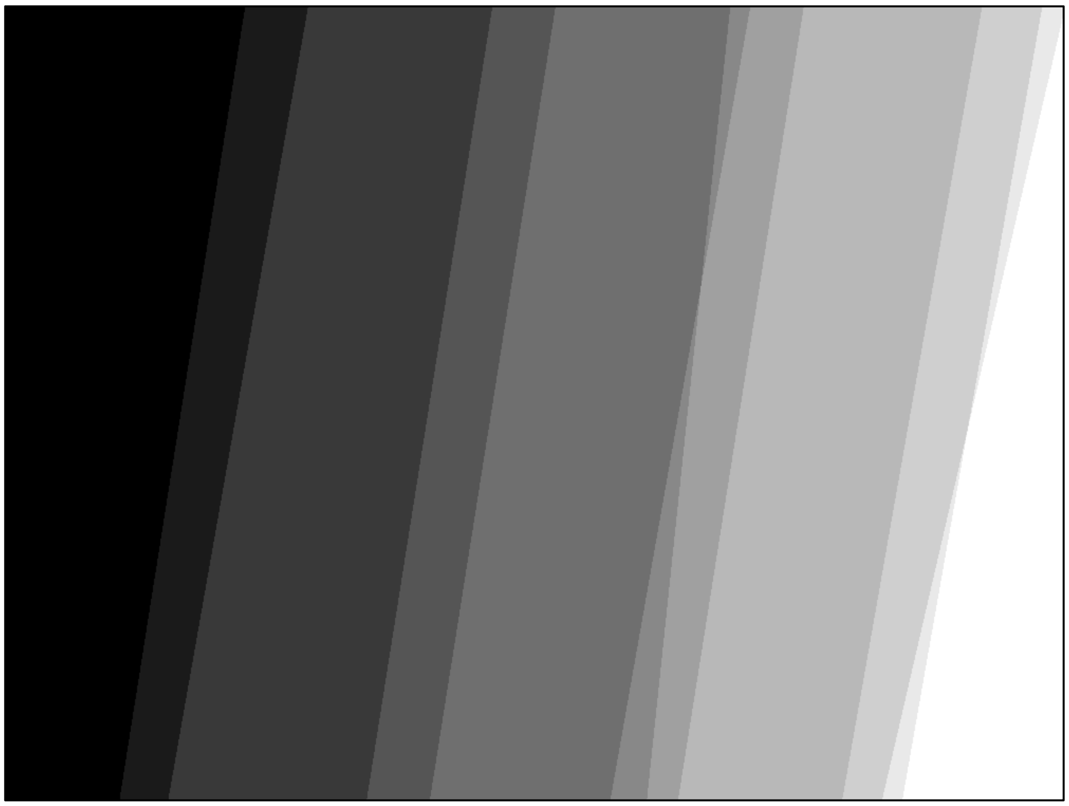


Figure 60: X-shaped intersection of two lines

Unfortunately, this creates small additional triangles. Recall that the mean area of all shapes created by the detected lines is used to distinguish between dirt rows and rows of plants, and this mean area will decrease due to these small unwanted shapes. To improve the algorithm, one solution is to expand the Gaussian filter to avoid these X-shaped intersections, or to invent a more reliable method than comparing against the mean area to create the final mask. Simply removing one of the intersecting lines is not a desirable solution, since in most cases where the intersection occurs, fully shading the region actually fits the shape of the border better than using either of the lines alone, which means the dirt is more completely masked out.

Figure 62 below shows an example where too few rows were masked out, and Figure 63 shows an extreme example where too much of the field was covered.

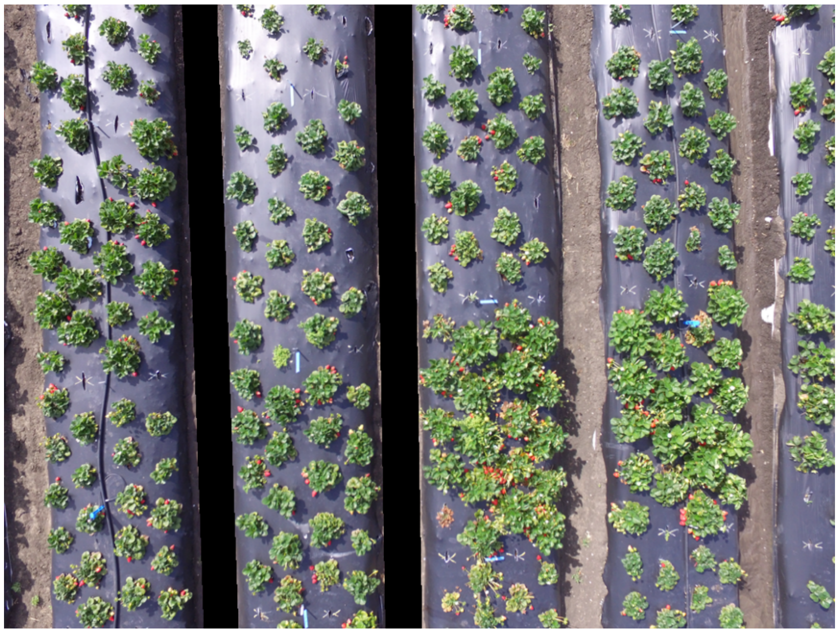


Figure 61: Too few rows are masked out

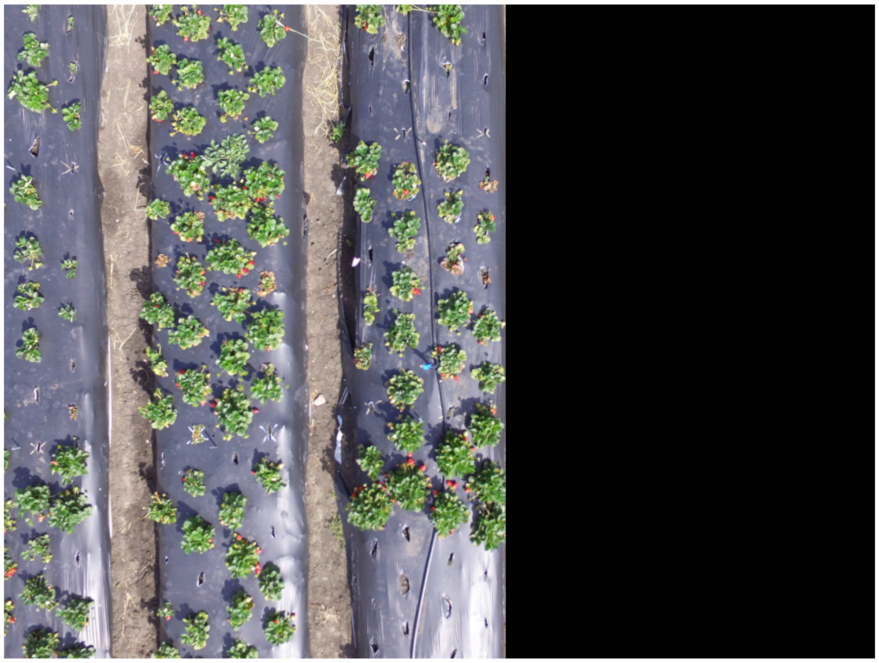


Figure 62: Areas of interest are masked out along with dirt rows

The latter case is due to a rare error where only two lines were found in the image, as shown in Figure 64. However, the Hough transform map in Figure 65 shows strong convergence, so it is unclear why the other lines were not detected as well.

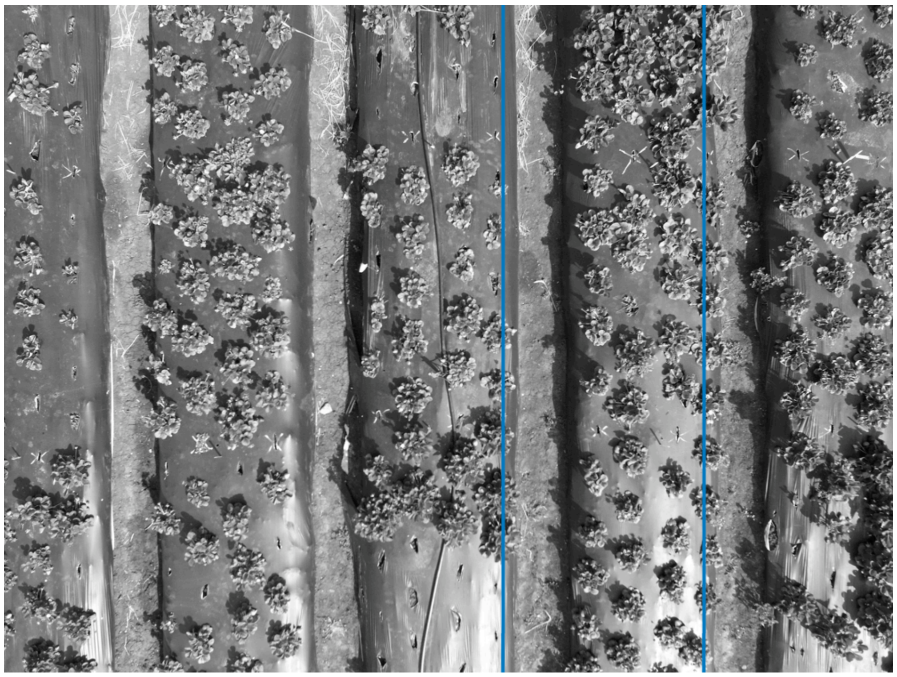


Figure 63: Only two lines were detected in this image, leading to the error in the previous figure

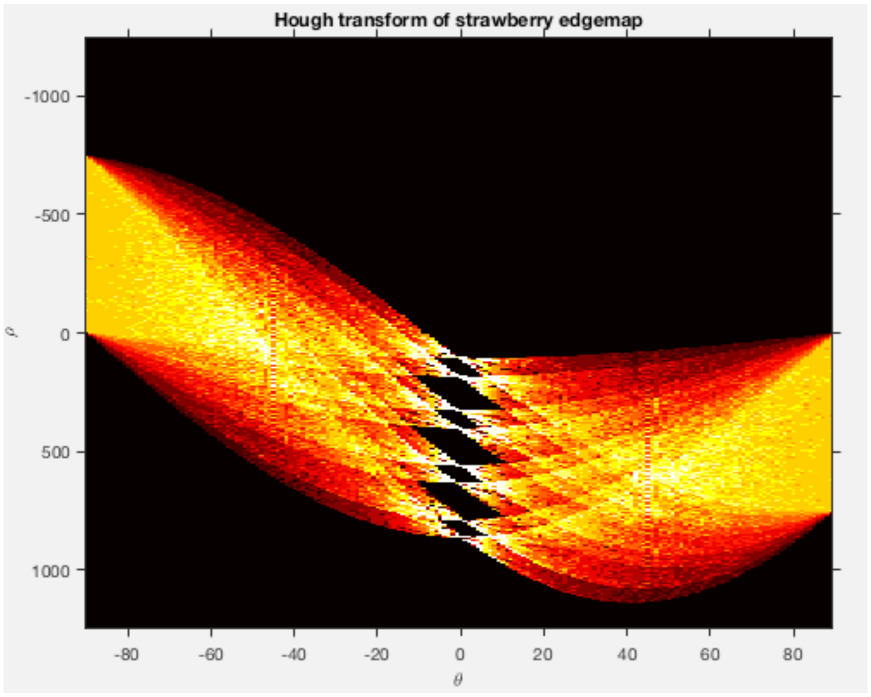


Figure 64: The rho-theta map shows strong convergence in 8 points, as expected.

For the purposes of this test, rows of fruit which were incompletely captured in the original image were ignored, regardless of whether they were masked out or not. Ideally, to avoid double-counting any fruit, these incomplete rows should probably be masked out in the future, either using this method or another. If the incomplete row is narrow and has a small area, it may distort the mean area of all rows as well.

3.5 OTHER CONSIDERATIONS

Ultimately, all computer vision techniques require human trial and error and many training samples, which means unsupervised learning is not yet possible for this stage. For example, the exact hue of the dirt will likely change from field to field. It is also possible that the dirt's hue may have too much variance in certain fields for the masking algorithm explained previously to work in all cases and all lighting. Therefore, much more generalization of the system is needed in order to apply it on a large scale.

4. PREDICTION

4.1 ARTIFICIAL SEASON DATA

In the absence of a complete season's worth of drone images from the Cal Poly Strawberry Center to analyze with the full system, it would be beneficial to make an artificial dataset using trends observed in the field. This was synthesized using a number of assumptions, primarily developed through watching a timelapse of strawberry growth [16] and observing surface-level trends in the growth of strawberries. It should be noted that these assumptions were put in place prior to obtaining hard data from the field; the resulting discrepancies are discussed in the next section.

1. It takes at least a week after planting for strawberry plants to bloom.

This is purely an estimation and depends heavily on outside variables such as the season, climate, and what kind of fertilizer is present. In reality, it was observed that it takes closer to a month for flowers to appear after planting. For this simulation, the starting state can instead be considered to be a plant about to bloom.

2. The appearance of flowers across the entire flower can be modeled as a Gaussian distribution over the period of 10 days.

This assumption is intended to account for especially early and late blooms. Once a strawberry flower blooms, it will likely persist for a few days before the petals fall off and the bud develops into a green strawberry.

3. There is a period of time between a flower being recognized and the bud being recognizable to the computer vision block as a green strawberry.

It is even more difficult than classifying a flower or a green strawberry to classify the intermediate stage between them. Therefore, there will be a short period of time between the leading edge of the flower stage's Gaussian curve and the leading edge of the green strawberry stage.

4. Red strawberries are picked by the farmer before they become overripe.

In the ongoing experiment at the Cal Poly Strawberry Center, this is not the case, which will result in helpful statistics on how long a strawberry can sit in the field before spoiling. However, in the case of the end user, all berries will be harvested prior to becoming overripe.

Figure 66 shows the progression of a strawberry from bud to flower to green strawberry to white strawberry to red strawberry, reinforcing a few of the earlier assumptions.

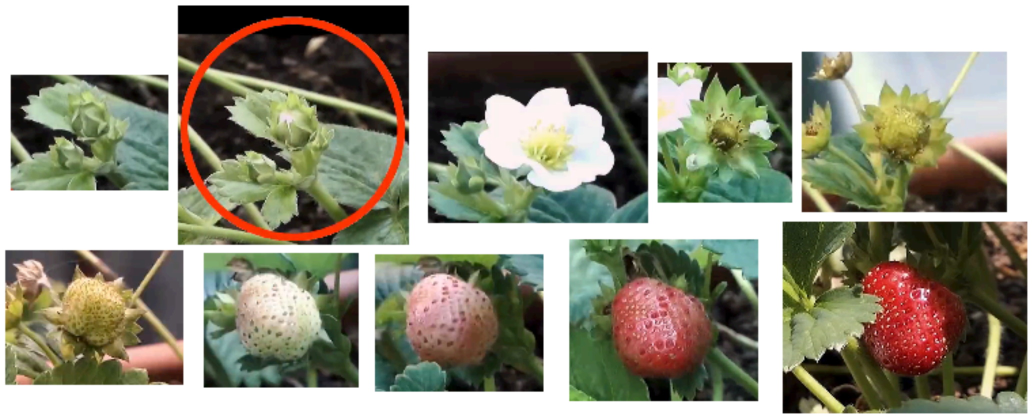


Figure 65: A strawberry's lifecycle [16]. Note that these images are not spaced at even time intervals.

Table 7 on the next page shows the ideal (no noise) table of strawberry growth for a made-up season, synthesized using the previous assumptions.

Table 7: Fake season based on 4 assumptions

Day	Flowers	Green	Red	Picked
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	1	0	0	0
9	3	0	0	0
10	10	0	0	0
11	25	0	0	0
12	42	0	0	0
13	50	0	0	0
14	42	1	0	0
15	25	3	0	0
16	10	10	0	0
17	3	25	0	0
18	1	42	0	0
19	0	50	0	0
20	0	42	1	0
21	0	25	3	0
22	0	10	10	0
23	0	3	25	0
24	0	1	42	0
25	0	0	50	0
26	0	0	42	8
27	0	0	25	25
28	0	0	10	40
29	0	0	3	47
30	0	0	1	49
31	0	0	0	50
32	0	0	0	50

From this concept, a Matlab script was written to generalize these assumptions and model a full season using Gaussian random variables. One advantage to this method is that once some a better estimate exists for the parameters, the simulation can run for an arbitrary number of seasons, and aggregate data over a longer period of time than is feasible by studying a field.

The inputs to this Matlab script are:

- K: number of seasons to run (default 10)
- Days: number of days in each season (default 35)
- Forgetting rate: if this parameter is greater than 1, the overall probability distribution will emphasize newer seasons. If this parameter is set to 1, all seasons will be weighted equally.
- Bloom range: 1x4 array. Defines the behavior of the portion of the season where blooms appear. The values of this array are (in days):
    - Minimum time until the first bloom (default 4)
    - Maximum time until the first bloom (default 7)
    - Minimum length of the bloom cycle (default 6)
    - Maximum length of the bloom cycle (default 14)
- Green range: 1x5 array. Defines the behavior of the portion of the season where green strawberries appear. The values of this array are (in days):
    - Minimum time until the first green strawberry (default 3)
    - Maximum time until the first green strawberry (default 5)
    - Minimum length of the green strawberry cycle (default 8)
    - Maximum length of the green strawberry cycle (default 18)
    - percentage of fruits that don't survive to this stage (default 10)

- Red range: 1x5 array. Defines the behavior of the portion of the season where red strawberries appear. The values of this array are (in days):
    - Minimum time until the first red strawberry (default 3)
    - Maximum time until the first red strawberry (default 5)
    - Minimum length of the red strawberry cycle (default 10)
    - Maximum length of the red strawberry cycle (default 22)
    - percentage of fruits that don't survive to this stage (default 20)

These default values are solely based on intuition. As such, they are likely to be extremely inaccurate. Still, this script gives some sense of what a full season of data could look like, ignoring measurement error, suboptimal weather, and seasonal fluctuations. As will be shown later, these factors are not at all trivial, so the usefulness of this simulation is limited to the parameters it is given

Note also that for the purpose of this paper, a "bloom" is described as the bud where white petals are visible; a "green strawberry" is described as a bud with visible seeds but no color change; and a "red strawberry" is a fruit which is ready for harvest.

Figure 67 shows one sample output of the simulation script. Blue (thinner) lines are the curves from the most recent season, while orange (thicker) lines are a distribution derived from all seasons together.

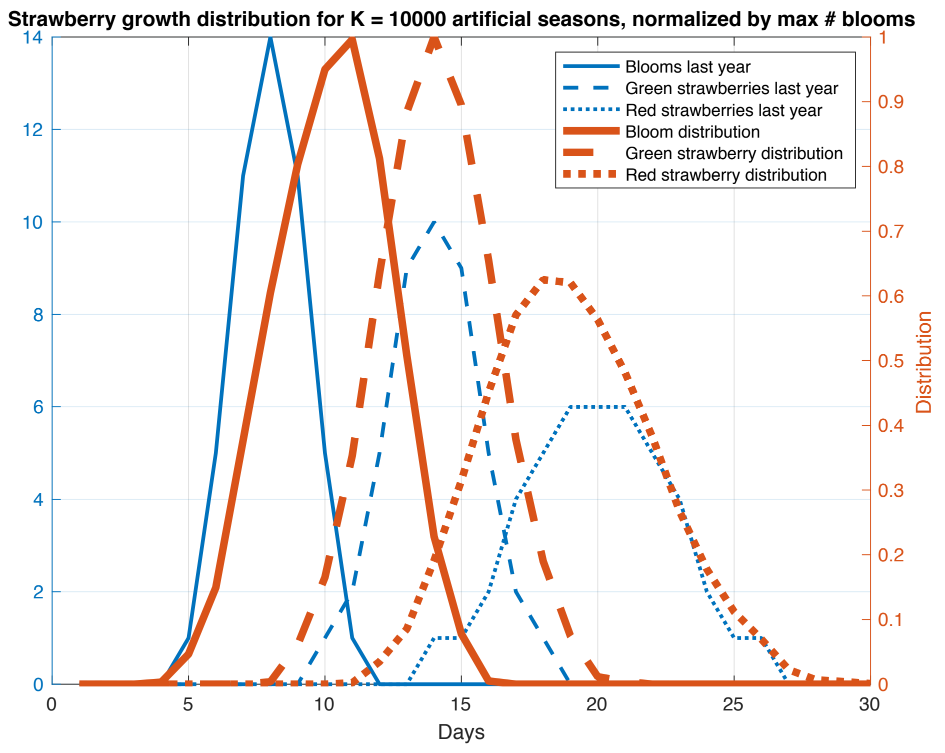


Figure 66: One possible output of the strawberry season simulation

The biggest shortcoming of this system is that it cannot currently account for the effect that the current season has on speeding up or slowing down strawberry plant growth. A more complete model would have a start date as input and would take into account the faster growth during the spring and summer, and slower growth during winter.

Once the error rates for the current computer vision techniques are known, it will be useful to look at modelling measurement noise in this artificial data and analyzing how many seasons it takes for the error in the model to fall to a negligible level. For example, if the strawberry detection error rate for one day's samples is 20%, assuming a white noise distribution, then averaging the distributions of two consecutive seasons gives a net error rate of 4%. In this way, the model becomes more reliable as the number of seasons reaches infinity.

4.2 ACTUAL DATA

With multiple teams analyzing the Cal Poly strawberry fields for multiple simultaneous projects, there are a few ongoing datasets to analyze. Appendix C contains photos of some regions of interest from Field 25, the same one from the earlier drone images. These images were taken by smartphone from January to April 2018 and outline a full season's worth of data, ignoring the phases of strawberry plant development prior to the production of flowers. From these image sets, objects were hand-counted according to four categories: flowers, green strawberries, pink strawberries, and red strawberries. Table 8 shows the data for row 9, column 14, in which 337 total objects were counted, and Table 9 shows the data for row 10, column 4, in which 489 total objects were counted.

Table 8: Row 9, column 14 data from January to April

Date	Day	Flowers	Green	Pink	Red
9-Jan	0	0	0	0	0
16-Jan	7	1	0	0	0
19-Jan	10	3	0	0	0
22-Jan	13	3	0	0	0
26-Jan	17	7	3	0	0
30-Jan	21	3	6	0	0
6-Feb	28	1	15	0	0
9-Feb	31	0	20	1	0
13-Feb	35	1	23	7	1
17-Feb	39	0	13	6	5
20-Feb	42	3	20	5	8

23-Feb	45	3	16	4	3
27-Feb	49	0	4	11	0
2-Mar	52	0	10	6	13
6-Mar	56	1	2	5	20
9-Mar	59	3	9	3	16
14-Mar	64	0	4	2	4
17-Mar	67	3	5	1	7
20-Mar	70	7	5	1	3
23-Mar	73	2	4	2	3

Table 9: Row 10, column 4 data from January to April

Date	Day	Flowers	Green	Pink	Red
9-Jan	0	2	0	0	0
16-Jan	7	2	4	0	0
19-Jan	10	3	6	0	0
22-Jan	13	5	5	0	0
26-Jan	17	3	3	0	0
30-Jan	21	3	2	0	0
6-Feb	28	6	10	0	0
9-Feb	31	2	19	0	0
13-Feb	35	4	21	2	0
17-Feb	39	2	18	5	4
20-Feb	42	1	21	5	4
23-Feb	45	0	15	11	2
27-Feb	49	1	21	15	6
2-Mar	52	2	13	16	12
6-Mar	56	1	8	12	26
14-Mar	64	5	7	6	22
17-Mar	67	7	3	7	19
20-Mar	70	8	3	5	27
23-Mar	73	6	10	0	31

Figures 68 and 69 show the plots for this data. They each roughly follow a bell curve for each stage besides the flower stage, which doesn't seem to follow any clear distribution through the season.

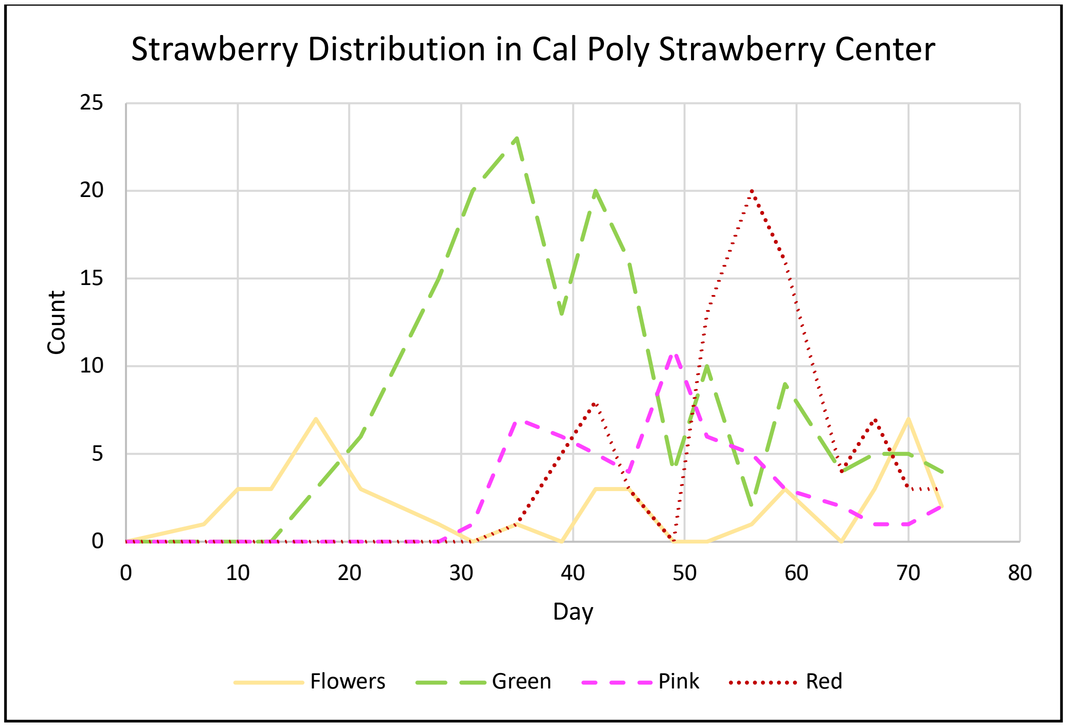


Figure 67: Plot of data from row 9, column 14

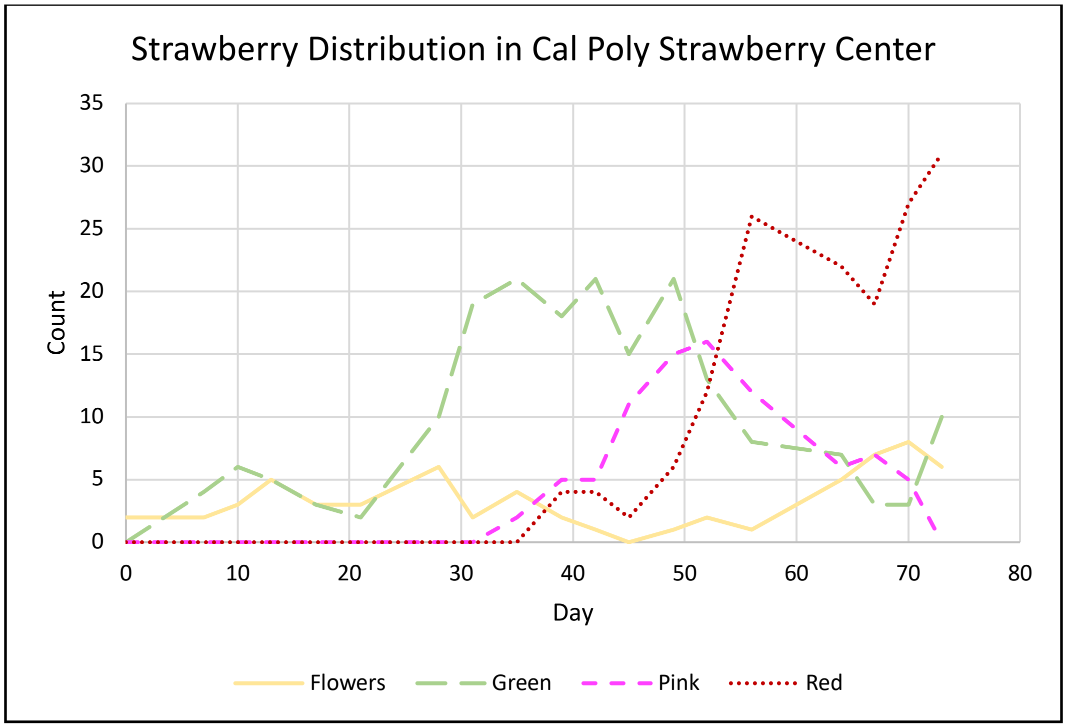


Figure 68: Plot of data from row 10, column 4

One aspect worth noting from both plots is that the area under the pink strawberry curve is much smaller than that of green or red strawberries. The pink strawberry curve's peaks also closely lead the red strawberry's. This is because strawberries spend very little time in this stage compared to the others. It may be useful for the model to reclassify these stages in a way that equalizes the time spent in each stage. It is also unexpected that the flower curve does not seem to follow a bell curve like the other stages; possibly, this is because flowers only bloom for a few days before developing into buds (technically green strawberries), and there is a high portion of flowers that do not become buds at all.

Manual object extraction is very difficult in this dataset due to a variety of factors. Variable lighting makes it difficult to find strawberries in the shade; samples taken on cloudy days were much easier to sort than samples taken in the afternoon with long shadows and glare, which alter the object's hue. Adverse lighting especially impacts pink strawberry recognition, since the presence of a slight red hue in the developing strawberry is what distinguishes the two.

The presence of white-colored plastic at the edge of the torn plastic may have led to some false positives in the flower category, since the white petals were one feature used to find flowers. Figure 70 below shows one example where torn plastic could easily be mistaken for a flower, especially due to the presence of difficult lighting.

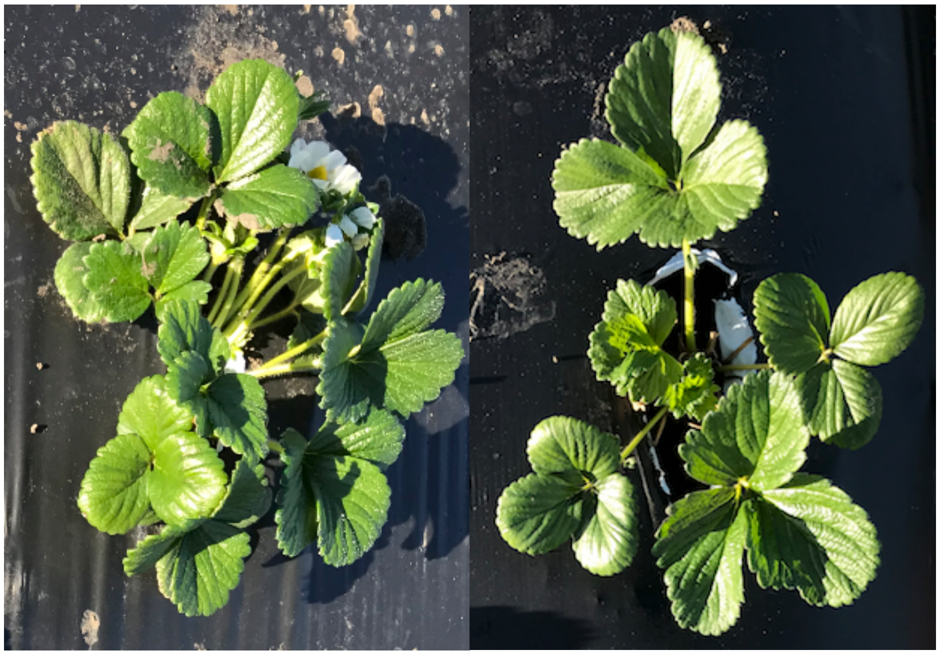


Figure 69: Flower object with white petals (left) and white-colored torn plastic (right)

Irregular sampling leads to some jaggedness of the model. Once this process is automated, it will be easy to obtain a sample each day, leading to better results and fewer spurious peaks. For now, samples are obtained biweekly (or sometimes weekly), putting them 3-4 days apart.

Double-counting is a large problem with this dataset. Figure 71 below shows three strawberries which are consistently seen in two photos each day. For the purposes of this experiment, they were indeed double-counted for consistency, since it is not known which other objects could have also been included in multiple photos. This problem would be largely mitigated with image mosaicking.

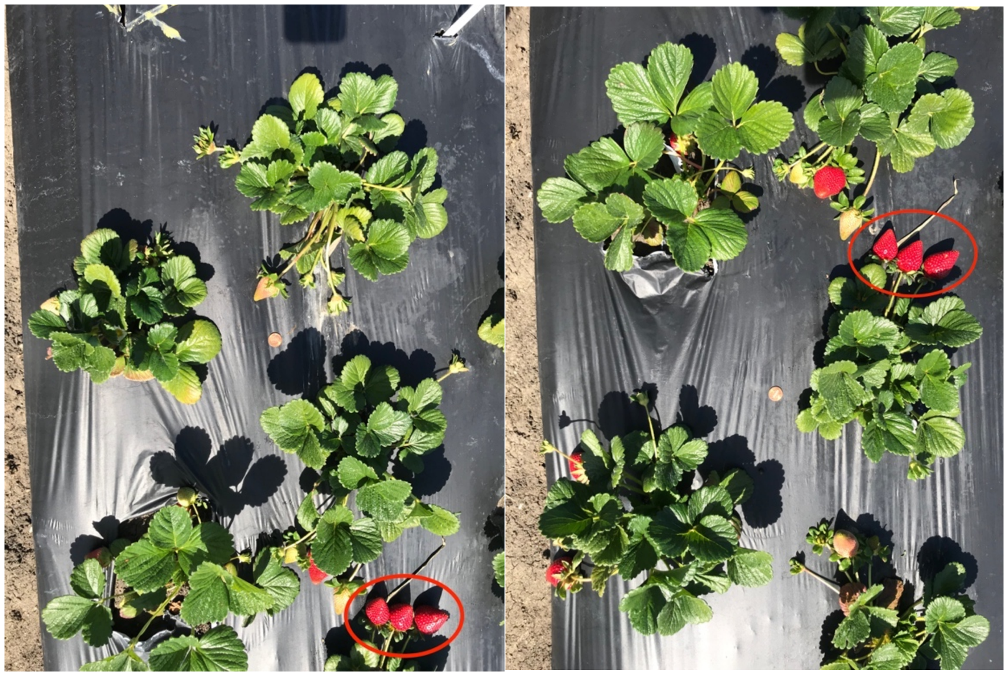


Figure 70: Three strawberries seen across two images

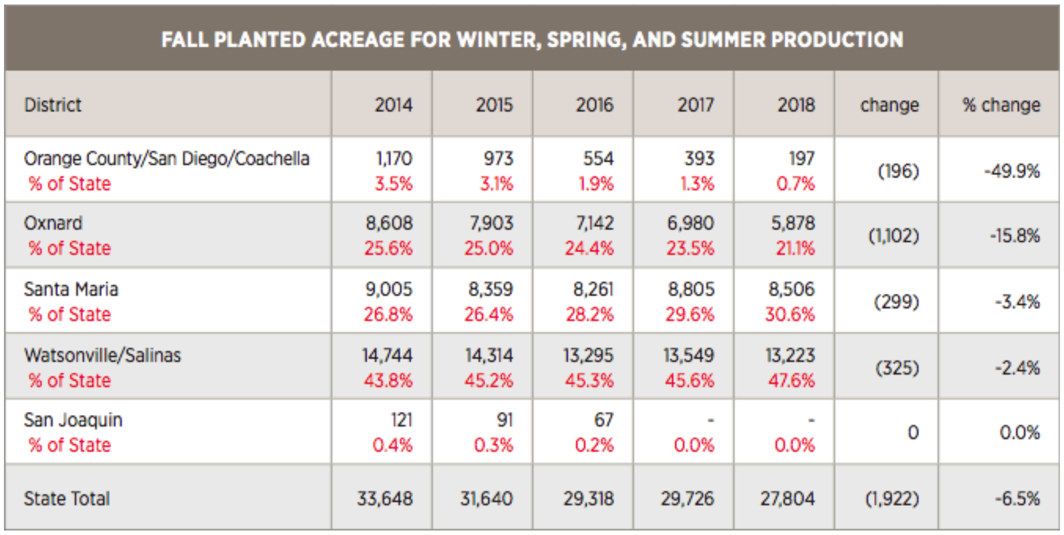
As a rule of thumb, it is much more difficult to employ computer vision when dealing with a problem which is challenging even for the human visual system. This should be kept in mind when creating a data acquisition system which has the best odds of successful object recognition.

4.3 LARGE-SCALE DATA

In an interview with Dr. Gerald Holmes of the Cal Poly Strawberry Center, it was stated that on average, it takes 25-30 days for a fruit to progress from a bloom to a red strawberry. This is consistent with real data found in Field 25, although the actual value varies wildly. An experiment that tracked individual fruits instead of counting the bed as a whole may reveal more statistically useful results. Dr. Holmes also stated that it takes about 3 months between planting and harvest in the winter, and about 2 months in the summer. For example, seeds planted in October 2017 in Field 25 at Cal Poly San Luis Obispo did not produce strawberries until late February or early March 2018.

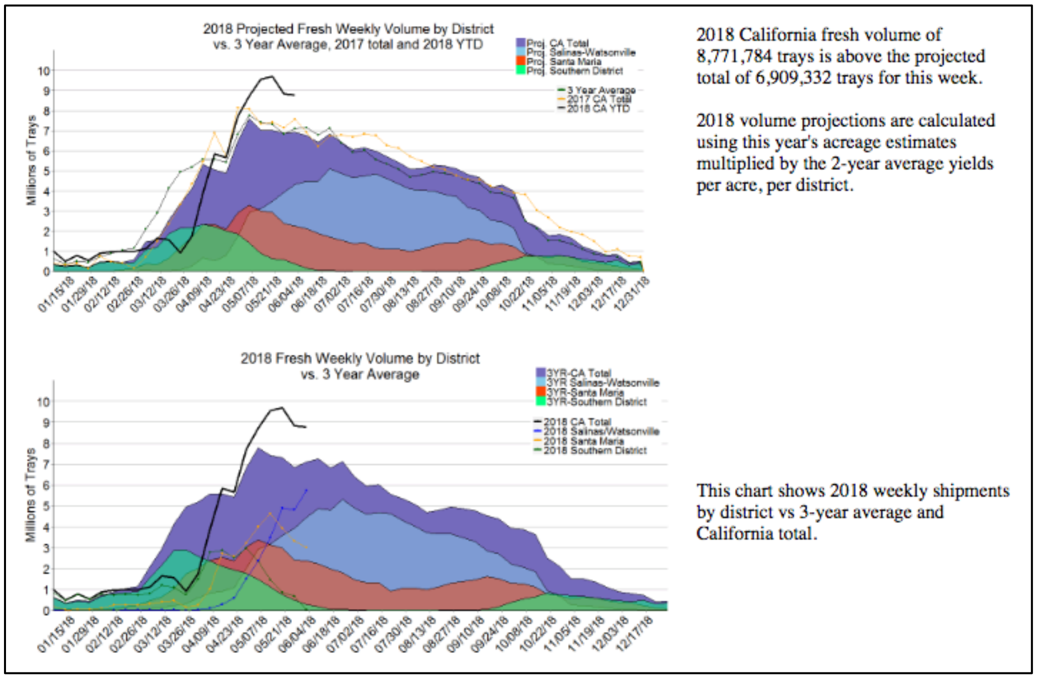
The California Strawberry Commission provides a large amount of real data on their website [20], including volume harvested per day. Weather records from each of the key strawberry-growing areas in California are available in daily reports as well, going back to 2014. Unfortunately, without additional info such as what varieties are in use and when and how many seeds are planted, production info is of limited use for this project. In addition, acreage fluctuates widely from year to year, which makes it even more difficult to directly compare between two years' harvests. Unfortunately, these figures are only available on a yearly basis. Table 10 below shows the acreage statistics from 2018.

Table 10: Strawberry Acreage in California, 2018 [20]



FALL PLANTED ACREAGE FOR WINTER, SPRING, AND SUMMER PRODUCTION							
District	2014	2015	2016	2017	2018	change	% change
Orange County/San Diego/Coachella	1,170	973	554	393	197	(196)	-49.9%
% of State	3.5%	3.1%	1.9%	1.3%	0.7%		
Oxnard	8,608	7,903	7,142	6,980	5,878	(1,102)	-15.8%
% of State	25.6%	25.0%	24.4%	23.5%	21.1%		
Santa Maria	9,005	8,359	8,261	8,805	8,506	(299)	-3.4%
% of State	26.8%	26.4%	28.2%	29.6%	30.6%		
Watsonville/Salinas	14,744	14,314	13,295	13,549	13,223	(325)	-2.4%
% of State	43.8%	45.2%	45.3%	45.6%	47.6%		
San Joaquin	121	91	67	-	-	0	0.0%
% of State	0.4%	0.3%	0.2%	0.0%	0.0%		
State Total	33,648	31,640	29,318	29,726	27,804	(1,922)	-6.5%

Figure 72 below shows an excerpt from the pink sheet report compiled during the week ending May 12, 2018. This shows the weekly yield for the year to date, compared against a 3-year average.



2018 California fresh volume of 8,771,784 trays is above the projected total of 6,909,332 trays for this week.

2018 volume projections are calculated using this year's acreage estimates multiplied by the 2-year average yields per acre, per district.

This chart shows 2018 weekly shipments by district vs 3-year average and California total.

Figure 71: Pink Sheet Report from the California Strawberry Commission, volume number 22 (week ending June 2, 2018)

This graph shows that most strawberry harvesting occurs between March and October, which proves the need for integrating the current date as a variable in the predictor model. However, this is still a very wide range, which varies even more between locations. Therefore, further analysis using data on this scale may not be as useful as considering a field in isolation, assuming its history is available.

5. FURTHER DEVELOPMENT

5.1 DATA ACQUISITION

It is inevitable that some methods of data acquisition exist that have not yet been considered for this project. Additional research should be performed to find methods of minimizing occlusion, perhaps through angling the camera better; minimizing cost through purchasing efficient parts to make the end solution affordable for farmers; maximizing image quality, by purchasing newer and better cameras; and minimizing the time required to take a full sample. Obtaining data outside the visible spectrum, such as infrared, may help the detection as well.

5.2 COMPUTER VISION

The Hough transform technique for masking out dirt rows requires fine-tuning on a field-by-field basis, so some generalization or unsupervised learning would help in applying this project to more cases. One idea is training a neural network to automatically extract strawberry objects. This could certainly serve as a good topic for a future thesis. As stated earlier, video mosaicking represents an open problem with the project as well.

5.3 PREDICTION

In the future, additional inputs such as temperature data, fruit size, field health, fertilizer types, local climate, etc. could be added to better guide the predictor. However, extensive research would be required to determine the effects of each of these variables on the harvest date and yield. In some cases, this research is already being performed [4], but these results are still a long way off from being reliable.

Depending on the final method of data acquisition, it may be possible to track individual plants and, more importantly, individual fruits through a season, rather than the full field. Doing so would generate more useful data about a strawberry's lifecycle and could even reveal growth trends based on field location. For example, unequal fertilization or a higher water table could result in strawberries growing faster in one area than another, which is a trend that full-field analysis cannot predict.

Another complication exists is the presence of over 90 variations of strawberries in the same field. These are planted to ensure a steady harvest, as some strawberries will ripen when others are still growing, but this makes building an accurate overarching model much more difficult. Figure 73 shows the different trials and varieties currently planted in Field 25 at Cal Poly San Luis Obispo.

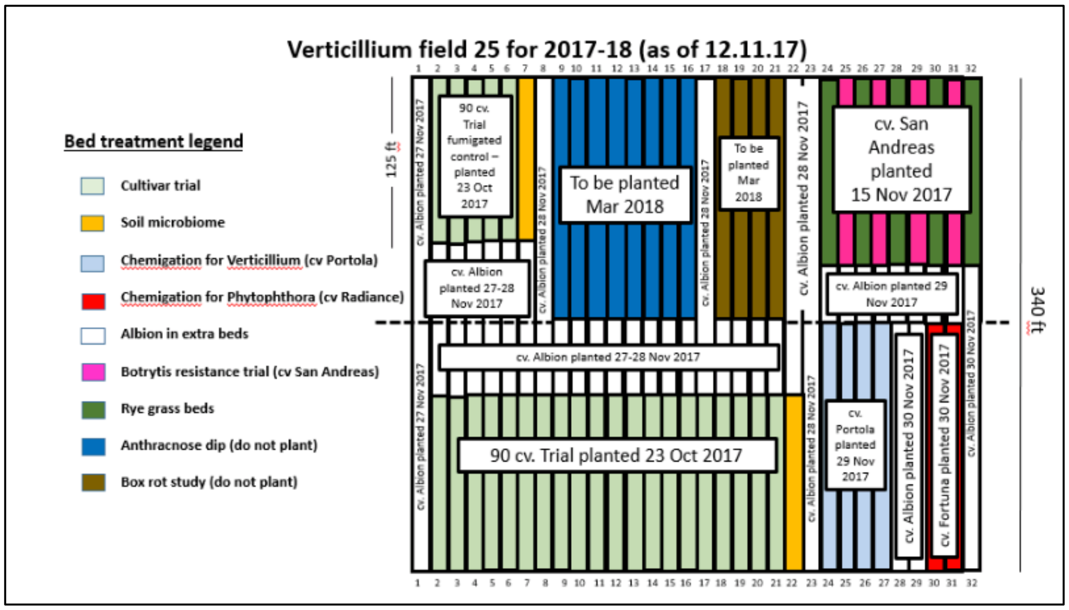

Verticillium field 25 for 2017-18 (as of 12.11.17)
Bed treatment legend
Cultivar trial
Soil microbiome
Chemigation for Verticillium (cv Portola)
Chemigation for Phytophthora (cv Radiance)
Albion in extra beds
Botrytis resistance trial (cv San Andreas)
Rye grass beds
Anthracnose dip (do not plant)
Box rot study (do not plant)
125 ft
340 ft
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
cv. Albion planted 27 Nov 2017
90 cv. Trial fumigated control – planted 23 Oct 2017
cv. Albion planted 27-28 Nov 2017
cv. Albion planted 28 Nov 2017
To be planted Mar 2018
To be planted Mar 2018
cv. Albion planted 28 Nov 2017
cv. San Andreas planted 15 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 30 Nov 2017
cv. Fortuna planted 30 Nov 2017
cv. Albion planted 30 Nov 2017

Figure 72: Map of strawberry varieties currently planted in Field 25

Given the number of experiments running simultaneously in this field, it may not be a good representation of what a real field might look like, but it appears to be a common practice to plant several varieties of strawberries [19].

Comparing the locations in Field 25 analyzed earlier, row 9 and column 14, and row 10 and column 4, there does not seem to be a significant disparity between their growth cycles. Dr. Gerald Holmes, director of the Cal Poly strawberry center, confirms that the different varieties do not exhibit completely different growth cycles. Regardless, it would be helpful to analyze each variety in isolation in the future, since building discrete models may be more helpful than building one for an entire field.

Unpredictable weather also has a large effect on strawberry growth. A sudden cold snap can kill all berries, and excessive rain promotes fungal growth and fruit decay. A "random event" type approach might be a worthwhile addition to the model; perhaps in the month of February, there would be a 0.5% chance each day that inclement weather increases the red strawberry loss rate by 40%.

In fact, a freeze in February 2018 at Cal Poly San Luis Obispo significantly affected the strawberry harvest until new berries reached maturity. According to Dr. Holmes,

> "Anytime temperatures reach 32F or below, the flowers and fruit are destroyed. All plants in the field were affected. Yield went to zero for 5 weeks until new blooms developed into fruit. We've had some frost every year but this year was worse than the previous 3. The areas affected most are those furthest from the ocean, as you might have guessed."

The end user should therefore be conscious of regional weather when considering the validity of the model's predictions.

6. CONCLUSION

This thesis proposed a system to predict the timing and yield of a strawberry harvest using computer vision techniques. The system consisted of three main blocks. For the first block, Data Acquisition, various methods were examined and compared. It was determined that the most promising implementation appeared to be an autonomous rover with cameras mounted on either side, due to the low need for user input and proximity between the cameras and the plants under test. For the Computer Vision block, different parameters for thresholding were investigated. The nRGB and Lab color spaces showed the most promising results for red strawberries. To correct the fisheye distortion problem present in some drone cameras, a method using a checkerboard image dataset was investigated. A method for masking out dirt rows using the Hough transform was also developed, to prevent false positives in strawberry detection. For the Prediction block, in the absence of real data from the Cal Poly Strawberry Center, synthetic data was constructed using a few guiding principles, and a probability distribution was built from multiple fake seasons of data. This was compared against real data from two plots in a strawberry field at Cal Poly San Luis Obispo, as well as trends from the strawberry industry in California.

The question remains: is digital imaging a viable solution for predicting a strawberry harvest? It is still difficult to conclude at this stage how well the full system will perform, but this approach certainly has many attractive attributes, such as being an automated, data-driven, and adaptable approach to forecasting. The computer vision block has excellent performance in recognizing red strawberries, although green strawberries are still a more difficult problem. Most importantly, the best method of data acquisition should be selected and assembled before the next season of strawberries is planted, to obtain a full season of data collected using the same method.

WORKS CITED

[1]: "Searching for Solutions: California Farmers Continue to Struggle with Employee Shortages." California Farm Bureau Federation: Federal Policy Division, 2017, pp. 1–9., www.cfbf.us/wp-content/uploads/2017/10/CFBF-Ag-Labor-Availability-Report-2017.pdf.

[2]: Daniels, Jeff. "From Strawberries to Apples, a Wave of Agriculture Robotics May Ease the Farm Labor Crunch." CNBC, CNBC, 8 Mar. 2018, www.cnbc.com/2018/03/08/wave-of-agriculture-robotics-holds-potential-to-ease-farm-labor-crunch.html.

[3]: Eaglescliffe, Beth. "Robot Strawberry Pickers to Replace Humans." ToughNickel, ToughNickel, 16 Dec. 2017, toughnickel.com/industries/Robot-Strawberry-Pickers-to-Replace-Humans.

[4]: Boivin, Carl, et al. "Developing a Simple Method to Predict Berry Harvest Volume for a given Day-Neutral Strawberry Production Field." IRDA, Jan. 2014, pp. 1–51., https://www.irda.qc.ca/assets/documents/Publications/documents/boivin-et-al-2013_rr_predict_strawberry_harvest_volume.pdf.

[5]: "Mobile Fact Sheet." Pew Research Center: Internet, Science & Tech, 5 Feb. 2018, www.pewinternet.org/fact-sheet/mobile/.

[6]: Horn, Leslie. "43 Percent of People Use Their Cell Phone As Their Primary Camera, Poll Finds." PCMAG, 27 June 2011, www.pcmag.com/article2/0,2817,2387677,00.asp.

[7]: Skafisk, Sven. "This Is How Smartphone Cameras Have Improved Over Time." PetaPixel, 16 June 2017, petapixel.com/2017/06/16/smartphone-cameras-improved-time/.

[8]: Strawberry, Mr. “Matted Row System.” Strawberry Plants. Org, 5 Dec. 2017, strawberryplants.org/2011/01/matted-row-system/.

[9]: “UAV Applications /// Drones for Inspection, Surveying & More.” Ascending Technologies, www.asctec.de/en/uav-uas-drone-applications/.

[10]: “Phantom 4 - DJI's Smartest Flying Camera Ever.” DJI Official, www.dji.com/phantom-4.

[11]: “GDU Byrd Premium 2.0.” GDU Official Store, store.gdu-tech.com/products/gdu-byrd-premium-2-0.

[12]: King, Photo. “DJI Phantom 4 Quadcopter: Camera & Photo.” Amazon.com : DJI Phantom 4 Quadcopter : Camera & Photo, www.amazon.com/DJI-CP-PT-000314-Phantom-4-Quadcopter/dp/B01CFXQZD0.

[13]: “GDU Byrd Premium 2.0 Quadcopter Drone.” Advexure, advexure.com/store/gdu-byrd-premium-2-0-quadcopter-drone/.

[14]: “Convert from HSV to RGB Color Space” MATLAB & Simulink, www.mathworks.com/help/images/convert-from-hsv-to-rgb-color-space.html.

[15]: “Decision Matrix.” ASQ, asq.org/learn-about-quality/decision-making-tools/overview/decision-matrix.html.

[16]: Frooxius. “Growing Strawberry Timelapse (40 Days in 15 Seconds).” YouTube, YouTube, 16 May 2014, www.youtube.com/watch?v=6mL_p2CXVVw.

[17]: “Find edges in intensity image – Matlab edge” MATLAB & Simulink, https://www.mathworks.com/help/images/ref/edge.html.

[18]: “Hough transform – Matlab hough” MATLAB & Simulink, https://www.mathworks.com/help/images/ref/hough.html?s_tid=doc_ta.

[19]: Strawberry. “Strawberry Varieties.” Strawberry Plants . Org, 19 Apr. 2018, strawberryplants.org/strawberry-varieties/.

[20]: "Market Data." California Strawberry Commission, www.calstrawberry.com/en-us/Market-Data.

[21]: "Fisheye Calibration Basics" MATLAB & Simulink, https://www.mathworks.com/help/vision/ug/fisheye-calibration-basics.html.

[22]: "Video Mosaicking" MATLAB & Simulink, https://www.mathworks.com/help/vision/examples/video-mosaicking.html.

[23]: "Feature Based Panoramic Image Stitching" MATLAB & Simulink, https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html.

[24]: "Orthophoto Map and 3D Model Generation" MAPS MADE EASY, https://www.mapsmadeeasy.com/.

[25]: "Powerful Drone & UAV Mapping Software" DroneDeploy, https://www.dronedeploy.com/.

[26]: "iPhone 7 Specs" Apple, https://www.apple.com/iphone-7/specs/.

[27]: Lin, Shizhuang, et al. "ZigBee Based Wireless Sensor Networks and Its Applications in Industrial - IEEE Conference Publication." IEEE Xplore Digital Library, Wiley-IEEE Press, 8 Oct. 2007, ieeexplore.ieee.org/document/4338898/.

[28]: Bouget, Jean-Yves, "First calibration example – Corner extraction, calibration, additional tools" Camera Calibration Toolbox for Matlab, Cal Tech, 15 Oct. 2015, http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html.

APPENDICES

A: Matlab code

https://drive.google.com/open?id=1TAwZqwuhyIbuvjcWobvhJrHnFpRwICyh

B: Test images from Chapter 3's Thresholding section

https://drive.google.com/open?id=1Em6lpS74WYlB0VAec41sGsyD7q5vvLb4

C: Image sets captured by smartphone for ground truth analysis in Chapter 4

https://cpslo-
my.sharepoint.com/:f:/r/personal/jzhang_calpoly_edu/Documents/Strawberry%20Yield%
20Prediction%20Project%202018/Data?csf=1&e=3ZHsSX

D: Fisheye distortion calibration dataset

https://cpslo-
my.sharepoint.com/:f:/r/personal/srchung_calpoly_edu/Documents/Strawberry%20Drone
%20Data/Checkboard%20Calibration?csf=1&e=KTm3eY

E: Mosaicking dataset

https://drive.google.com/open?id=1weSN9WqWa9IslY3pBe3lj5QBICmHnmzO