UNIVERSITY OF CALIFORNIA, MERCED

Path Planning in Vineyards

Andres Torres Garcia

A technical report submitted in partial satisfaction of the requirements for the degree of

Master

Committee in charge:

Professor Stefano Carpin, Chair

Professor Shawn Newsam

Professor Marcelo Kallmann

Summer 2018

Abstract

Automated agriculture will play an important role in the coming years. Water scarcity and manual labor shortage are two important problems to solve in the next decade. One solution to those two problems is to use robots that can efficiently inspect and maintain optimal growing conditions for fields of crops. In this technical report, we present an algorithm that creates a path for a robot to follow through a field of crops. To demonstrate the efficacy of this algorithm, we focus on vineyards. We use a linear actuator to take soil samples while the robot is traversing a vineyard. Autonomous robot navigation is accomplished using the ROS 2D navigation stack.

1 Introduction

Precision in agriculture is becoming more important due to water scarcity in the future. The problem of inefficient use of water in agriculture is exacerbated by rapid population growth. Environmental scientists predict that water scarcity will be a major issue in the coming decades[12].

This project was part of the RAPID (Robot Assisted Precision Irrigation Delivery) and was funded by USDA NIFA. The project consisted of developing a mobile robot capable of maneuvering autonomously between crop rows in vineyards while gathering data, such as soil moisture, from selected spots in the process of navigating through the field.

The body of this report is organized into *Material* and *Methods*, which describes the physical materials employed and the methods used for each step of the project; *Results* which presents the raw data and the calculated data; *Discussion* which draws conclusions about the data, points out limitations and suggests areas of future work. *References* are listed at the end of the report. The report is the result of a project carried out 2017-2018 in Merced, California at the University of California, Merced.

2 MATERIALS AND METHODS

2.1 MATERIALS

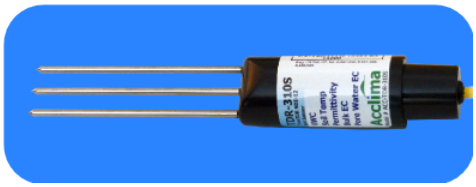
The following hardware was used to implement the autonomous navigation:

- Husky unmanned outdoor field robot
- Two Topcon Hyper GPSs
- One Qstarz GPS
- One IMU
- One SICK laser

2.1.1 Soil Moisture Devices

In addition to navigating autonomously through the crop rows, the robot also needed to take soil samples from selected spots. The robot was equipped with a ZABER linear actuator and attached to it a soil moisture sensor ACCLIMA TDR-310S. Figure 1 shows the two of them. The code for these two devices was implemented as ROS services. Figure 2 shows the description of the ROS services of each device.

(a) The Zaber A-LAR-E Series linear actuator



(b) The ACCLIMA TDR-310S soil moisture sensor

Figure 1: The soil moisture sensor and the linear actuator

int8 cmd
\- - -
int8 response

(a) The actuator ROS service

\- - -
int8 response
float32 moisture_reading

(b) The soil moisture ROS service

Figure 2: The soil moisture and linear actuator ROS services

The linear actuator and a device which measured soil moisture attached to the end of the actuator worked together in the following manner: Once the robot was in a spot selected for soil sampling, it called the ROS service to lower the actuator. After the actuator had been lowered to a suitable level and taken a soil sample, the robot called the moisture sensor service to measure the moisture level. The reading from the soil moisture sensor took less than a second. Once the reading was finished, the robot lifted the actuator to its initial/home position then continued navigating.

2.2 METHODS

In this case there was a robot on the ground moving through roads. The roads were the paths between the rows of field crops. For a robot to be able to traverse crop rows, it was crucial to create a path plan. In order to build a path plan, a map of the terrain was required. The terrain map had to be created either from satellite imagery or from drone images. In this work, the map was created from images gathered from Google Map satellite imagery API.

At an early stage of this project, a technique for map creation was implemented. This implementation used Canny Edge Detection [1] and the Hough transform [7] to detect crop row edges. Figure 3 shows the flowchart of the implementation just described.

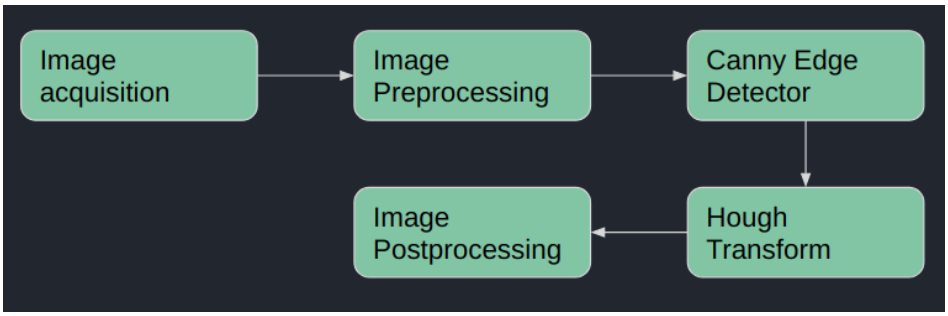


Figure 3: Previous method flowchart

This map creation technique worked well in detecting lines, but those lines could be either edges of roads or edges of crop rows. Figure 4 shows an output image resulting from this method.

Figure 4: Previous method result. Here the detected lines in red are the shade of each crop row.

In the process of writing an algorithm that creates a path for the robot to traverse through a field of crops, a new method for detecting roads was developed. Road segmentation from satellite imagery was done using: color-based segmentation [4], K-Means Clustering [6] and a common classifier k-nearest neighbors algorithm(k-NN) [3].

2.3 Related Work

In the literature, most of the methods proposed for crop row detection in the literature employ either vision-based agricultural machinery guidance systems [11], [8] or images taken with unmanned aerial vehicles (UAVs) [10], [2] with superior resolution. The work found in the literature that most closely related with our work [9], used panchromatic WorldView-2 Satellite Imagery which provided 50 cm resolution. That level of resolution is far superior to the resolution that can be seen on images gathered from the Google Map

API. In our work, we developed a method of detecting paths between crop rows so that a robot could navigate autonomously along the crop rows using imageries freely available from Google Map API.

2.4 Assumptions

Before delving into detail, let's be aware of the following assumptions, namely, all the vineyards tested in this project were horizontally oriented to the north grid, the vineyards were rectangular, the vines ran from east to west, the crop rows were equally spaced and the images only contained vines.

For the rest of this section we will show the list of steps utilized to generate a series of waypoints that the robot had to follow to traverse a vineyard. In the first phase, a series of images that compounded a whole vineyard was gathered. The second phase was to determine where the roads were. Roads were where weeds grew and where free space existed. The free space between the crop rows was detected using basic algorithms of image processing and computer vision. In the third phase, a series of waypoints was calculated from the roads to later sort for autonomous robot navigation. Here are the sequential steps that were used. Figure 5 shows the flowchart followed.

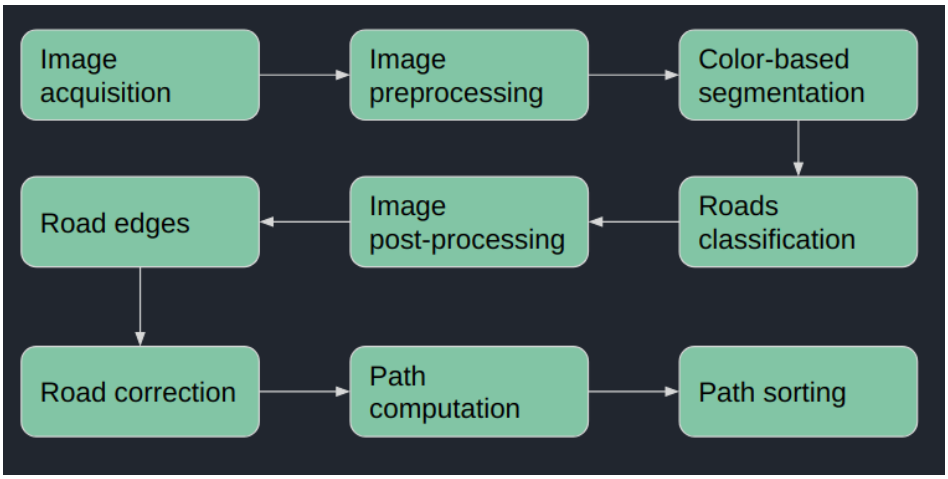


Figure 5: Crop Row Detection Flowchart

2.5 First Phase

2.5.1 Image Acquisition

The images acquired during this step were gathered from the Google Map API. A Python script that uses functions from the Google Map API to assemble a series of images that compound a whole vineyard was developed. For the purpose of this algorithm, the images were not stitched together. (See further explanation in section 2.6.3). The inputs of this script were the four corner points that defined the rectangular layout of a vineyard.

The script also output metadata for each of the images. This was how a metadata file looked :

- Image format: PNG
- Zoom level: 20
- Image size: 1320
- Top left GPS coordinate of the image: lon=-120.21432, lat=36.84329612604934
- Meters per pixel: 0.14929106712341308

The number of images depended upon the area of the vineyard, the zoom level and the image size. Figure 6 shows a sample image.

Figure 6: Sample Image

2.6 Second Phase

2.6.1 Road Detection

The roads, that were between the crop rows, were detected using simple methods of image processing and computer vision. First, the images were preprocessed by removing the Google banner that was on every image downloaded from the Google Map API. Second, the images were color-based segmented and the data was trained and classified by using a supervised learning algorithm. Third, the images were post-processed by computing morphological operations on each image to remove noise from the roads.

2.6.2 Image Preprocessing

Figure 4, clearly depicts a Google banner in the bottom left part of the image. This banner had to be removed for proper RGB segmentation, therefore part of the image was removed to get rid of that banner.

The output image of this step is shown in Figure 6.

2.6.3 Color-based Segmentation

The roads were easily distinguished from the crop rows and the shade because of the distinct variation in pixel coloring. The crop rows in the pictures gathered contained mostly green pixels. The roads, between each crop row, were mainly light brown pixels, and the shade was generally black or close to black pixels.

For the color-based segmentation, K-means was used to perform the pixel clustering. As previously stated, the value of $K = 3$ was chosen because there were three evident classes, namely, crop row, roads, and shade.

- Crop row: mostly green pixels
- Roads: mostly light brown pixels
- Shade: mostly black pixels

Figure 7 shows these three classes. The three images resulting from this clustering, namely, crop row: mostly green pixels; roads: mostly light brown pixels; and shade: mostly black pixels, are shown in Figure 8. Figure 8 shows the original image (top left) alongside three images which correspond to each of the three clusters (top right: crop row; bottom left: roads; bottom right: shade).

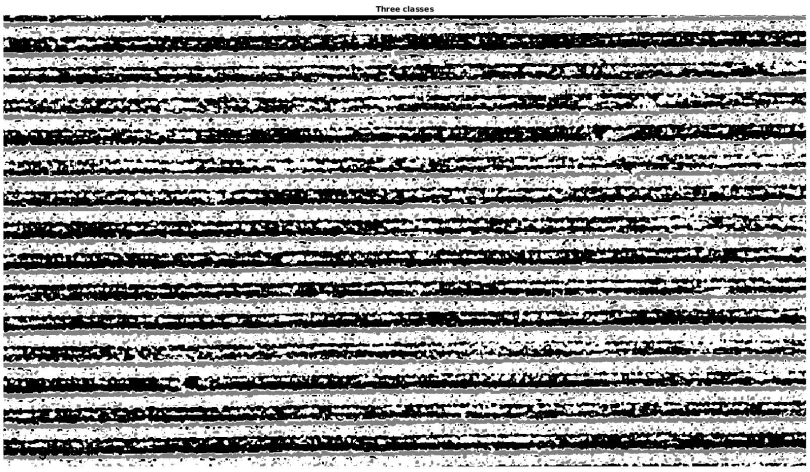
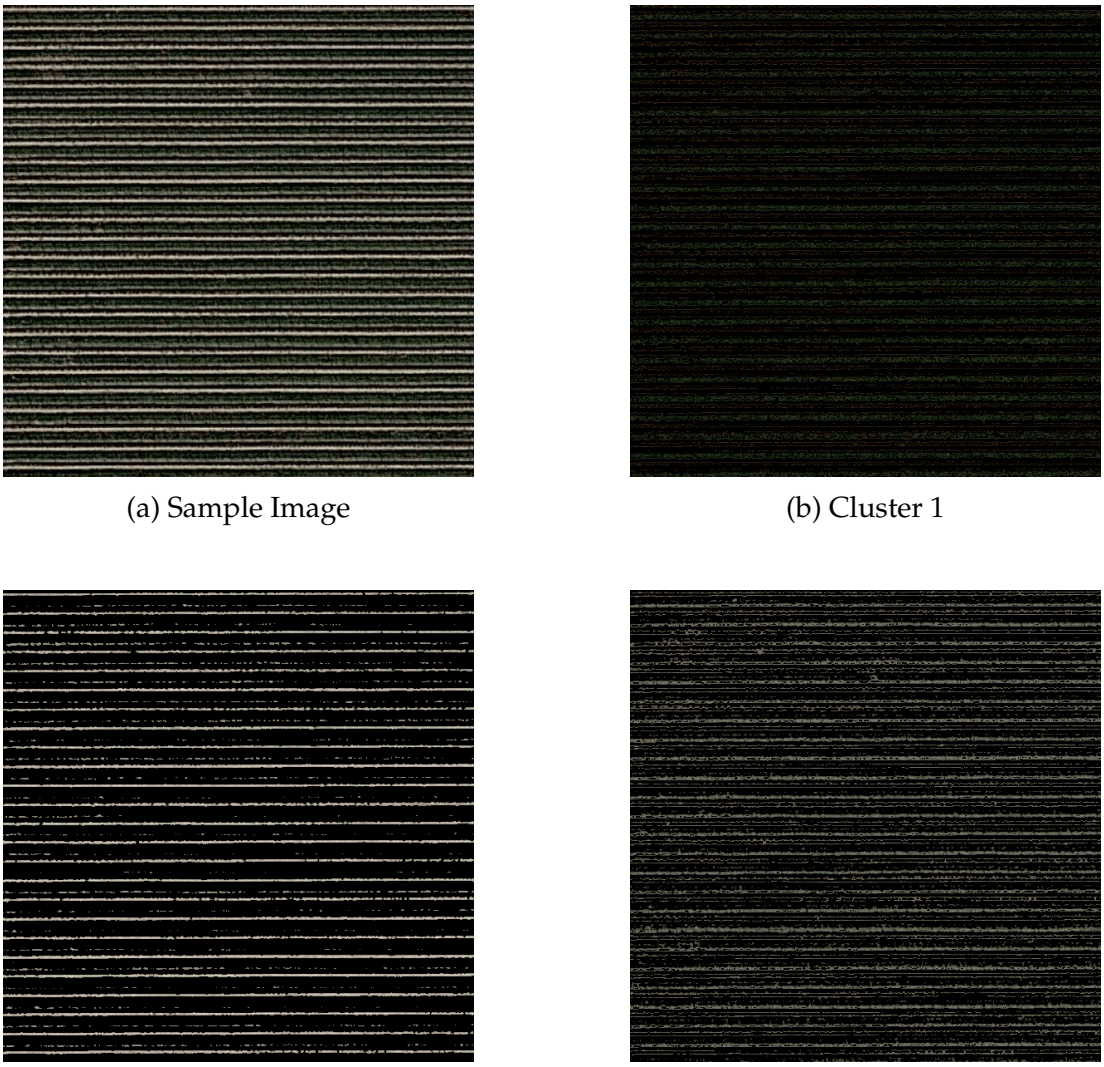


Figure 7: Three classes in sample image: Shade in gray, Roads in white and Crop rows in black



(a) Sample Image (b) Cluster 1

(c) Cluster 2 (d) Cluster 3

Figure 8: Color-based Segmentation of the sample image

2.6.4 Roads Classification

The output of K-means was three RGB clusters. However, having these three clusters, the program did not know which cluster was the one that corresponded to the roads. Therefore, we needed to use a classifier algorithm: k-NN. k-NN uses the heuristic of classifying points close in distance to certain known class. We used the Euclidean distance as a metric. This algorithm needed a dataset to make predictions. We worked with datasets of 5 images. The larger the dataset of user-selected clusters, the more accurate the subsequent automated classification of clusters. Sometimes roads contained weeds or the weed pixels were dark light brown instead of green. Since every vineyard was different, this dataset was created at runtime. The program asked the user to classify the cluster that contained roads for the first 5 images of the vineyard. Having the dataset of user-selected clusters, the program could classify all the clusters which followed by itself

using the k-NN classifier algorithm. In the following example, the image with roads was the image located at the bottom left in Figure 8.

2.6.5 Image post-processing

Figure 9 shows the salt and pepper noise in the image after classification. This noise was due to the fact that not all the pixels in the crop rows were green. Moreover, not all the pixels in the roads were light brown because of the presence of untrimmed weeds. Figure 10 shows the binary image without the salt and pepper noise.

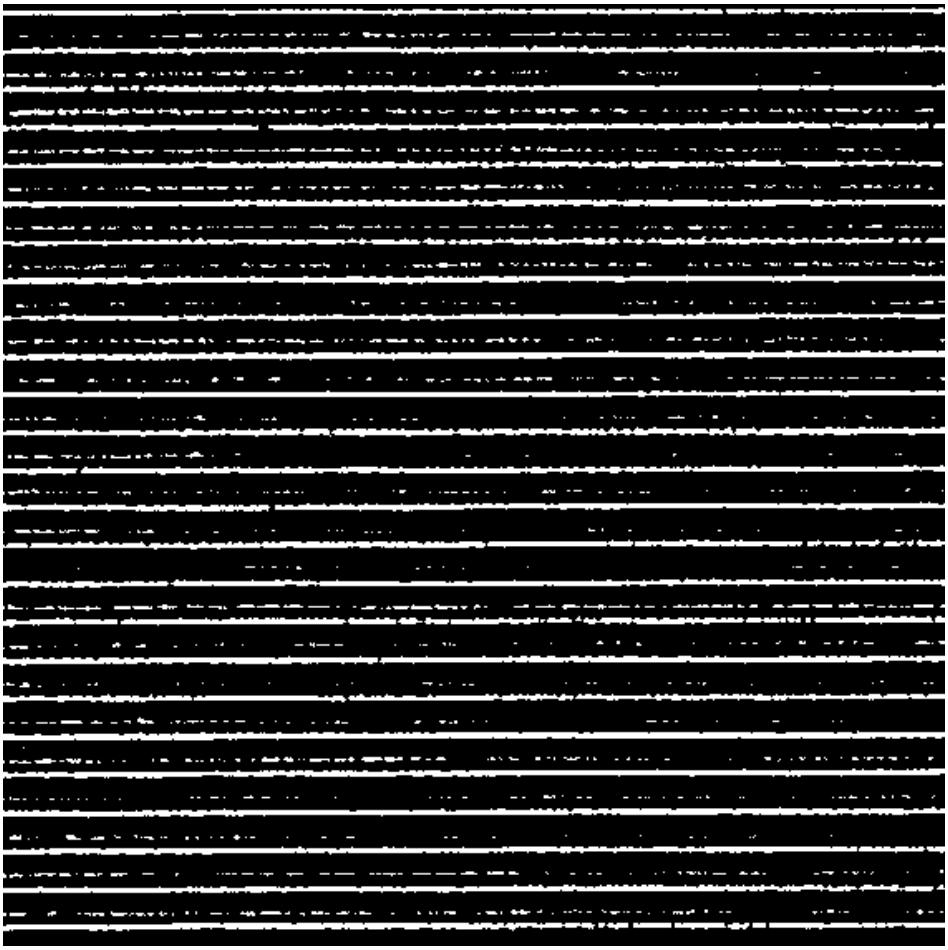


Figure 9: Salt and pepper noise in roads blobs

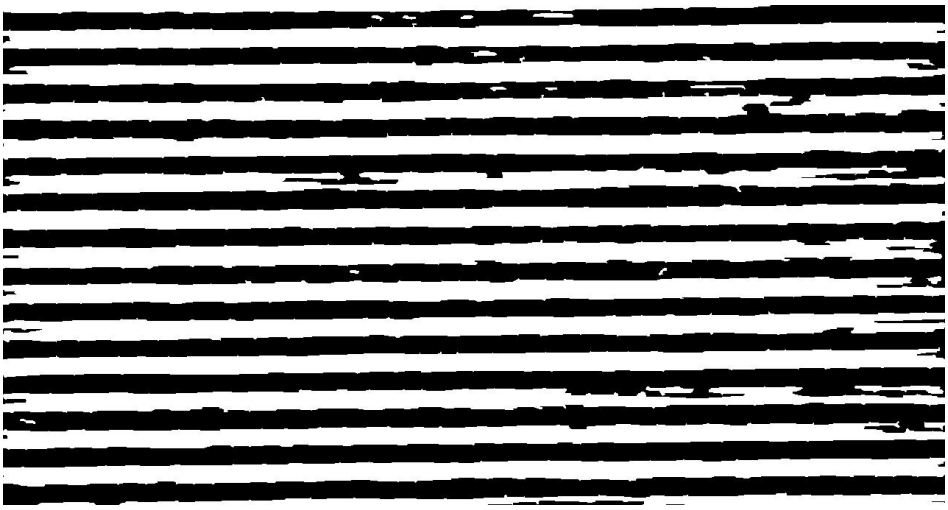


Figure 10: Without Salt and pepper noise in roads blobs

After removing the salt and pepper noise, the image still might have had noise in the form of blobs. The noisy blobs and the half road blobs were normally smaller in both overall area and width than the road blobs. These relative size features were used as an input for the K-means. In this particular instance, the value of $K = 2$ was used. The two classes were: (1)road blobs, and (2) half road blobs and noisy blobs.

Figure 11 shows the blobs. Yellow corresponds to road blobs, blue corresponds to half road blobs and red corresponds to noisy blobs. The half road blobs and the noisy blobs were removed, leaving only the road blobs in the image. We kept the cluster that contained the blobs which had the biggest areas. Figure 12 shows the resulting image.

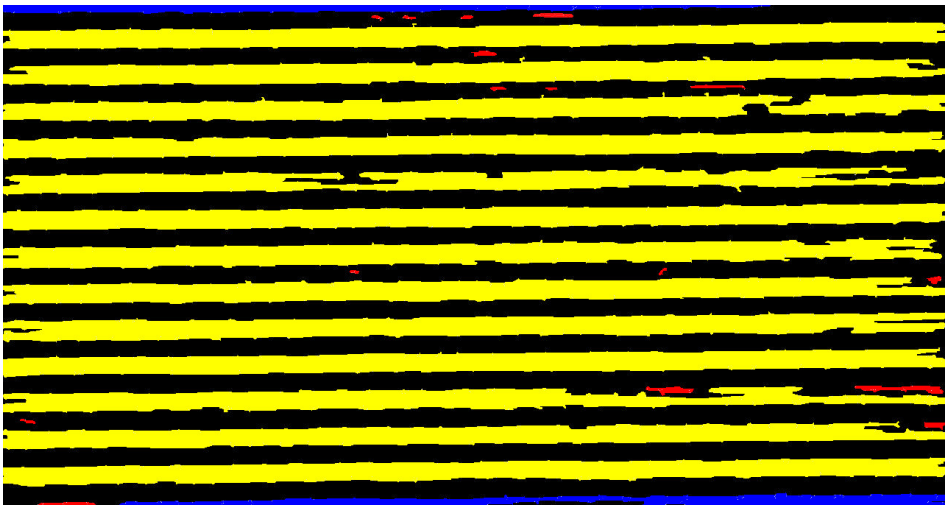


Figure 11: Noise in road blobs

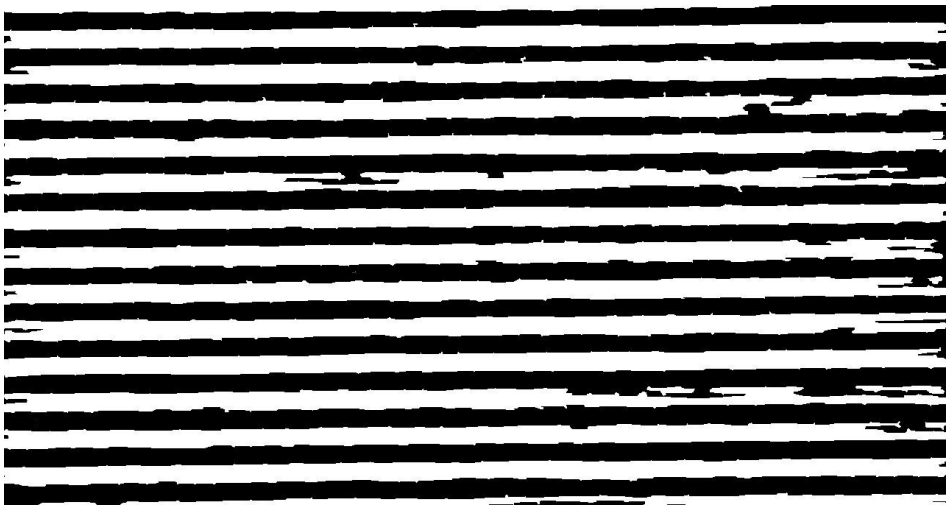


Figure 12: Road blobs

2.7 Third Phase

2.7.1 Straight line edge computation

Given the road blobs from the color-segmentation, now the task was to calculate their straight line edges. The straight line edges for each road blob (top edge and bottom edge) were computed using RANSAC[5]. RANSAC was very useful for parameter estimation of 2D models from a set of observed data points. The topmost and bottommost points from each road blob were used as input for this algorithm. Algorithm 1 shows how to compute the straight line edges and Figure 13 shows the sample image with the straight line edges on each road detected.

Algorithm 1 Straight Line Edges algorithm

```
1: procedure GetRoadEdges
2:     for each blob ∈ blobs do
3:         topedge ← topmost points of blob
4:         botedge ← lowest points of blob
5:         topedge ← RANSAC(topedge)
6:         botedge ← RANSAC(botedge)
7:         store topedge and botedge
```

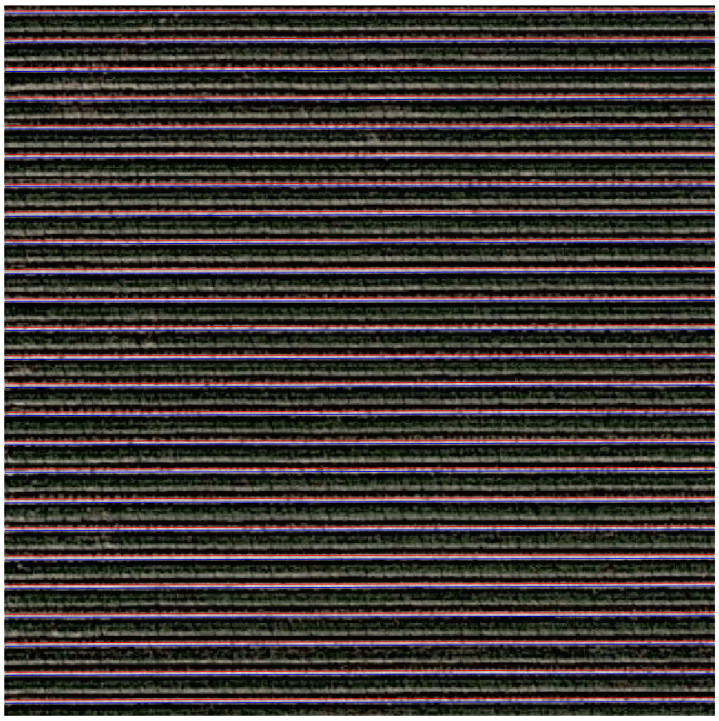


Figure 13: Straight line edges on each road detected. Red lines are top edges and blue lines are bottom edges respectively.

2.7.2 Road Correction

Sometimes the algorithm was not able to detect all the roads from all the images, so the number of roads in the images needed to be corrected. First, the algorithm gathered important information from all the images, such as the width of the roads, the slope of the roads and the space between roads. Second, using that information, the algorithm checked if there were missing roads in the image. Note that the roads ran east-west. When we refer to a pair of edges, we are referring to the north edge and south edge of an east-west road. The method to detect missing roads was the following: The algorithm started checking from the northernmost road detected. The road was defined by a pair of parallel edges (north edge and south edge). From there, parallel edges were drawn above them using the information previously gathered. If the algorithm was successful, the drawn edges would be inside the image. And in that case those drawn edges were added to the image. These steps were repeated until the drawn edges were not inside the image. The same process was repeated but instead of introducing drawn edges above the ones in the image, parallel lines were drawn below the southernmost road detected in the image. The final step was to loop through all the pairs of edges detected in the image and see if there were any missing edges to be filled in so that in the end, the resulting image would have roads with intact edges.

2.7.3 Waypoint computation

Computing the waypoints on the road was readily achieved by calculating the medial axis between the two road edges. While doing that, the pixel coordinates for each image were converted to their equivalent in GPS coordinates. Figures 14 and 15 shows the waypoints in pixel coordinates and in GPS coordinates respectively.

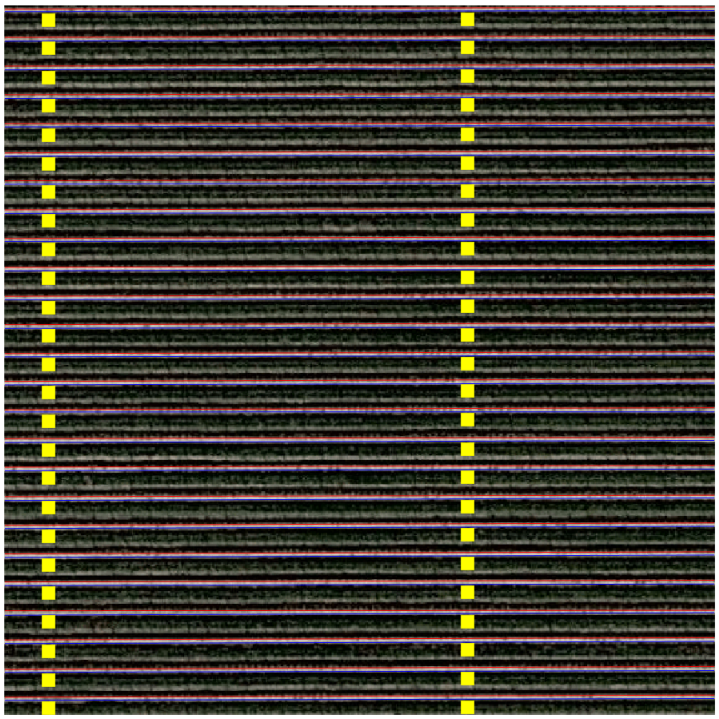


Figure 14: Waypoints in pixel coordinates in yellow

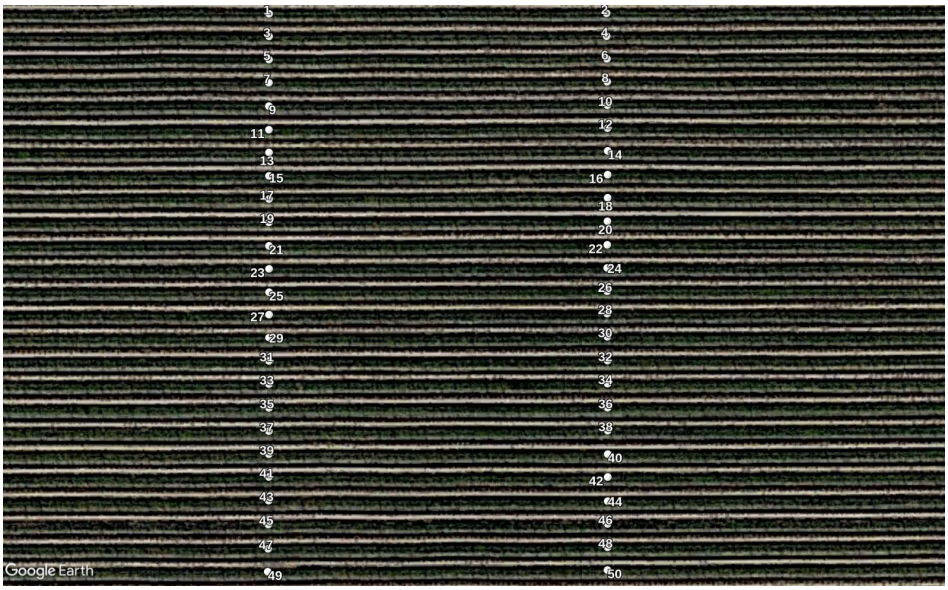


Figure 15: The GPS coordinates numbered in ascending order

As you can see in Figure 14, the yellow pixel coordinates, are located below the edges of each row. This is due to the fact that the Google Earth images were taken with perspective. This perspective causes part of the roads be occluded by vines. Therefore, the road detection phase of the work only detected the parts of the roads that were not occluded. Having the upper edge of the roads detected, we shifted the pixel coordinates by half of the width of the roads in pixels to move the pixel coordinates at the center of the corresponding road.

2.7.4 Waypoint sorting

Each image was preprocessed, color-segmented, trained and classified. After processing all the images individually, there was a need to sort the GPS points from all the images for autonomous waypoint robot navigation. That was one of the reasons that we did not stitch all the images together. Instead of creating a high resolution image, a series of individual images was processed and subsequently their respective waypoints were put in order. Another reason for not stitching all the images together was to avoid the Mercator projection error that incrementally increases as an image gets bigger.

Ordering the points belonging to the same crop row in a row and later sorting them in a waypoint fashion required multiple steps. First, they were sorted by their latitude coordinate. Second, we calculated the distance of each waypoint to the top left corner of the vineyard. From that collection of distances an array resulted. Third, we computed the difference between adjacent distances in the array. Fourth, we looped through each distance difference adding waypoints to a row until a difference greater than a positive threshold was found. Finally, each row was ordered by its longitude coordinate by intercalating them in ascending and descending order respectively. This form of sorting is referred to as zigzag. In this report a threshold value equal to 100 was used. Figure 16 shows sorted in zigzag order.

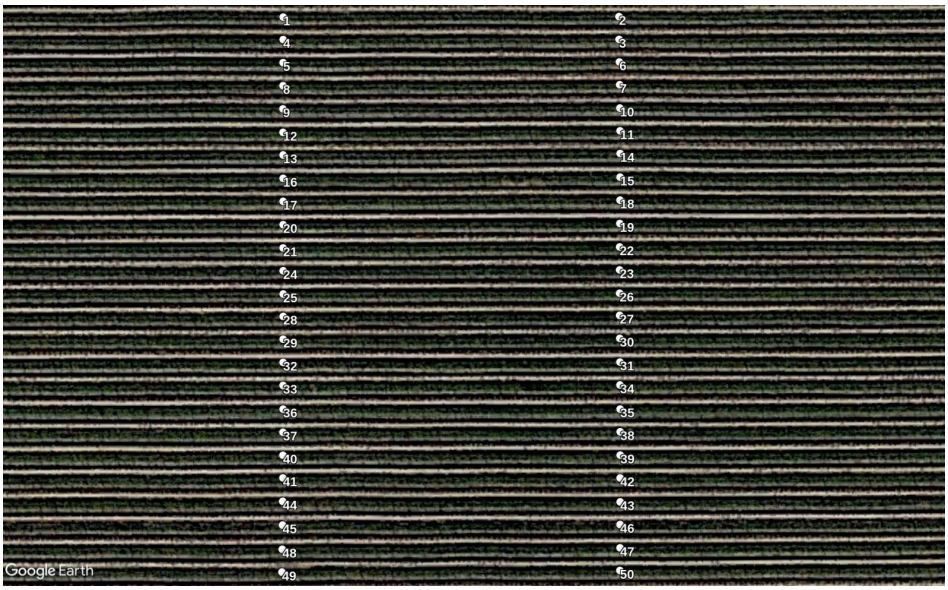


Figure 16: GPS points sorted in zigzag order

2.7.5 Row selection

For autonomous robot navigation in vineyards, normally, the robot does not need to go through every crop row. That is why it was required to have an option to select specific rows. Figure 17 shows selection of every two row.

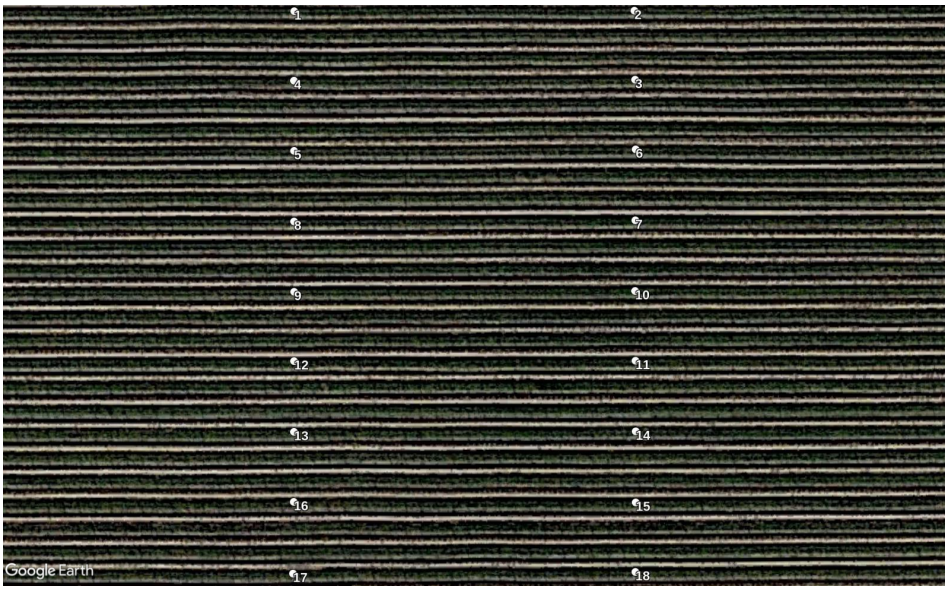


Figure 17: Selection of every two row

2.7.6 Waypoints connection

This is how each road was dealt with. After calculating the waypoints of a given road, we connected those waypoints with waypoints outside of the vineyard at both ends of that

road. This calculation was necessary in order to allow the robot to move from one road to another. For each road, we shifted the first waypoint by a negative number of meters, and we shifted the last waypoint by a positive number of meters. Figure 18 is an example showing the connecting waypoints for two adjacent parallel roads.

(a) Left end (b) Right end

Figure 18: Connecting waypoints in both ends for two roads

3 RESULTS

3.1 Test Images

In order to evaluate the accuracy of this algorithm, we gathered images from the following vineyard: Ripperdan Ranch with coordinates $Latitude : 36°50'23.6"N, Longitude : 120°12'43.8"W$. Figure 19 shows the satellite image of the vineyard.

Figure 19: Ripperdan Ranch vineyard highlighed in red

For this vineyard, we did not gather any single image containing the entire vineyard, but rather we collected a series of images of all parts of the vineyard. We split the vineyard

into the upper half of the vineyard and the lower half of the vineyard. Figure 20 shows this division. Some parts of the land did not contain vines or had noise such as smoke.

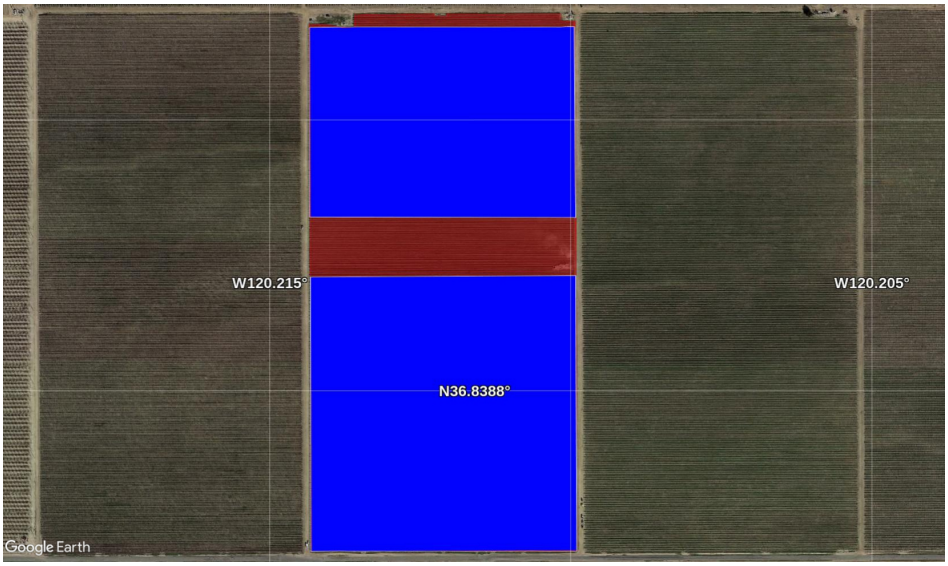


Figure 20: Ripperdan divisions in blue

3.2 Outputting GPS Coordinates

The road detection and waypoint computation were performed for each of the images of the vineyard. The resulting waypoints, after calculating the GPS coordinates for each of the images, is shown in Figure 21. Finally, Figure 22 shows the waypoints of the upper half after sorting the GPS waypoints in zigzag order and selecting every twenty-fourth row.

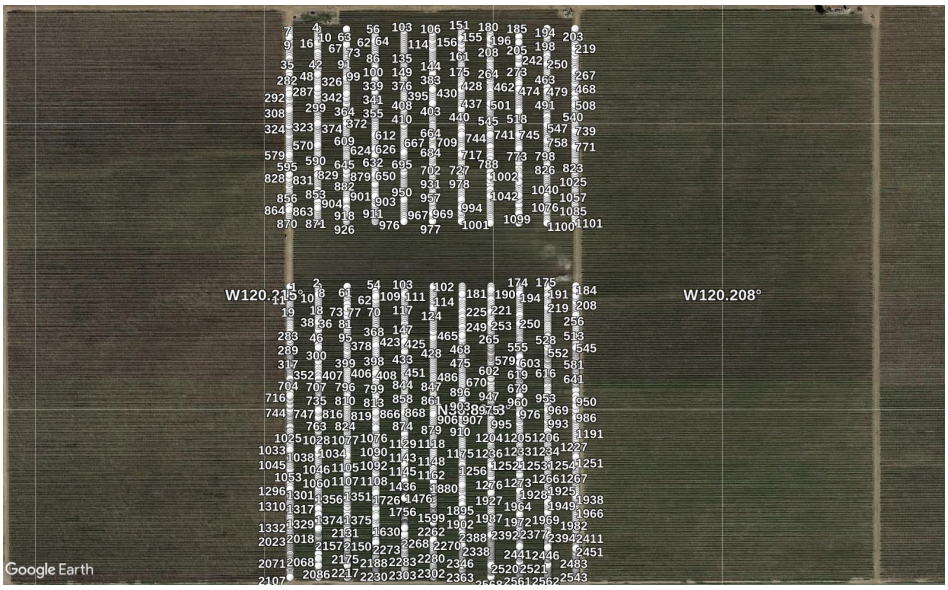


Figure 21: GPS points

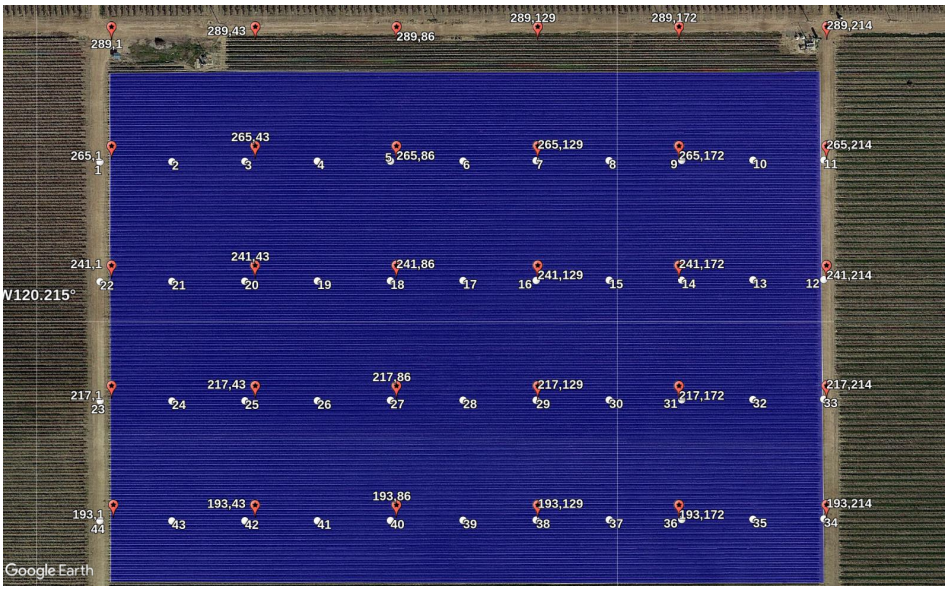


Figure 22: GPS waypoints of the upper half vineyard

4 DISCUSSION

4.1 Conclusions

An algorithm was developed that outputs a series of waypoints from satellite imagery. The waypoints helped ground robots navigate through vineyards. The algorithm used basic techniques in image processing and computer vision to detect roads in vineyards, compute the road edges and calculate a list of GPS waypoints to follow.

4.2 Limitations

Although it had some limitations, e.g. noise in satellite imagery; the need to select a dataset in the beginning of the program; and the acceptance of only north oriented rectangular vineyards, it was fairly robust. As long as the algorithm detected one road per image, it could easily compute the other roads using the meta-information e.g. road width, space between roads and edge slope from the detected road. The reason for this computational ability was the fact that crop rows are equally spaced in vineyards as in any other standard crop field. Therefore, we can safely assume that the computed calculation is correct. Moreover, the limitation that the algorithm is sensitive to noise in the images and the limitation that the algorithm only accepts rectangular vineyards that are oriented to the North Pole could be easily overcome by preprocessing the images before putting them into the algorithm. However, the limitation that the algorithm depends on the user to be trained in the selection of the pixels that correspond to the roads is more difficult to overcome. One naive solution might have been to assume that the pixel roads were going to be the lighter-colored pixels in the image. Most of the time the light brown pixels represented the roads, however this was only the case when the weeds were trimmed.

When weeds were not trimmed they could show up in the image as green pixels, like the crop rows which also appeared as green pixels.

Another important limitation was the fact that satellite imagery has projection, i.e, the images are not taken directly above vineyards. One of the consequences of this projection is the creation of a certain level of road occlusion. This road occlusion caused the algorithm to only detect partial rows. This was easily solved by knowing the width of the roads in advance.

4.3 Future work

Future work includes overcoming these three limitations. The first limitation is that the algorithm is sensitive to noise in the images. For example, if the image has some sort of smoke over the vineyards, it can affect the RGB segmentation. The second limitation is that the algorithm depends on someone to train it by selecting the pixels that correspond to the roads. The third limitation is that the algorithm only accepts rectangular vineyards that are oriented to the North Pole.

References

[1] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[2] Ana I. de Castro, Jorge Torres-Sánchez, Jose M. Peña, Francisco M. Jiménez-Brenes, Ovidiu Csillik, and Francisca López-Granados. An automatic random forest-obia algorithm for early weed mapping between and within crop rows using uav imagery. *Remote Sensing*, 10(2), 2018.

[3] D. A. Forsyth and J. Ponce. Example: A nonparametric classifier using nearest neighbors. In *Computer Vision a modern approach*, pages 469–470. Pearson, 2012.

[4] D. A. Forsyth and J. Ponce. Image segmentation by clustering pixels. In *Computer Vision a modern approach*, pages 268–269. Pearson, 2012.

[5] D. A. Forsyth and J. Ponce. Ransac: Searching for good points. In *Computer Vision a modern approach*, pages 302–305. Pearson, 2012.

[6] D. A. Forsyth and J. Ponce. Segmentation using k-means. In *Computer Vision a modern approach*, pages 272–273. Pearson, 2012.

[7] D. A. Forsyth and J. Ponce. The hough transform. In *Computer Vision a modern approach*, pages 290–293. Pearson, 2012.

[8] Guoquan Jiang, Zhiheng Wang, and Hongmin Liu. Automatic detection of crop rows based on multi-rois. *Expert Systems with Applications*, 42(5):2429 – 2441, 2015.

[9] Melkamu Meseret-Alemu. Automated farm field delineation and crop row detection from satellite images, 2016.

[10] M. Pérez-Ortiz, P. A. Gutiérrez, J. M. Peña, J. Torres-Sánchez, F. López-Granados, and C. Hervás-Martínez. Machine learning paradigms for weed mapping via unmanned aerial vehicles. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.

[11] C. Tu, B. J. van Wyk, K. Djouani, Y. Hamam, and S. Du. An efficient crop row detection method for agriculture robots. In *2014 7th International Congress on Image and Signal Processing*, pages 655–659, Oct 2014.

[12] J. Wallace. Increasing agricultural water use efficiency to meet futurefood production. *Agriculture, ecosystems and environment*, 82(1):105–119, 2000.