

**Potato Field Drone Analysis**

**James Currier**

**Abstract**

An overview of the potato field drone analysis project. The purpose of the project was to gather better statistics on the planting of the potato fields so that improvements in planting accuracy could be made to maximize the yield of each field. This was done with the use of drones and image processing in the Python 3 programming language.

**Introduction**

Having the correct distance between potato plants in a field is essential for maximizing yields. Currently, the distances between plants are being measured manually on a ten-foot strip of each field. This small sample size does not adequately represent the planting of the entire field. To get a better estimate of the distances between plants in a field, it was proposed that drones be used to analyze large sections of the field, to give a much better representation of the planting accuracy. Identifying problems in each field using drones is the first step to improving each field’s yield.

**Approach**

A drone was flown over the fields to take photos at a height of 20 meters. The timing of our sample collection was important, as the plants had to be at the correct stage of growth. If the growth was too little then the plants would not be visible from the drone, but if the growth was too much then the plants would be overlapping, and it would be difficult to distinguish individual plants. Some of the images were then preprocessed by straightening them and removing trees or other objects that are not part of the field. This was done to make it easier for the program to handle the photographs. The program was then used to process the images. The program outputted the analyzed images, and additional data about each image.

**Algorithm**

The algorithm, written in Python 3, takes a photo of the field as an input, and outputs photos with information drawn on them, as well as two excel sheets with additional information. The algorithm first translates the entire picture into an array. It then iterates through the array, looking for green pixels. When it finds a green pixel, it finds the green pixels near it, and considers the whole cluster of green pixels a plant. After finding all the plants, it begins to make rows out of the plants. It does this by finding the angle of the plants in the photo, and then searches the image for local minimums of green, along lines of this angle. These local minimums are the ditches between the plants. After finding the ditches, the algorithm completes the rows by organizing plants between the ditches. To find the distances between the plants, the median number of pixels between rows is found. Since the rows are on average 36 inches apart, the number of pixels to 36 is a ratio to convert pixels to inches. The number of pixels between plants in each row is then counted, and converted to inches by the pixels to inches ratio. Finally, the image is drawn on using the Python Imaging Library (PIL), and the additional information is exported to CSV files.

**Pseudocode**

Convert the image to a 2D array of RGB pixels

Iterate through each pixel in the array

If the pixel is green

Find the cluster of green pixels near it

Consider this cluster a plant

Find the average slope of the field

Find local minimums of green in the field along lines of this slope

Split the plants into rows between these local minimums

Sort all of the rows

Create a line of best fit for each of the rows

Find the distances between each line

Define a pixel to inches ratio as the median of these distances to 36 inches

Iterate through each row of plants

Find the number of pixels between this plant and the next one

Convert this to a distance in inches using the pixels to inches ratio

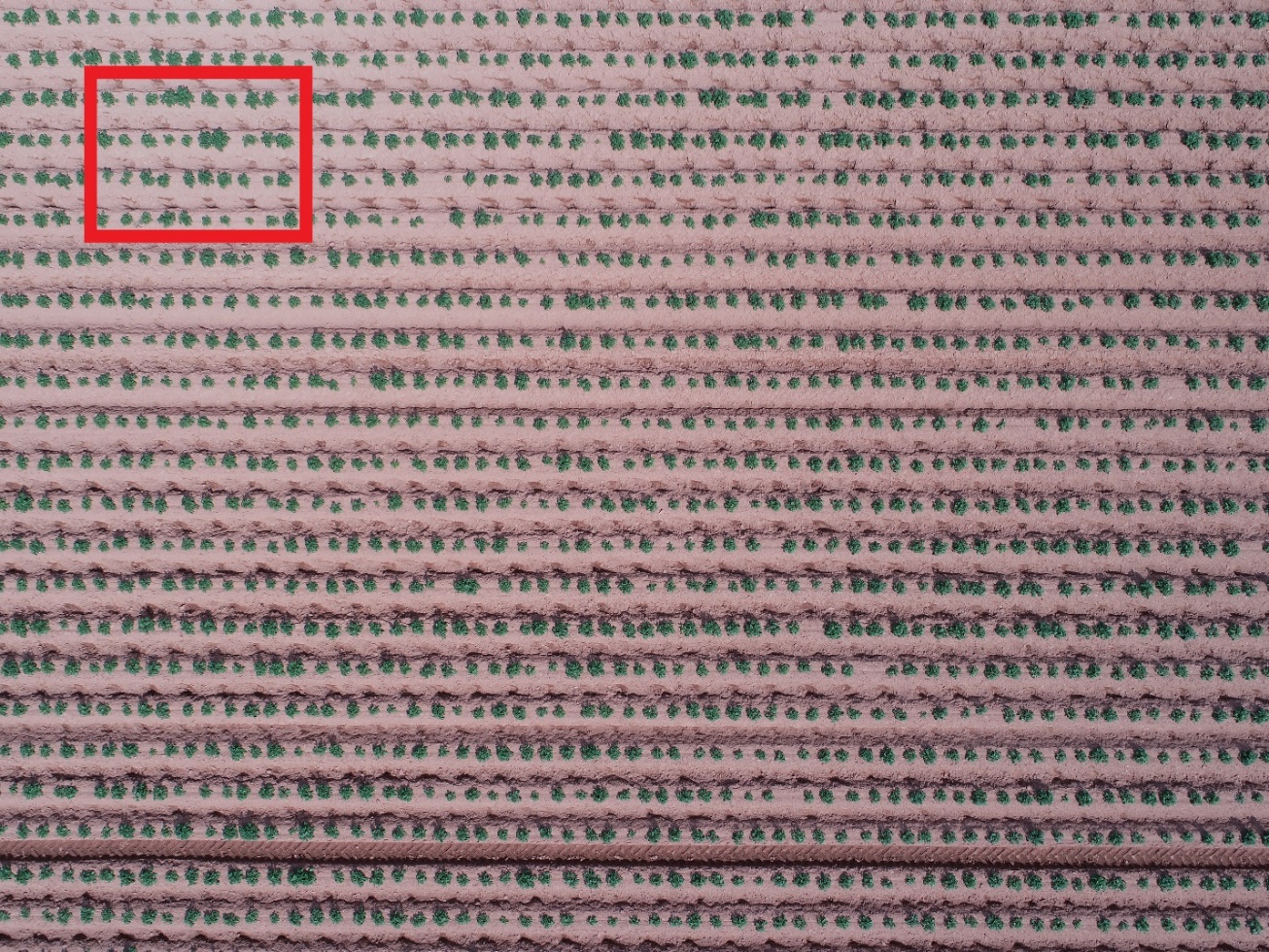
Draw plant centers, rows, and distances on the image

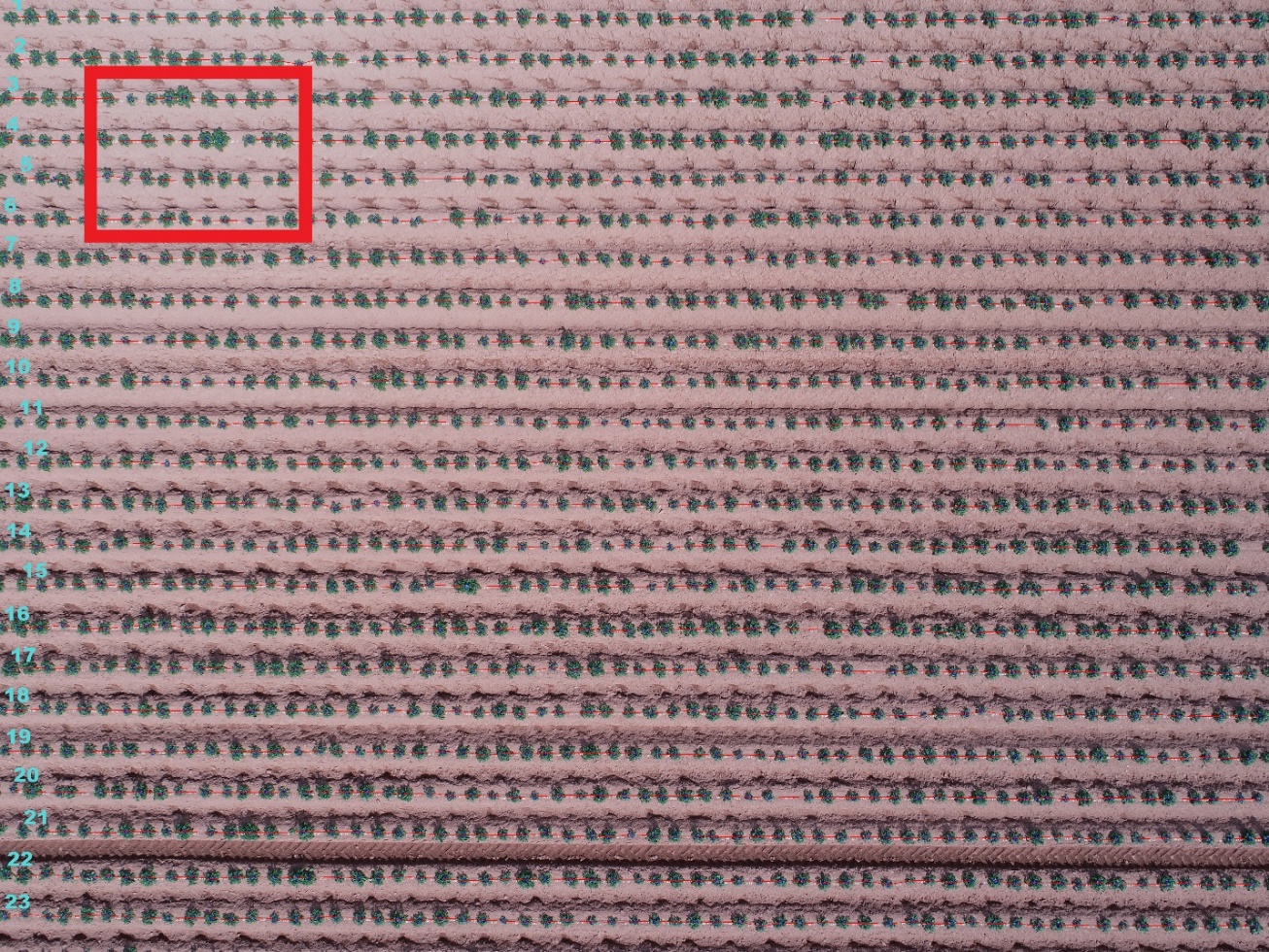
Calculate statistics about the field

Output the analysed image and statistics

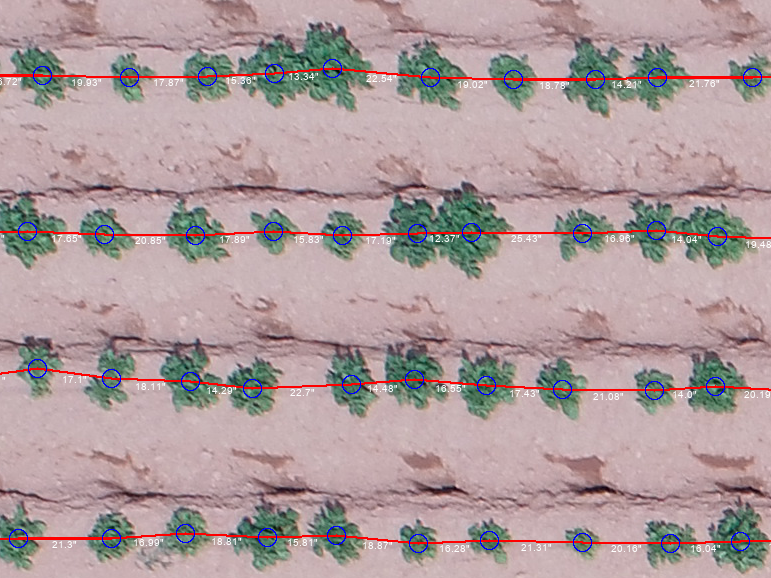
**Example Imagery**

This image was taken with 4K resolution at a height of 20 meters.

**Original Image:**

**Algorithm Output:**

**Original zoomed in on the red box:**

**Algorithm output zoomed in on red box:**

**Additional information outputted to CSV file:**

Distance Information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Gap ID | Distance (Inches) | Row ID | Latitude | Longitude | Image Name |
| 1 | 16.76 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 2 | 19.26 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 3 | 19.05 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 4 | 16.51 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 5 | 18.59 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 6 | 18.1 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 7 | 16.09 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 8 | 16.96 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 9 | 18.55 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 10 | 18.57 | 1 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |

**1355 more rows**

Row Information

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row ID | Ideal Number of Plants | Actual Number of Plants | Average Gap (Inches) | Median Gap (Inches) | Standard Deviation (Inches) | Latitude | Longitude | Image Name |
| 1 | 69 | 62 | 17.99363 | 18.09755 | 3.370222 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 2 | 69 | 61 | 18.30366 | 17.97149 | 3.316894 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 3 | 69 | 61 | 18.37137 | 17.87438 | 3.421595 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 4 | 69 | 61 | 18.30486 | 18.11421 | 2.932955 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 5 | 68 | 61 | 18.12904 | 18.26988 | 3.280627 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 6 | 69 | 60 | 18.62742 | 18.3266 | 3.38308 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 7 | 69 | 61 | 18.31997 | 18.68286 | 2.442246 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 8 | 69 | 60 | 18.70603 | 19.03443 | 3.755257 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 9 | 69 | 61 | 18.41405 | 18.55777 | 2.533758 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |
| 10 | 69 | 60 | 18.70521 | 18.33805 | 3.109029 | 46.47725 | 63.63119 | DJI\_0009\_ChappelSiloWhole.JPG |

**13 more rows**

**Accuracy**

The algorithm was manually audited by randomly selecting 8 images. As shown below, the results are impressive. It is noted that the lower accuracies are from fields that are too emerged (e.g. plants are touching). T-test significance is set to 99%:

|  |  |
| --- | --- |
| **Min Accuracy** | 98.237% |
| **Max Accuracy** | 100.000% |
|  |  |
| **Average Accuracy** | 99.15% |
| **Margin of Error** | 0.78326% |
| **Lower Confidence** | 98.36% |
| **Upper Confidence** | 99.93% |

**Future Improvements**

* Autonomous preprocessing - automatically straighten images and remove trees and other unwanted features.
* Better handling of edge cases - allow the algorithm to do more with plants on or next to the edge of the photo.
* Faster Processing - speed up the algorithm.