Pumpkin Counting Miniproject
Large Scale Drone Perception

Karina Bay Bloch (Kablo17) & Nicoline L. Thomsen (nthom18)

17/03-2022

Figure 1: `https://solidstarts.com/foods/pumpkin/`

Contents

1 Introduction **2**
1.1 Technical Specifications 2

2 Colour Based Segmentation **2**

3 Counting Objects **3**

4 Generate An Orthomosaic **5**

5 Count In Orthomosaic **6**

6 Endnotes and Discussion **7**

7 Conclusion **8**

8 Bibliography **8**

1 Introduction

This group mini-project regarded with dealing with how to estimate the largest and most efficient number of pumpkins in a known field with the help of a large sample of UAV images, that was taken of this pumpkin field with surroundings included. The purpose was to use several methods within the range of spotting the pumpkins in the field, while disregarding the rest of the landmarks, and counting the pumpkins. To process the information from each picture more easily, the images were to be combined into an orthomosaic, which should only contain the field of the pumpkins.

Source code for this project can be found on Github: `https://github.com/PositiveBeat/LSDP-Miniprojects-22`.

1.1 Technical Specifications

The images used in this project were taken during spring, at an altitude of approximately 140 meters above mean sea level. The camera model was a senseFly S.O.D.A. [1], with a focal length of 10 mm, an F-stop of f/2.8, exposure time of 1/1000 sec, and an ISO of 800.

2 Colour Based Segmentation

The chosen colour space to use for the segmentation is HSV, which categorises colours in hue, saturation and value. To detect pumpkins the hue is what is of interest, and that value being independent of the saturation or value of the colour is advantageous. Before any more processing are to be done, a Gaussian blur is applied to the image to reduce noise.

To determine if a colour is close to a specific reference, and can be classified as a pumpkin, a distance metric should be decided. Euclidean distance is an option, however by using the Mahalanobis distance, the variance of the colour is taken into account. This is the method chosen for the segmentation.

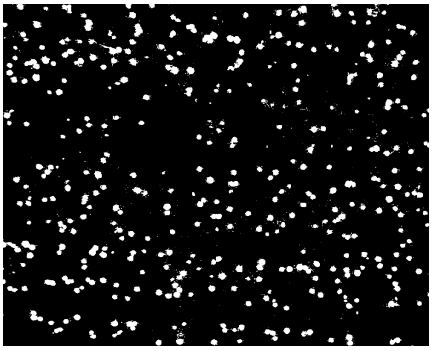
Obtaining the reference colour is done by annotating a sample image of pumpkins with a red colour, and using it to create a bitmask. This bitmask is then used to isolate the desired colours from the image. From this the average and co-variance matrix are calculated, to be used to calculate the Mahalanobis distance in the online computation. The annotation on the image was taken at different places on the image, to account for different lighting, and the pumpkins were annotated only in the middle, to avoid sampling colours from the surrounding field. The inversed co variance matrix used in this project is denoted in equation 1, and the average in equation 2.

$$cov^{-1} = \begin{bmatrix} 0.1425055 & 0.00316633 & -0.00380055 \\ 0.00316633 & 0.00282081 & 0.00163333 \\ -0.00380055 & 0.00163333 & 0.00575833 \end{bmatrix} \tag{1}$$

$$average = \begin{bmatrix} 17.49497487 & 155.94974874 & 247.45477387 \end{bmatrix} \tag{2}$$

Using OpevCVs function `threshold()`, the image is segmented into a binary image with the Mahalanobis distance. This is shown in figure 2. This binary image is used for further analysis in the following sections.

(a) Picture of pumpkins



(b) Segmented picture of pumpkins

Figure 2: Showcase of the function using Mahalanobis distance to segment pumpkins

3 Counting Objects

To accurately detect blobs in the segmented image from section 2, the image should firstly be filtered, to remove most of the noise, that was not filtered out by the Gaussian blur. This is done by morphological closing followed by an -opening. This has the effect of removing small fragments in the image. OpenCV provides the function `morphologyEx()` with the input flag `MORPH_CLOSE` and `MORPH_OPEN` to complete these operations. The effect of which can be seen in figure 3, where the morphological filtering has been applied to the binary image from figure 2b.

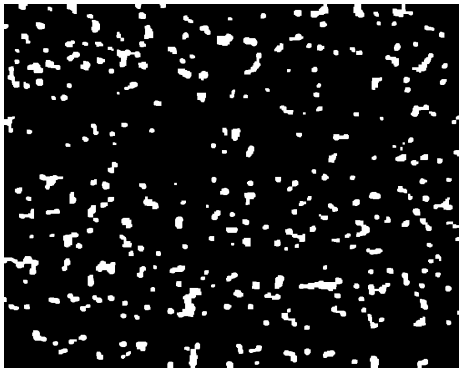


Figure 3: Segmented image after morphological closing

After this operation contours are detected by using OpenCVs function `findContours()`. At this point, there are still detected contours, that are too small to be considered pumpkins. Examining the histogram in figure 4, which shows the square root area of the contours, the first two bins seems to differ from the trend created by the remaining bins. The value after the second bin is 4.5, and this number is used as a threshold to remove any contours below this value.

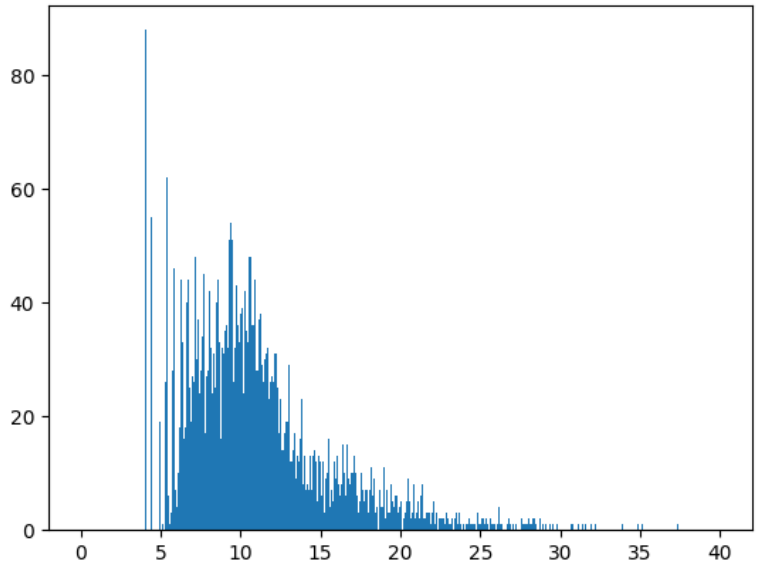


Figure 4: Histogram of the square root area of the detected contours in a sample image

A lot of the detected contours are in fact clusters of multiple pumpkins, and should contribute to the total count by the size of the cluster. To identify clusters, the method used is to realise that pumpkins are inherently shaped like circle or eclipses, and therefore it is possible to test for 'circle likeness'. Using equation 3 [2], which assume a value of 1 for perfect circles. Using a threshold of 0.75, it is possible to distinguish between single pumpkins and clusters of multiple. For each cluster the size of that contour is used to determine how many pumpkins it consists of, based on the average size of all the single contours.

$$\frac{4\pi \cdot area}{perimeter^2} \tag{3}$$

After these operations, the total number of pumpkins in the image has been counted. Figure 5 shows the image after processing, having each detected contour marked, the red contours are the single pumpkins counted as one, and the blue are clusters counted as either one or more. The black number on each cluster represents the algorithms guess as to how many pumpkins are in the cluster.

Figure 5: Detected pumpkins in the sample image

4 Generate An Orthomosaic

The purpose of constructing an orthomosaic is to combine multiple orthoimages, which in this case was 222 images, to acquire a combined image, that still contains information from each singular image, but where geometric distortion should be corrected and the colour of each image should be balanced. Since the purpose of this project is to detect as many pumpkins as correctly as possible by using an orthomosaic, the end result should not show any distortion while still retaining its high image quality and resolution. To construct an orthomosaic in this project, the program Pix4DFields [3] was used, since the purpose of this project is to count pumpkins and not construct orthomosaic generators. However, the process of generating an orthomosaic in Pix4DFields looks as such:

- All images are placed upon a satellite picture at positions matching their image data of place, time and size. At this point of time, it is easy to see the correlation between the pictures, but lighting and perspectives makes it look a little off.
- The distortions are removed, and each image are being assigned with key points (more than 1000), like edges, irregularities and otherwise. These key points are what defines where and how to combine each image together with each other.
- Together with the generation of the orthomosaic the lighting of each image is being blended together to make it look as real as possible.
- The orthomosaic should be done, and it should be visible in the satellite image, which gives and opportunity to compare.

Figure 6: An example of one image out of the 222 images combined into an orthomosaic. Landmarks, such as the building, are not needed and are removed when the orthomosaic are being reduced to just the pumpkin field

This method also ensures, that distances are preserved, and therefore the orthomosaic can be used for measurements. For this project, only the pumpkin field is necessary and Pix4DFields gives the possibility of removing the unnecessary parts, which was done and can be seen in figure 7. All that would be needed now, would be to divide the orthomosaic into several smaller images of the pumpkin field to count try and count the pumpkins effectively.

Figure 7: The Orthomosaic after the removal of all unnecessary buildings and landmarks. Altough much was removed, the file itself is still very large, and should be cut in size to be more easy to work with.

5 Count In Orthomosaic

As described in the last part, working with the orthomosaic as a whole would take a lot of processing power, so cutting the image into tiles would prove to be more efficient. To cut them into tiles, the Python package `rasterio` was used to perform operations on the orthomosaic with tile based approaches. Rasterio is a GDAL and Numpy based Python library designed to make working with geospatial raster data more productive and fast. At first, the size of each tile was manually decided to be 1/12 of the size of the orthomosaic in the x-dimension, and 1/7 in the y-dimension. That would mean that there should be 84 tiles in total to be used for pumpkin scanning, each with a size of `383x1001` pixels. To look for -and count the pumpkins, the methods from section 3 were used. An example of a tile can be seen in figure 8.

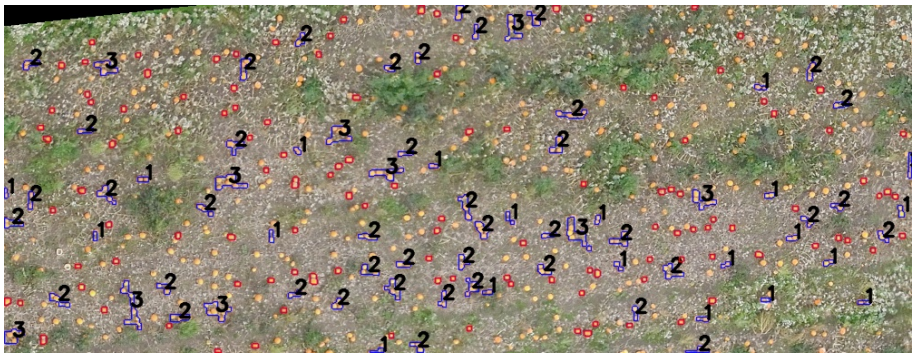


Figure 8: Detected pumpkins in a tile of the orthomosaic

Figure 9: Detected pumpkins in a tile of the orthomosaic

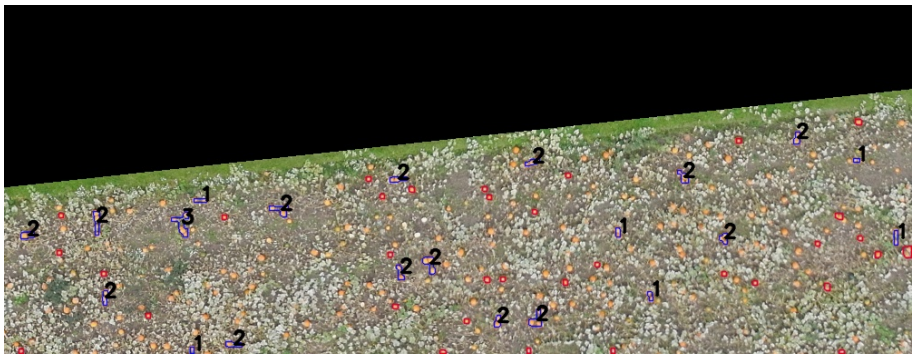


Figure 10: Detected pumpkins in a tile of the orthomosaic

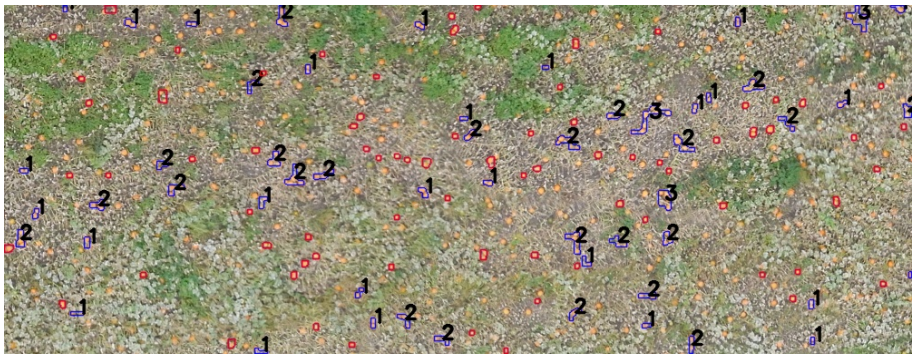


Figure 11: Detected pumpkins in a tile of the orthomosaic

The final number of counted pumpkins is 7274, which is considered an underestimate, since the figures show that many pumpkins are not taken into account in the program. A reason for this might the choice of reference colour, which came from a test image instead of the orthomosiac.

6 Endnotes and Discussion

Improving Pumpkin Detection

The different methods for removing noise are incapable of removing all noise, and also risks removing genuine pumpkins. This decreases the accuracy of the method.

Another important factor, that impacts the quality of the detection, is the reference colour distribution. This was sampled using a few pumpkins spread over a large image. Only a few pumpkins was needed, and it seemed to begin detecting the field itself when too many pumpkins were annotated.

However this means not all pumpkin colours were included in the reference colour distribution. It is a possible reason for not detecting some pumpkins.

Not Counting Duplicates

The current method of processing the orthomosaic does not take into account the possibility of counting certain pumpkins twice, if they are at the edges of the tiles. A better implementation would be to overlap these tiles, and defining a seam line between the tiles. Then only count the contours which centre of mass is within the current tiles seam line.

7 Conclusion

The finalised algorithm was able to detect pumpkins well in a sample image, also handling clusters of pumpkins. However in the orthomosaic the detection was not as promising, as it had been for the sample image. It is speculated that this is due to poor sampling for the reference colour. Thus the system needs to be improved in order to be ready to help a farmer with the task of estimating the number of pumpkins in a field.

8 Bibliography

[1] SenseFly. sensefly s.o.d.a. `https://www.sensefly.com/camera/sensefly-soda-photogrammetry-camera/`. Online; accessed 12 March 2022.

[2] Henrik Skov Midtiby. Object recognition. University Lecture, 2021.

[3] PIX4D. Pix4dfields advanced agriculture mapping software for aerial crop analysis and digital farming. `https://www.pix4d.com/product/pix4dfields`. Online; accessed 17 March 2022.