Abstract:

For this research project, we will be examining the use of three various computer vision detection algorithms in the context of detecting the area of various leaves picked from the black cottonwood tree. The black cottonwood tree is invaded by a particular bug named Orgya leucostigma (White-marked moth). The amount of damage this bug does on this tree varies between the distinct subspecies of the black cottonwood tree. The damage done on each tree can be approximately measured by looking at the how much of a leave is eaten overtime by a White-marked moth. Therefore, to study which subspecies of the cottonwood tree is affected by the White-marked moth bug, calculating the area of each leave in an accurate manner is essential. Four different area detection algorithms were used to estimate the size of a leaf. These algorithms were a global thresholding, RGB thresholding, rg chromaticity and a canny algorithm. Each algorithm inspected six different images. Each of these images included four leaves. The algorithms had to outline the perimeter of each leaf and had to calculate how many pixels each leaf contained for each image. The accuracy of the algorithm is based on how well the algorithm can outline the original leaf objects, since the area in terms of pixels of the original leaf objects is not known. Based on the results of our tests, the RGB thresholding algorithm preformed the best, followed by the global thresholding algorithm. The Canny algorithm was the third best at area detection and the rg chromacity algorithm preformed the worst.

Introduction:

Image recognition techniques can be traced back as early as 1949 when Donald Hebb wrote the book "The Organization of Behavior". This book discussed how the interaction of neurons can be used to determine how human being make decisions. This idea of neuron-to-neuron interaction was not converted to machine decision making until 1957 when an Aeronautically engineer named Frank Rosenblatt built he "Mark 1 Perceptron". The Mark 1 perception was used for image detection. Since the algorithm was built a rustic machine with low computing power it was able to correctly detect very few images. It was not until the 21st century when an abundant amount of computing computer became widely accessible did image recognition algorithms become accurate and viable. By 2014, engineers at google were able to deploy an algorithm that could recognize and verify humans in an image with the same accuracy of a human. As time progressed, detection algorithms have become more accurate and quicker to implement. Because of this, these image recognition algorithms have been used in a wide range of applications such as area calculation, face detection, and default product detection.

As computers become more powerful, and as more capital is invested to improve image detection algorithms, these algorithms are still plagued by the same problems as they were when these algorithms were invented. Image detection algorithms still struggle to produce accurate results when images are placed into backgrounds that are homogenous to the objects colour. In addition, the angle of the camera and the lighting used to take an image can distort the outcome of an image recognition algorithm.

Even with these issues, image recognition algorithms can be used to greatly benefit the human and business world from everything from fraud detection to leaf destruction by a white marked moth.

Global Threshold Algorithm:

The first algorithm that was used to detect the area and outline of leaves from the cottonwood tree was the global thresholding algorithm (denoted as “global_threshold” in pycharm). The global thresholding algorithm converts the inputted image into a grey image where the perimeter of the leaf is outlined. The steps this algorithm takes is as follows:

1. The original image is cropped so that the four-leaf objects become the focal point of the image
2. The cropped image is then converted to greyscale
3. The image is processed using the “createCLAHE” algorithm
4. The local threshold is determined by taking the lowest value between the two peaks determined by the histogram. This can also be approximately determined by setting the derivate of the histogram function to zero.
5. The contours (outlines of the leaves) of the converted greyscale image are determined using the “cv.findcontours function”
6. The contours found are then filtered to between a certain size to reduce any noise (both large and small).
    a. The filtering is conducted using the contourArea function.
7. The images with their new outlines are re-shown and plotted.
8. The total average area of the leaves are calculated by taking the area of each contour summing them and dividing by the number of the leaves in the image (this variable Is passed by the user).

The summarize the above steps images 1-3 show the process of the algorithm forming the histogram, finding the threshold, and converting this image to a black and white image where the leaves become black circles, then finally the algorithm forming the perimeters around the original leaf objects. This algorithm was able to correctly partition four of six images with minimum error. As can be seen in image 4, one of the leaves has no perimeter around it, representing the error in the algorithm. In addition, in image 4, the top left leaf is bounding an area of the leaf that is not part of the leaf. The reason for this is the leaf is casting a shadow which is creating some noise that can be filtered out by a threshold since the colour scheme of the shadow is like the leaf’s colour scheme. A threshold of 101 was chosen for all images as it created the most accurate outline for all leaf objects.

Image 1:

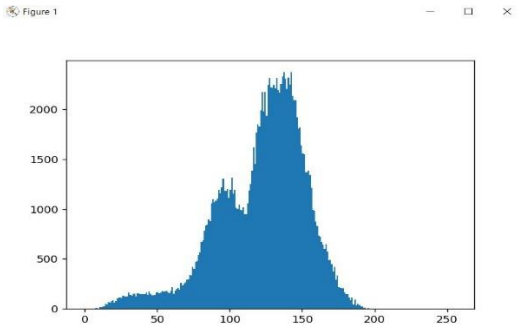


Image 2:

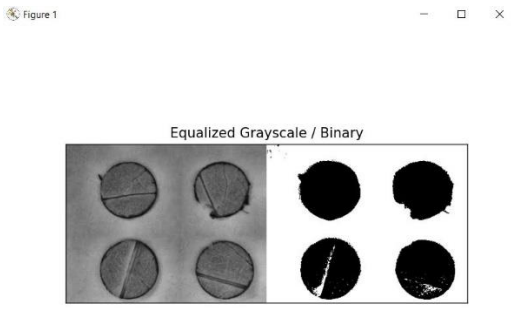


Image 3:

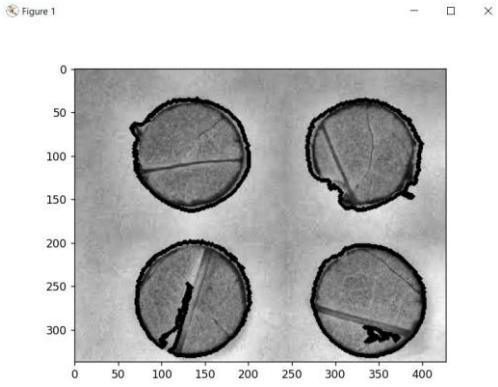
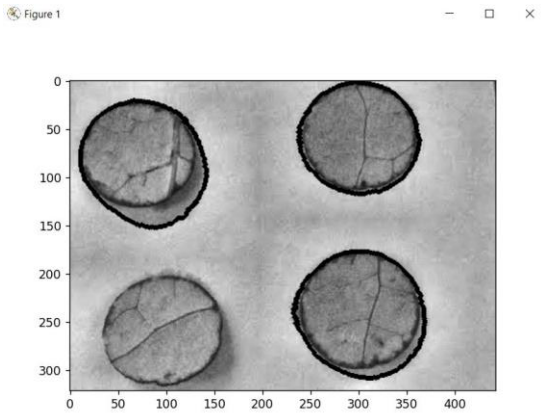


Image 4:



RGB Colour Algorithm:

The second algorithm that was used to detect the area and outline of leaves from the cottonwood tree was the RGB thresholding algorithm (denoted as “three_colour” algorithm in pycharm). The three_colour algorithm converts the inputted image into a grey image, blurs the background of the image to minimize shadows and then converts the image to a black and white image based on a threshold. The steps this algorithm takes is as follows:

1. The original image is cropped so that the four-leaf objects become the focal point of the image
2. The cropped image is then converted to greyscale
3. The image is pre-processed using a blurring algorithm
4. The threshold range for each colour scheme is determined by the taking the range of pixel values from a patch of a leaf object.
5. A nested for loop, goes through the greyscale image and converts pixels that are within the threshold to white and the other pixels to black.
6. The black and white pixels are replotted and shown next to the original image.
7. The total average area of the leaves is calculated by summing up the number of white pixels in the new black and white image.

As can be seen in images 5-7, this algorithm does a good job at creating a representative outline of the are the leaf objects cover compared to the background it covers. Unlike the global thresholding algorithm, it does not completely miss a leaf object in any of the six test images.

This algorithm does struggle when there is a shadow cast on the leaf objects as can be seen in image 6. The algorithm does a good job when the leaf is different shades of green. In image 7, the top left leaf has dark green part on the leaf object since this part of the leaf is dead. Because of this, the differences in shades of green causes the thresholding algorithm not to pick up this portion of the leaf. This can be an advantage or disadvantage depending on how a person deems what the area of a health leaf is.

Image 5:

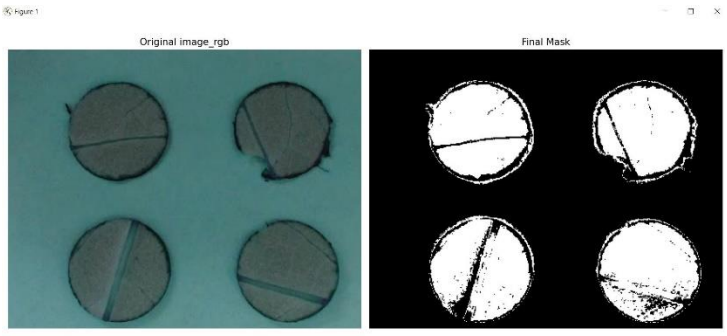


Image 6:

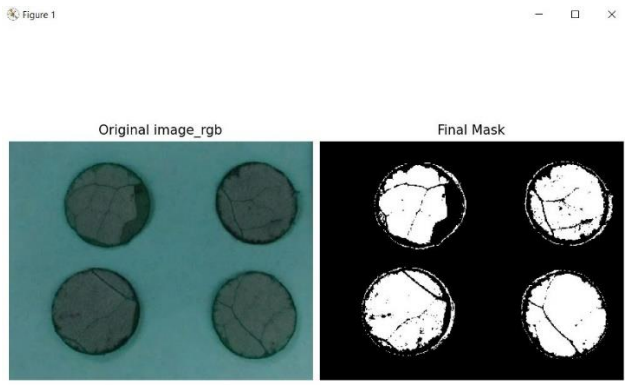
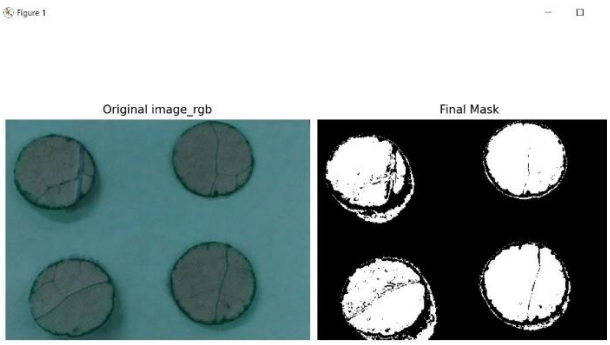


Image 7:



<u>Canny algorithm:</u>

The third algorithm used was the canny algorithm form opencv. The process of the algorithm was as follows:

1. The original image is cropped so that the four-leaf objects become the focal point of the image
2. The cropped image is then converted to greyscale
3. The edges (perimeter of the leaf objects) of the greyscale object is determined by the "cv.canny algorithm"
4. The contours (outlines of the leaves) of the converted greyscale image are determined using the "cv.findcontours function"
5. The contours found are then filtered to between a certain size to reduce any noise (both large and small).
    a. The filtering is conducted using the contour Area function.
6. The images with their new outlines are re-shown and plotted.
7. The total average area of the leaves is calculated by taking the area of each contour summing them and dividing by the number of the leaves in the image (this variable Is passed by the user).

The algorithm will eventually convert the original image to a blank and white image where the white pixels are the outline of the four-leaf objects. As can be seen in the three images below, the canny algorithm fails to make a solid boundary around where the leaf should be. If the contours are not accurate the underlying area calculation will not be accurate as well.

Image 8:

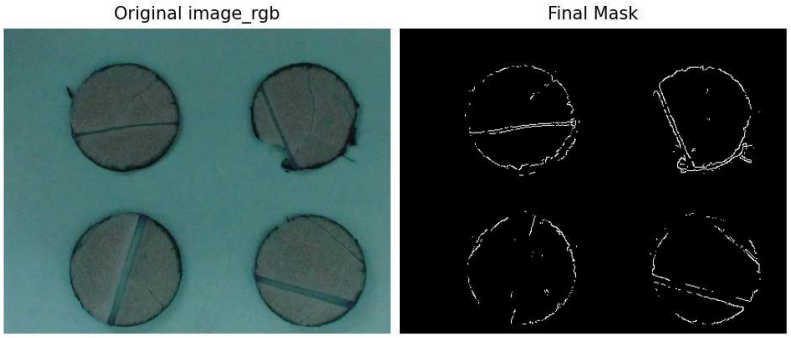


Image 9:

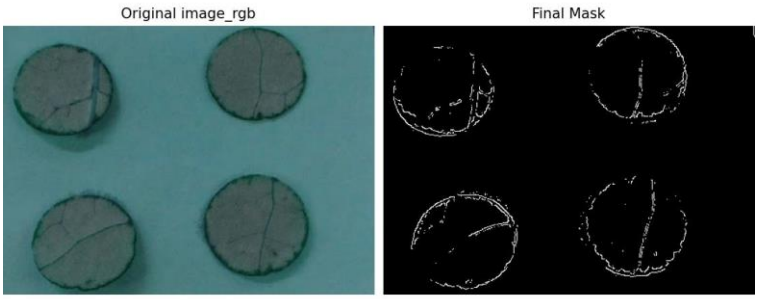
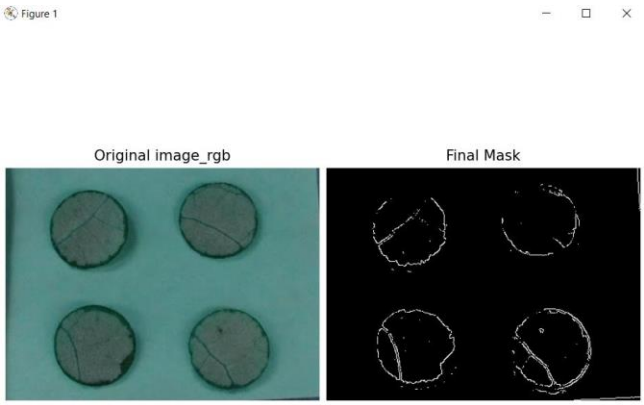


Image 10:



RG Chromaticity Algorithm:

The last algorithm used to calculate the area of the leaf was the RG chromaticity algorithm (titled "mask_algo" in python notebook). This algorithm takes a representative patch of the leaf object and preforms a gaussian statistical analysis to determine if a given pixel should be converted to white or black. A pixel will be converted to white if the RG ratio is within the gaussian distribution based on the leaf patch and black otherwise. The algorithmic process can be summarized as follows:

1. The original image is cropped so that the four-leaf objects become the focal point of the image
2. The cropped image is then converted to greyscale
3. The "patch" is taken from one green area on a leaf object. The patch is used for each testing image
4. The RG ratio for the whole image is compared to the RG ratio from the patch threshold
5. A nested for loop, goes through the greyscale image and converts pixels that are within the threshold to white and the other pixels to black. A pixel is converted to white if the RG ratio is within 3% of the patch ratio.
6. The black and white pixels are replotted and shown next to the original image.
7. The total average area of the leaves is calculated by summing up the number of white pixels in the new black and white image.

The algorithm does a decent job of outlining the leaf area (as can be seen in image 12-14) but the algorithm did a very poor job with some images. For example, in image 11, the algorithm almost colored three quarters of the image white when only half should be colored. The issue with this algorithm lies in the fact that the background will form a similar RG ratio to the leaf object and the patch chosen may be represented of all leaf object colorings. Image 15 is the patch ratio chosen.

Image 11:

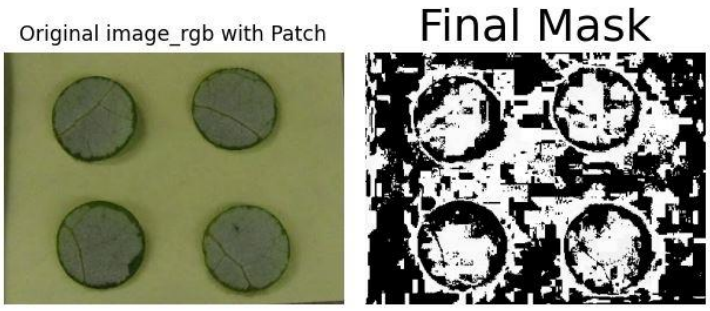


Image 12:

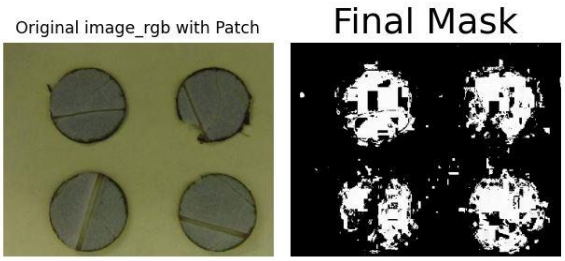


Image 13:

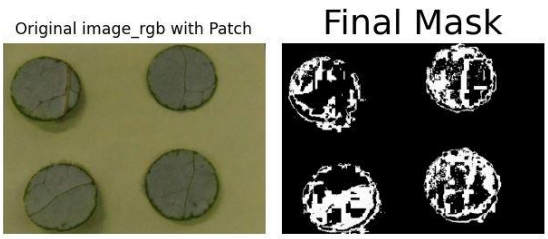


Image 14:

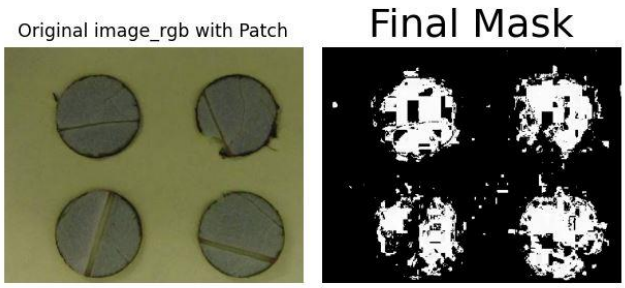
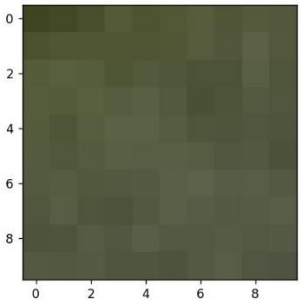


Image 15:



Results Conclusion and Future Work:

As can be seen from the above fifteen images the ranking from best to worse in terms of best outlining the leaf objects in the six images are:

1. RGB
2. Global
3. Canny
4. RG Chromaticity

I believe the RGB threshold worked the best since we were able to look at the chosen images and specifically choose a threshold that would fall into the colour scheme that would bets match the leaf objects. If the original images were not cropped this algorithm would have performed well since certain colours that fell within the range would have been picked up in the background. For example, when this algorithm was tested on images containing a brownish background, the whole brown background was picked up.

In the future, I would of have liked to automate this process much more by doing the following preprocessing work:

1. Take a picture from a camera at a 90-degree angle
2. Controlling lighting to prevent shadow casting onto the image
3. Using a very light or dark background to contrast with the leaf objects
4. Placing a standard dime into the picture

By taking the above four steps we could greatly minimize and background noise from colors and shadows. The dime can be used as a reference point when calculating the real-life size of the leaf objects. For example, we could compare the per pixel amount of each leaf object to the pixel amount of the dime object and get a real-life measurement down to the millimeter. Using a camera at 90-degree angle will minimize any angle bias distorting the per pixel measurement.