# How to Estimate the Area of Every Leaf

Jayce Zhang
*Northeastern University*
*Khoury College*
Vancouver, BC, Canada
zhang.jies@northeastern.edu

***Abstract*—This article's aims to research three ways to estimate the area of every leaf, which would help the biological scientist decrease a lot of time measuring the leaves and trying to compute the areas of the partially eaten leaves.**

***Index Terms*—Image Recognition, Threshold Technique, Edge Detection, Color Chromatic.**

## I. Introduction

Researchers at the Department of Biological Sciences at the University of Toronto (Links to an external site.) are studying how plants defend themselves against invaders.
The first step is to quantify the damage suffered by each variety. In order to do this, they collect several leaves from different varieties available in British Columbia and crop them into circles of known diameter.
Then, they place these circles in trays and wait for the bugs to attack.
Every week, they need to collect the leaves from every tray and estimate what percentage of every leaf has been eaten by the bugs. They do this with a large number of leaves from several varieties of back cottonwood tree.
As we know, estimating this percentage gets a bit challenging since estimating the area of each semi-destroyed leaf is not straightforward.
The scientists would spend a lot of time measuring the leaves and trying to compute the areas of the partially eaten leaves. Even though this is part of doing research, these researchers could be more productive if they could use their time for more creative tasks while automating the process of continually estimating the area of every leaf.

## II. Find three ways to estimate the leaf's area

In this section, we will use three different method to calculate the area. From the picture library of leaves, I select three images as the sample, which have different background and shapes.

### A. Thresholding Technique

I will use the thresholding technique to find the leaves, because there are obviously difference between the leaves and background. [1]
Before that, images are pre-processed by function I wrote at the Lab1, which will help the images get the larger contrast between the target and background Firstly, We should read and trans it to the grayscale image(Fig.1). Then We should get and display the histogram(Fig.2), which could help us determine a good threshold to separate foreground from background.

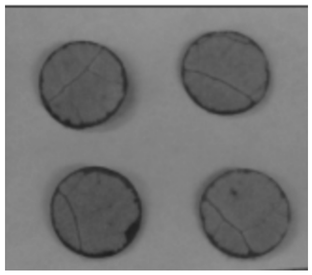
Fig. 1. Grey picture

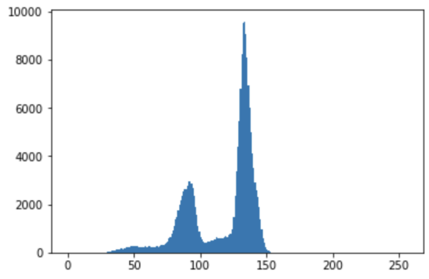

Fig. 2. Histogram

According to the histogram, we could easily determine the threshold = 100, that is the obvious. Then, output the binary image and the original one. And we will get the Binary picture(Fig.3) Our threshold estimate did not result in the desired output so we equalize the histogram to see if this approach yields better results.
We go for the Contrast Limited Adaptive Histogram Equalization (CLAHE) method since our background is not uniform.(Fig.4)
As we can see, after using the equalized image, we got better results.(Fig.5) Next, we should calculate the area of these leaves.

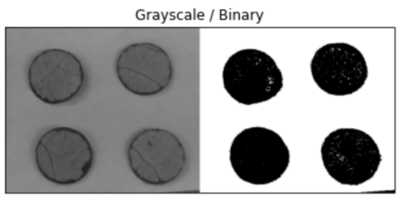

Fig. 3. Grayscale and Binary

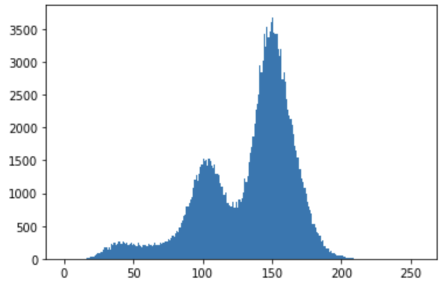

Fig. 4. CLAHE

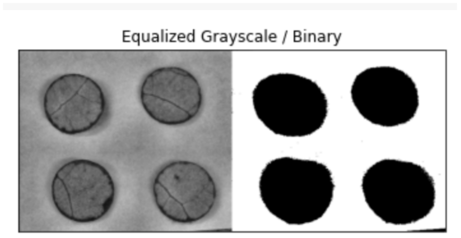

Fig. 5. Optimized Grayscale and Binary

We use the Fig.5 and Fig.1 to note the boundary of the four leaves(like Fig.6), then we could use the function of contourArea in opencv to calculate the area.

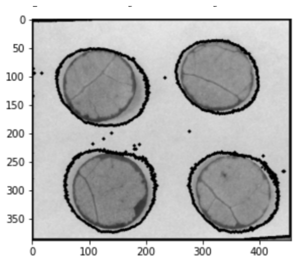

Fig. 6. Thresholding Output

### B. Edge Detection

Firstly, let's talk about the Sobel operator, which is the basic of the Canny. Firstky, we should reduce the noise through the Gaussian filter, which would help to prevent to note the noise as the edge point and get the better results.
Next, When the image is smoothed, the derivatives Ix and Iy w.r.t. x and y are calculated. [2]
According to the gradient calculation, we could get two result from X direction and Y direction. We will gain two pictures based on different gradient calculation.(Fig.7 and Fig.8)

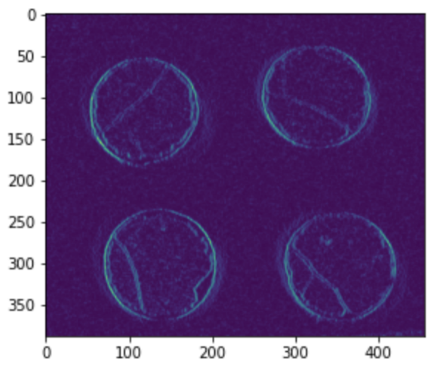

Fig. 7. Sobel-X

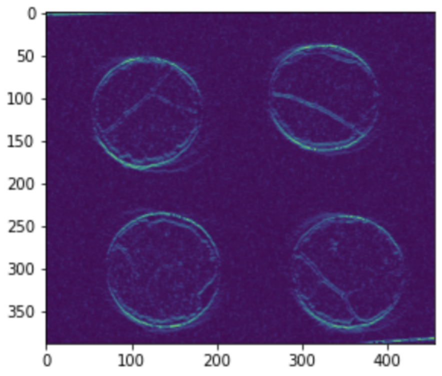

Fig. 8. Sobel-X

For the better result, we could combine both of them, it's obvious that we could get a better result.(Fig.9)

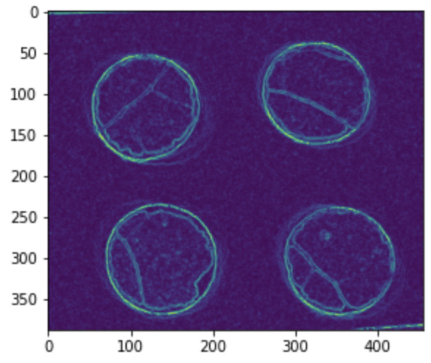

Fig. 9. Sobel-Combined

Now, let's talk about the Canny. There is only one mainly different between the Canny and Sobel,that canny will

do another process to promote the result: Non-Maximum Suppression.
Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges.
The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.
From comparing the Fig.9 and Fig.10, after doing non-maximum suppression, the edge will be given greater prominence. [3]

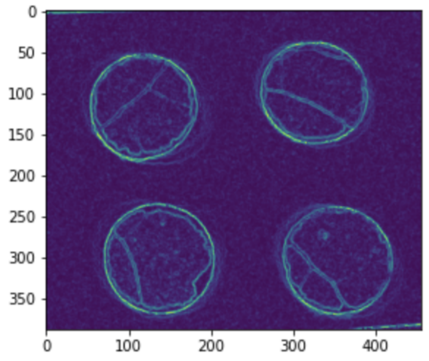

Fig. 10. Canny output

### C. Color Chromatic

I hope to use the color difference to divide the leaves and background. I plan to use the HSV to deal it. Firstly, we should read the picture.
Then we should change the RGB picture to the HSV picture.

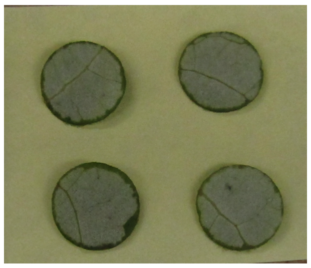

Fig. 11. Origin

According to the showed picture(Fig.12), we found that only the edge of the leaves is recognized, the leaves' center show the same(similar) color as the background. So, we can not use the HSV to find the leaf. After deeply researching the HSV, we found: because the background is yellow, in the HSV array, green's value of S and V is almost same as yellow, there is a little different in value H. So, we could not use it to found the leaf. [4]

I found that we could use the Y Cr Cb as the characteristic to recognize the leaf. Because I found that the Y and Cb 's

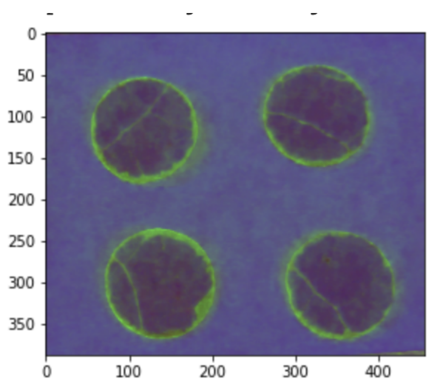

Fig. 12. HSV-Picture

value of green has the obvious different with yellow. Even we considered the compatible with the difference between white and green, which is little different in value Y, we need to extend the limit of Y, we also could get the result. Then we could use it to calculate the area.

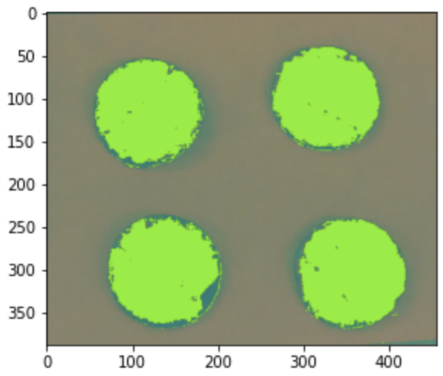

Fig. 13. Color Chromatic - Output

## III. Discussion

### A. Sphere of Application

For the Thresholding Technique, the most important thing is contrast between the target and background. So, if the background's color is similar to the target, this method will not be effect, like the next picture.

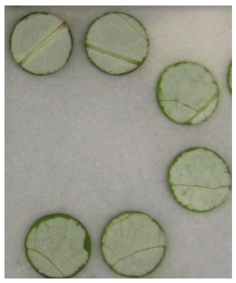

Fig. 14. White-Green

As we can see, I could not find the obviously difference between the target and the background. So, if it doesn't have the obviously difference between the target and the background, we could not use this method.

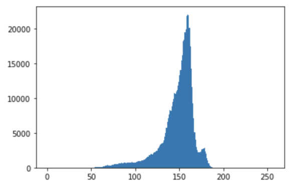

Fig. 15. White-Green's Histogram

Although the edge detection also depends on the contrast between the target and background, it would get a much more better result than thresholding, due to the complex mathematical treatment. As you can see(Fig.16), it could also get the relativity edges of the Fig.14.

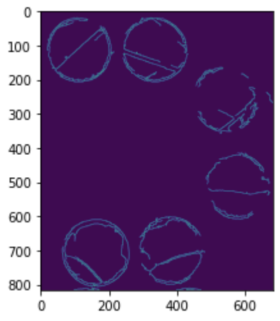

Fig. 16. Edge Detection of White-Green

For the Color Chromatic, it could output nice results in the most cases, unless at the Fig.14, because the white's values in the YCrCb much like the green. At the same time, I found that if I used the pre-processed images to do, I would get a worse result for Fig.14. Because in the YCrCb, white's Y much like the green after I boost lightness. You can see the output at Fig.17.

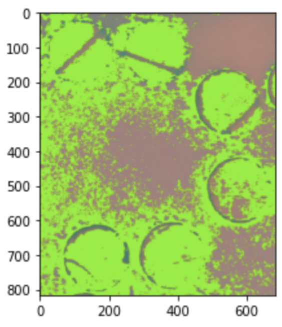

Fig. 17. Color Chromatic of White-Green

### B. *Effective*

In this section, we used the rate of the leaves in the whole picture to evaluate the effective. The reason I used the rate

TABLE I
COMPARISON OF THREE METHODS

| | **Name of the methods** | | |
|---|---|---|---|
| | ***Thresholding*** | ***Edge Detection*** | ***Color Chromatic*** |
| Rate | 0.413 | 0.245 | 0.277 |
| Accuracy Rate | 0.68 | 0.79 | 0.89 |

Benchmarkis0.312.

to judge the effect, during the processing of this picture, the image sizes would be changed.
According to this table, we could get: the thresholding would extend the edge of the leaf, so it will output a larger size than the original. The other two method will output a better results.
The exact value of the rate is 0.312, which I used PS to get it.
So, as we can see, the Color Chromatic got the best result.

### C. *Automated or Not*

For the thresholding, it needs us to manually determine the threshold. If we want to make it as automatically, we need another development, but it also adapts particular situations.
For the edge detection, when we plan to calculate the area of the leaves, we meet some difficult. Because of the leaf vein and rough edges, there are too many edges to calculate the size by contourArea function. So, we must manually set up limit to cut them off.
For the Color Chromatic, it adapts the most case, so that it can be fully (or almost fully) automated.

## REFERENCES

[1] Thresholding (image processing), from wikipedia.
[2] Sobel operator, from wikipedia.
[3] Sofiane Sahir, Canny Edge Detection Step by Step in Python — Computer Vision.
[4] YCbCr, from wikipedia.