School of Computer Science

Introduction To Image processing Coursework

Mustafa Mehmood
20306551
hcymm3

15th April 2022

Table of Content

1. Image Pre-Processing

To begin with, I started with Image pre-processing as it is a very crucial step before Image Segmentation. The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis tasks. The main pre-processing procedures that were applied were sharpening, retrieving the greenness from the image, filtering and Binarizing the Image

1.1 Sharpening

Our goal with image sharpening is to make digital images look sharper.Although minor, a camera shake or an out-of-focus photo can also be corrected with sharpening. A sharpened image's edges are better defined.The red and green channels were sharpened to emphasise the transitions or discontinuities in their intensities. However, the disadvantage to over sharpening is that it makes the noise in the image more noticeable than needed. It can lead to Sharpening lines, Sharpening spotsand Sharpening grain.

1.2 Extracting Greenness

To extract the greenness from the image, I had two ideas in mind. Firstly using the greenness formula from the lecture: we extract the green Channel, the red channel, and the blue channel, and then we use the formula: “ greenness = (greenChannel-(blueChannel+redChannel)/2”” to get the greenness the result is in follows

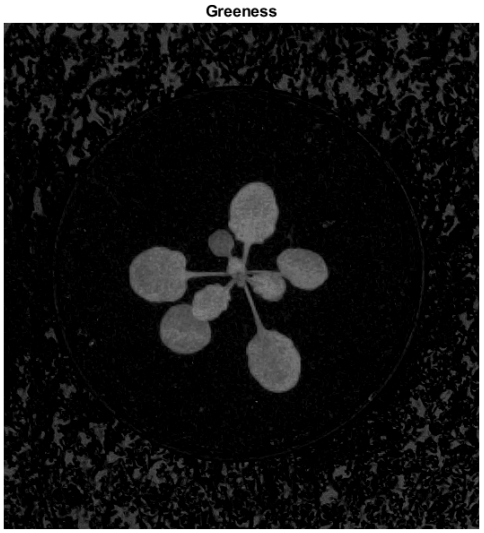
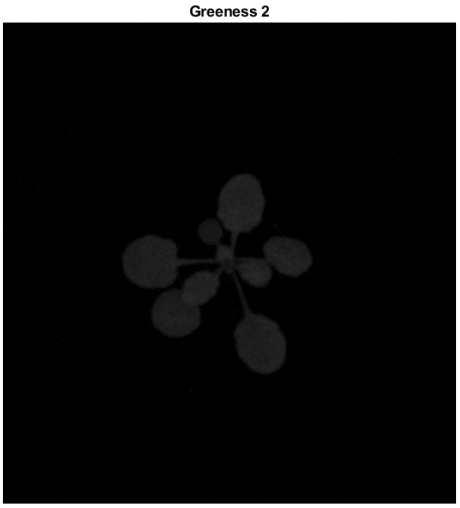


Figure 1.1.1 Greenness 1 (plant001)

Another way to do this, which also gave a cleaner-looking image was to use imsubtract inbuilt function of Matlab by using the greenChannel and gray Scale of the image. It subtracts each element in array Y from the corresponding element in array X and returns the difference in the corresponding element of the output. The result is shown *Figure 1.1.2*



Greenness 2: *Figure 1.1.2 Greenness 2*

1.3 Binarizing

The next pre-processing action on the image was to binarize the image and like before i stumbled upon two ways to do it, First was to use thresholding, where we change the pixels of an image to make the image easier to analyse. My approach to determining the threshold is to first apply this condition Greeness(greeness>0) which returns a row vector comprising the values of the Greenness array where the greenness value is greater than 0, and then take the mean of the column vector. Mean is used to find the optimum threshold for the values greater than 0., If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value. Another much easier way to do this was to use the inbuilt function of matlab called “imbinarize”. imbinarize computes a threshold for each pixel using the local mean intensity around the neighbourhood of the pixel. It uses a neighbourhood size of approximately 1/8th of the size of the image (computed as 2*floor(size(I)/16)+1). . imbinarize(I) creates a binary image from 2-D or 3-D grayscale image I by replacing all values above a globally determined threshold with 1s and setting all other values to 0s. By default, imbinarize uses Otsu's method, which chooses the threshold value to minimise the intraclass variance of the thresholded black and white pixels. Both ways produced the same effect.
But for Binarizing we need to properly choose the threshold value, that's why for plant 1 and plant 2 this gives good results but for plant 3 since there are shadows it does not give proper results

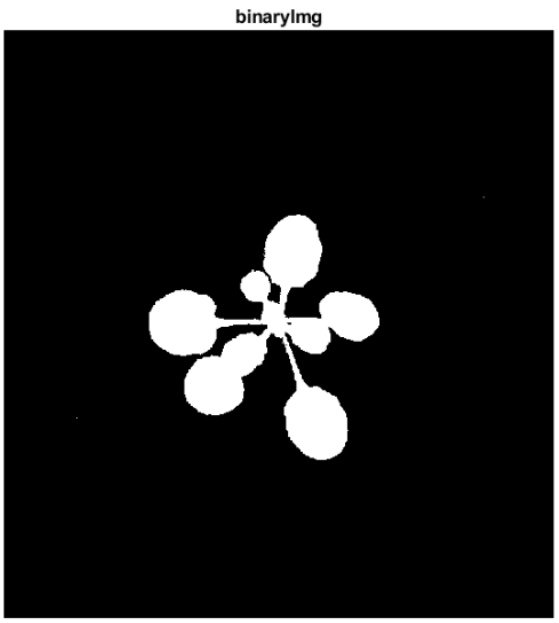


Figure 1.1.3 binaryImg

1.4 Filtering

Succeeding binarizing was filtering for enhancing the image to emphasise certain features and remove other features which are not required for leaf detection. While there are a plethora of techniques for filtering, the ones that I used and that seemed fit for this project are detailed next.

To denoise and remove most of the salt and pepper noise, Median filtering was used. There is a high degree of effectiveness in using median filters to reduce random noise, particularly when the noise amplitude probability density has long tails and periodic patternsmedfilt2(I) performs median filtering of the images in two dimensions. Each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image. The results, before and after for image 1 is shown below:

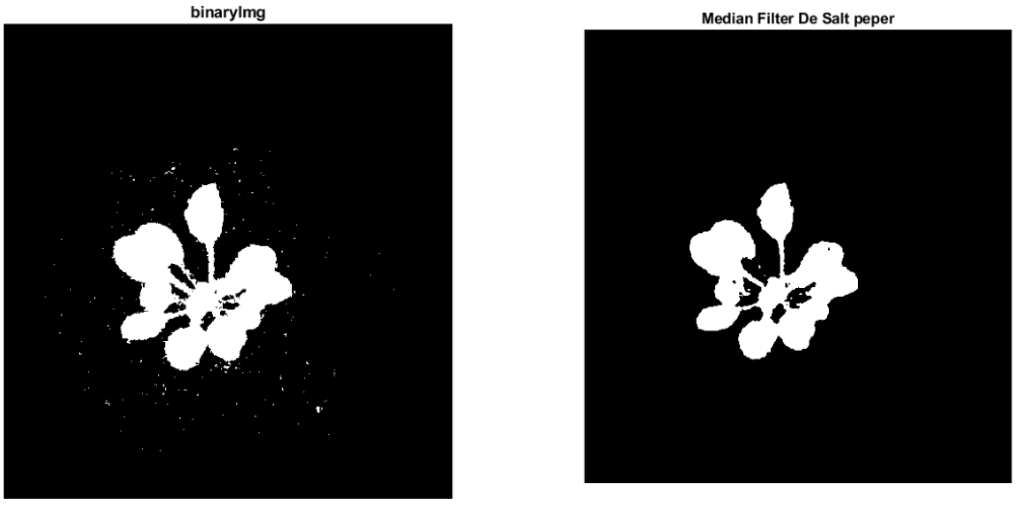


Figure 1.3.1 median filter

After de-nosing the unwanted regions in the images is removed and t all the connected components are extracted within the built in function bwareafilt was used. It extracts all connected components (objects) from the binary image BW, where the area of the objects is in the specified range, producing another binary image. bwareafilt returns a binary image

containing only those objects that meet the criteria. This will remove any other green unwanted areas from our images which could be caught too as a result of thresholding.

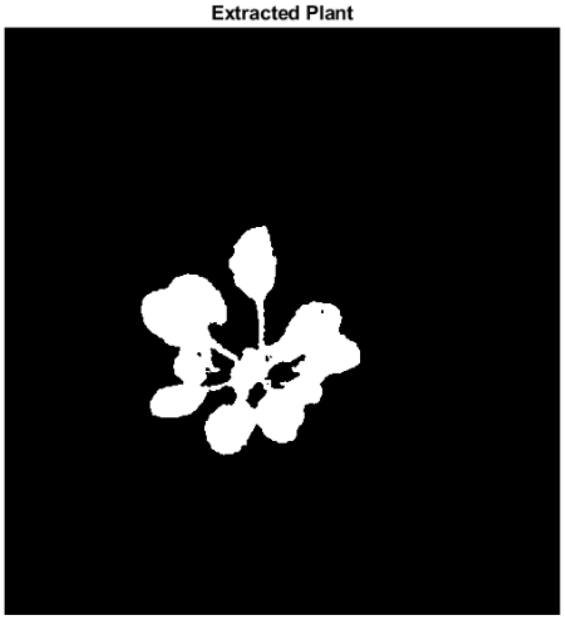


Figure 1.3.2 Extracted Plant using bwarefilt

1.5 Erosion

To somewhat smooth out the edges, erosion was performed on the images imerode(I,SE) erodes the grayscale, binary, or packed binary image using the structuring element SE which in my case was a disk of size 2. This is done to filter out the smaller and separate overlapping or bounding objects in our image

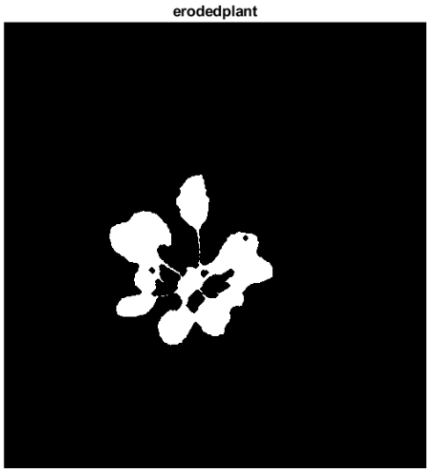


Figure 1.4.1 Eroded Plant

Foreground Improvement
Using bwareaopen, very small dots can be removed. As a result, they are removed in the foreground, so we complement the image before and after bwareaopen.

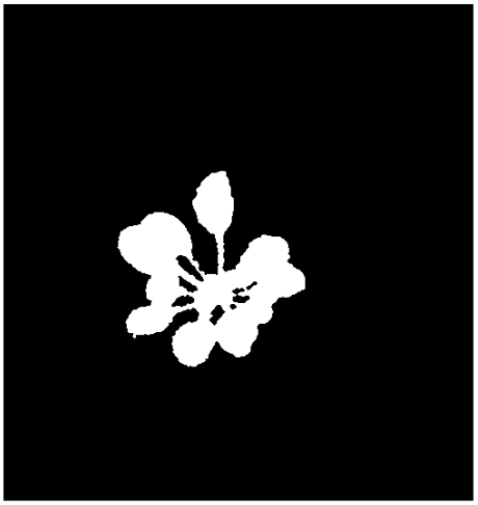


Figure 1.3.2 Improved foreground

2. Testing Pre-Processing

For testing the images color was put back into the eroded images only in white areas. Here are the results for all images

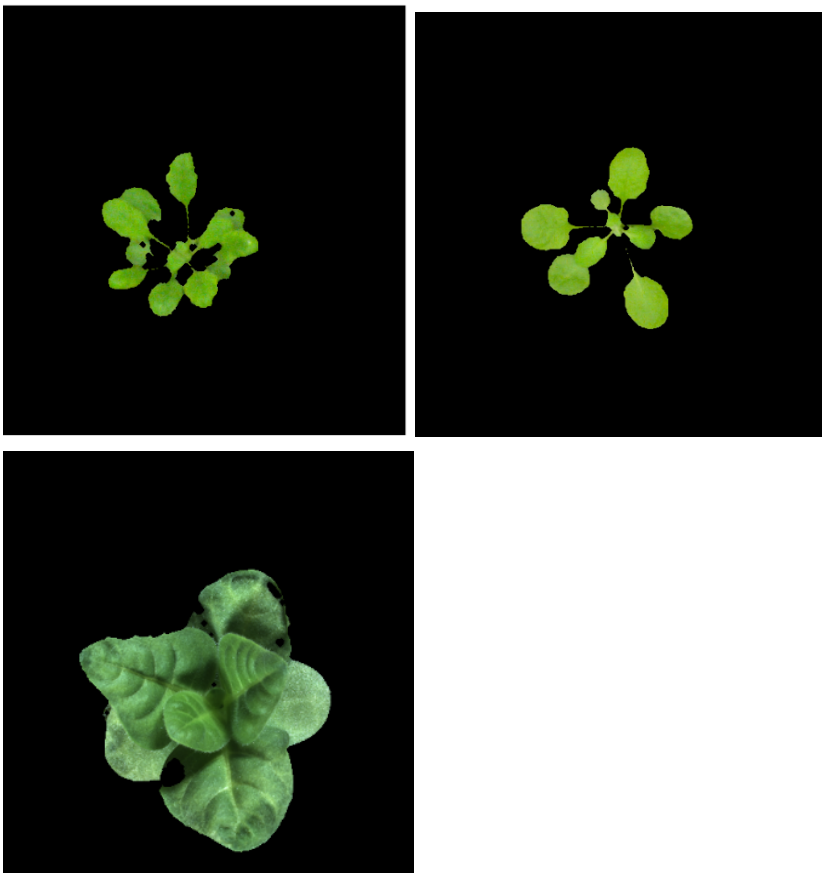


Figure 1.1.1 PreProcess testing

3. Image Segmentation

3.1 Watershed segmentation

Finally, for segmentation, a region-based method known as Watershed segmentation was applied. The image segmentation method based on region processing called the Watershed algorithm has many advantages. With a watershed algorithm, global segmentation, border closure, and high accuracy are achieved. An outline can be one pixel wide, connected, closed, and located to the nearest pixel. Based on this concept, a gray level image is visualized as its Topographic representation, which includes three fundamental notions: minima, catchment basins, and watershed lines. In watershed segmentation, ridges and valleys are defined as the topography of an image. Gray values or gradient magnitudes define the elevation of the landscape. The watershed transformation decomposes the image into catchment basins based on its 3D representation. Catchment basins' boundaries are determined by the points of steepest descent terminating at each local minimum. An image is decomposed into its constituent pixels by the watershed transform. Each pixel is assigned either to a region or a watershed. Several small regions result from noisy medical image data. This phenomenon is called "over-segmentation". Effective filtering is therefore necessary. A general segmentation approach may be based on the watershed transformation in several different ways.

In an image like this, there are roughly circular, touching blobs of leaves. The distance transform can be used to identify the objects in an image by looking at its "catchment basins." It computes the Euclidean distance transform of the binary image BW

Figure 3.1.1 Applying bwdist function

Next, we compute the watershed transform

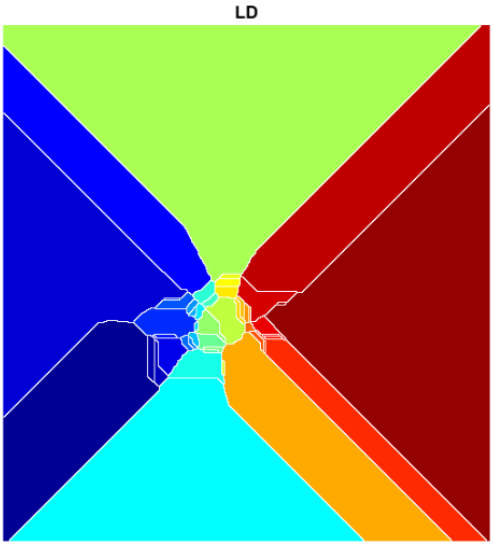


Figure 3.1.2 LD

Ld == 0 corresponds to the white ridgelines of the watershed. By changing the corresponding pixels in the background, we can segment the binary image based on these ridgelines.

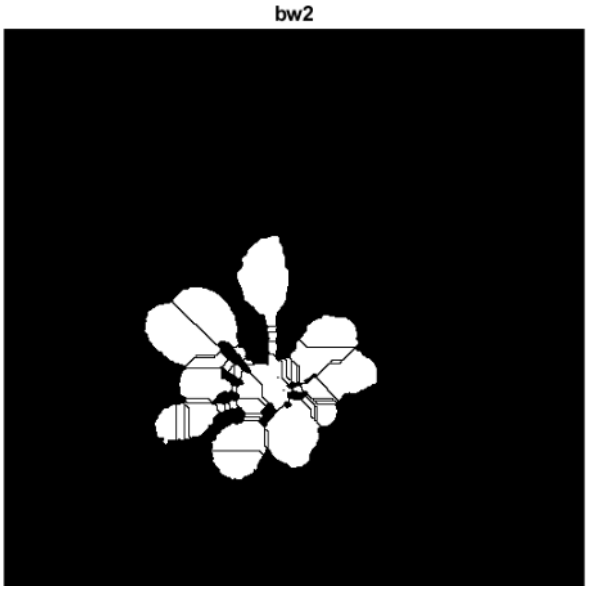


Figure 3.1.1 bw2

Overcoming problem of Over Segmentation

This raw " watershed transform" oversegments the image. Each local minimum, no matter how small, becomes a catchment basin. A common trick, then, in watershed-based segmentation methods is to filter out tiny local minima using imextendedmin and then modify the distance transform so that no minima occur at the filtered-out locations. This is called "minima imposition" and is implemented via the function imimposemin. A minimal imposition is implemented with the function imimposemin. By modifying the distance transform, it will appear only at the desired locations, then repeat the watershed process.

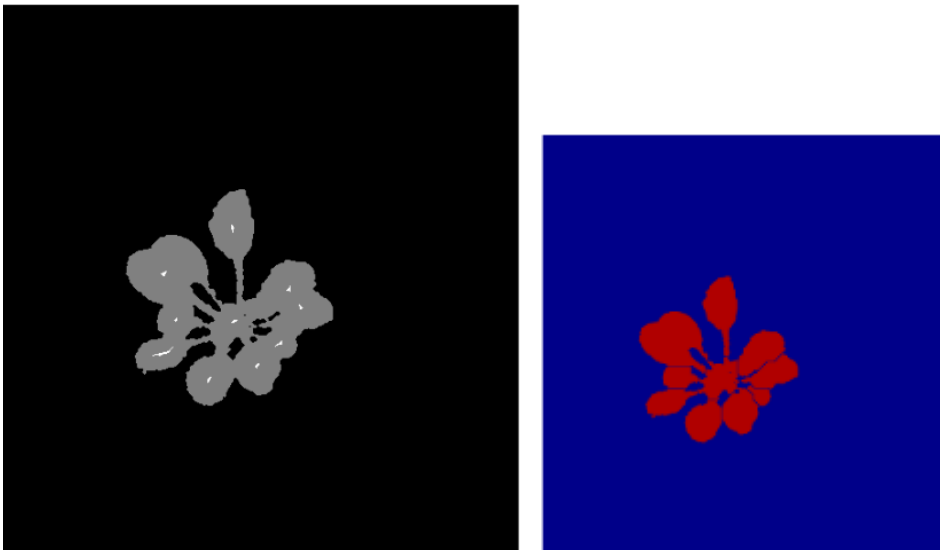


Figure 3.1.1 Applying a mask using imextendedmin

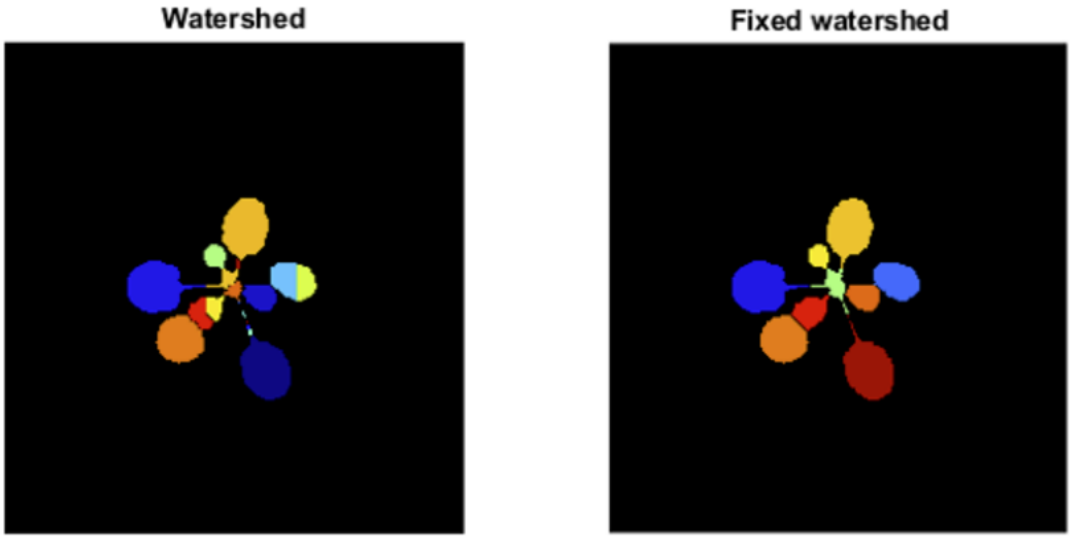


Figure 3.1.2: Comparing normal watershed with fixed watershed.

3.2 Color Randomization

It should be random for the program to generate different colors for each leaf after segmentation. Firstly, I created a simple random series of data, which returns a random array of numbers in order for the resulting images to consist of random colored leaves. Then a color map was made out of matrix values which define colors for graphic objects such as images and patch objects.

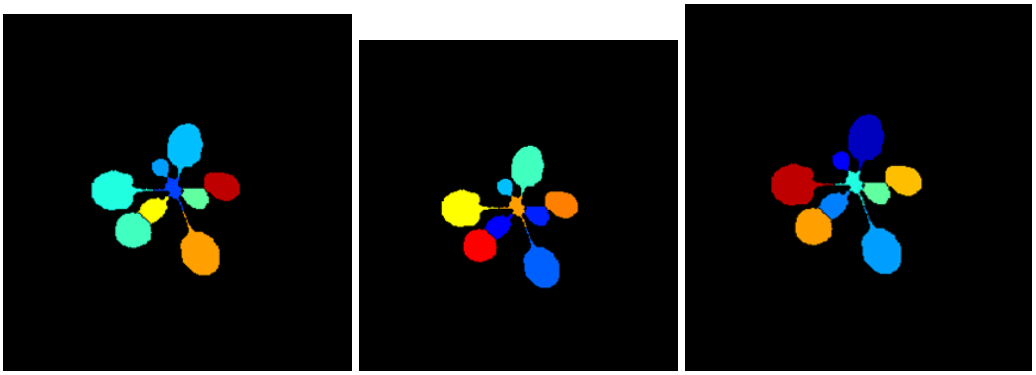


Figure 3.2.1: Showcasing randomness of each leaf colour.

4. Final Results

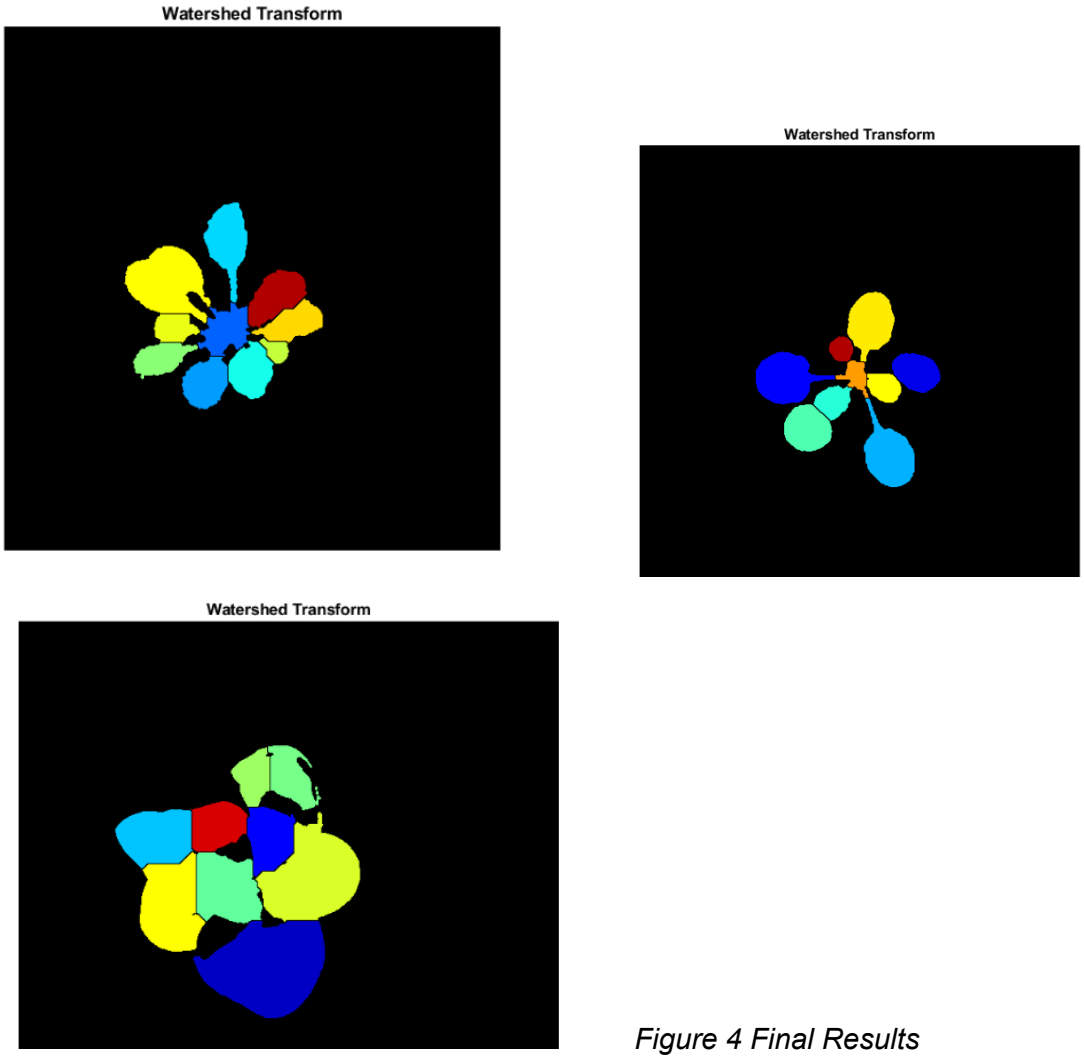


Figure 4 Final Results

In the case of the image of plant2, Watershed produced the best results as it did not detect any overlapping leaves or noise, so all the required regions were correctly identified.Due to the noise in image 1 and the poor quality of that image, the watershed detected almost all of the leaves, but there were some overlaps.Our algorithm accurately segments the outer edges of the leaves in Image 3, but the results were below average.This is caused by the presence of overlapping leaf segments, leaf veins, and shadows in our image, resulting in oversegmentation because the watershed will segment regions along its gradient, which means some of these sharp elements will act as natural borders during segmentation.

5. Weakness of the process

As can be clearly seen from Figure 4 Final Result the process is inoperable with Plant 3 and doesn't produce the desired output. There are multiple reasons for this. The features which contribute to this fact are that there are shadows in the leaves due to which image binarizing doesn't work properly and finding a proper threshold value is hard secondly Leaf veins are sharp. If the image has too many "sharp" elements. Watershed will segment regions following gradients. Since the mage has too many sharp elements, they act as natural borders during segmentation resulting in improper segmentation since it is trying to segment shadows and leaf crack veins.

6. References

https://medium.com/spidernitt/image-preprocessing-why-is-it-necessary

https://blog.roboflow.com/why-preprocess-augment/

https://en.wikipedia.org/wiki/Channel_(digital_image)

https://www.geeksforgeeks.org/image-sharpening-using-laplacian-filter-and-high-boost-filtering-in-matlab/

https://www.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html