Introduction:

I think this assignment is about the code being able to take in certain images and estimating the percentage of ground pixels in a given image. The code will take in a certain number of files containing both their label and original image. A nested for loop will run through each of the pixels and their colors, and store whether that colored pixel is a ground pixel or not based on the label image. Having that information, the code will then be able to take all pixels within that region to find what colored pixels associate with ground pixels in the images to formulate a percentage of ground pixels for the remaining images provided. The more data there is, the more accurate the results will be when displaying which pixels identified as ground pixels on the testing image(s).

Method:

The method used for this assignment was to take one or more images to train along with its label image and store in a matrix the exact color of the pixel along with whether it was represented as a ground pixel or not stored in two separate variables. By using more images to train, this will allow more pixels to be added to the matrix to make it clearer as to if the rgb value of the pixel truly represents a ground pixel or not. After training the necessary images, each element in the matrix was divided by the total amount of pixels total to show a percentage of which pixels are ground and which are not. Using this information, while testing the other image(s), the code was able to identify which pixels are ground pixels by seeing if the percentage of the pixels pertaining to that exact rgb value were higher as a ground pixel or not. If most of the pixels with that rgb value were considered ground pixels, then this image would also have a ground pixel and was displayed on the original picture as red pixels and the label image as white pixels.

Experiments:

Figure 1: Training images 2 and 3. Testing image 1

Precision: 0.787838
Recall: 0.222556
Micro F-Score: 0.347069
Macro F-Score: 0.347069

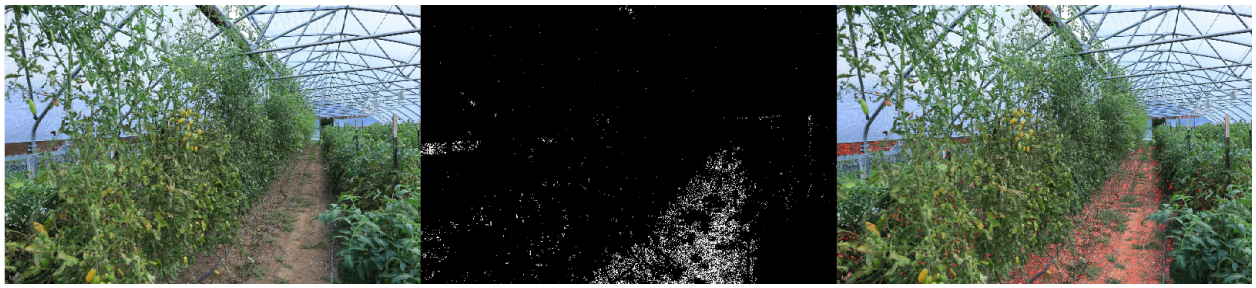


Figure 2: Training images 1 and 3. Testing image 2
Precision: 0.775887
Recall: 0.678895
Micro F-Score: 0.724158
Macro F-Score: 0.724158

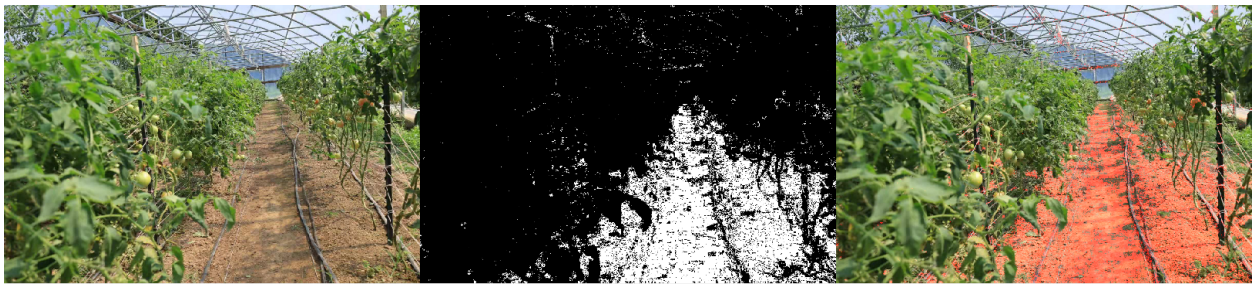


Figure 3: Training images 1 and 2. Testing image 3
Precision: 0.902111
Recall: 0.425828
Micro F-Score: 0.578557
Macro F-Score: 0.578557

Figure 4: Training images 1, 2, and 3. Testing image 1
Precision: 0.936104
Recall: 0.482160
Micro F-Score: 0.636485
Macro F-Score: 0.636485

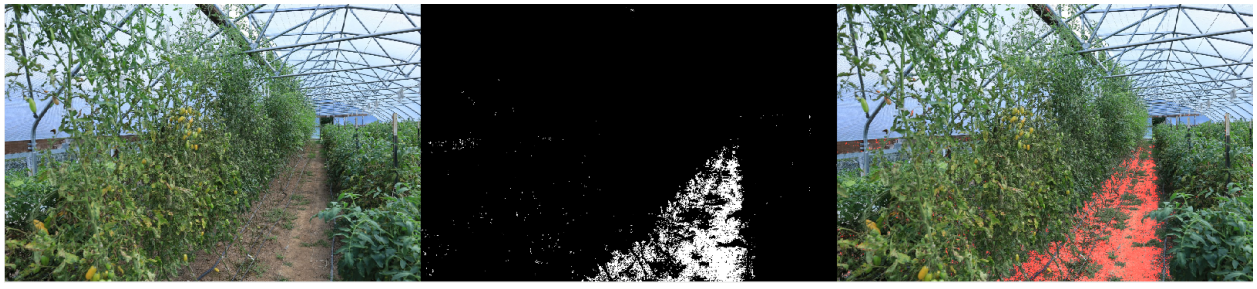


Figure 5: Training images 1, 2, and 3. Testing image 2
Precision: 0.857683
Recall: 0.682505
Micro F-Score: 0.760132
Macro F-Score: 0.760132

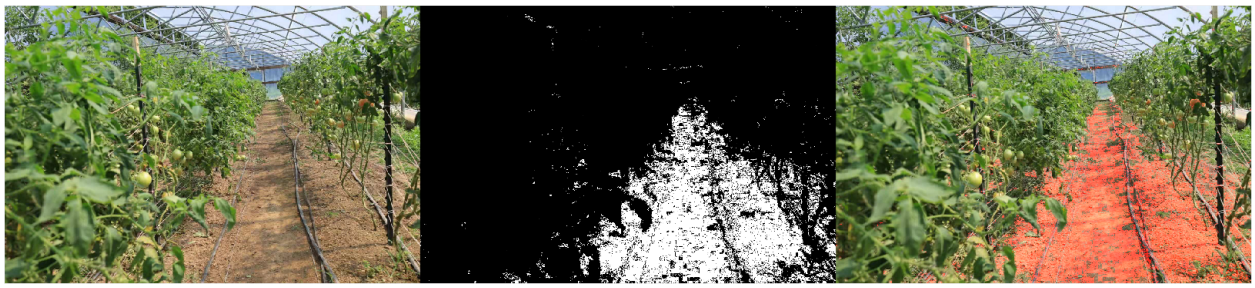
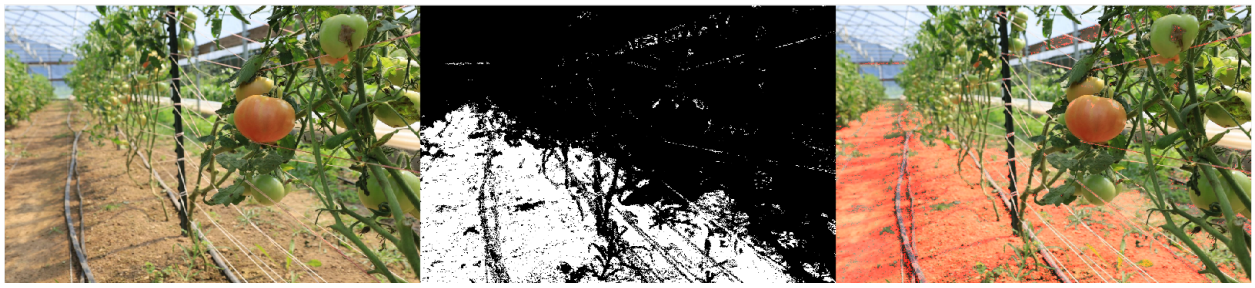


Figure 6: Training images 1, 2, and 3. Testing image 3
Precision: 0.951142
Recall: 0.740738
Micro F-Score: 0.832857
Macro F-Score: 0.832857



Discussion:
Like mentioned before, I noticed that having more data by using more testing images causes the detection of ground pixels to be more accurate. Therefore, this code will have an easier time detecting ground pixels correctly when the percentages of the ground pixels are more exact. Notice how in each of the pairs of testing images, if there was an additional training image, the precision of the testing image would go up. For testing image 1, it went from about 78% to around 93%. Testing image 2 went from 77% to 87% precision and testing image 3 went from 90% to 95%. In addition to this, each of the recall values also went up for each image going from two to three testing images. Not only does this show how using more data leads to more accuracy, but also leads to a higher F-Score, which basically states the same idea. It is clear that the more data there is, the more accurate the results will be when detecting whether pixels are ground pixels or not. The method appeared to have worked well in the sense that there was between an 87% and 95% precision rate.