CSE 353 Assignment 1
Ground Region Detection Report

Chengzhi Dong
ID 112890166

Due September 16, 2021

1 Introduction

A robot is being designed to help farmers harvest vegetables and fruits in a greenhouse. This is an important ability for the mobile robot to understand which direction is the correct path to move, so the robot will not destroy any plants and facilities by crushing over them. In this assignment, I am using Bayes' rule to design an algorithm that can detect where the ground is for a mobile robot to move. Using the images of the greenhouses as a training set, the algorithm can well estimate the ground and non-ground area in the new pictures taken by the robot.

2 Method

My algorithm, which is coded in Python, is primarily based on the Bayes' rule:

$$Pr(y|x) = \frac{Pr(x|y)Pr(y)}{Pr(x)}$$

It uses the RGB values in each pixel of the greenhouse to estimate. (Note: I use Opencv to read the image, so instead of RGB, the values in my code are in the order of BGR). The training data set consists of the sample images of the greenhouse and their ground truth labels. In the case of ground region detection, the variable y represents a pixel is a ground pixel or a non-ground pixel, and the variable x represents the pixel's RGB values. Therefore,

$$y = \begin{cases} 0 & \text{if this is a non-ground pixel} \\ 1 & \text{if this is a ground pixel} \end{cases}; \qquad x = \Big\{ (R,G,B) \quad s.t. \quad R|G|B \in [0,255]$$

The prior $Pr(y)$ represents the probability of ground/non-ground pixels in the images. This can be obtained by count all the ground/non-ground pixels on the

training images and divide them by the total pixels of the training images:

$$Pr(y): \begin{cases} Pr(y=0) & \text{the probability of a non-ground pixel on the image} \\ Pr(y=1) & \text{the probability of a ground pixel on the image} \end{cases}$$

The likelihood $Pr(x|y)$ or $Pr((RGB)|y)$ represents the probability of the RGB value of a pixel given that it is a ground/non-ground pixel. This can be obtained by created two 3-D arrays of size $256 \times 256 \times 256$ (one for ground, one for non-ground), and each cell of the array represents an RGB value. Count all the ground/non-ground pixels with specified RGB and store them into the array. Then, divide the count in each cell by the total pixel of ground/non-ground:

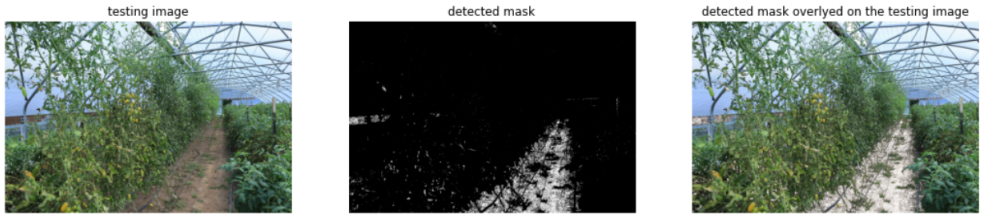
$$Pr(x|y): \begin{cases} Pr((rgb)|y=0) & \text{the prob. of a rgb value given it is a non-ground pixel} \\ Pr((rgb)|y=1) & \text{the prob. of a rgb value given it is a ground pixel} \end{cases}$$

For testing, I need to calculate and compare the posteriors. The posterior means the probability of a pixel is ground/non-ground given that its rgb value is x. The posterior can be calculated by Bayes' Rule, $P(y|x) = \frac{Pr(x|y)Pr(y)}{P(x)}$. When I compare $P(y=1|x)$ and $P(y=0|x)$, I am actually comparing $\frac{Pr(x|y=1)Pr(y=1)}{P(x)}$ and $\frac{Pr(x|y=0)Pr(y=0)}{P(x)}$. Notice that the evidence $Pr(x)$, the prob. of a pixel having rbg value of x, is on both side, and they will cancel each other out when comparison. Therefore, the evidence $Pr(x)$ is unnecessary in this case. I only need the likelihood and prior to estimate a pixel is ground or not given its rgb value. In the code, the algorithm create a 2-D array with the size of the image, and initialize each cell to zero (non-ground). Then it compare each cell's $Pr(x|y=1)Pr(y=1)$ and $Pr(x|y=0)Pr(y=0)$ base on its rgb value. If $Pr(x|y=1)Pr(y=1) > Pr(x|y=0)Pr(y=0)$, the algorithm detect the pixel as ground pixel and change the value in the 2-D array to 1. Then, I calculated the true positives, false positives, false negatives, precision, recall, and F-score using the formulas.

3 Experiment

I tested my code using "Tunnel-02" as the training image set, and "Tunnel-01" and "Tunnel-03" as the testing image set. Here is my result:

```
Result Analysis for testing image 1 :
True Positives = 68827
False Positives = 15895
False Negatives = 108135
Precision = 0.8123863931446378
Recall = 0.38893660785931444
F-score = 0.5260313966463368
```



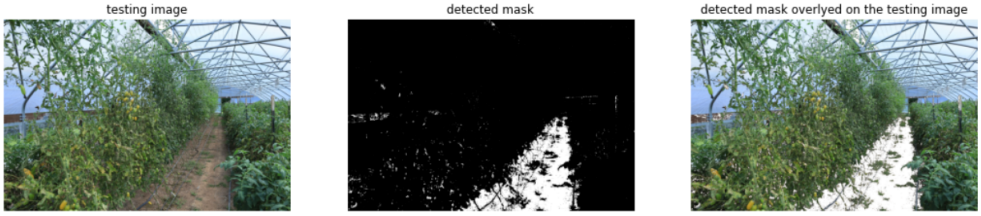

```
Result Analysis for testing image 3 :
True Positives = 428110
False Positives = 54621
False Negatives = 213804
Precision = 0.8868500262050707
Recall = 0.6669273454076403
F-score = 0.7613246846782763
```


```
----------------------------------------
Final Result Analysis:
Total true positives = 496937
Total false positives = 70516
Total false negatives = 321939
Overall precision = 0.8757324395148145
Overall recall = 0.606852563758127
Micro F-score = 0.7169106323246502
Macro F-score = 0.6436780406623066
```

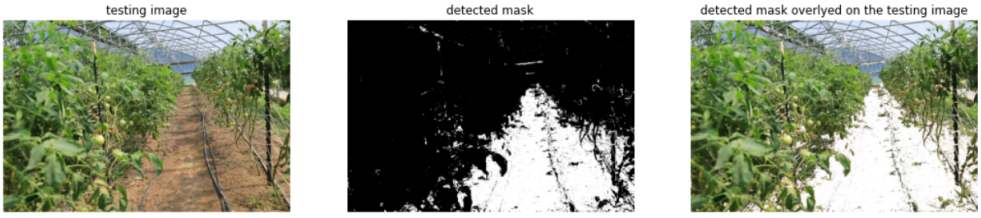
The precision, recall, f-score for Tunnel-01 are 0.81, 0.39, and 0.53, respectively. The precision, recall, f-score for Tunnel-03 are 0.89, 0.67, and 0.76, respectively. The overall precision, overall recall, micro f-score, and macro f-score are 0.88, 0.61, 0.72, and 0.64, respectively. The testing results are decent. Although there is only one image in the training set, most of the scores are above 0.5 with exception of Tunnel-01's recall. Just by looking at the result images of the detected mask with our human eyes, we can easily and clearly determine which areas of the images are ground. The algorithm is working as I expected.

I test the algorithm again with "Tunnel-01", "Tunnel-02", and "Tunnel-03" all as training set and testing set. Here is my result:

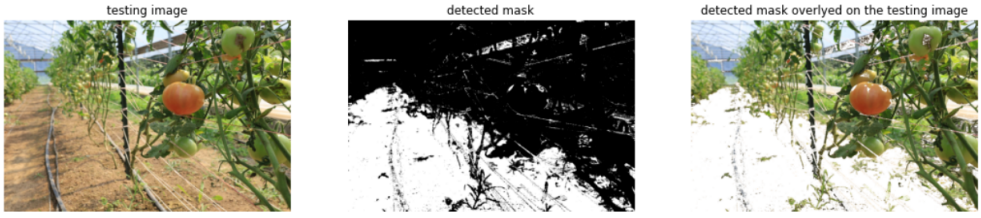
```
Result Analysis for testing image 1 :
True Positives = 125916
False Positives = 19634
False Negatives = 51046
Precision = 0.8651047749914119
Recall = 0.7115425910647484
F-score = 0.7808453638934364
```




```
Result Analysis for testing image 2 :
True Positives = 317528
False Positives = 85592
False Negatives = 58031
Precision = 0.787676126215519
Recall = 0.8454810029848838
F-score = 0.8155555755324081
```




```
Result Analysis for testing image 3 :
True Positives = 559016
False Positives = 63552
False Negatives = 82898
Precision = 0.8979195846879378
Recall = 0.8708580900245204
F-score = 0.8841818230706329
```



```
----------------------------------------
Final Result Analysis:
Total true positives = 1002460
Total false positives = 168778
Total false negatives = 191975
Overall precision = 0.8558977765407202
Overall recall = 0.839275473340952
Micro F-score = 0.8475051285617242
Macro F-score = 0.8268609208321592
```

This time I used all three images as the training set. As shown above, the result improved by a significant amount. Most of the scores are above 0.8, and we can see in the detected mask images that more ground pixels are labeled compared to the last test. The more images there are in the training set, the more accurate the estimation is. This algorithm did a great job in detecting ground regions.

4 Discussion

When I am comparing $Pr(x|y=1)Pr(y=1)$ and $Pr(x|y=0)Pr(y=0)$, I had to decide whether to use $>$ or $\geq$ for comparison. I tried both and found out that $>$ works a lot better than $\geq$. The main reason is the limited training data set. There is not enough data for all RGB values, which will result in many cells in the RGB arrays remains 0. Comparing 0 to 0 is meaningless and count it as a ground pixel will reduce the correctness of the estimation.

One more thing I realize with my algorithm is that it is slow. Every run takes approximately one minute to execute. This speed is too slow for a real-life application. The mobile robot may need to take pictures and make ground region detection every few seconds. One way to improve that is to increase the range per RGB value. Now, I am using a $256 \times 256 \times 256$ matrix to store and calculate the probability. This is too precise and the detection does not need to be that accurate. I may reduce the matrix to $50 \times 50 \times 50$, and set a range for each cell, so all RGB values within the range belong to that cell. This improvement will decrease the running times of my algorithm by a lot.

We also can introduce more variables into the algorithm to make it more reliable. For example, instead of using a normal png image, we can use a radar camera that can record the distance and temperature of the object in the photo. Adding distance and temperature variables can make the algorithm to be more accurate.