James Donnelly
Jfd235
ECE5242
13 February 2023

Project 1 Report

Introduction

This report outlines the problem, processes, and results surrounding the first project of ECE5242. This project served as a refresher of standard probability concepts for me, as well as an introduction into building generative models, which is something I had very little prior experience with. I am leaving this project with what I believe to be a newfound and solid understanding of these topics, which I hope to make evident throughout the report.

Problem Statement

The problem, simply put, is to build a model that locates traffic cones in a set of images, and outputs the location of the cones as well as their distances from the camera. Given to us was a directory of training images, as well as the distance of the cones in each image.

Approach Description

The general idea was to use a multivariate Gaussian distribution in conjunction with Bayes' Theorem to determine the probability of a certain pixel being a cone based on its color. The first step in achieving this was understanding the inverse of the problem. That is, given a certain pixel is a cone, what colors is this likely to be? This probability could then be flipped using Bayes' Theorem to find our desired output.

To begin the calculations, it was decided that an RGB color space would be used. This was done for convenience, as reading in the image files using *matplotlib* and *imread* provided RGB values for each pixel in a range from 0 to 1. The pixels being read in this format allowed for the construction of a multivariate Gaussian distribution for the red, green, and blue values for each pixel. This is simply done by calculating the log likelihood for the trained data. The data was trained using RoiPoly in two classes, one class for cone pixels, and another for all pixels that aren't a cone. A third class was also trained to capture pixels that were similar in color to the cones, such as red carpets and chairs, but this did not improve the model enough to be worth the computation, so it was not used in the final program.

After attaining the output of the multivariate Gaussian distribution, Bayes' rule was used to flip the probability to the desired output, as mentioned earlier. With the math and computation out of the way, now came the question of finding suitable threshold values and tweaking the model. As the output of the model was a probability ranging from 0 to 1, multiple thresholds were experimented on using a holdout set from the training data. In the end it was found that a very high threshold (0.999999) was most effective in properly discerning the cone pixels from the rest to create a mask. On top of this, erosion and dilation were used to get rid of any unwanted noise in the mask, leaving only the desired cone pixels. The values used for erosion and dilation were experimented with using a holdout set, and an erosion disk of 4 pixels coupled with a dilation disk of 8 pixels were found to be optimal. To build bounding boxes and determine the location of the cones, code from the regionprops documentation was used to find the boxes and their centroids. In addition, simple shape analysis was used to make sure that the box height was higher than its width, as that was the general shape of the cones given in the training data.

Finally, the cone height was determined by examining the number pixels in the cone's height against its distance from the camera in the training data, as the pinhole model states this relationship exists. This allowed for a linear regression that fit the data for ease of computation as shown in the figure below.

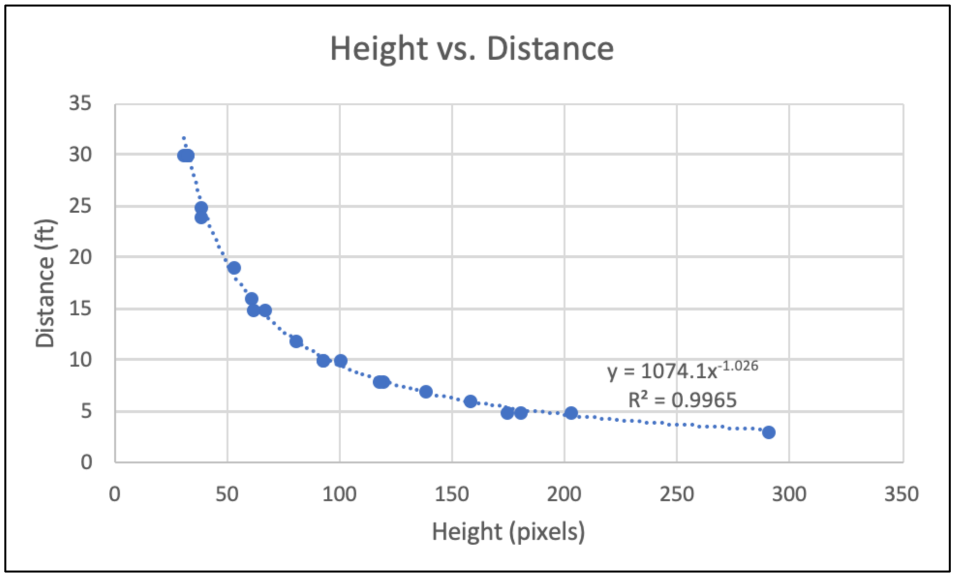


Figure 1: Cone Height vs. Distance

The height of the bounding boxes were then used to determine the distance from the camera.

As the Gaussian distribution proved to be effective in locating the cones in the training set through holdout testing, it was ultimately decided that a mixture model would not be used, as the desired results were achieved without the need for further complexity. The gaussians for the different classes can be seen in the figure below. Red, green, and blue lines represent their respective colors, solid lines represent the cone class, and dotted lines represent the not cone class.

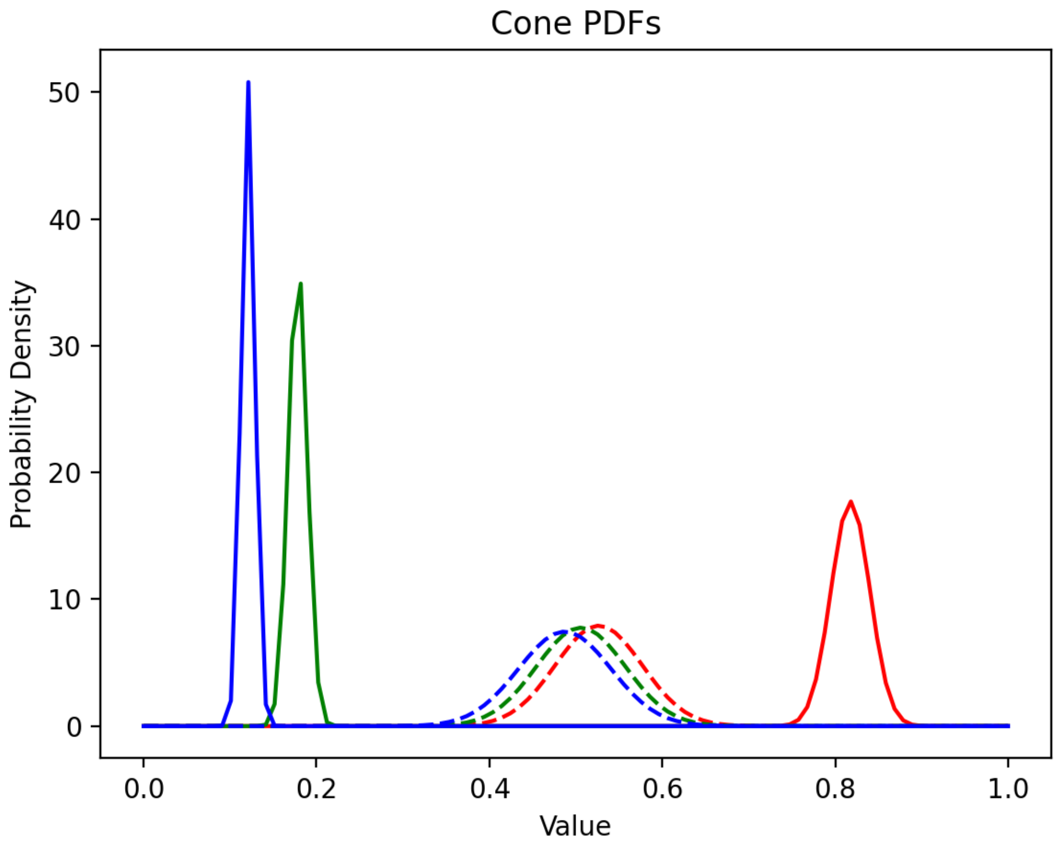
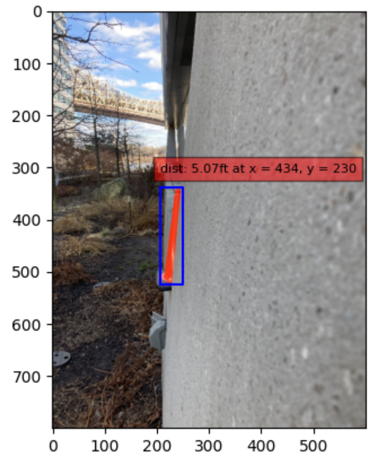


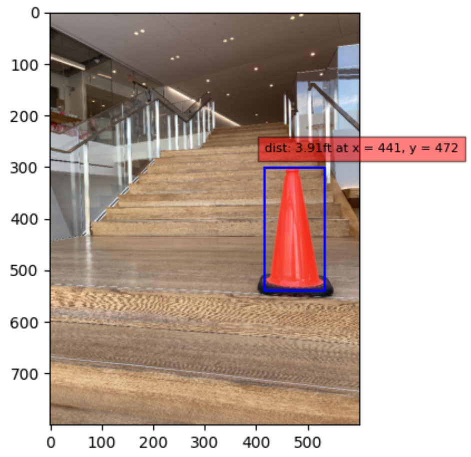
Figure 2: PDFs

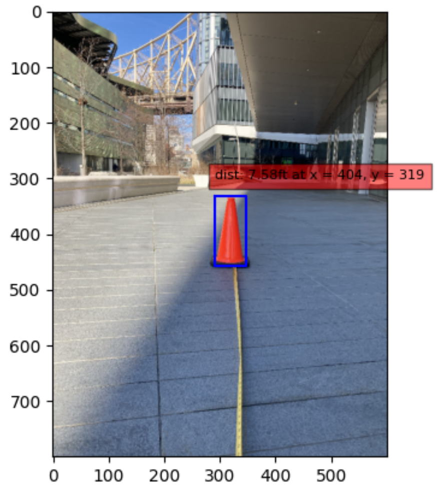
The process was optimized by pre-calculating a lookup table, to store the probability output of every possible RGB value, quantized to the most significant 6 bits. This sped up runtime from a few minutes per picture all the way down to a few seconds per picture.

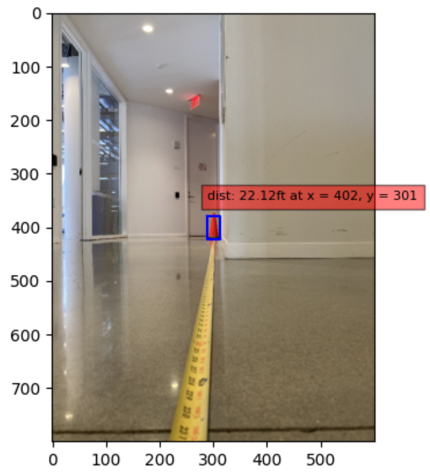
Results

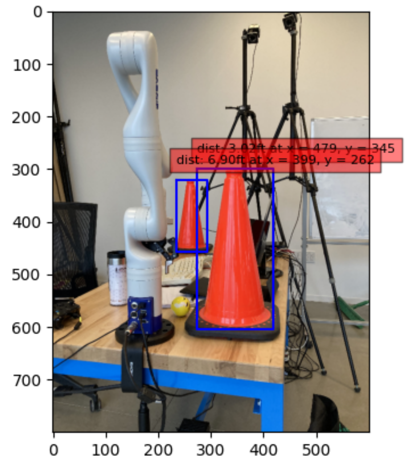
The output of the program was successfully able to identify all of the cones in the test images as shown in the figures below.











Figures 3-7: Model Output

Shape statistics were only needed for these pictures for taking out the exit sign in the background of one of the images. All other cases were handled by erosion, dilation, and thresholding. The picture with the farthest cone appears to be slightly overestimated for distance, as the bounding box height is a few pixels too short for the cone. However, all of the others appear to fit the cones fairly well.

Overall, I’m very happy with the final result of the project. Having come in with such limited knowledge and experience on the topics, I was very proud to see a model that worked so well, as well as to have learned as much as I did.

References

https://scikit-image.org/docs/