ECE 276A:Sensing and Estimation
Project 1

Akanimoh Adeleye

I. Abstract

***Abstract*–This paper covers an approach to using a Gaussian Discriminate Model to classify pixels and theoretically group similar classified pixels to determine the location of a blue bin object.**

II. Introduction

The classification of different groups is a useful tool and has applications in many different fields. A common yet important example is the case where a hospital patient is being diagnosed and we need to tell if a tumor is malignant or benign. Similarly there are many other cases where being able to group some unknown item is helpful directly or indirectly towards a goal.

There are many different classifiers in machine learning and while often similar in their approach, they can have negligible to drastically different classifications. The general break up of machine learning models fits into two groups, discriminate and generative. A discriminate model uses past data to create a decision boundary between the classes. This can be thought of as a function or a line that distinguishes all classes. A generative model is more generative and rather, tries to model the actual distribution of each class.

In order to identify and bound a specific object - a blue bin - within a set of images, we take the approach of first classifying a each individual pixel then grouping pixels to see if they match with our expected object. The rest of this paper explains our approach towards this.

III. Problem Formulation

A. Classification

The objective of this method is to use pre-labeled example pixels to create a model capable of predicting a label given new examples.

For this we use Gaussian Discriminate Analysis(GDA). Let $p(y, X|w, \theta)$ where y is a set a labels, X is a set of pixels, θ and w are parameters. We expand this joint probability as follows:

$$p(y, X|\theta, w) = p(y|\theta)p(X|y, w) = p(y|\theta)\prod_{i=1}^{n} p(x_i|y_i, w) \tag{1}$$

This is possibly because the parameter θ is independent of X and similarly w is independent of y. This is the general form of GDA. We are now free to attribute a distribution to each probability as we see fitting. For our problem we choose to stick with a categorical and normal distribution (which transform this general from to a true GDA).

$$p(y|\theta) = \prod_{i=1}^{n}\prod_{k=1}^{K} \theta_k^{\mathbb{1}(y_i=k)} \tag{2}$$

$$p(x_i|y_i = k, w) = \phi(x_i; \mu_k, \Sigma_k) \tag{3}$$

The parameters of these models can be determined by starting with a random non-zero value an performing Maximum Likelihood Expectation (MLE). When it is not easily done, we can solve a constrained optimization problem instead. The process for this is outside the scope of this paper but result in the follow:

$$\theta_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = k) \tag{4}$$

$$\mu_k = \frac{1}{n}\sum_{i}^{n} X_i \tag{5}$$

$$\Sigma_k = \frac{1}{n}\sum_{i}^{n} (X_i - \mu_k)(X_i - \mu_k)^T \tag{6}$$

With this, given a new example x^* we can find y^* by:

$$y^* = argmax_{y\epsilon(1..K)} log\theta_k + log(\phi(x_i; \mu_k, \Sigma_k)) \tag{7}$$

IV. Technical Approach

This section gives an overview to the algorithms implement. We go also discuss why the where chosen and go over there implementation in detail

A. Classifying the Pixel

When first looking at the problem it seemed like either a discriminative or generative approach would work well given the enough data. We choose to implement a generative model simply out of preference.

At first we believed a Gaussian Naive Bayes (GNB) model would work well and would be relatively simple to implement. Upon implementation we decided that although given enough data the model might still be able to classify correctly, because we know the dimensions are inherently not i.i.d, it would be best to use another model.

We chose to instead use Gaussian Discriminat Analysis because it relives this assumption and was close to GNB.

In order to build the GDA model, the necessary sample data must first be collected. This meant creating a matrix of pixels (X) along with another matrix of corresponding labels (y).

At first we tried to use the *roipy* package but do to system incompatibilities, the images were hand cropped using Mac photo editor. Each region cropped was place in a folder corresponding to its category: *Blue bin , Grey, Not blue*.

OpenCV was used to read in all cropped images and handle image manipulation. Before creating the sample data x and label y, the images where transformed from BGR color space to HSV. After this, each cropped image was simply iterated over for all it's pixel dimensions and all pixels were stored to numpy array while numpy array to indicate what folder (category) the pixel belong to was simultaneously created.

With this the needed examples and labels where created and our weights where trained using the method described in section III.

Once the model was fully trained we moved on to our main object of classifying unseen pixels. Much like with training, the image was read in with OpenCV and transformed to the HSV color space. Then, traversing thorough the image pixel by pixel, we classified whether or not it belong to Blue class by choosing the class that satisfied equation (7). To make processing faster we found it helpful to filter out all pixel values outside of the range of known hsv blue color space, with some tolerance. With this we where able to successfully classify pixels.

We note that due to unfamiliarity, it took quite a sometime to understand that the multivariate gaussian probability density function was different from the single variable gaussian probability density function as well as a few other tidbits. In the future we hope to collaborate earlier to save time.

Determining the Bounding Box

After identifying which pixels belongs to what class. Our approach to bounding and identifying the blue bin was as follows: using the OpenCV class SimpleBlobDetector() or Contours(), identify regions of blue pixels that are close to each other within the masked image. Calculate the height of each blob as well as the width. If the ration of that blob is within a certain tolerance as well as the area of that blob, consider this a bin and draw a rectangle from the center of the blob. Although we were not able to fully implement this class, we believe this approach would solve this problem partially. There are edge cases where the bin in not fully visible or is obstructed that we did were not able to cover.

V. Results

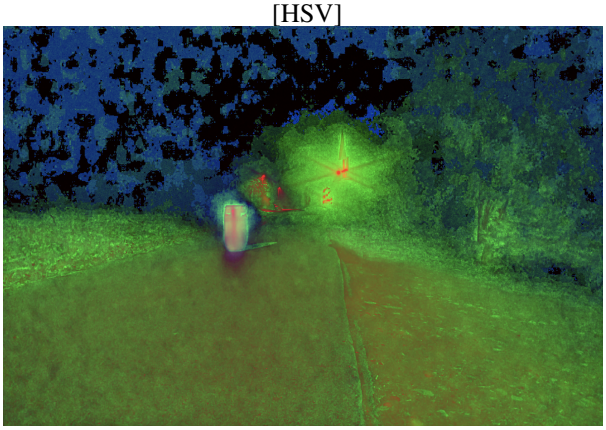
In this section we display our results from prepossessing our images as well as our color classification. We conclude with a brief discussion the our process. The final weights for our model is shown below along with two example images that show our method from reprocessing step to the segmentation step.:

```
Theta:
  [0.07604698 0.3319201  0.59203291]
Mu:
 [[ 87.99369347 159.5087348  111.97058752]
 [ 22.25998531  45.5541095   72.96446487]
 [ 37.42076222  33.20938344  98.1416361 ]]
Sigma:
 [[[1761.9792303  3084.40613586 2080.36619263]
  [3084.40613586 7603.74179424 4709.38447107]
  [2080.36619263 4709.38447107 4832.9376605 ]]

 [[1027.21251521  357.9862781   333.47551349]
  [ 357.9862781  2163.02411076 1460.92860276]
  [ 333.47551349 1460.92860276 5608.06310424]]

 [[2108.71426005   81.94912445 1762.75815146]
  [  81.94912445 1359.41903419  153.9425878 ]
  [1762.75815146  153.9425878  6072.95113855]]]
```

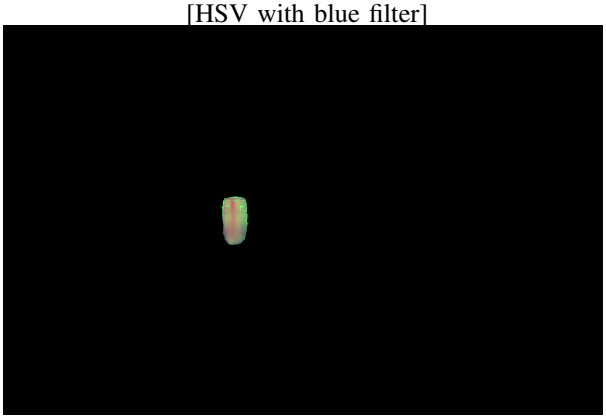
The first half of the method implemented works well but I know their are cases where the segmentation fails and believe this could be improved by collecting more data.

[BRG]

[HSV]



[HSV with blue filter]



[Segmented]

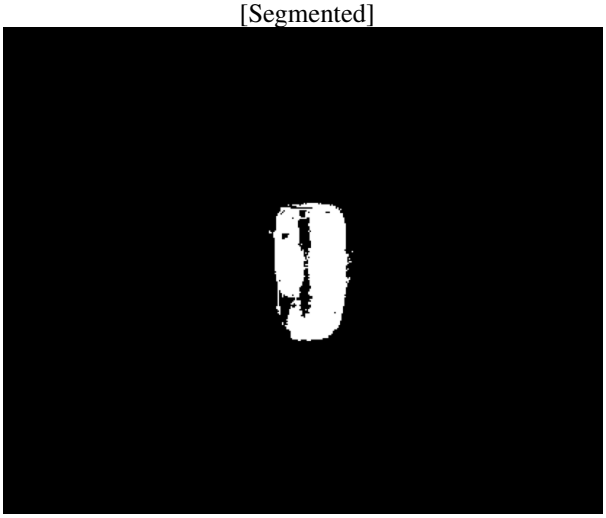
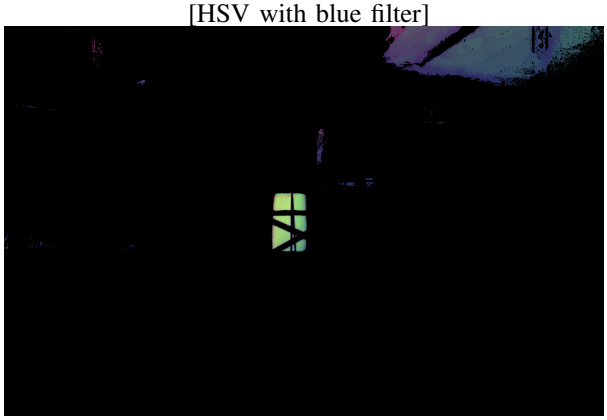


Fig. 1.

[BRG]

[HSV]

[HSV with blue filter]



[segmented]

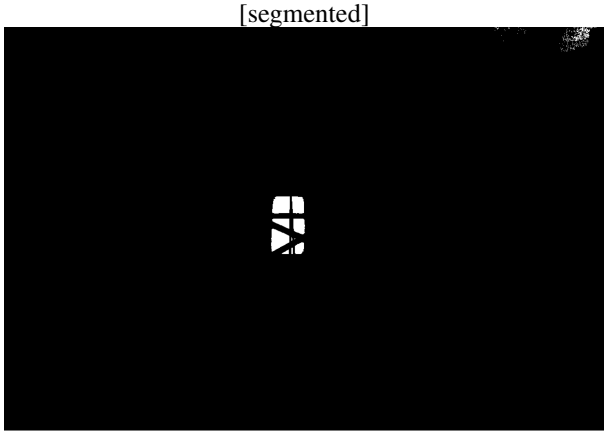


Fig. 2.