

# ECE 276A: Project 1

## Colour Classification and Recycling Bin Detection

Chaitanya Tambat  
Department of Mechanical and  
Aerospace Engineering  
University of California San Diego  
ctambat@ucsd.edu

**Abstract—** This project discusses the training and execution of a probabilistic color model from pixel data to distinguish among red, green, and blue pixels. It further discusses training and executing a probabilistic color model to recognize recycling-bin of blue color and using it to segment unseen images into blue regions using Python and its libraries.

**Keywords—** segmentation, classification, Python, Python-libraries, probability, Gaussian Naïve-Bayes, pixels, class

### I. INTRODUCTION

Machine learning and artificial intelligence continue to be active research fields focused on realworld problems. Machine learning uses computers to make predictions based on the provided data set or previous experience. Using machine learning methods such as supervised and unsupervised learning, we can process large data and solve classification problems.

High-level (semantic) image classification can be achieved by analysis of low-level image attributes geared for the particular classes. In this paper, we have proposed a novel application of the known image processing and classification techniques to achieve such a high-level classification of color images.

Regions of interest (ROI) usually means the meaningful and important regions in the images. The use of ROI can avoid the processing of irrelevant image points and accelerate the processing.

The project deals with training a probabilistic model to interpret and process image data in form of pixels and achieve the aim of determining regions of interest. The part 1 of this project deals with pixel classification into three classes and part 2 deals with using similar model for image processing to obtain particular regions of interest.

### II. PROBLEM FORMULATION

#### Problem 1

##### A. Pixel Reading

The first step of this problem is to read each pixel and obtain its RGB values to train a probabilistic model. Each RGB values of a single pixel image from the red, green and blue classes needs to be determined. The

values obtained are 8-bit data and range between 0-255. We convert each value to 0-1 range by dividing it by 255 and this data is stored in the data structures.

##### B. Setup of training model

Once the data for all the pixels from the training data is collected, this data is used to train a probabilistic model. The decision of the validation pixels will be based on this data and the model trained using it. We will be using a Gaussian distribution-based probability model the Gaussian Naïve Bayes classification by determining the mean and variance of the distribution and also the posteriori probability of the element. The class of each pixel is determined based on this probabilistic model.

Equation...

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Figure 1: Gaussian Distribution

$\mu$  is the mean of the class data and  $\sigma$  is the standard deviation of this class.

##### C. Classification of pixel using trained model

Based on the trained model, each pixel from the validation set is chosen and the decision of its class is based on the trained model and its own RGB values.

The initial probabilities denoted by  $\mathbb{D}(\theta)$  are found out by the number of sample pixels of each class. This is an initial guess for the pixel class. They can be represented as  $P(1)$  for class red,  $P(2)$  for class green, and  $P(3)$  for class blue.

The post probability of the selected pixel is found out in terms of the distribution of each class and the highest probability is selected. These probabilities are denoted by  $P(Y|1)$ ,  $P(Y|2)$  and  $P(Y|3)$  which represent the probability of the pixel belonging to a class given the pixel's RGB values.

The classification of the pixel will give an output in the form of the class value it will belong to. In our case, the classes in problem 1 are namely Red (1), Green (2)

and Blue (3). The chosen pixel from the validation set will be put in one of the either class.

The pixel classification will involve a probabilistic distribution, producing unexpected results in cases. The data obtained from the pixels might also influence this.

The problem formulation in problem 1 is similar to problem 2 and hence we can use the same approach to determine the pixel class in problem 2. The only difference in each of the problems is that we need to decide the class for each pixel in problem 2 and just one pixel in problem 1.

### Problem 2

#### A. Multiple Pixel Reading

The problem faced in this part is dealing with multiple pixel data from a single image.



Figure 2: Image 1 from set of training images

From figure 2, we can see that the image has multiple pixels. The data that needs to be extracted from this and collect the data for training our model. From this image we need to select only the pixels that we need to train our model. We will use the roipoly function from the Roi library. The approach to this is discussed in the technical approach section below.

#### B. Setup of training model

Once the data for all the pixels from the training data is collected, this data is used to train a probabilistic model. The process of highlighting the blue bin region will be based on the data collected from these pixels from a data set of 60 images.

#### C. Segmentation of Blue bin regions from other

Based on the training data, the blue bin regions in each image of the test need to be separated from the other regions. We will be using a binary image segmentation to plot the segmented image using the Mathplot library. The blue bin regions will appear as a different colour using imshow.

#### D. Creation of bounding boxes

The last part of the problem 2 deals with binding the blue bin region using bounding boxes. The method is discussed in the next part of this report.

## III. TECHNICAL APPROACH

### Problem 1

#### A. Pixel reading

Each pixel's RGB value is read using `cv2.imread(os.path.join(folder, filename))` But the value obtained using this is in the BGR format and hence we convert it to RGB using the following: `cv2.cvtColor(img, cv2.COLOR_BGR2RGB)` Once the RGB values of each image from the training set are obtained, it is stored in a data structure.

#### B. Setup of Training model

The data obtained from reading these pixels is used to train a probabilistic model. The probabilistic model used in this project is based on the Gaussian Naïve-Bayes probability distribution. The Gaussian distribution uses the Probability Density Function to determine the probability of an element belonging to a certain class.

Each class (R G and B) in this case, will have its own distribution and based on this we will calculate the posteriori probability of a pixel belonging to a particular class. The initial probabilities denoted by  $\Theta(\theta)$  are found out by the number of sample pixels of each class. After this the Probability distribution function (PDF) of each class is found out and this generates the respective PDFs.

The post probability denoted by  $P(Y|1)$ ,  $P(Y|2)$  and  $P(Y|3)$  determine the probability of the pixel belonging to a class. The decision is made based on the highest probability value and the pixel is allocated to the class with the highest post probability value.

#### C. Classification of Pixel

The values of  $\mu$  mean of the selected pixel and  $\sigma$  the standard deviation are found out for each class 1, 2, 3 and each value of R, G, B. The probabilities of each pixel are used to find the posteriori probability using the formula

$$P(Y|1) = \Theta_1 \times \text{PDF}_{1r} \times \text{PDF}_{1g} \times \text{PDF}_{1b}$$

Where  $\Theta_1$  is the initial probability of class1 and PDF values are found out using putting the values of mean and variance and the RGB values of the selected pixel. The decision of the class is made based on the highest

value of the probabilities found out using the above formula.

The verification of the pixel classifier is done using finding the value of data structure Y which stores the value of the class number represented by 1,2 or 3 respectively.

The accuracy of the system is determined using the number of pixels predicted correctly from the validation set to the number of total pixels in it.

In the given problem, we have tested our model on a validation set with all blue elements, hence to obtain the accuracy of our model we will have to check the elements in Y and see how many of them represent 3, i. e. the Blue class.

## Problem 2

### A. Multiple Pixel Reading

In this part of the project, we dealt with images containing multiple pixels. Although we had multiple pixels representing the same class, we selected a small region to train our model to represent a specific class.

We created four classes namely, Bin-blue, Not-bin-blue, Green, Brown. The Bin-blue (1) class included training data which represented all pixels that constitute the blue colour of the bins in all the training images.

Similarly, the Not-bin-blue (2) class consisted of all the data of pixels that were blue but did not represent the blue bins' colour.

The Green (3) class consisted of all the shades of yellow, green and other lighter colours from the training images.

The Brown (4) class consisted of all the shades of red, brown and other darker colours from the training images set of 60.

To select the region of interest, we used the RoiPoly function from python, but limited the number of pixels extracted from each image to 200. Although the pixel selection was limited, we got a very distributed and even data as the pixels selected were randomly selected 200 pixels.

### B. Setup of Training Model

The data obtained from reading these pixels is used to train a probabilistic model. The probabilistic model used in this project is based on the Gaussian Naïve-Bayes probability distribution. The Gaussian distribution uses the Probability Density Function to determine the probability of an element belonging to a certain class.

Each class (1, 2, 3 or 4) in this case, will have its own distribution and based on this we will calculate the posteriori probability of a pixel belonging to a

particular class. The initial probabilities denoted by  $\Theta(\theta)$  are found out by the number of sample pixels of each class. After this the Probability distribution function (PDF) of each class is found out and this generates the respective PDFs.

The post probability denoted by  $P(Y|1)$ ,  $P(Y|2)$  and  $P(Y|3)$  and  $P(Y|4)$  determine the probability of the pixel belonging to a class. The decision is made based on the highest probability value and the pixel is allocated to the class with the highest post probability value.

The pixel belonging to class Bin-blue (1) is stored as 1 and any pixel belonging to other classes (2, 3 or 4) is stored as 0 in a data structure named bin.

### C. Image Segmentation

In the prior step, we have stored value of each pixel in an image as either 1 if it belongs to the Bin-blue class or 0 if it belongs to any other class.

Using the imshow function a segmented image is obtained where the pixel with data 1 is highlighted and others are not highlighted.

In figure 3 we are shown a validation image on which we are going to test our code.



Figure 3: Image 1 from Validation set.

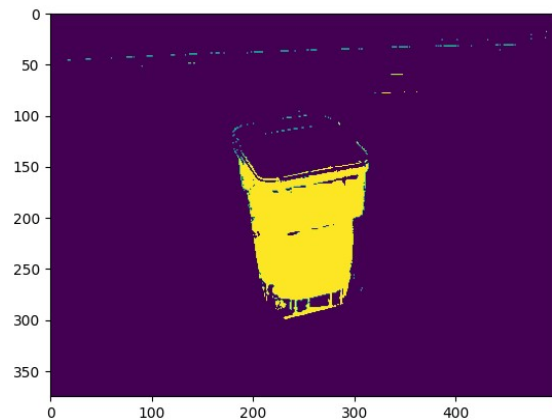


Figure 4: Segmented Image using imshow function

#### D. Bounding Boxes

As shown in figure 4, the segmented image is obtained using plotting the binary data structure which stores 1 and 0's.

The bounding boxes are obtained using the label and regionprops function from the skimage.measure library.

The bounding box for the figure 4 image is obtained as shown in figure 5.

image



Figure 5: Bounding Box on the bin from test image 1

#### IV. RESULTS

##### Problem 1

We obtained an accuracy of 97.59% in detecting the blue pixels from the validation set.

The issue which might've caused the loss of 2.5% accuracy is the fact that some pixels might have an RGB value very close to that of the R and G class. The mean, variance and probability values of each class and the respective RGB are shown in the figures below.

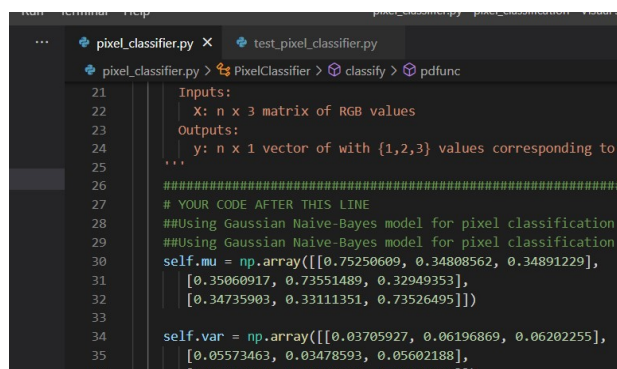


Figure 6: Mean, Variance and Theta of Training sets

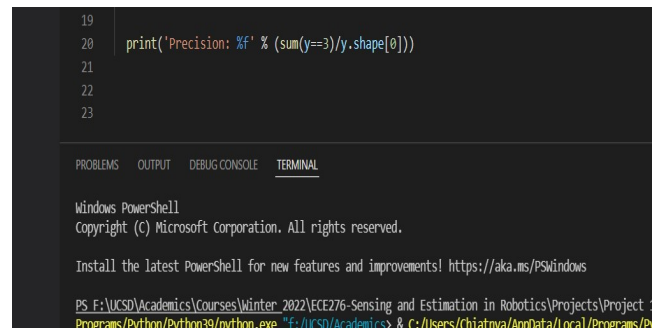


Figure 7: Precision on testing of validation set

##### Problem 2

In problem 2, we obtained segmented and bounding box images as shown in figure 4 and figure 5. This image was detected accurately as it did not contain any other blue areas which might have a RGB value very close to the Bin-blue class (1).

But in image 70 from the validation set, there was a blue colour which was very close to the Blue-bin class and hence, despite not being a bin, the code executed a bounding box for the swimming pool and hence we obtained an accuracy of 0% due to this. The figure 8 shows the original image and the figure 9 shows where the bounding box was executed.



Figure 8: Original test image

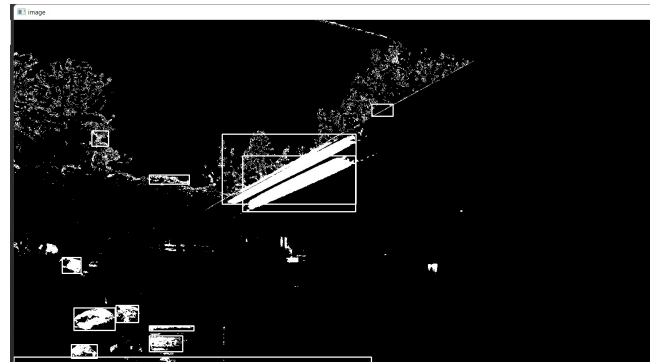


Figure 9: Unexpected bounding boxes for above image



For eliminating these unnecessary bounding boxes, we put a filter on the height and width ratio of the bounding boxes limiting it to a minimum of 1.3. The code for the same is as shown below in figure 10

```
h=abs(prop.bbox[0]-prop.bbox[2])
    w=abs(prop.bbox[1]-
prop.bbox[3])
    if 1.3*h>w and
40000>h*w>10000:
```

Figure 10: Height width ratio code for bounding boxes

Similarly, for test image 65, the blue bin was partially hiding behind the black bin and hence, the bounding box ratio condition was not satisfied and the bounding box was eliminated while calculating the accuracy of the box. Figure 11 shows the original image and figure 12 shows the bounding box on the image.



Figure 11: Original test image 65



Figure 12: Bounding boxes formed on the image

These two issues are a result of similar RGB configuration of the bin blue class (1) and the RGB configuration of the images detected in the images. For eg the swimming pool.

As the two images mentioned above have issues, on testing the bounding box accuracy using the (X0,Y0) and (X1,Y1) values of the bounding box with the accurate values, we get 0% accuracy in these two images and 100% accuracy in the other images. The accuracy for all the images is printed below in figure 13.

```
19 union_area = (box1[2]
20 return inter_area/uni
21
22
23 def compare_boxes(true_
24 ...
```

PROBLEMS OUTPUT DEBUG CONSOLE

```
/AppData/Local/Programs/Python/P
1/ECE276A_PR1/bin_detection/tes
The accuracy for 0061.jpg is 100
The accuracy for 0062.jpg is 100
The accuracy for 0063.jpg is 100
The accuracy for 0064.jpg is 100
The accuracy for 0065.jpg is 0.0
```

Figure 13: Accuracy of bounding box in validation set

The mean, variance and theta values of the classifier are mentioned in the figure 14 below.

```
27 mask_img = a binary image with 1 if the pixel in the original
28 ...
29 #####
30 # YOUR CODE AFTER THIS LINE
31
32 mu = np.array([[0.12305833, 0.26567634, 0.64023492],
33 [0.42882381, 0.57094613, 0.75512574],
34 [0.28762304, 0.34534259, 0.21421107],
35 [0.54224106, 0.43418504, 0.38058496]])
36
37 var = np.array([[0.01222447, 0.02025565, 0.03034596],
38 [0.06225271, 0.05990138, 0.05171289],
```

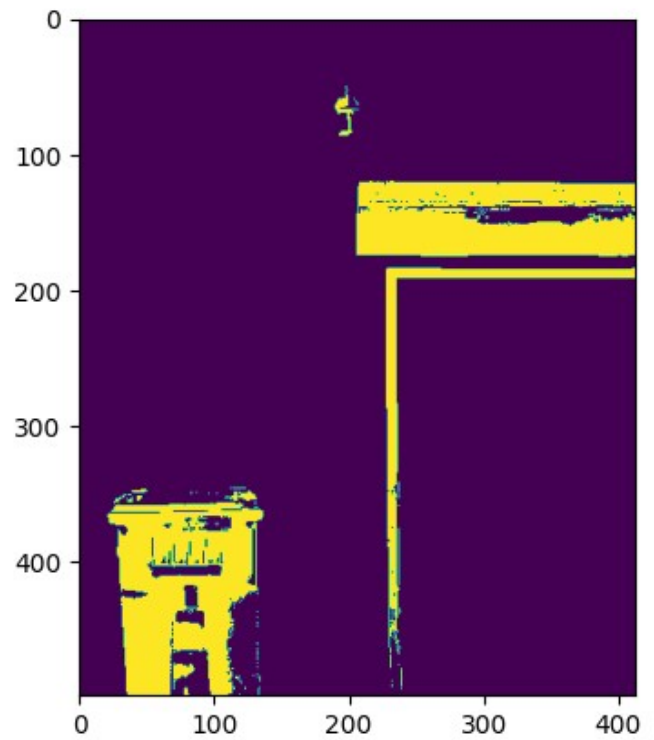
Figure 14: Mean, variance and probabilities of all the four classes and RGB.

The failure of the two images can be attributed to the closeness in data of the RGB values. This could have been resolved by using the YUV colour coding scheme by segregating the bin-blue from the other blue class. Although this might not have always been true as the colour of the swimming pool in the case of image 70 is the same and might have same YUV values. The bounding boxes for various other images alongwith their coordinates are shown below.

## V. REFERENCES

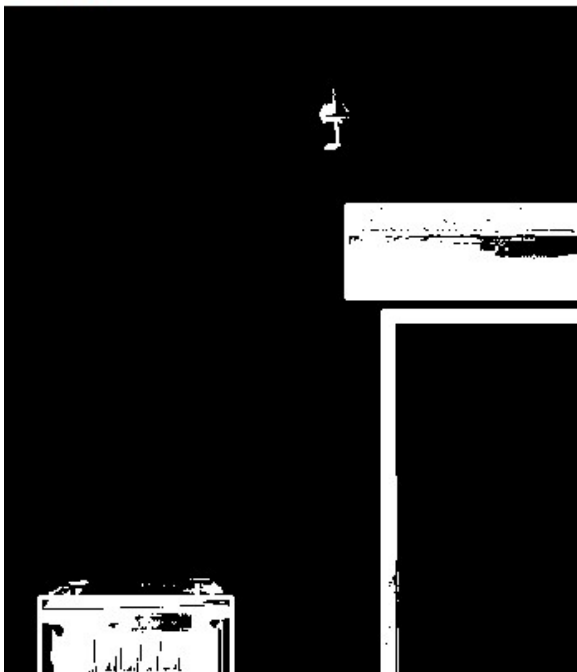
- A. Nikolay Atanasov, University of California San Diego, Lecture Notes- ECE276A, Winter 2022
- B. Q. Zhang and H. Xiao, "Extracting Regions of Interest in Biomedical Images," *2008 International Seminar on Future BioMedical Information Engineering*, 2008, pp. 3-6, doi: 10.1109/FBIE.2008.8.
- C. [https://climserv.ipsl.polytechnique.fr/documentation/idl\\_help/Overview\\_of\\_Working\\_with\\_ROIs.html](https://climserv.ipsl.polytechnique.fr/documentation/idl_help/Overview_of_Working_with_ROIs.html)
- D. Liu, Guoxu, Shuyi Mao, and Jae H. Kim. 2019. "A Mature-Tomato Detection Algorithm Using Machine Learning and Color Analysis" *Sensors* 19, no. 9: 2023. <https://doi.org/10.3390/s19092023>

Validation Figure 62

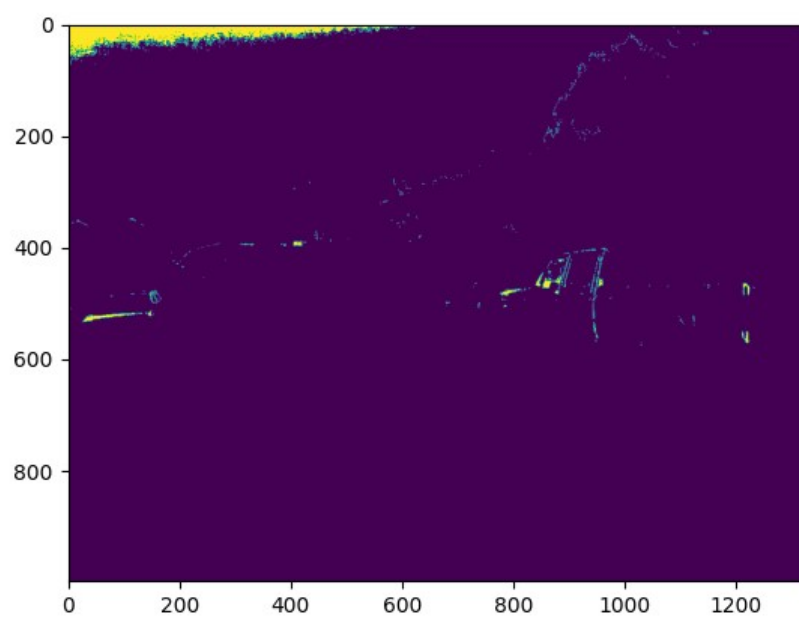


Bounding Box coordinates: [[119, 498, 120, 499]]

image



Validation Image 66

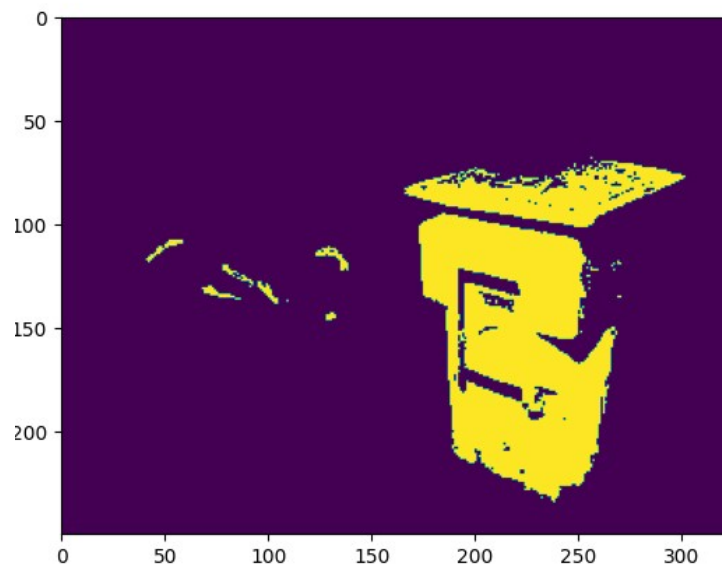



Bounding Box Coordinates: `[[1028, 574, 1030, 576]]`





Validation Image 63



 image

