ECE 276A Project 1: Color Segmentation

Brian Wilcox, University of California at San Diego, MS, Electrical and Computer Engineering
bpwilcox@ucsd.edu

I. Introduction

Object detection is a critical functionality in machine learning and robotic applications. Finding a target object amongst a collection of pixels (e.g. an image or video) is a non-trivial task. In the context of robotics, it is important to identify relevant objects for interaction with the environment. Color segmentation is a process in object detection which learns and classifies the color of each pixel. In this report, a multivariate Gaussian model is trained to classify the pixels of images containing a red barrel into several color classes. The red barrels are detected by grouping the regions of the barrel's red labeled pixels into bounded boxes and determining whether these boxes satisfy a "barrellness" threshold. The distance to the barrel from the camera was learned by training a linear regressor on the training data of known distances and the width and height of the detected barrels. This method was evaluated on a validation and test set.

II. Problem Formulation

In this project, we want to learn how to detect red barrels in an image. We also learn to approximate the distance of a barrel from the camera.

A. Training

Given training images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$ composed of pixels $x \in \mathbb{R}^3$, color classes $\mathcal{C} = \{1, ..., k\}$, and known distances to a red barrel in the image $\mathcal{D} = \{d_1, d_2, ..., d_n\}$, we want to train a classifier to determine the color class $y \in \mathcal{C}$ of a given pixel x and a regression algorithm to determine the distance to a detected red barrel.

B. Testing

Given a test image T, we want to classify every pixel x and create closed bounded boxes $\mathcal{R} = \{R_1, R_2, ...R_p\}$ of these classified pixels which belong to the color class $y = red\ barrel$.

Now, given closed bounded boxes $\mathcal{R} = \{R_1, R_2, ...R_p\}$ of color class "red barrel", we want to determine whether for any R_i, it is a barrel or not, i.e. $z \in \{barrel, not\ barrel\}$. For any detected barrels, we also want to use regression to determine the distance of the barrel $d \in \mathbb{R}$ from the camera.

III. Technical Approach

The training data consisted of fifty 900x1200 pixel image files (*.png). Each image contained at least one red barrel among different background and orientations. The training data included the distances to the barrels from the camera. Data was split into a training set and validation set (20% of set or 10 images). A testing set consisted of ten images of size 2448x3264 pixels (which was converted to size 900x1200) also containing at least one red barrel.

A. Labeling

In order to facilitate the training of a classifier for color segmentation, the training image set was labeled into color classes. The color classes chosen for the set were $\mathcal{C} = \{red\ (barrel), red\ (not\ barrel), yellow, brown\}$. Closed regions were manually drawn (via roipoly[1]) for these classes in each image. Darker colors in the images were included under labels for class "brown" while lighter colors were included under labels for the class "yellow". Careful attention was given to draw regions containing only the outline of the barrel as the "red (barrel)" color class. The label for "red (not barrel)" consisted of any red in the image that was not a part of the barrel. These regions created labeled masks for each color class.

B. Training Gaussian Model

Given the labeled dataset, a multivariate Gaussian classifier model $G(x, \mu, \Sigma)$ is trained such that

$$P_{X|Y}(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\}$$

where $x \in \mathbb{R}^3$ is a pixel with RGB color dimensions, μ_i is the mean of class i, and Σ_i is the variance of class i.

To train the the Gaussian classifier, the maximum likelihood estimate for μ_i^{mle} and Σ_i^{mle} is calculated using the pixel data for each class $D^i = \{x_1^i, ..., x_{N_i}^i\}$ such that

$$\mu_i^{mle} = \underset{\mu_i}{\operatorname{argmax}}\, P_{X|Y}(D^i; \mu_i, \Sigma_i)$$

$$= \frac{1}{N_i} \sum_j^{N_i} x_j^i$$

$$\Sigma_i^{mle} = \underset{\Sigma_i}{\operatorname{argmax}}\, P_{X|Y}(D^i; \mu_i, \Sigma_i)$$

$$= \frac{1}{N_i} \sum_j^{N_i} (x_j^i - \mu_i)(x_j^i - \mu_i)^T$$

Additionally, the prior probabilities of each color class are estimated using the relative proportion of class label counts via

$$P_Y(i) = \frac{N_i}{N_{total}}$$

[1] https://github.com/jdoepfert/roipoly.py

The computed parameters will be used for classification of a pixel into a color class $y \in \mathcal{C}$.

C. Classifying Pixels

Once we have the trained parameters of our Gaussian model, we classify pixels of the validation and test sets using the Bayes decision rule (BDR) for a Gaussian classifier. Here, we use the log-based BDR

$$i^*(x) = \underset{i}{\text{argmax}}[\log P_{X|Y}(x|i) + \log P_Y(i)]$$

$$= \underset{i}{\text{argmax}}[-\frac{1}{2}(x-\mu_i)^T(x-\mu_i) - \frac{1}{2}\log(2\pi)^d|\Sigma_i| + \log P_Y(i)]$$

Using the BDR above, every pixel x for any test image is classified into a color class to create a color segmented mask array denoting which pixel belongs to which class. This classification is applied to the validation set, test set, as well as the training set (to be used for training linear regression discussed later).

D. Detecting Red Barrels

The color segmented masks from the Gaussian classifier are filtered for the "red (barrel)" color class. This creates a mask for only the classifed red barrel color. Now, we aim to detect where the red barrel is in a test image.

The "opening" processing technique (erode followed by dilate in opencv) was used to process the red barrel mask in order to remove noise. Using the findcontours function in opencv, regions of red barrel color pixels are grouped into separate contours. Because the barrels are sometimes partially covered or split up by other objects in the scene, a method found online[2] to join contours which are close to each other by a distance threshold was utilized.

Bounded boxes were found around the contours of red barrel pixels by the minAreaRect() function in opencv. This method creates a rectangle of width w and height h around the contour with the minimum area.

The above algorithm so far was performed on the training set to find bounded boxes in the images. In order to create a metric for "barrellness", we aimed to find the average ratio (height to width) of the barrels in the training data. Using an arbitrary threshold, the widths and heights of detected barrels in the training set was found to calculate the ratios.

Using the average ratio of red barrels from the training set, a barrelness score was computed for each bounded box in $\mathcal{R} = \{R_1, R_2, ...R_p\}$ as $barrelness = |ratio_{R_1} - ratio_{avg}|$. The detection rule using this score was as follows:

$$DetectBarrel = \begin{cases} 0, & barrelness < 0.25 \\ 1, & else \end{cases}$$

The threshold of 0.25 seemed to work well for detection. Thus, the barrels (and their respective bounded boxes) in the test sets are detected using this approach.

E. Linear Regression for Distance

Simple linear regression was used to determine the distance to the red barrel in each image. We chose the feature for regression to be the inverse of the width and heights of each barrel in the training set. This was extracted from the bounding boxes found in the training data by performing the detection method as explained in the previous section. Here, we use the linear model

$$Y = Xw$$

where $Y \in \mathbb{R}^{n\times 1}$ is the vector of distances to the barrel, $X \in \mathbb{R}^{n\times 2}$ is the feature vector for widths and heights of the barrel, and $w \in \mathbb{R}^{2\times 1}$ is the weights vector. Given Y and X extracted from the training data, we find the maximum likelihood estimate (equivalent to least square approach) of the weights w by

$$w^{mle} = (X^TX)^{-1}X^TY$$

.

With the parameter for simple linear regression computed, we can estimate the distance to the barrel $\hat{y}$ for a test image given the inverse of the width and height of a detected barrel $x_{test} \in \mathbb{R}^{1\times 2}$ as

$$\hat{y} = x_{test}^T w^{mle}$$

This regression is performed for the validation and test set to find their respected distances for all detected barrels.

IV. Results

A. Training Parameters

The trained parameters from the Gaussian classifier were as follows:

$\mu_{red(barrel)}$	$[168.518, 43.525, 51.498]^T$
$\mu_{red(notbarrel)}$	$[151.501, 89.401, 70.207]^T$
μ_{yellow}	$[172.670, 163.079, 146.264]^T$
μ_{brown}	$[63.104, 66.207, 54.682]^T$
$\Sigma_{red(barrel)}$	$\begin{bmatrix} 3055 & 1097 & 1241 \\ 1097 & 776 & 797 \\ 1241 & 797 & 882 \end{bmatrix}$
$\Sigma_{red(notbarrel)}$	$\begin{bmatrix} 3479 & 2065 & 1448 \\ 2065 & 1748 & 1532 \\ 1448 & 1532 & 1596 \end{bmatrix}$
Σ_{yellow}	$\begin{bmatrix} 2165 & 2099 & 1920 \\ 2099 & 2220 & 2244 \\ 1920 & 2244 & 2601 \end{bmatrix}$
Σ_{brown}	$\begin{bmatrix} 1775 & 1421 & 1341 \\ 1421 & 1632 & 1454 \\ 1341 & 1454 & 1510 \end{bmatrix}$
$P_Y(red\ barrel)$	0.061
$P_Y(red\ not\ barrel)$	0.102
$P_Y(yellow)$	0.304
$P_Y(brown)$	0.533

Note: Variance values are rounded to whole number

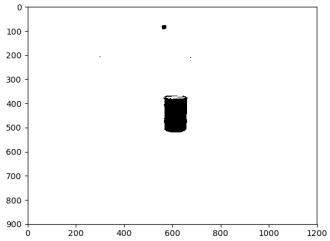
For linear regression, the trained weight w_{mle} was

$$w^{mle} = [346.982, 87.396]^T$$

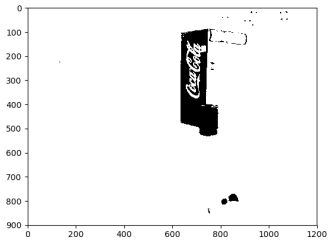
[2]https://dsp.stackexchange.com/questions/2564/opencv-c-connect-nearby-contours-based-on-distance-between-them

B. Color Segmentation

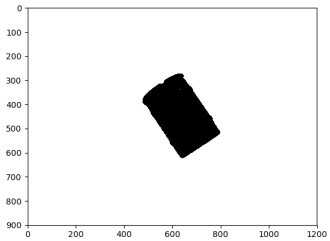
Below we present the color segmentation masks for select test images. The masks are segmented only for the "red barrel" color that is used for detection.



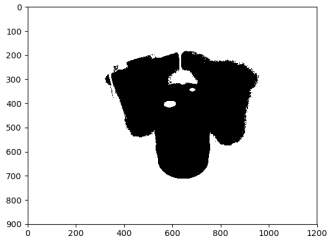
Mask for ImageNo [001]



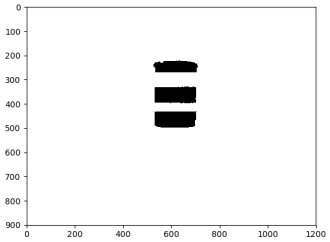
Mask for ImageNo [003]



Mask for ImageNo [004]



Mask for ImageNo [006]



Mask for ImageNo [008]

The segmentation masks in general corresponded well to the red barrel color except on a couple exceptions. In test images [003] and [006], which represent the presence of a red coca-cola machine and a reflection of the barrel, the Gaussian classifier is unable to distinguish between the red barrel color and the red non-barrel class. Apparently, the color of the coca-cola and the reflection of the barrel is very too similar to the red barrel color for the classifier. In cases like [008], the mask has segmented red barrel pixels that are split into multiple regions. This is an example where the method which combines close contours is useful to identify the regions as a single barrel.

C. Validation Results

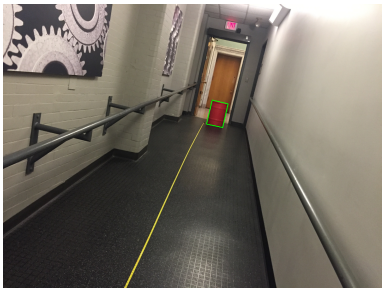
The trained models and detection algorithm were first implemented on the validation set of ten images. Below are the results, showing the bounded box of the detected barrel in the image (green box in figure). The respective bounding box coordinates and computed distance to the barrel are also displayed in the image caption.

ImageNo = [7.2.png] , Vertex_1 = [622 444] , Vertex_2 = [622 355],Vertex_3 = [682 355] , Vertex_4 = [682 444] , Distance = [6.7650115]

ImageNo = [7.3.png] , Vertex_1 = [1014 343] , Vertex_2 = [945 337] , Vertex_3 = [954 235] , Vertex_4 = [1023 241] , Distance = [5.85235281]

ImageNo = [8.3.png] , Vertex_1 = [605 540] , Vertex_2 = [605 463] , Vertex_3 = [655 463] , Vertex_4 = [655 540] , Distance = [8.0746527]

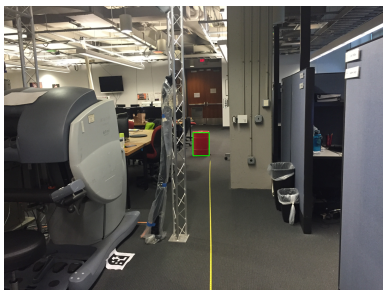
ImageNo = [7.4.png] , Vertex_1 = [583 542] , Vertex_2 = [581 452] , Vertex_3 = [639 451] , Vertex_4 = [640 541] , Distance = [7.01776478]



ImageNo = [8.4.png] , Vertex_1 = [695 391] , Vertex_2 = [644 380] , Vertex_3 = [662 304] , Vertex_4 = [712 316] , Distance = [7.86830543]

ImageNo = [7.6.png] , Vertex_1 = [518 525] , Vertex_2 = [459 525] , Vertex_3 = [459 438] , Vertex_4 = [518 438] , Distance = [6.8856023]

ImageNo = [8.5.png] , Vertex_1 = [611 525] , Vertex_2 = [611 449] , Vertex_3 = [661 449] , Vertex_4 = [661 525] , Distance = [8.08958707]



ImageNo = [8.2.png] , Vertex_1 = [598 474] , Vertex_2 = [598 401] , Vertex_3 = [649 401] , Vertex_4 = [649 474] , Distance = [8.00077503]

ImageNo = [9.2.png] , Vertex_1 = [616 700] , Vertex_2 = [616 629] , Vertex_3 = [656 629] , Vertex_4 = [656 700] , Distance = [9.90547937]

ImageNo = [9.3.png] , Vertex_1 = [570 376] , Vertex_2 = [570 309] , Vertex_3 = [614 309] , Vertex_4 = [614 376] , Distance = [9.19037379]

ImageNo = [003] , No Barrel Detected

This approach performed very well for the validation set. All the red barrels were detected successfully and the distances to each barrel are close to known actual distances. Therefore. the Gaussian classifier and regression appeared to work decently.

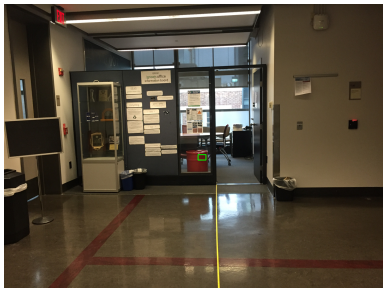
D. Test Results

Here we present the results for the ten test images. As with the validation set, the bounded box of the detected barrel is shown in green while the bounding box coordinates and computed distance to the barrel are displayed in the image caption.

ImageNo = [004] , Vertex_1 = [636 631] , Vertex_2 = [461 370] , Vertex_3 = [632 256] , Vertex_4 = [807 517] , Distance = [1.97106513]

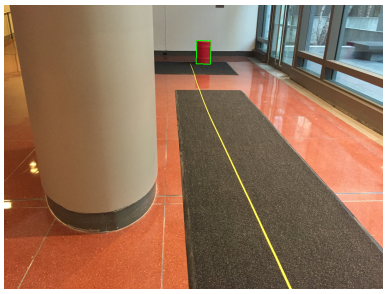
ImageNo = [001] , Vertex_1 = [566 521] , Vertex_2 = [566 379] , Vertex_3 = [659 379] , Vertex_4 = [659 521] , Distance = [4.34645388]

ImageNo = [005] , Vertex_1 = [679 514] , Vertex_2 = [613 514] , Vertex_3 = [613 420] , Vertex_4 = [679 420] , Distance = [6.18704744]

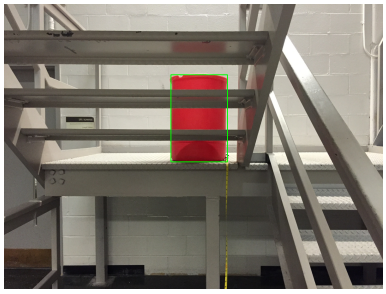


ImageNo = [002] , Vertex_1 = [639 494] , Vertex_2 = [614 494] , Vertex_3 = [614 477] , Vertex_4 = [639 477] , Distance = [23.90654492]

ImageNo = [006] , No Barrel Detected

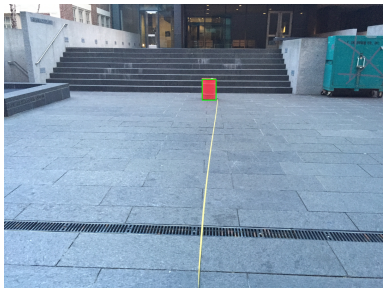


ImageNo = [007] , Vertex_1 = [653 187] , Vertex_2 = [608 186] , Vertex_3 = [610 112] , Vertex_4 = [655 113] , Distance = [8.79305824]



ImageNo = [008] , Vertex_1 = [528 498] , Vertex_2 = [528 223] , Vertex_3 = [707 223] , Vertex_4 = [707 498] , Distance = [2.25625048]

ImageNo = [009] , Vertex_1 = [633 667] , Vertex_2 = [633 572] , Vertex_3 = [688 572] , Vertex_4 = [688 667] , Distance = [7.22872136]



ImageNo = [010] , Vertex_1 = [627 304] , Vertex_2 = [627 238] , Vertex_3 = [671 238] , Vertex_4 = [671 304] , Distance = [9.21013601]

In 8 of the 10 images, a bounding box indicated a detected barrel is found. For image [002], it appears the bounding box doesn't cover the entire barrel and thus the distance is also far off. This error is likely due to the color segmentation mask for this image, meaning the classifier didn't detect those surrounding pixels as part of the "red barrel" color class. The approach failed to identify the barrels in images [003] and [006]. Referring to their color segmentation mask for the red barrel color, the bounded box found for the region did not satisfy the barrelness score (i..e the ratio of the box was very different than the average ratio of the barrel). Perhaps another method of determining barrelness would be useful using some prior information. Images [008] and [009] are a good demonstration that though the barrel is partially blocked by other obstacles in the scene, the algorithm is able to join the close red barrel regions to identify the whole area as one singular barrel. The distances computed via linear regression also seem to be reasonable compared to those in the training set. Overall, the detection algorithm appears to perform well except for in images with objects very close to the red barrel with about the same color.