Color Segmentation

Hsiao-Chen Huang
Department of Electrical and Computer Engineering
University of California San Diego
hsh030@eng.ucsd.edu

***Abstract*—This project presented approaches using Gaussian mixture for the pixel-wised classification model. Our goal is to detect the blue barrels in an image by predicting each pixel which is belong to blue barrel or not. In this work, we basically followed the Bayesian decision rule(BDR), and compute the maximum likelihood estimates for the parameters of the class conditional densities under the Gaussian assumption. The results are shown in the final section.**

I. Introduction

Color segmentation is a part of object detection or recognition problem. It is widely used in robotic system, autonomous driving or others. Knowing the environment around is important for a machine and help for extended complicated functions such as object tracking and semantic analysis.

In this project, we try to solve a simple classification problem with color recognition. We train a probabilistic color model from 46 images training data and use it to detect a blue barrel, and draw a bounding box around it. We divide this task in to three steps. First, we hand-label our training images and build a color classifier for barrel-blue and non-barrel-blue. Based on computing the class conditional densities of color space(YCrCb) value under mixture Gaussian assumption, we can predict the class by Bayesian decision rule. Second, we need to filter the region in color mask to delete the false noise region. Third, we identify the coordinates of a bounding box of a detected barrel in the image frame on the test image.

II. Problem Formulation

With Gaussian classification problem, we define the image data as $\mathbf{X} = \begin{bmatrix} \mathbf{x}_Y \\ \mathbf{x}_U \\ \mathbf{x}_V \end{bmatrix}$, where $\mathbf{x}$ is a pixel vector with image size in a color space. Here we use 46 training images with dimension (800, 1200, 3). 3 is the dimension of color space. The labeled image is defined as $\mathbf{Y}$. The class of color is defined as $i \in \{BarrelBlue, NonBarrelBlue\}$. Now, we want to get the optimal class of each pixel as

$$i^* = \text{argmax}_i P_{Y|X}(i|x) \tag{1}$$

We know the Multivariate Gaussian is

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

$\mu_\mathbf{i} = \begin{bmatrix} \mu_{Yi} \\ \mu_{Ui} \\ \mu_{Vi} \end{bmatrix}$ is the mean of pixel vector $\mathbf{x}$ in class i, and $\mathbf{\Sigma}_i$ is the covariance of matrix $\mathbf{X}$ with all pixels in class i. First, we need to compute the log-based Bayesian decision rule

$$\begin{aligned} i^* =& \text{argmax}_i[P_{X|Y}(x|i)P_Y(i)] \\ =& \text{argmax}_i[\log P_{X|Y}(x|i) + \log P_Y(i)] \\ =& \text{argmin}_i[d_i(x,\mu_i) + \alpha_i] \\ =& \text{argmin}_i[(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) + \log|\Sigma_i| \\ & - 2\log P_Y(i)] \\ =& \text{argmin}_i[x^T\Sigma_i^{-1}x - 2\mu_i^T\Sigma_i^{-1}x + \mu_i^T\Sigma_i^{-1}\mu_i \\ & + \log|\Sigma_i| - 2\log P_Y(i)] \\ =& \text{argmin}_i[x^T w_1 x - 2w_2 x + w_3] \end{aligned}$$

$$\text{, where } \begin{cases} w_1 = \Sigma_i^{-1} \\ w_2 = \mu_i^T\Sigma_i^{-1} \\ w_3 = \mu_i^T\Sigma_i^{-1}\mu_i + \log|\Sigma_i| - 2\log P_Y(i) \end{cases}$$

We choose the pixel as barrel-blue when

$$\begin{aligned} & x^T w_{1(bb)} x - 2w_{2(bb)}x + w_{3(bb)} \\ & \qquad > x^T w_{1(nb)} x - 2w_{2(nb)}x + w_{3(nb)} \end{aligned} \tag{3}$$

bb stands for the class of barrel-blue, and nb stands for the class of non-barrel-blue.

III. Technical Approaches

A. Color Segmentation

For color segmentation, we first hand-label our 46 training images, and get corresponding 46 binary mask images with 0 for non-barrel-blue pixel and 1 for barrel-blue pixel. We write our code with **Roipoly** library. While running, the image will show in a window. The user can start by left-click on the edge of the region, and right-click when closing the region. It will automatically save a labeled binary image with hand-selected region.

In the training step of build our classification model, we load the training images and convert RGB to YCrCb

color space. In order to calculate the parameters such as mean and variance in each class of our model, we need to separate all the pixels into two classes $\{\mathbf{X}_{(bb)}, \mathbf{X}_{(nb)}\}$. We can get the prior probability of each class by computing

$$P_Y(i \in BarrelBlue) = \frac{\text{\# of barrel-blue pixels}}{\text{\# of total pixels}}$$
$$P_Y(i \in NonBarrelBlue) = \frac{\text{\# of non-barrel-blue pixels}}{\text{\# of total pixels}}$$

We reshape the data $\mathbf{X}_i$ in each class into three pixel vector $\begin{bmatrix} \mathbf{x}_{Yi} \\ \mathbf{x}_{Ui} \\ \mathbf{x}_{Vi} \end{bmatrix}$, and the mean μ_i and variance Σ_i of $\mathbf{X}_i$ in each class can be easily obtained. Finally, we save all the parameters in to model.mat file for the next prediction step.

In the prediction step, given a test image $\mathbf{X}$, we convert it into YCrCb color space and reshape into size $(3, length \times width)$. After loading the model parameters, we can predict the result by computing the BDR in Eq.3 and obtain a binary mask image.

B. Barrel Detection

For barrel detection, we first remove noise regions in the mask image. All the regions smaller than 0.1% of the image size are considered as noise regions and be deleted. Then, find all the external contours in mask images, and iterative compare the similarity of their shape with a sample barrel shape. If the similarity <20%, we will save the bounding box $(x1, y1, x2, y2)$ of the region into our result, where $(x1, y1)$ is the left-above coordinate and $(x2, y2)$ is the right-below coordinate of the rectangle box. This region can be consider as detected barrel.

IV. Results

The training result of our color classification model is shown as following. The prior probability of each class is

$$P_Y(i \in BarrelBlue) = 0.0124$$
$$P_Y(i \in NonBarrelBlue) = 0.9876$$

The mean vector of each class is

$$\mu_{(bb)} = \begin{bmatrix} \mu_{Y(bb)} \\ \mu_{U(bb)} \\ \mu_{V(bb)} \end{bmatrix} = \begin{bmatrix} 67.04 \\ 101.01 \\ 163.64 \end{bmatrix}$$
$$\mu_{(nb)} = \begin{bmatrix} \mu_{Y(nb)} \\ \mu_{U(nb)} \\ \mu_{V(nb)} \end{bmatrix} = \begin{bmatrix} 100.82 \\ 132.62 \\ 121.04 \end{bmatrix}$$

The covariance matrix of each class is

$$\mathbf{\Sigma}_{(bb)} = \begin{bmatrix} 1057.19 & -345.81 & 237.52 \\ -345.81 & 343.36 & -295.23 \\ 237.52 & -295.23 & 358.25 \end{bmatrix}$$
$$\mathbf{\Sigma}_{(nb)} = \begin{bmatrix} 3571.90 & 17.01 & 5.47 \\ 17.01 & 72.54 & -50.75 \\ 5.47 & -50.75 & 74.05 \end{bmatrix}$$

We predict our segmentation results through the above model, and obtain the segmented mask images in the left-hand side. Then, detect the possible barrelness region, and draw the bounding boxes on the original test image in the right-hand side. The results in different conditions are shown as following.

A. Correct Results

In figure 1-3, no matter the image is in the bright or dark condition, or contain other blue items in the background, we all get the accurate bounding boxes. We can see that there are some noise regions and non-barrel-blue regions in the mask image, but after detecting barrel, they will be removed in the final prediction.

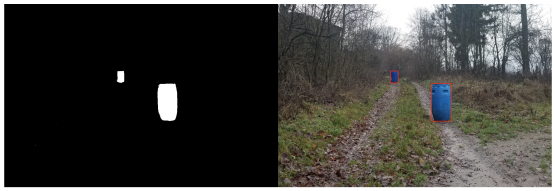

Fig. 1. The correct result of detecting blue barrels.

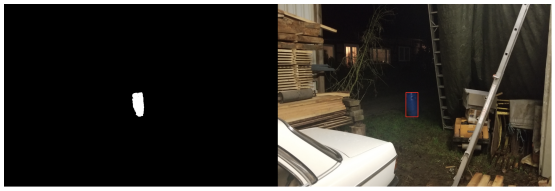

Fig. 2. The correct result of detecting a blue barrel in dark condition.

Fig. 3. The correct result of detecting a blue barrel with other blue items in the background.

B. False Results

In figure 4-6, we did not get will performance in these conditions. For figure 4, the other blue item is too close to the blue barrel that two regions are connected.

Therefore, it is too difficult to distinguish. For figure 5, the shape of other blue item is similar with the barrel, so it is also selected by our barrel detector. For figure 6, the blue barrel hide behind the branches. From the mask image, we can see that the blue region is cut by the branches into small and non-barrelness regions. Hence, those can not be detected as blue barrel in our final result.

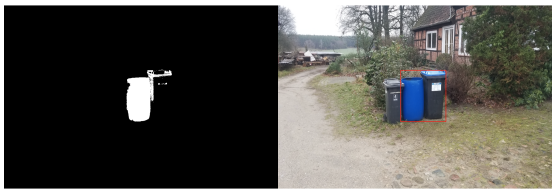

Fig. 4. The false result of detecting a blue barrel besides another blue item.

Fig. 5. The false result of detecting a blue barrel with another barrelness blue item.

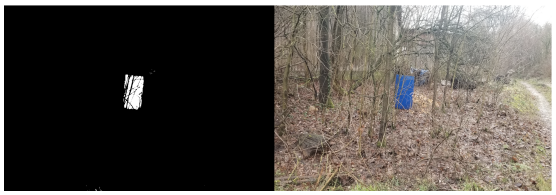

Fig. 6. The false result of detecting a blue barrel behind other items.

C. Discussion

To improve the performance, we may implement opening and closing operator on the mask image to obtain a proper input for our barrel detector. Opening operator helps remove small protrusions or thin connections and closing operator helps remove holes. This will be applied in the future work.