Color Segmentation

Abhijeet Singh

***Abstract*— This report describes the application of color based image segmentation to segment red barrels from real world images. We use Gaussian Mixture Models to train on 50 images and build a classifier that segments the red barrel pixels from the image. We use the segmented image to get a bounding box on the image and calculate its centroid as well as the distance of the barrel from the camera. We achieve a recall value of 0.96 during cross validation of our model. We also mention the results on test data in the last section.**

I. INTRODUCTION

Image segmentation is the process of dividing pixels in an image into a set of classes. These classes have some unique characteristic generally distinguishable by pixel data. This project is about image segmentation based on color information. The aim is to correctly segment a given object and label its bounding box and the object centroid. In this project we are given 50 images containing a red barrel at various locations in the real world and the objective is to build a classifier that can segment the red barrel on new test data. As a final task, we have to estimate the distance of the barrel from the camera.

There are several ways to perform segmentation in an image like thresholding method, clustering based approaches, edge detection, histogram based methods etc. Here, we discuss two learning approaches to perform the segmentation based on color information from each pixel. These are namely Gaussian distribution based and Gaussian Mixture Model (GMM) based, which are discussed in the next section.

II. DESCRIPTION

Various machine learning methods typically model the objective with some loss function which is then optimized to learn the parameters of the function to fit the data. They are usually "supervised" in the sense that first the parameters are learned from some "labeled" training data which is finally tested on a new set of data to evaluate the performance.

A. Labeling the data

The data was labeled using the "roipoly" tool. This tool basically allows us to select regions in the image which we can use to specify which region is "red", "white" etc. This way, the labeled pixels can be assigned a color which can be used to train the model for color based classification. For each set of labels, we learn the parameters of a distribution that encodes that color class. For example, we learn the distribution of the "red" class using the labels obtained above and the models described next.

*This work is part of the Project 1 of the ESE 650 class Learning in Robotics at University of Pennsylvania, taught by Prof. Daniel Lee

B. Modeling the data

Here we use learning methods to model the distribution of object colors in an image. The aim is to correctly segment the red barrel in the image. If we are able to learn the distribution of the "red" color of the barrel, and are able to distinguish it from other colors in the image, we will be able to classify each pixel in the image into red barrel and non-red barrel categories. We can thereafter collect all the pixels classified as "red barrel" and connect them to get an object which will represent our segmented red barrel from the image.

We use two approaches to model the color distributions in the image.

1) Gaussian Model: In this method we model each color class as the probability distribution function of a Gaussian Distribution.

$$p(x|color) = \frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\mu)^T\mathbf{\Sigma}^{-1}(x-\mu)\right)$$

In the above equation, D represents the dimension of the space, which is 3 for typical color spaces such as RGB, YUV etc. Assuming that each pixel realization is independent of the other pixels given the color class, we can easily estimate the parameters μ and Σ above using Maximum Likelihood Estimation (MLE). This gives,

$$\mu^* = \frac{\sum_i^N(\boldsymbol{x}_i)}{N}$$
$$\Sigma^* = \frac{\sum_i^N(\boldsymbol{x}_i-\mu)(\boldsymbol{x}_i-\mu)^T}{N}$$

These values can be obtained directly from the labeled training data and plugged into the Gaussian distribution equation to obtain a model for each of the color class.

2) Gaussian Mixture Model (GMM): GMMs assume that the data comes from a set of Gaussians, combined in certain proportions.

$$p(x|color) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\mu_k,\Sigma_k)$$

For each color, there are three types of parameters - the mixture probabilities, mean values and covariance matrices. These parameters are estimated for each mixture. Here also we do a Maximum Likelihood Estimation to compute the parameters, but there is no closed form solution for GMM parameters. Hence we use the Expectation Maximization algorithm to evaluate the parameters of the distribution for each class.

The Expectation step evaluates the membership of each pixel into each of the mixtures.

$$\gamma(z_{nk}) = \frac{\mathcal{N}(x_n|\mu_k, \Sigma_k)}{\Sigma_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}$$

The Maximization step re-estimates the parameters using the above membership values.

$$\mu_k^{new} = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma(z_{nk}) x_n$$
$$\Sigma_k^{new} = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma_k(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$$
$$\pi_k^{new} = \frac{N_k}{N}$$

Where $N_k = \Sigma_{n=1}^{N} \gamma(z_{nk})$. These steps are repeated until the log-likelihood of the data converges.

$$\ln p(X|\pi, \mu, \Sigma) = \Sigma_{n=1}^{N} ln\{\Sigma_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)\}$$

The above log-likelihood is guaranteed to converge at a local maxima. This way, we get the parameters of our mixture model.

C. Predicting for new data

Once we have one of the above model that captures the distribution of a particular color, we can use Bayes rule to calculate the probability of each pixel belonging to each of the color class.

$$P(color|\boldsymbol{x}_i) = \frac{P(\boldsymbol{x}_i|color)P(color)}{P(\boldsymbol{x}_i)}$$

Here $\boldsymbol{x}_i$ represents each pixel in the new (test) data. For each class, we get the probability of the pixel belonging to that particular class, and we can simply choose the color with the maximum probability to make a hard classification for each pixel.

D. Bounding Box and Centroid

Bounding box was calculated using the skimage regionprops() method. This method finds the connected components of the image and creates a rectangular bounding box for each such component in the image. This function also outputs the centroid of the bounding box, which is almost the same as the centroid of the barrel because the barrel is of a cylindrical shape. We use this method after we estimate, for each pixel, whether it belongs to the red barrel or not. This estimation step (which is basically the Bayes rule from above) gives a mask of the image with the red barrel pixels highlighted. This mask is used as input to the regionprops() method which outputs the connected components in the image. Note that since the classifier is not perfect, it might happen that there are a number of false connected components in the result of regionprops. These false components have to be discarded based on the expected geometeric properties of the barrel. These are described in more detail in the Discussion section.

E. Barrel distance calculation

The distance of the barrel from the camera can be calculated using regression given the distances in the training images. Suppose the distance of the barrel from a camera of focal length f be d and the pixel height of the barrel in the image be h_p. Using the pin hole camera model, the similar triangles give,

$$\frac{h_p}{f} = \frac{\text{Actual barrel height}}{d}$$

From above, d can be calculated as a function of h_p since the other variables are constant. The exact relation can be found using regression.

III. Discussion

This section goes more into the details used in implementing the above work flow.

A. Labelling the data

We labeled the data into 6 classes - the red color of the barrel, the red color elsewhere, white color (including bright shiny surfaces), black color (including dark surfaces and object), green, blue. We basically started with only the red barrel color in the beginning when we were using a single Gaussian for predictions, but later extended into other colors as we moved to using multiple Gaussians for different colors, and then later for GMMs. The decision to label colors was basically an iterative process. When using multiple Gaussians or GMMs we take the class which gives the maximum probability for each pixel. When we used a smaller set of colors, there were areas in the image that were being classified incorrectly as "red" (hard classification has to classify into one of the available classes). This way we deduced which color to label next.

Further into the process, there were instances when we felt that the certain pixels have to be relabeled to account for some of the incorrect classifications. This way, the quality of the labeled data improved iteratively.

TABLE I
LABELED DATA: NUMBER OF PIXELS FOR EACH COLOR

Red barrel	893593
White & shiny surfaces	2511771
Red non-barrel	1952134
Black/Dark surfaces	2927642
Green	2558816
Blue	235451

B. Predicting the classes

We started by using a single Gaussian distribution to model the red barrel color class and used a threshold on the resulting probability to classify a pixel into "red barrel" class or "non red barrel" class. The threshold was adjusted empirically after testing on unseen data. The results are summarized in Table II. These values were observed in the YCbCr color space.

TABLE II

SCORES FOR SINGLE GAUSSIAN

Recall	0.652
Precision	0.760
F-measure	0.702

We use precision, recall and f-measure (harmonic mean of precision and recall) values to estimate the score of a particular algorithm with certain parameters. The precision is calculated as the ratio of number of pixels classified correctly to the total number of pixels classified as "red barrel" pixels. The recall is calculated as the ratio of the number of pixels classified correctly to the actual number of pixels that were "red barrel" pixels. In general, there is a trade-off between the precision and recall curves - here we would ideally want to have a high recall value (while keeping a decent precision) so that most of the barrel pixels are classified correctly. The rest of the noise can be pruned while drawing the bounding boxes, where the geometry of the connected components can be used to infer whether a set of connected pixels form a barrel or not.

The score for precision, recall, f-measure have been calculated by cross validation on the training data, and taking the average values after multiple runs. We have used 10-fold cross validation by training on 90% of the data on each run and testing on the remaining 10% data.

Next we tried using multiple Gaussians to model the different color classes present in the images. This involved estimation of the Gaussian parameters - mean and covariance for each color class. For each pixel, the probability of it coming from each of the color class is calculated. This step uses the Bayes rule described in the previous section. Here, we assumed a uniform prior for each of the color class. Further, we thresholded the pixels calculated as belonging to "red barrel" class on the probability of them coming from the class. It basically means that after calculating the probabilities for each class and taking the argmax over the colors, we classify each pixel into one of the color classes. Thereafter, for the pixels classified in the "red barrel" class, we require a minimum confidence (the threshold value) for it to be finally considered as part of the "red barrel" color class. The results are summarized in the table III.

TABLE III

SCORES FOR MULTIPLE GAUSSIANS IN DIFFERENT COLOR SPACES

	RGB	YCbCr
Recall	0.927	0.959
Precision	0.812	0.816
F-measure	0.866	0.882

We performed the tests in two color classes - YCbCr and RGB. We see that the barrel is better distinguishable in the YCbCr color space. Henceforth, we used the YCbCr color space to train the rest of the algorithms.

Gaussian Mixture Model use a number of Gaussian distributions combined with certain probability to model the distribution of each class. We cross validated on the number of mixtures and the results are summarized in the table IV, V. This cross validation was performed for the case in which we used a full covariance matrix as well for the case of diagonal covariance matrix.

TABLE IV

SCORES FOR GMM WITH DIAGONAL COVARIANCE MATRIX

	k=2	k=3	k=4	k=5
Recall	0.938	0.946	0.927	0.919
Precision	0.536	0.481	0.446	0.467
F-measure	0.682	0.638	0.602	0.619

TABLE V

SCORES FOR GMM WITH FULL COVARIANCE MATRIX

	k=2	k=3	k=4	k=5
Recall	0.962	0.964	0.963	0.952
Precision	0.813	0.704	0.598	0.598
F-measure	0.881	0.814	0.738	0.735

Here "k" represents the number of mixtures used for each class. It is taken to be the same for each of the class. As we can see from the results, the GMM with full covariance outperforms the GMM with diagonal covariance matrix. The recall values for diagonal covariance are good as the precision can be improved by further using a threshold on the "red barrel" pixels. But the full covariance GMM does slightly better on the recall front as well, and hence we use the full covariance matrix.

Table V gives the scores for different number of mixtures as well. We see that the recall value is best for $k = 3$, but the precision value falls off rapidly as k increases. This seems to be because of the fact that there are a lot more parameters to be estimated as k increases. We see $k = 2$ has the best precision and a comparable recall value.

Our final model uses the Gaussian Mixture Model with 2 Gaussian mixtures for each color class. We use the YCbCr color space to perform our computations. Since we had a good amount of data (see Table I) we could train the GMM model with full covariance matrix and get good cross-validation results.

The results on the test set have been summarized in the next section.

C. Barrel bounding box

The barrel bounding boxes were calculated using the regionprops method, with 2-dimensional connectivity as described previously. Some heuristics, based on geometry, had to be used to find the bounding boxes that correspond to the red barrel. For this, the bounding boxes were sorted according to the filled area inside them and only the first few boxes were chosen so that they contribute to 70% of the total filled area of all the boxes. This was done to remove the small noise points that were classified incorrectly as the red barrel pixels. The value of 70% was chosen empirically after observing the top bounding boxes. Since the correct red barrel bounding box will have most of the pixels classified as belonging to the class of "red barrel", it will ideally be the

only object that is kept in this step (if there are two barrels, roughly of same size, they contribute roughly 50% each and hence both will be kept in this step).

But it happens that there are objects that have a color very close to the "red barrel" that will not be filtered in the above step. There can be other sources of noise as well. So next, we sort the remaining points based on the extent (ratio of filled area to the area of the bounding box) of each connected component and use that to filter the rest of the noise. The extent is helpful because of the shape of the barrel which is cylindrical. This means that the red barrel should have a very high value of extent. Even when it is tilted, the extent will be quite high, but the threshold in the step has to be carefully set so as not to classify other similar colored objects as the barrel. We came up with empirical thresholds on the area and the extent of the barrel, and also for the other barrels, if there are more than one. We used both these values to classify a bounding box as containing a barrel or not containing a barrel.

There were scenarios when there was some object in front of the table because of which the entire barrel was not captured in one bounding box. For example, if the barrel is kept below a table, one leg of the table might come in the image in front of the barrel because of which the continuity of the pixels labeled as red barrel breaks and hence the pieces get assigned to different bounding boxes.

We used the value of extent once again to merge these two bounding boxes. The idea is that the two pieces of the barrel will, in most cases, have a similar extent value - which will be close to the extent value when they are merged. This is simply due to the fact that the barrel is cylindrical. If two bounding boxes are able to satisfy the above condition, with some empirical threshold on the "closeness" desired, we can merge two bounding boxes into one without merging two separate barrels or objects into one barrel.

D. Barrel distance calculation

Linear regression was used to calculate the distance of the barrel from the camera. We used the equation described in the previous section. It basically describes a linear relation of distance with the inverse of pixel height of the barrel in the image (Fig 1). Since linear regression has a closed form analytical solution, we used it to calculate the parameters of the equation - the "slope" of the line and the bias term ($y = mx + c$). In place of the pixel height, we used the square root of the area of the barrel in the image. This is because the barrel has a fixed sized base and hence the pixel height will be a multiplicative factor away from the square root of the area, which will be found by regression. Area of the barrel was easy to estimate from the labeled data as the number of pixels marked as "red barrel". This approach of using the area is also resilient to the tilt of the barrel, hence provides a good estimate of the barrel distance. While predicting the distance for new barrels, we use the filled area inside the bounding box as an approximation to the barrel area.

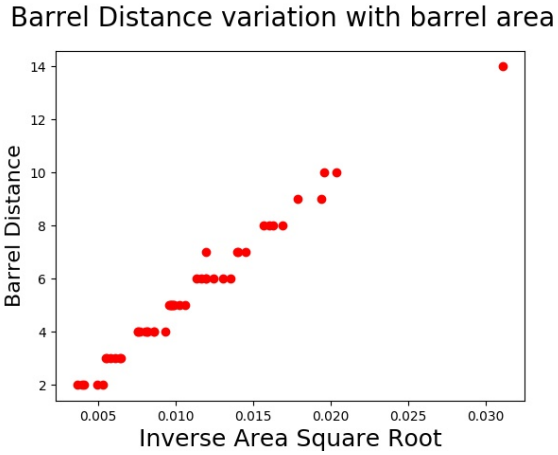

Fig. 1. Plot of barrel distance variation with area

IV. Results

This section contains the results on the test set. The model was trained from the set of 50 training examples, as described in the previous section. Figure 2 - 11 show the detected bounding boxes. The centroid for each barrel is also labeled in the figure.

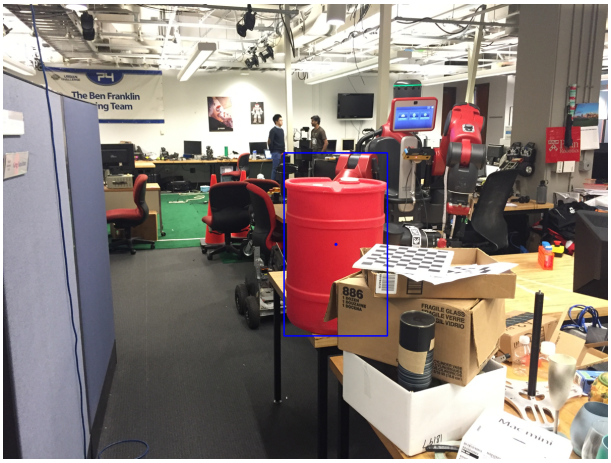

Fig. 2. Image 001.png

Fig. 3. Image 002.png

Fig. 6. Image 005.png

Fig. 4. Image 003.png

Fig. 7. Image 006.png

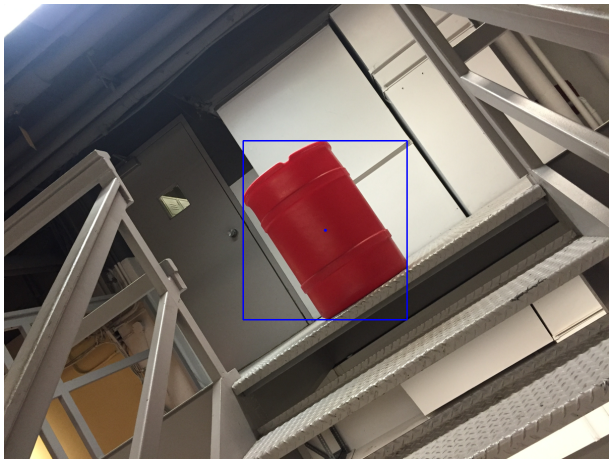

Fig. 5. Image 004.png

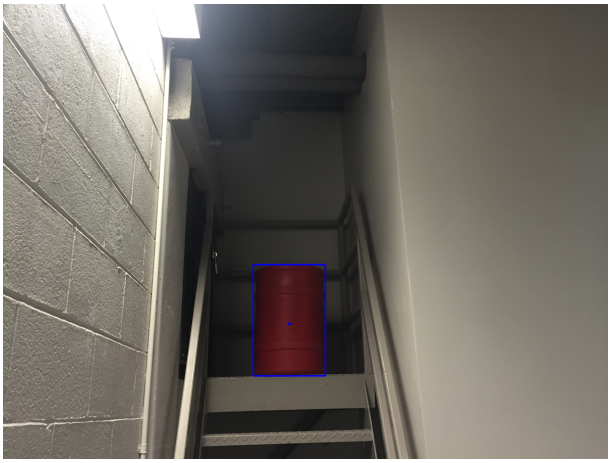

Fig. 8. Image 007.png

Fig. 9. Image 008.png

Fig. 10. Image 009.png

Fig. 11. Image 010.png

The distance and centroid information for each of the barrel is listed below -

Distance for 001.png: 2.4618
Centroids for 001.png: (659.5, 475.0)

Distance for 002.png: 2.2285
Centroids for 002.png: (618.5, 427.0)

Distance for 003.png: 2.8050
Centroids for 003.png: (696.5, 459.44)

Distance for 004.png: 2.1439
Centroids for 004.png: (636.5, 447.0)

Distance for 005.png: 6.2238
Centroids for 005.png: (645.0, 462.0)

Distance for 006.png: 9.9279
Centroids for 006.png: (632.5, 427.5)

Distance for 007.png: 3.0327
Centroids for 007.png: (567.5, 631.6)

Distance for 008.png: 11.6247
Centroids for 008.png: (741.5, 474.0)

Distance for 009.png: 10.5981
Centroids for 009.png: (673.5, 429.0)

Distance for 010.png: 5.6055
Centroids for 010.png: (664.5, 413.0)

The image segmentation is good in most of the examples, with a few notable exceptions. Figure 4 has two boxes, though only one of them has been selected. This is probably due to the tight threshold parameters we had placed on the potential barrels - although both of them were segmented correctly (as can be seen in Figure 12). We could detect multiple barrels while doing cross validation, but somehow fail here due to the fact that the second barrel is not completely visible and hence the tight bounds for drawing the bounding boxes failed here.

Fig. 12. Segmented Image 003.png

Barrels in Figure 2 and 11 have some red objects around them, which have been erroneously classified as belonging to the barrel. This gives them bigger bounding boxes.

For the other figures, the algorithm gives good results. One notable example is that of Figure 3 which has two segmented portions (Figure 13) and we are successfully able to merge them.

Fig. 13. Segmented Image 002.png

The distance estimate seem to give a wrong estimate for the Figure 9. This is because we used the number of red pixels in the barrel to calculate the area, which fails here because the box is slightly hidden. We should have rather used the bounding box height and width to get a more reasonable estimate.

In hindsight, we could have used some image processing tools to clean up the noise in the image. At present, we do not use any image processing tool and rely on geometry to detect barrels from noise and performing the bounding box merging step etc.

Overall, it was a fun project and we really enjoyed working on it!

REFERENCES

[1] Nasrabadi, Nasser M. "Pattern recognition and machine learning." Journal of electronic imaging 16.4 (2007): 049901.

[2] Roipoly: https://github.com/jdoepfert/roipoly.py

[3] Regionprops: http://scikit-image.org/docs/dev/api/skimage.measure.html