Color Segmentation of objects using Gaussian Models

Sai Krishnan Chandrasekar
Graduate Student, Robotics
University of Pennsylvania
Email: saikrish@seas.upenn.edu

***Abstract**: Segmentation of objects is an important part of computer vision and robotics for numerous purposes like object detection, recognition, mapping etc. Color based segmentation is one commonly used segmentation technique where the system is trained to detect and segment specific objects based largely on their color. In this project, a single Gaussian model was trained for the color of a barrel: Red. This Gaussian model was used to detect and segment barrel(s) in images.*

1 Introduction

The main goal of this project was to detect red colored barrel or barrels in a given set of images. This was achieved by training a single Gaussian based machine learning model that segmented barrels from images and also identified their height, width and distance from the camera. This type of color based segmentation that also estimates parameters of the object is particularly useful for visual odometry based robots, path planning, localization and mapping. This was coded up on Python and OpenCV libary was used for image processing.

2 Problem Formulation

The problem at hand was divided into 5 parts:

1. Hand Label training images to train the system for "Barrel-Red", i.e., red color intensities that could be barrels.
2. Estimate their parameters, i.e., estimate the mean and covariance of this Gaussian distribution of red color intensities.
3. Identify all regions in the image that had the same red intensity as the barrel.
4. Identify barrel or barrels among those regions.
5. Estimate their height(s), width(s) and distance(s) from the camera.

3 Hand Labeling the data

The given 50 images were divided into training set and testing set. Training set consisted of 35 images and the testing set consisted of 15 images. "Roipoly" function was used along with Pylab and OpenCV to create a user interface for hand-labeling. In this interface, the user could select regions of interest (referred to as ROI from now on) that were "Barrel-Red".

These pixel values got appended to a array that was used for further processing. At the end of this stage, a complete array was formed. This array had over 30,000 pixels that were designated "Barrel-Red" by the user.

4 Estimation of Parameters.

The next step in the pipeline was to estimate the parameters of the Gaussian distribution that was fit onto the pixels. Since this is a Unimodal Gaussian, only two parameters were needed to express the Gaussian: Mean and Covariance.

The mean and covariance of the "Barrel-Red" pixels were calculated using in-built functions of the Numpy library.

5 Identification of Barrel-Red regions

This step identified all regions in the image, irrespective of size or shape that "could" be a part of the barrel. A probability density function was computed for each image where every pixel got assigned a probability of being "Barrel-Red". All pixels that whose probability was at least $\frac{1}{17.5}^{th}$ of the maximum probability of that image was deemed as "Barrel-Red". The figure $\frac{1}{17.5}$ was computed using cross-validation.

6 Identifying Barrels

Before the barrels are identified, the images undergo morphological pre-processing.

6.1 Morphological Processing

The images are blurred with a median filter of kernel size 5 to remove stray "Barrel-Red" pixels. The morpholog-

ical operation of "closing" is performed on these images to fill in small gaps in the images. This showed a noticeable improvement in barrel detection accuracy.

6.2 Contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. Contours are the main method of barrel detection used in this project. Contours are detected after thresholding the image to make it binary. However, not all contour points are stored. Only 4 crucial points per contour is saved. This showed a noticeable improvement in speed.

6.3 Differentiation Barrel from non barrel region

A region was deemed a barrel if 2 conditions were met:

1. If the area of the contour was at least $\frac{1}{4.4}^{th}$ of the largest contour in the image. The figure was computed using cross validation.
2. If the height to width ratio of the bounding rectangle of the contour was within 1.15 and 2.69.

These conditions ensured that regions that were rectangular but not a barrel were eliminated. However, these conditions caused the model to overfit the training data (more on that in the "Things to improve" section).

Thus, barrel(s) were identified using these steps.

7 Estimating Parameters of the barrels

7.1 Height, width

These are returned directly from cv2.boundingRect() function.

7.2 Depth

Since camera parameters weren't provided, the focal length was estimated using the distance provided. Height and width were made use separately and thus, 2 focal lengths were calculated, using the formulae:

$$F_{width} = \frac{(PxD)}{W}$$

$$F_{height} = \frac{(PxD)}{H}$$

where F is the focal length, P is the height or width in pixels, D is the distance we know and H & W are the height and width of the barrel respectively.

Using these two focal lengths, 2 distance estimates were calculated, using the same formulae. The 2 distance estimates were averaged to output the final distance estimate or depth of the barrel from the camera.

This method performed better than linear regression on the height and width.

7.3 Corner and Centroid Coordinates

Using outputs from cv2.boundingRect() function, the bottom left and top right coordinates (x and y) were calculated. Similarly the centroid coordinates (x and y) were also calculated.

8 Results

The model performed well on all the given 50 images. However, it performed well on just 4 of the final 10 test images. The reasons are enumerated in the "Things to improve" section. Below are a few choice results from the validation set and from the final test set.

A table of all the 60 images is listed after the final test set results. This contains all the parameters of all the barrels.

8.1 Validation set Results

The following 7 images are from the initial 50 images released to us. These images are being displayed because they're quite similar to the images in the final test set. A couple of these images are just interesting images, so they're being showcased here.

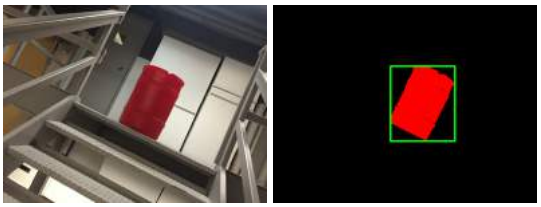

Fig. 1. 2.14.png

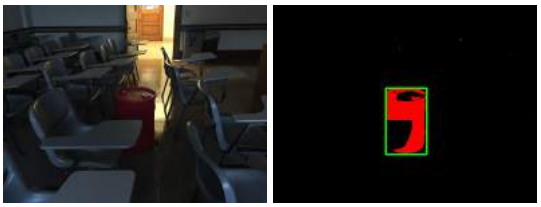

Fig. 2. 2.2.png

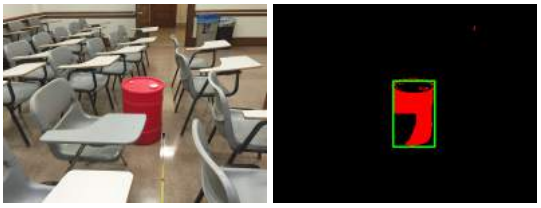

Fig. 3. 2.2.png

Fig. 4. $2_3 1$.png

Fig. 5. 3.11.png

Fig. 6. 5.9.png

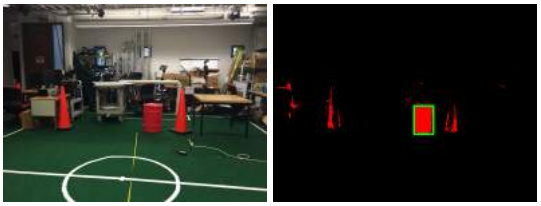

Fig. 7. 5.4.png

8.2 Final Test set results

The following 10 images are from the final test set containing 10 images that were released to us. All 10 of these images are being displayed because they all have something interesting to convey about the algorithm. In most of these 10 images, barrels while detected were not drawn boxes around and classified as barrels. This is because the conditions that were imposed on a ROI to be a barrel overfit the training data. This is discussed in detail in "Things to improve" section.

Fig. 8. 001.jpg

Fig. 9. 002.jpg

Fig. 10. 003.jpg

Fig. 11. 004.jpg

Fig. 12. 005.jpg

Fig. 13. 006.jpg

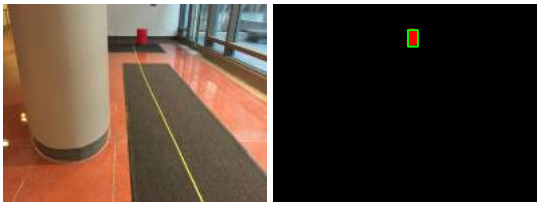

Fig. 14. 007.jpg

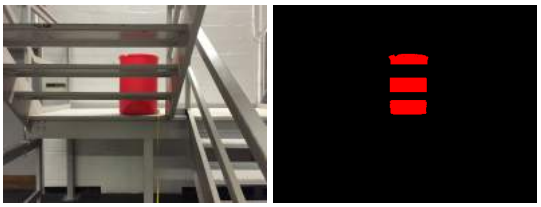

Fig. 15. 008.jpg

Fig. 16. 009.jpg

Fig. 17. 010.jpg

8.3 Barrel Parameters

8.3.1 Initial set

This is a compilation of the results. The results tabulated are the image name, the coordinates of the bottom left of the barrel, the top right coordinates and the depths. It should be noted that all images have been resized to $400 * 300$. All 50 images are listed below here:

Name	Bottom Left	Top Right	Depth(s)
10.1	(205,157)	(218,137)	11.24
10.4	(203,139)	(217,119)	10.83
14	(201,81)	(210,66)	15.63
2.14	(177,205)	(275,92)	1.73
2.2	(170,226)	(232,125)	2.29
2.3	(181,214)	(244,115)	2.29
2.6	(125,217)	(210,89)	1.74
2.8	(164,204)	(240,98)	2.02
2-3.1.a	(63,239)	(149,108)	1.71
2-3.1.b	(302,173)	(367,88)	2.44
3.1	(204,217)	(252,137)	2.93
3.10	(174,192)	(225,128)	3.18
3.11	(188,172)	(229,105)	3.46
3.2	(178,202)	(221,132)	3.308
3.4	(209,210)	(261,128)	2.77
3.8	(179,121)	(225,49)	3.15
4.1	(188,222)	(222,169)	4.27
4.10	(195,106)	(228,52)	4.30
4.2	(253,132)	(290,78)	4.05
4.3	(117,164)	(154,111)	4.09
4.4	(190,176)	(221,126)	4.61
4.5	(183,194)	(217,138)	4.16
4.6	(243,187)	(291,130)	3.48
4.7	(184,174)	(217,120)	4.31
4.8	(187,169)	(220, 115)	4.31
5.1	(192,169)	(217,129)	5.73
5.10	(190,170)	(216,127)	5.43
5.11	(202,188)	(231,143)	5.02
5.2	(191,174)	(226,127)	4.47
5.3	(209,186)	(236,145)	5.45
5.4	(213,196)	(242,154)	5.19
5.5	(193,195)	(220,151)	5.26
5.6	(244,187)	(272,142)	5.11
5.9	(242,191)	(269,149)	5.38
6.1	(211,162)	(232,129)	6.89
6.3	(194,147)	(218,111)	6.16
6.4	(264,178)	(289,137)	5.67
6.5	(203,151)	(225,121)	7.06
6.6	(130,159)	(154,129)	6.78
6.7	(85,190)	(110,158)	6.42
6.8	(200,176)	(219,141)	7.07
7.2	(208,151)	(229,119)	6.99
7.3	(316,114)	(341,83)	6.53
7.4	(195,182)	(214,153)	7.73
7.6	(159,175)	(172,148)	9.82
8.2	(200,160)	(218,134)	8.38
8.3	(202,181)	(219,155)	8.62
8.4	(217,130)	(238,103)	7.63
8.5	(206,176)	(221,151)	9.38
9.2	(206, 235)	(220,211)	9.91
9.3	(191,127)	(206,104)	9.76

8.3.2 Final Test set

This is a compilation of the results. The results tabulated are the image name, the coordinates of the bottom left of the barrel, the top right coordinates and the depths. It should be noted that all images have been resized to $400 * 300$. All 10 images are listed below here:

Name	Bottom Left	Top Right	Depth(s)
001	(190,176)	(221,126)	4.61
002			
003			
004			
005			
006			
007	(204,64)	(219,39)	9.38
008			
009	(211,225)	(231,191)	6.97
010	(210,102)	(224,81)	10.57

9 Analysis of results

All barrels in the 50 validation set images were detected. The bounding boxes and depth were fairly accurate as well. Bounding boxes sometimes had an error of ± 10 pixels. The depth sometimes had an error of ± 0.5m. However, therein lies the problem.

9.1 Things to improve

1. Overfitting: The model overfit the train data. The algorithm failed in the stage of differentiating between ROIs that had barrel-red color and actual barrels. To refresh the memory, the algorithm categorized an ROI as barrel if it met the following two conditions:
   (a) If the area of the contour was at least $\frac{1}{4.4}^{th}$ of the largest contour in the image. The figure was computed using cross validation.
   (b) If the height to width ratio of the bounding rectangle of the contour was within 1.15 and 2.69.

   The figures $1.15, 2.69$ and 4.4 were all obtained by cross validation for the validation set, but were too well tailored for the data at hand. The conditions were too stringent. They failed on the testing set. For example, barrels in images 002, 004 and 005 were detected once the height/width ratio was relaxed a bit and allowed to be between 1 and 2.75.
2. Orientation: All bounding boxes were straight rectangles and orientation wasn't taken into account. Instead of fitting just straight rectangles, orientation should have been taken into account. This would have caused the barrel in 004 to be detected.
3. Region Analysis: Multiple "Barrel-Red" regions weren't analysed. Region-growing was implemented, which grew the barrel over small regions of obstacles like in image 3.11.png. This was implemented using the "closing" method. However, this failed on image 008. Instead of "Closing", different ROIs should be taken into consideration and seen if in combination, they form a barrel.
4. Illumination Invariance: Illumination invariance wasn't handled since a few images in the training set like images 2.2.png, were dimly lit images. Thus, barrel-red pixels of those intensities came through to the Gaussian model. However, there was no brightly lit image like Image 005. This caused the algorithm to fail on image 005.
   A simple fix would be to apply "Contrast Limited Adaptive Histogram Equalization" (CLAHE). CLAHE takes local regions into consideration and performs histogram equalization on them. If this was performed on all images before the hand labeling stage and during the prediction stage, illumination invariance might be resolved. CLAHE in action:

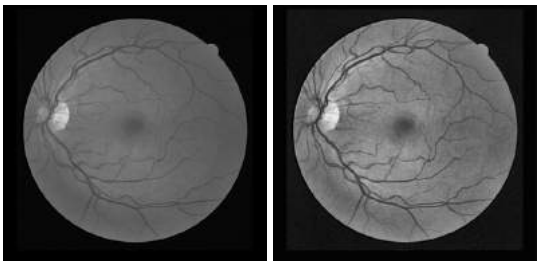

Fig. 18. A sample image demonstrating CLAHE

5. GMM: Gaussian Mixture Models weren't implemented since good results were obtained using just a single Gaussian. However, implementing a GMM might help with overfitting since more lenient and more generalized conditions might be imposed.

10 Conclusions

The project was to segment barrels from images using GMMs. While a GMM wasn't implemented in this project, a single Gaussian performed well, despite the overfitting and other issues listed above. While the performance could have been better, a lot was learnt from this project. The hazards of overfitting or of designing an algorithm that's too tailored for the data at hand was well understood.

On the whole, the project was a great learning experience.