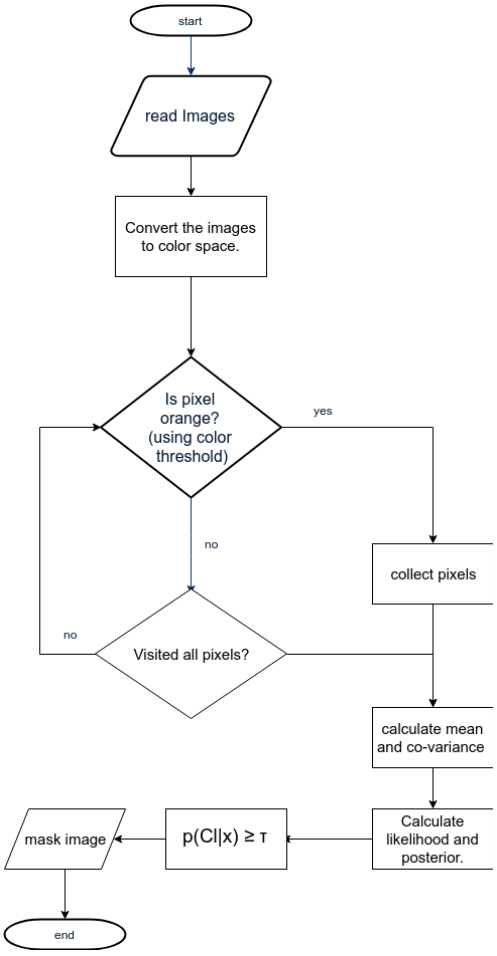
CMSC 426 - project 1

Color Segmentation using GMM

1. Single Gaussian:

In order to perform color segmentation, we need the probability distribution for the orange pixels. The procedure is as follows:

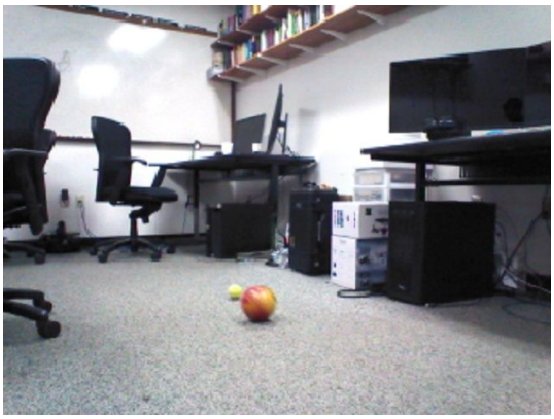
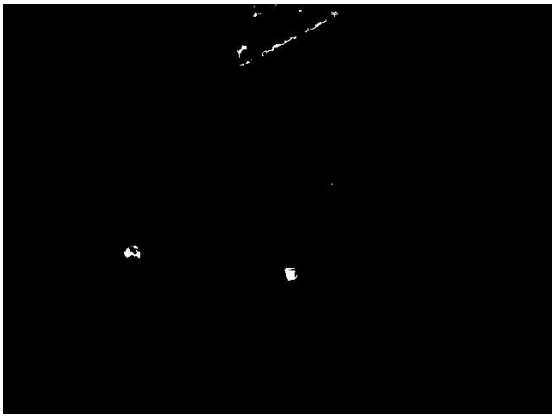
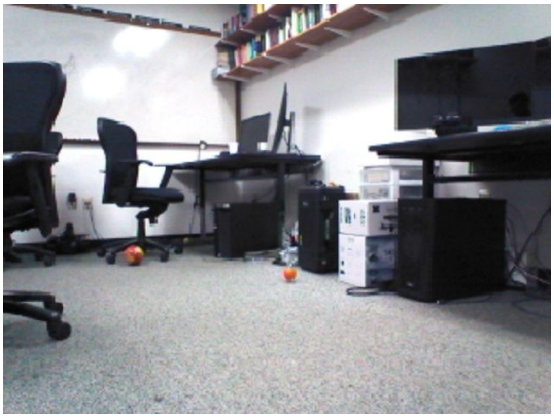
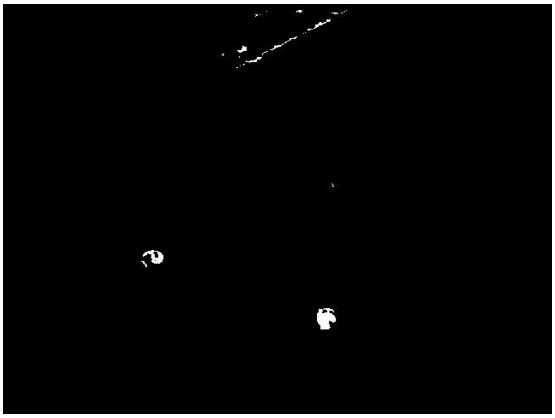
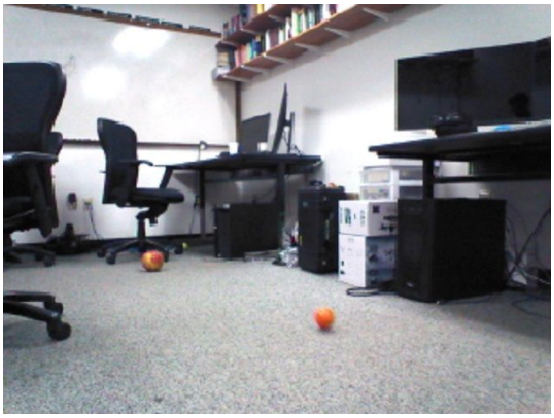


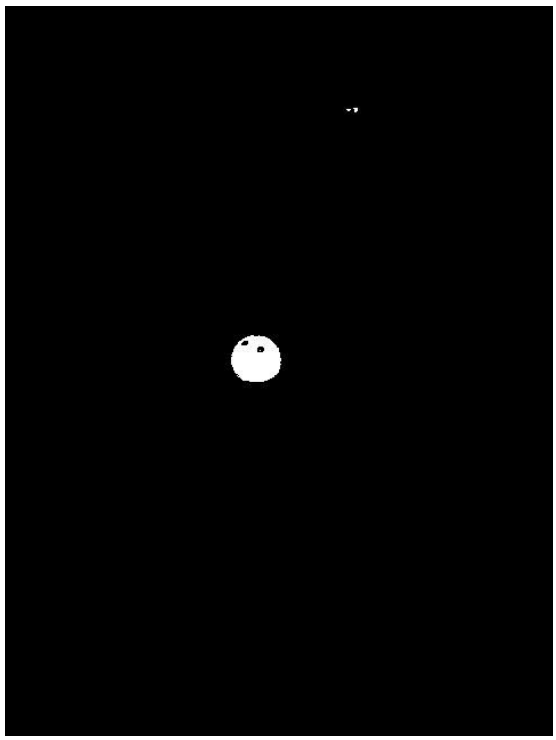
To find the probability distribution we need to find likelihood of orange pixels. For this we need to collect all the orange pixels. Using the color thresholding app in the matlab the minimum and maximum values of each channel of l*a*b* colorspace were determined, so that the process clustering of orange pixels can be automated. This could also be done using ‘roipoly’ function of matlab.

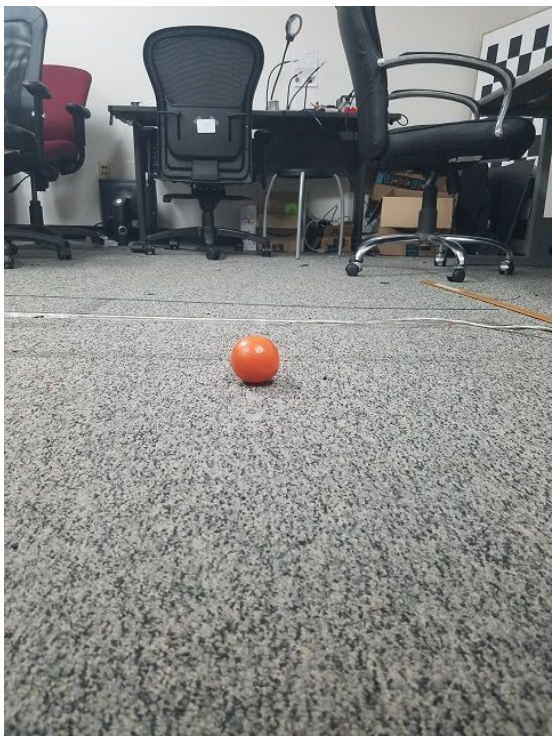
Now, the covariance and mean for the orange pixels can easily be calculated. Using the mean and covariance, posterior and likelihood of the test images is calculated. After thresholding posterior by a value defined by us, we get the masked images where the orange ball is detected.

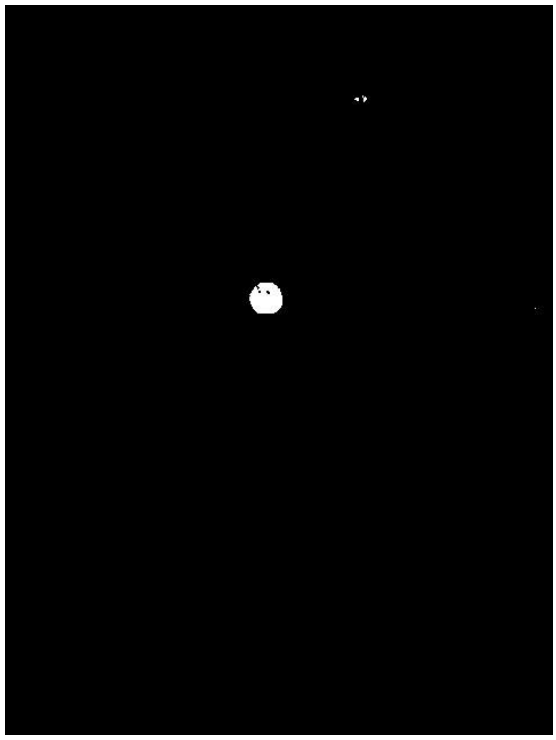
- Prior(p(C_l)) = probability of the pixel being orange= 0.5 in our case
- Likelihood(p(x|C_l)) = given that the color is orange, likelihood is the probability of the pixel being the value 'x'. This is essentially a normal/gaussian distribution which is calculated by using the function 'mvnpdf' in matlab.
- Posterior(C_l|x) = probability that a pixel is of color orange.
- Posterior = $p(Cl|x) = \frac{p(x|Cl)*p(Cl)}{\sum_{i=1}^{l} p(x|Ci)p(Ci)}$
- Posterior is directly proportional to Prior*Likelihood. Since we are thresholding the Posterior, we don't care about the denominator as we aren't interested in the absolute value of Posterior.
- Thresholding :$p(C_l|x) \geq \tau$,Where τ is the threshold value set by us, which signifies the confidence score.
- **Choice of color space**: After tweaking parameters in color thresholder app in matlab, it was clear that l*a*b* color space was perfectly able to capture the ball for the given database. RBG color space gave partial fit for the ball. HSV also provided some good results but not superior to l*a*b. Further, **Ignoring the 'l' color space in l*a*b can reduce our computations** since we only have to operate on a* and b*. Here L* represents the lightness and a* and b* represent the green–red and blue–yellow color components. Reading up further on wikipedia revealed that CIELAB (lab color space) was designed to be **perceptually uniform with respect to human color vision**, meaning that the same amount of numerical change in these values corresponds to about the same amount of visually perceived change. Thus it was more intuitive to choose this color space.

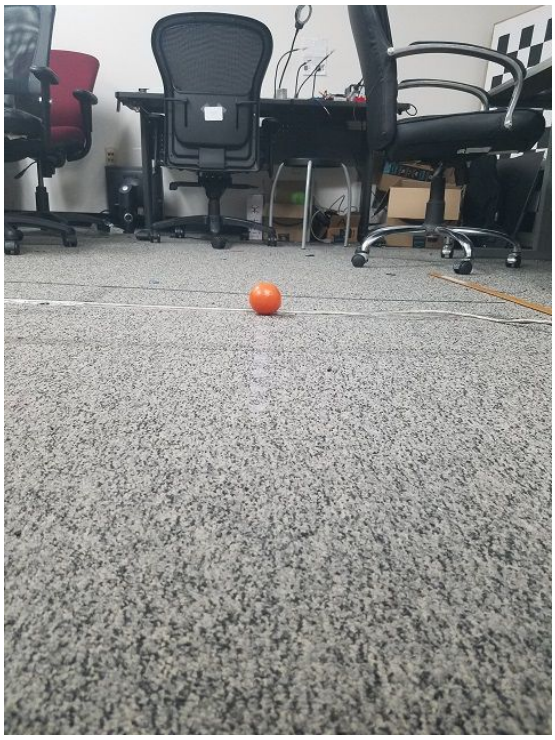
Results of single gaussian on the **test data:**

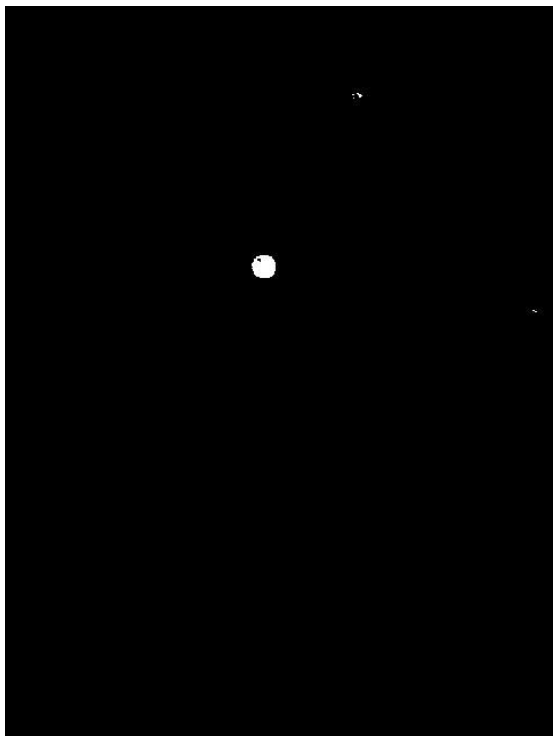







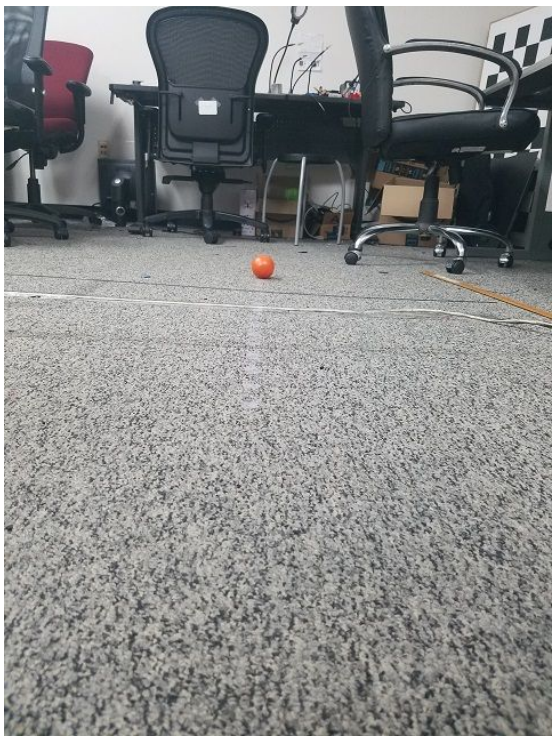


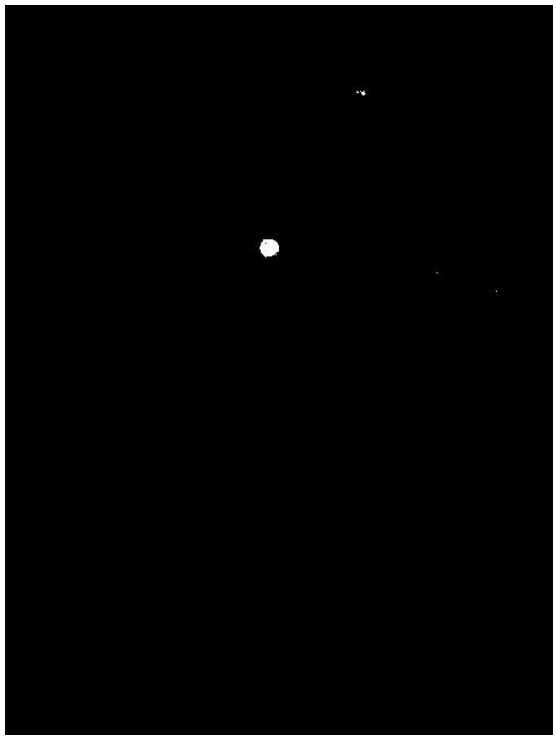


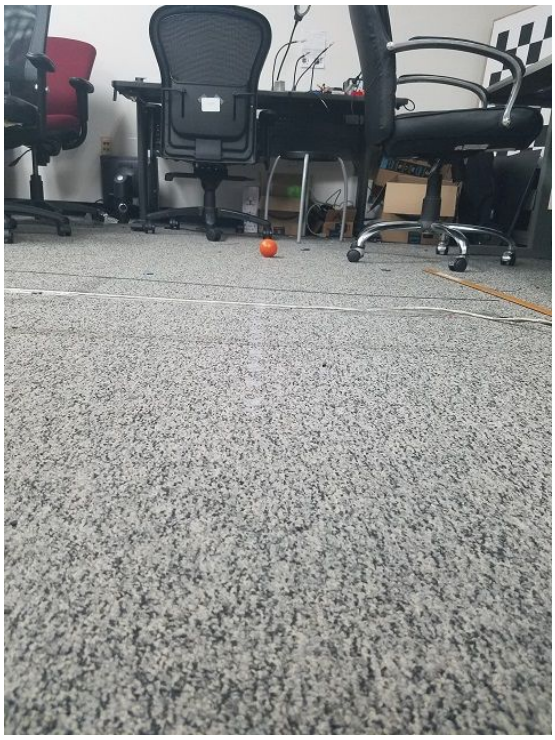


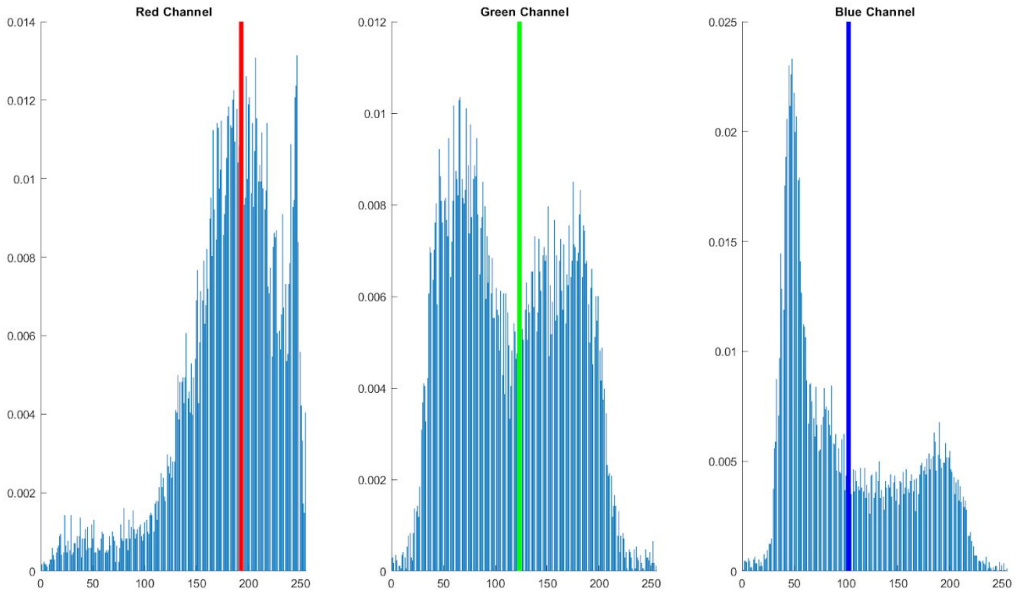


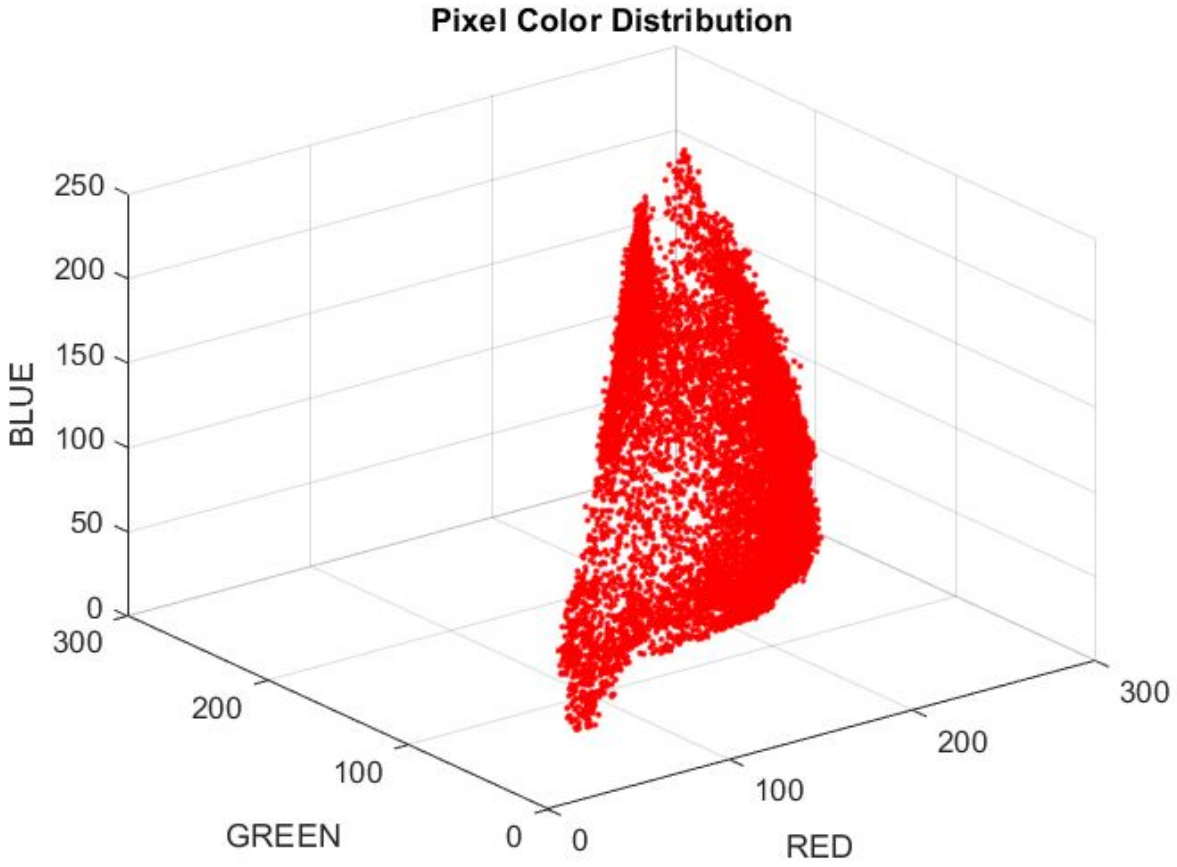









Red Channel
Green Channel
Blue Channel
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0.025
0.02
0.015
0.005
50
100
150
200
250



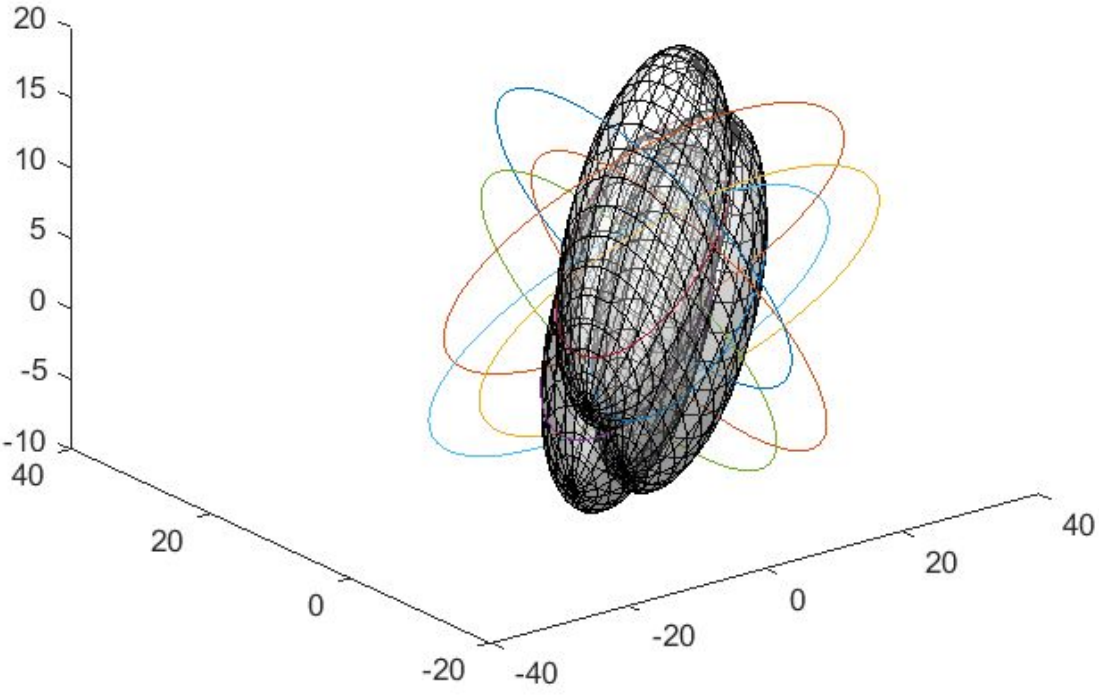
2. Gaussian Mixture Model

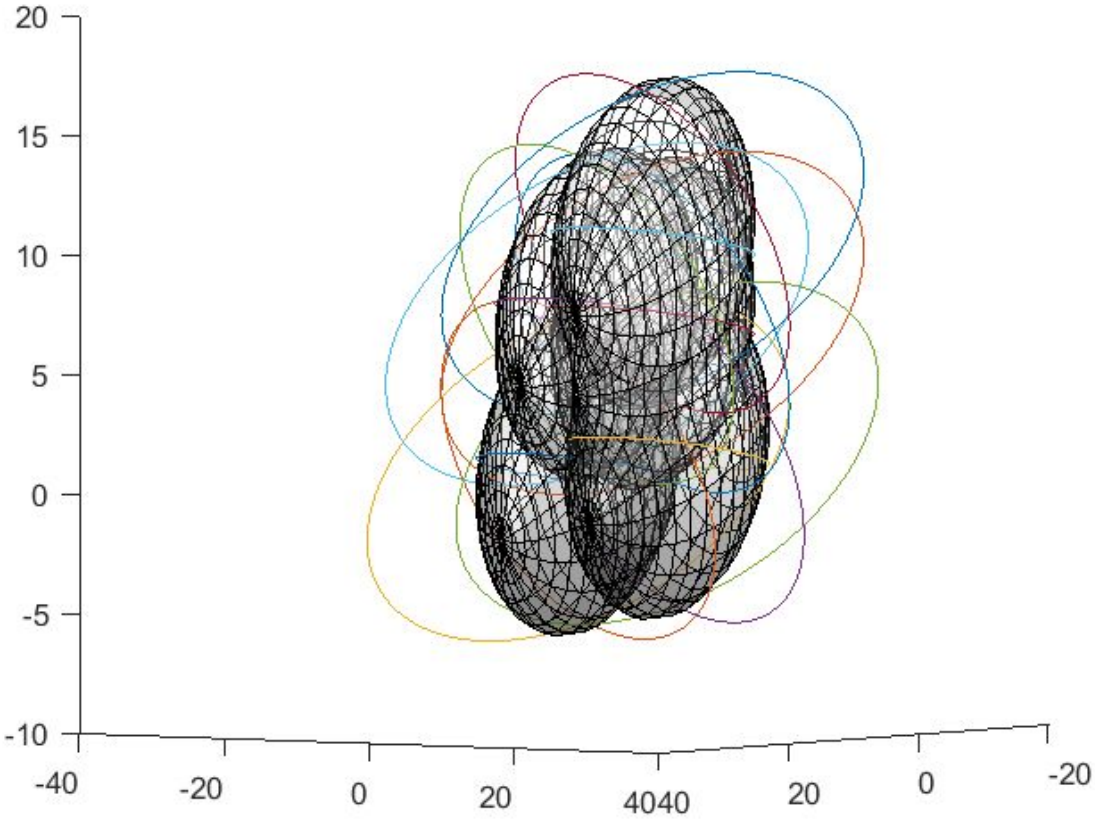
Single gaussian model isn't perfectly able to capture the color space entirely if there there is change in lighting conditions as the color may not be bounded by a single ellipsoid. In order to capture the entire color space in all kinds of lighting, a mixture of ellipsoids are used to segment the color. Now, instead of finding a single ellipsoid, we need to find multiple, say k, ellipsoids. This can be done by Maximum likelihood estimation.

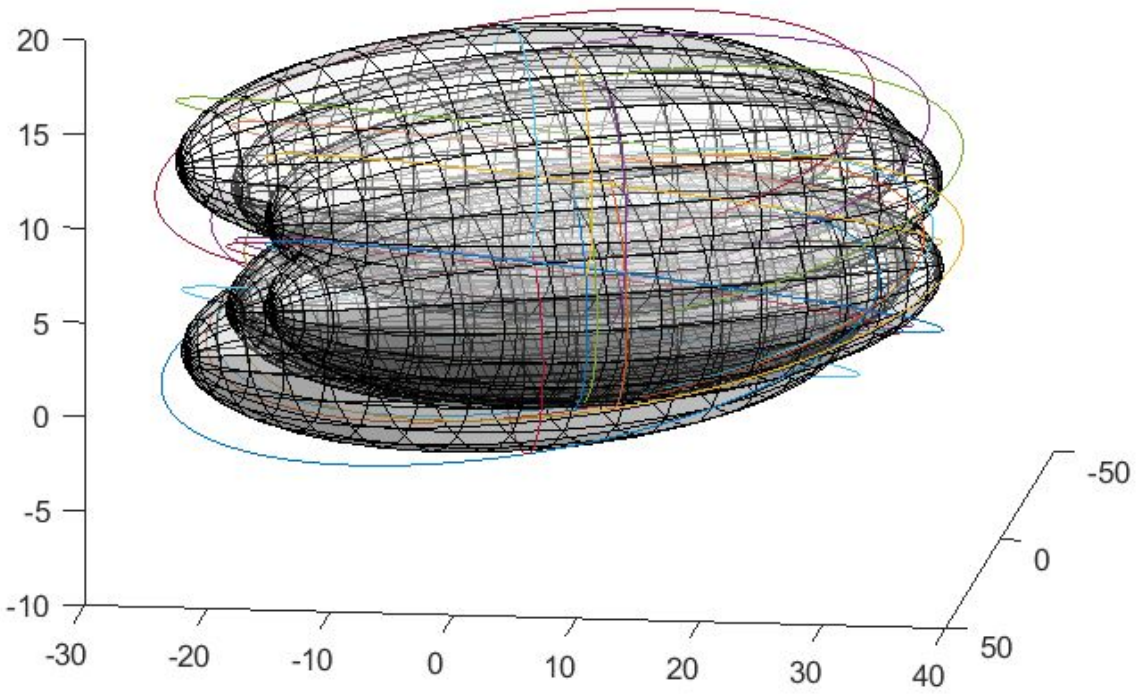
The entire process can be divided into 2 steps: Estimation step and Maximization step. Both the steps are repeated until convergence.

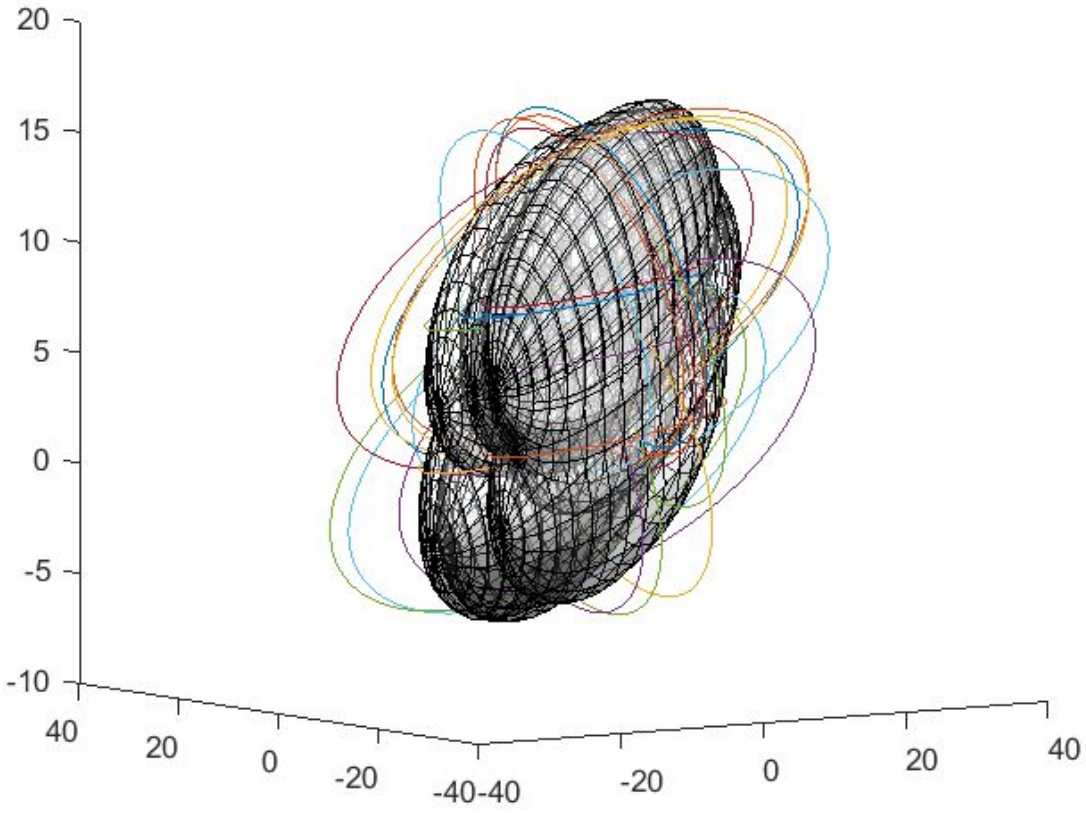
- Initialization method: **Initializing mean** was done randomly using the function rand in matlab. **Initializing covariance** was a bit trickier since it has to be positive semi definite. So, it was only initialized by calculating cov of all the images(pixels x 3 dimension), elements so that it is PSD and its inverse exists.
- **Number of gaussians in GMM -** There is an optimal number for number of gaussians. For our dataset we selected the number of gaussians as **7**. If the number of gaussians are fewer, we aren't able to entirely capture the color space, so our algorithm underfits and the accuracy is low. On the other hand, if we increase the number of gaussians,

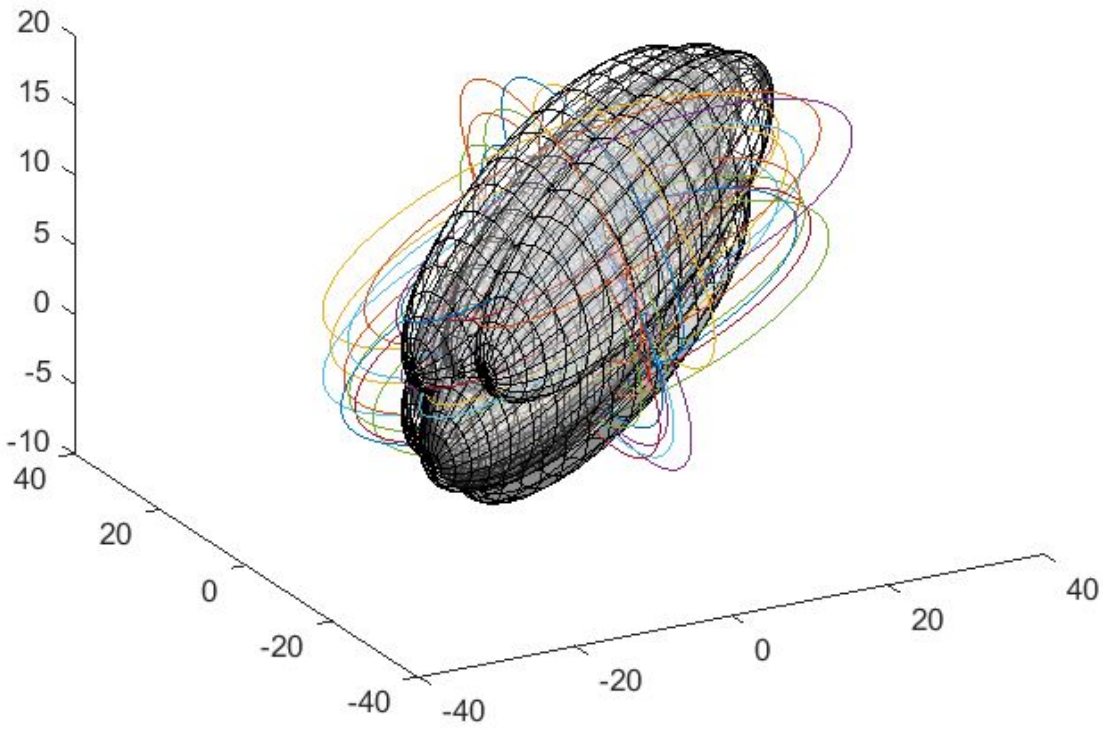
even though the accuracy increases on the current dataset, the algorithm tends to overfit , which may not be a good generalization. Overfitting will lead to poor prediction.


20
15
10
5
0
-5
-10
40
20
0
-20
-40
-20
0
20
40


20
15
10
5
0
-5
-10
-40
-20
0
20
4040
20
0
-20


20
15
10
5
0
-5
-10
-30
-20
-10
0
10
20
30
40
50
0
-50


20
15
10
5
0
-5
-10
40
20
0
-20
-40
-40
-20
0
20
40



GMM vs single gaussian

Since, single gaussian gives only one ellipsoid, which is same as having a single cluster in case of GMM. In such cases, the model that predicts the underfits the data as the color space can't be captured by a single ellipsoid. The results of GMM are better as it gives us multiple ellipsoid which captures the color in different lighting condition with more robustness. There is an optimal number for number of gaussians. For our dataset we selected the number of gaussians as **7**. On the other hand, if we increase the number of gaussians, even though the accuracy increases on the current dataset, the algorithm tends to overfit , which may not be a good generalization. Overfitting will lead to poor prediction.

3. Distance estimation and cluster segmentation results:

Distance estimation table:

Image name	Estimated distance

1.jpg	326.9
2.jpg	333.3
3.jpg	261.7
4.jpg	87.4
5.jpg	123
6.jpg	185
7.jpg	243
8.jpg	265

GMM results: