# CMSC 426 Computer Vision

# Project 1

# Color Segmentation using GMM

---

**Siyuan Peng**
**Da Shen**
**Daoyi Wang**

## Our choice of color space for this project:

We chose the RGB color space because we are familiar with it and it is widely used because it corresponds to the mechanisms of human retina.

# 1. Single Gaussian

## What we did:

- We initialized tau = 0.00000017 as our threshold for the purpose of identifying orange pixels and applying masks on the testing images; we also set prior = 0.5 as if in a common uniform distribution.
- In the training process, we extracted the orange pixels from training images and obtained the mean and covariance matrix of the Gaussian model from those orange pixels.
- During testing, we first computed the likelihood of the pixels belonging to the class orange, and the prior
- Then, we obtained the posterior, which is the probability of a pixel being orange, using likelihood and prior; We were able to classify the pixels into the orange class if posterior exceeds the threshold tau.
- To visualize the result of testing, we applied a mask on the test images that display the orange balls.

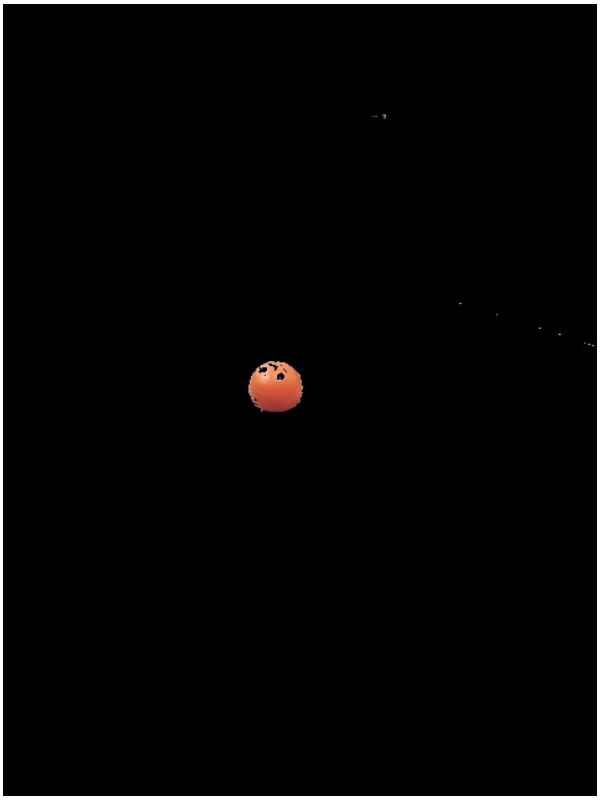
## Results on the test images:

1) Test Image 1

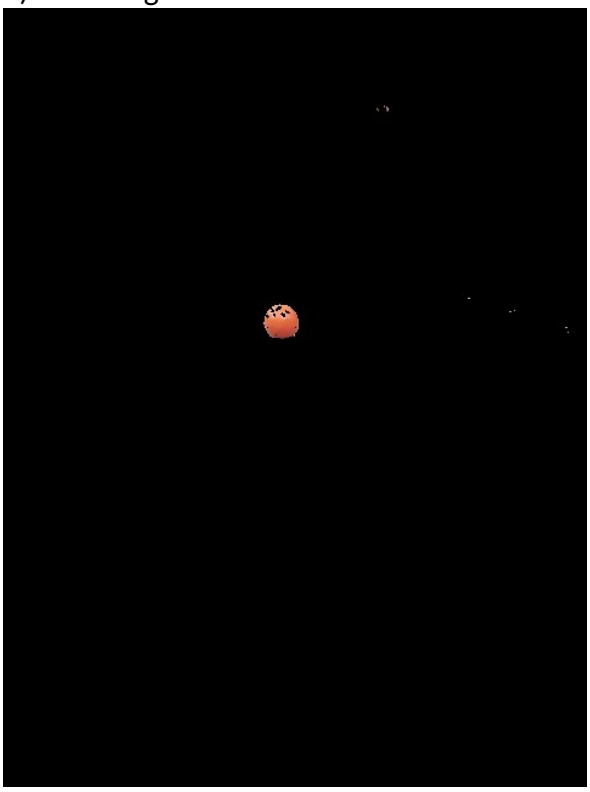
2) Test Image 2

3) Test Image 3

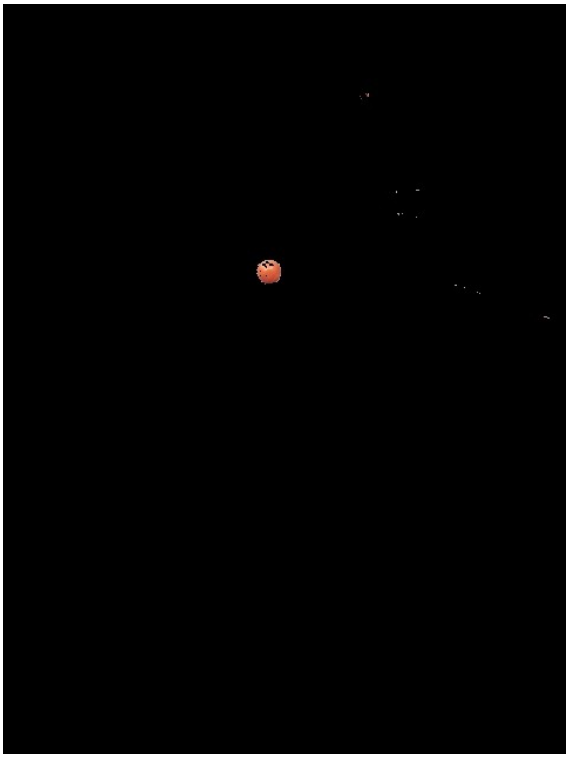
4) Test Image 4



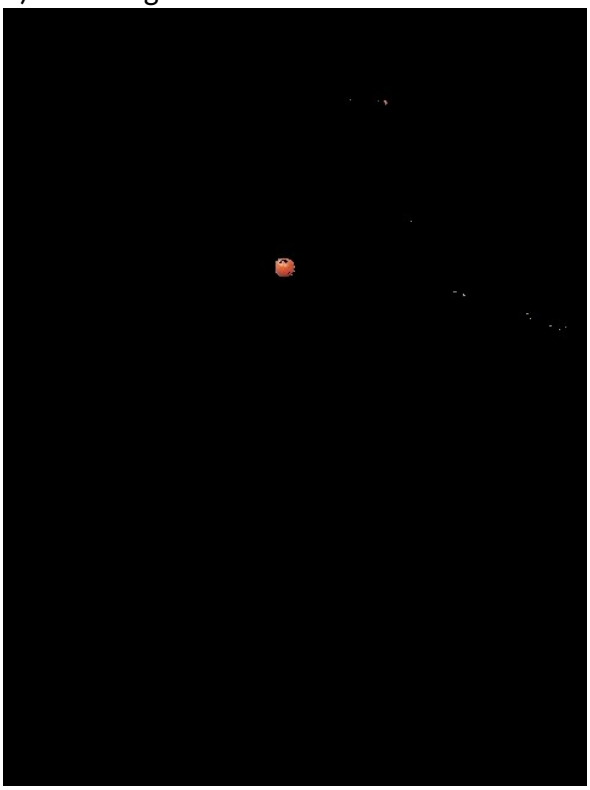
5) Test Image 5



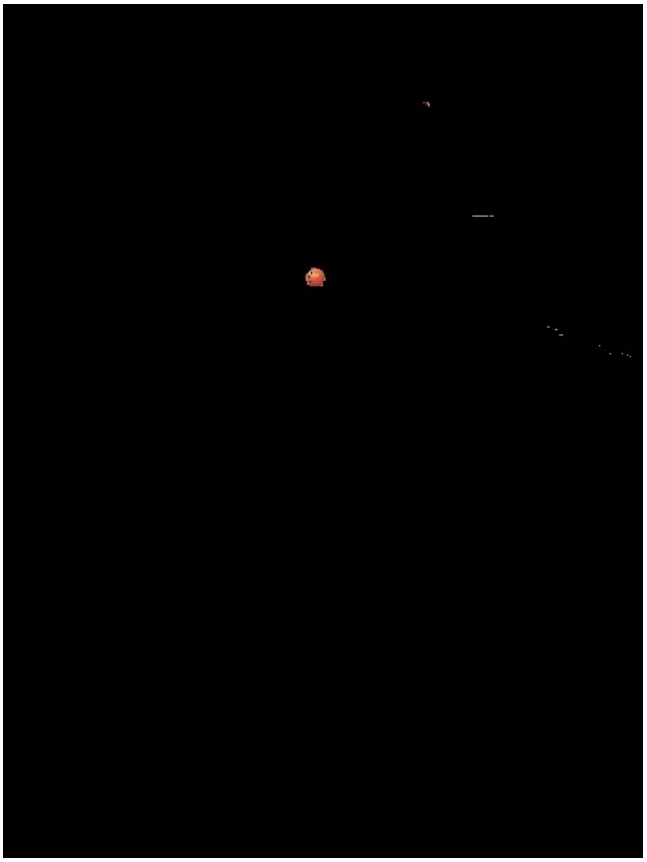
6) Test Image 6



7) Test Image 7



8) Test Image 8



## Problems we encountered:

- At the early stages of working on this project, we did not have a full understanding of the training process of the Gaussian model, so we mistakenly used all the pixels to train the model instead of only the orange pixels. By re-evaluating our algorithm and researching on this subject, we realized that it was essential to extract and use only the orange pixels for training.

- Another difficulty we encountered is setting a good threshold value (tau). Through trail and error, we were able to decide on our tau value that gives a satisfying result.

# 2. Gaussian Mixture Models

**Our choice of initialization method:** Random initialization for scaling, mean, and covariance for each cluster.

**Our number of Gaussians:** K = 20

## What we did:

- In trainGMM, we first randomly initialized the parameters: scaling, mean, and covariance.
- In the expectation step, we calculated the likelihood and obtained each cluster's weight using the scaling parameter and the likelihood.
- In the maximization step, we updated the parameters through an iterative process until convergence. At convergence, the resulting parameters are stored in a file named "weights" for future use.
- During testing, in testGMM, we calculated the likelihood and posterior for each cluster and classified the pixels in the orange class if the posterior exceeds the threshold.
- We applied a mask and produced a masked image for each of the testing images.

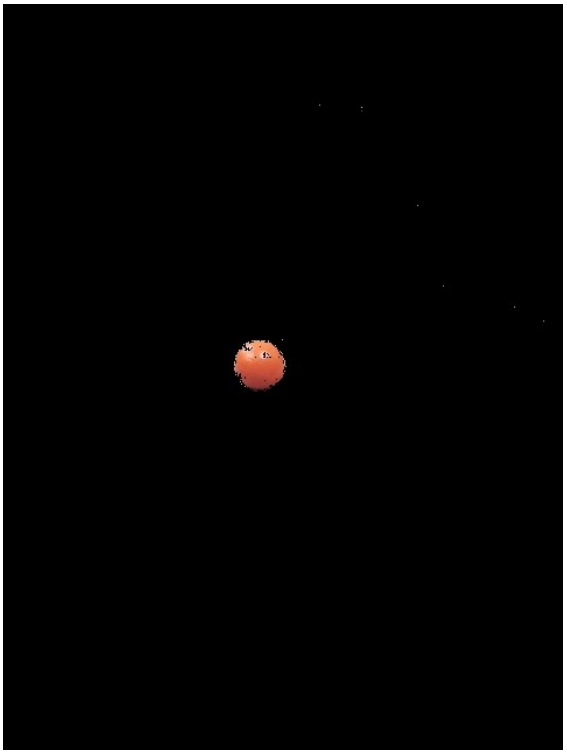
## Results on the test images:

1) Test Image 1

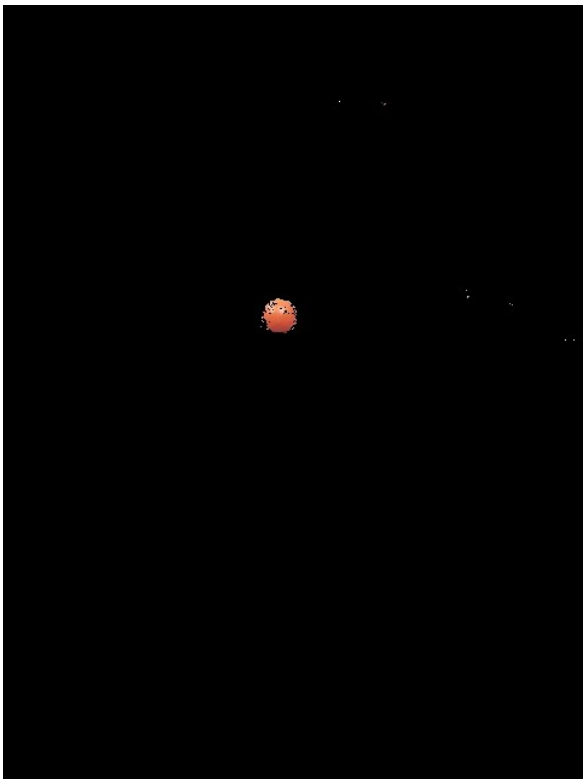
2) Test Image 2

3) Test Image 3

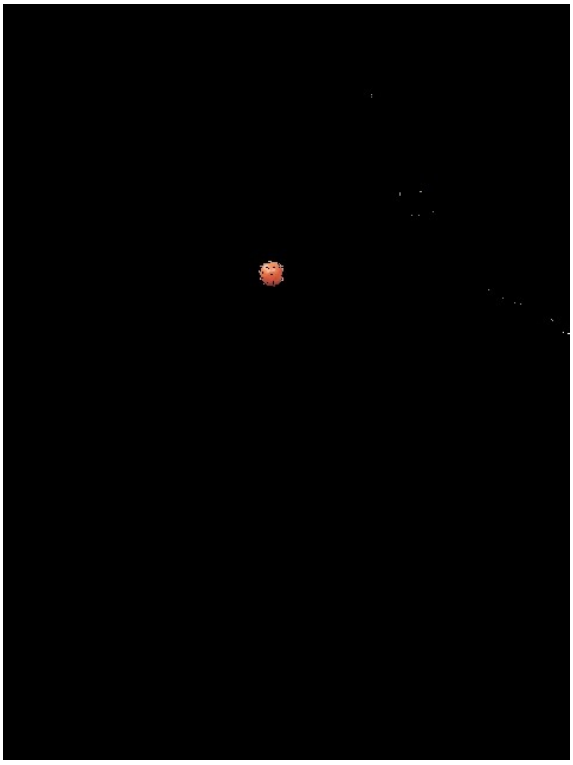
4) Test Image 4



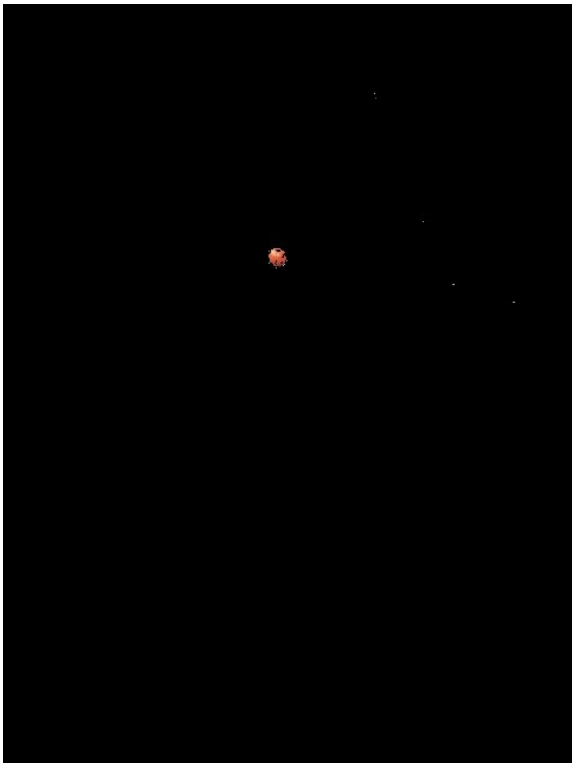
5) Test Image 5



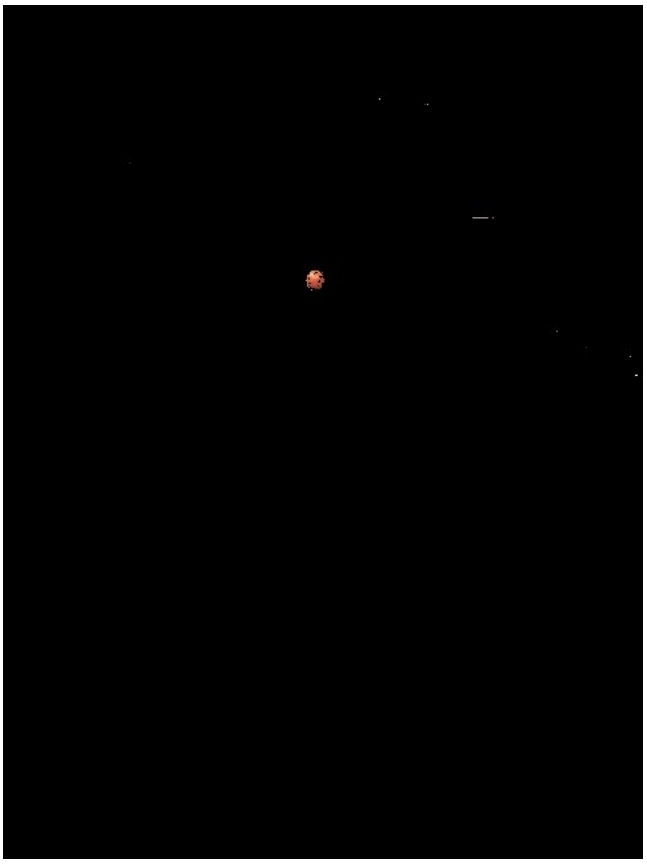
6) Test Image 6



7) Test Image 7



8) Test Image 8

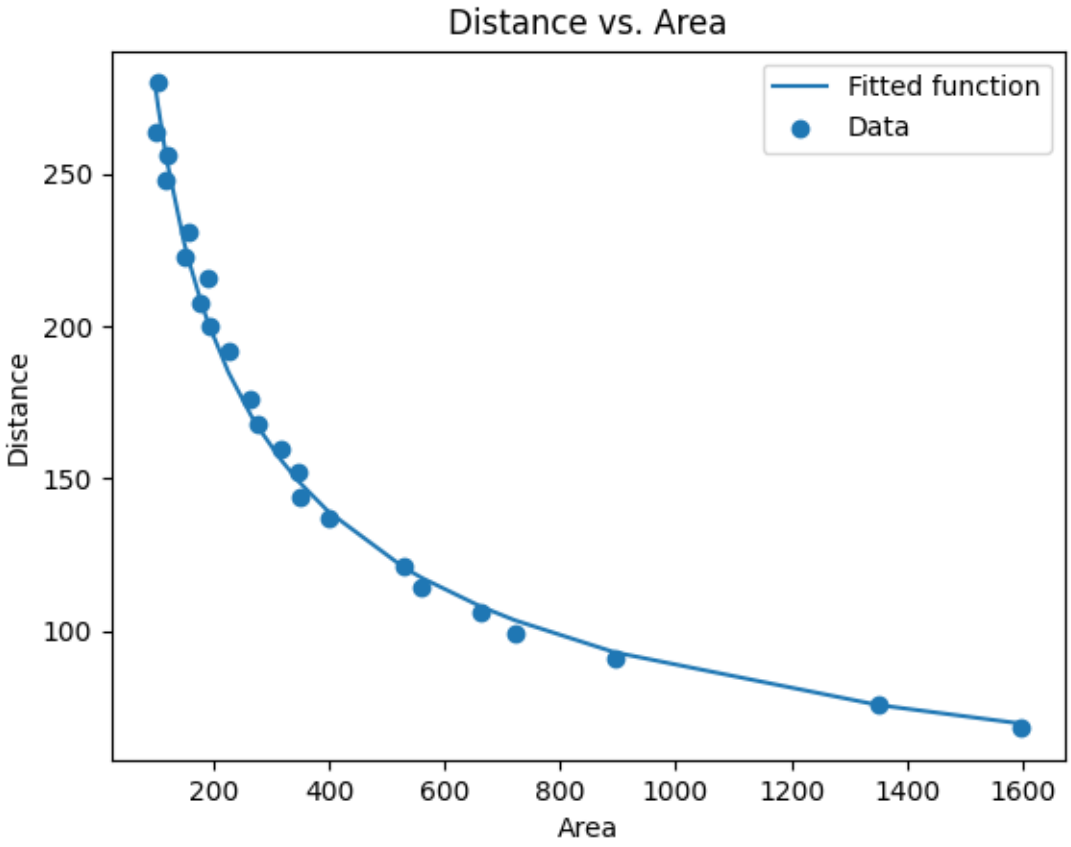


## Problems we encountered:

- When we first wrote out the code for the training algorithm, we iterated through every pixel to calculate the cluster weight on each data point. The processing of iterating by pixels were extremely time-consuming and we were even not able to finish training. Then we realized, though cluster weight of each pixel needs to be calculated, vectorization can still be applied to do those computations at once. Solving this issue by vectorization significantly improved the performance of the training.
- We encounter zero division error when computing the covariance matrix in the maximization step. We found that a mean's R value too close to 225 will cause the determinant to be 0, so we adjusted the values accordingly to avoid zero division.

## Distance Prediction:

- Based on the knowledge that the orange ball's area on the image and its distance to the camera may have an inverse square relationship, we defined a fitting function inverse_square() to train the model and find the best-fit parameters.
- We used the areas and distances of the training images provided to train a model that best-fit these data pairs.
- Using the trained model, we made prediction on the distance of all test images and stored the denoted images under results/distances.



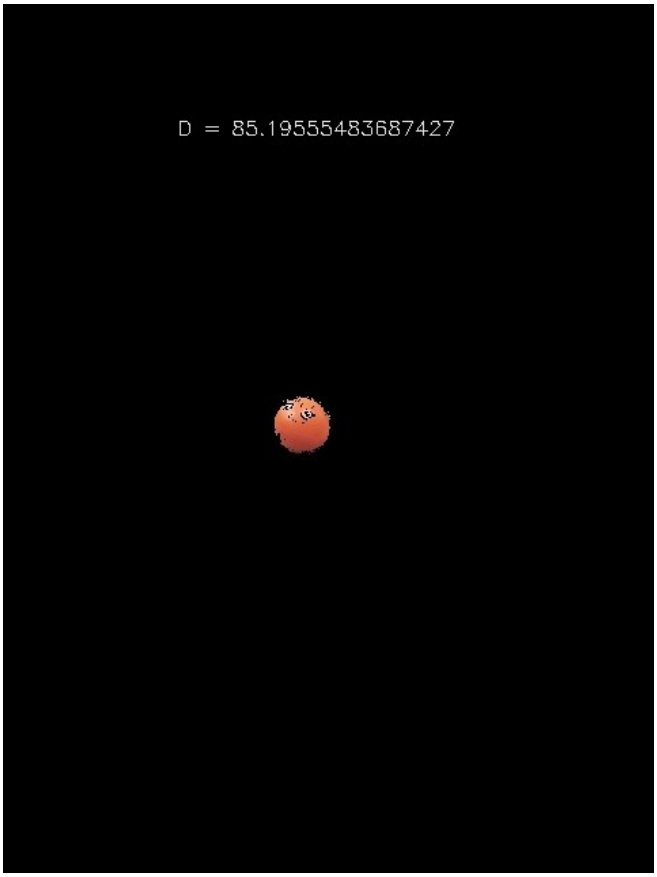
## Distance Prediction on the test images:

1) Test Image 1

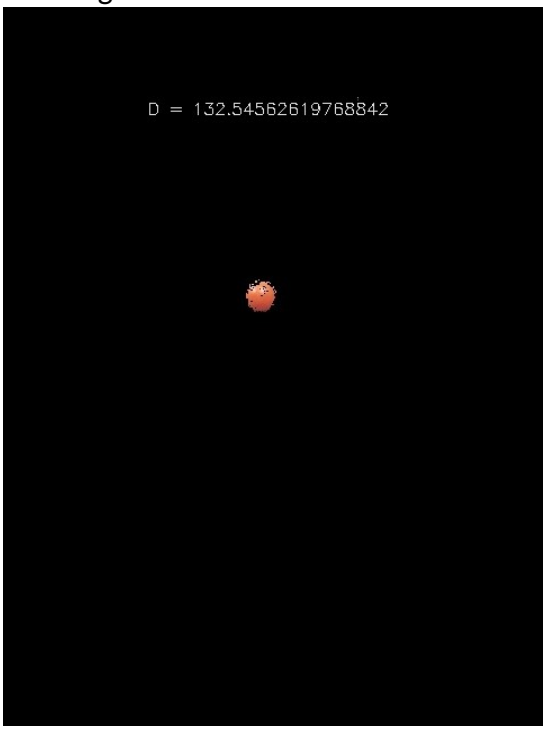

2) Test Image 2

3) Test Image 3

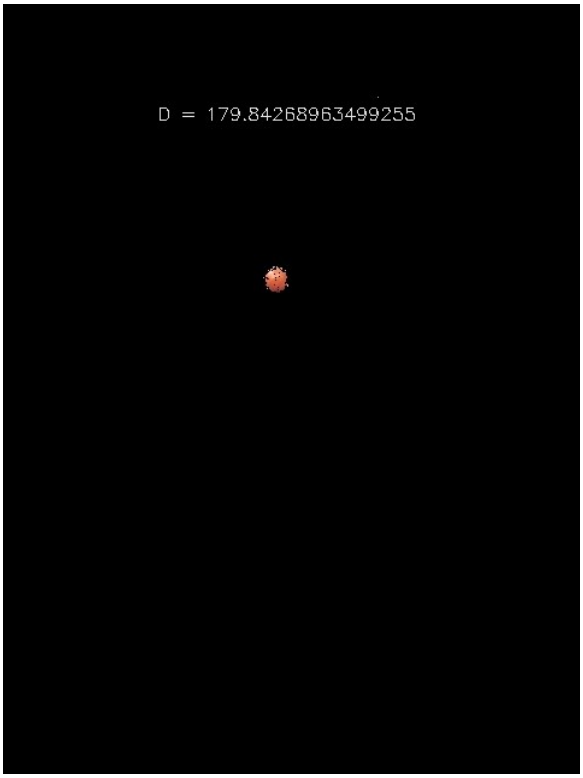

4) Test Image 4



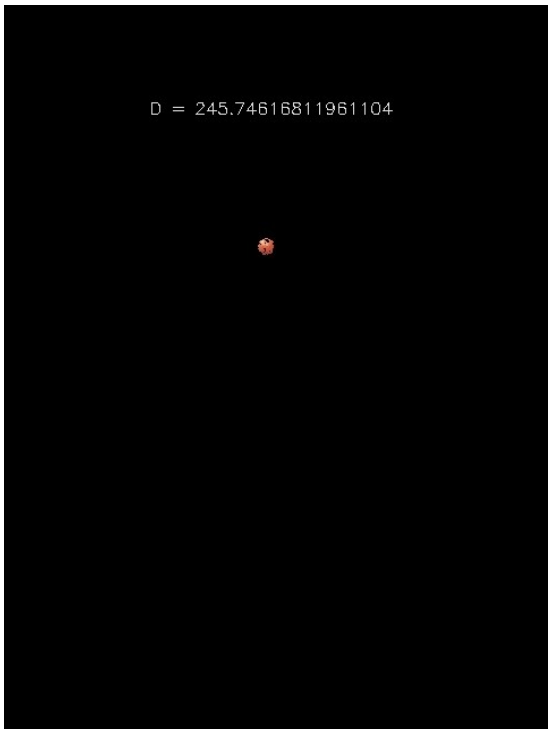
5) Test Image 5



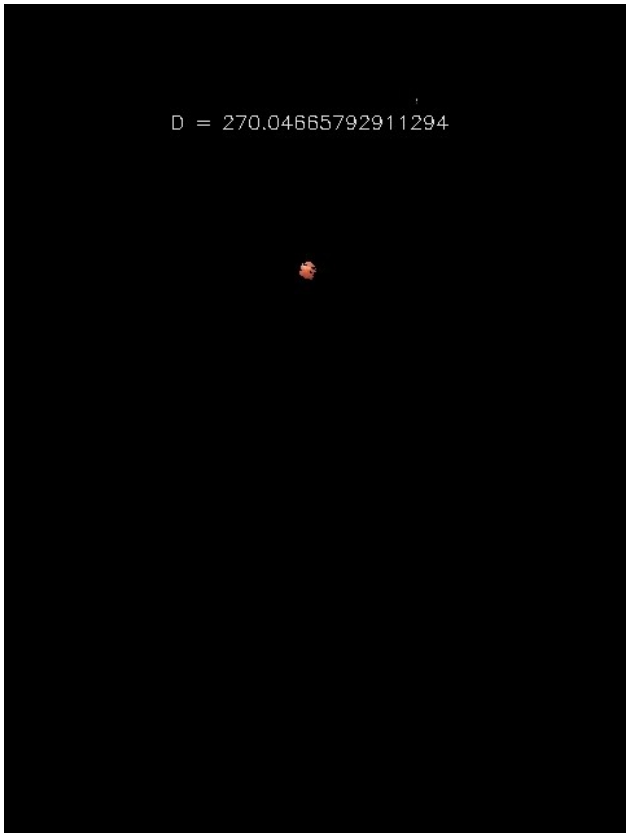

6) Test Image 6



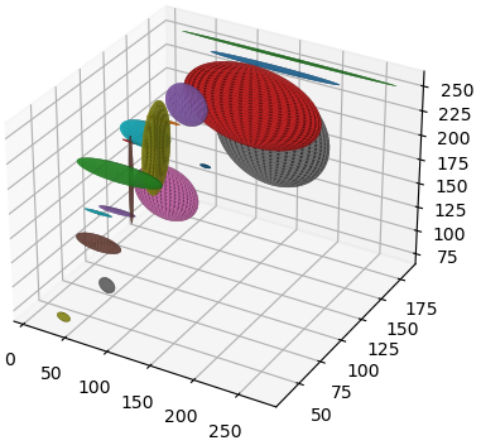
7) Test Image 7




8) Test Image 8



### Ellipsoids:




We used k=20 and fitted 20 ellipsoids in the RGB color spaces. We used python matplotlib.pyplot package for plotting.

## 3. Why is GMM better than Single Gaussian:

In our implementation, GMM performed better at finding the pixels of the orange ball. In general, GMM performs better at color segmentation for a few reasons:

- Colors under different lighting conditions may not be well-bounded by one ellipsoid.
- GMM is more robust than single gaussian in different lighting conditions.
- We are able to more finely capture the object by fitting multiple gaussian models

## 4. Examination on the Algorithms

### Single Gaussian:

Strengths:

- Simpler implementation
- Lower computational cost

Limitations:

- Does not perform well with noise, such as different lighting conditions and other orange colored objects.
- For example, in test images 1, 2, and 3 where another orange object is present, single gaussian performed poorly and included many pixels on the other object. In other test images, single gaussian was unable to identify some small parts of the orange ball where different lighting such as shadows were present.

### GMM:

Strengths:

- More robust to different lighting conditions than single gaussian.
- Fitting multiple gaussian models allow capturing more details of the object.

Limitations:

- More complexed to implement, higher computational cost
- Random initialization may make it take longer to converge.
- Choice of threshold requires manual tweaking.
- Though better than single gaussian, GMM is not completely robust to noise. The first three test images show that it cannot distinguish well between the orange ball and the apple, which means it lacks robustness to outliers. Also, different lighting conditions still affects its performance.