CMSC426 COMPUTER VISION

PROJECT 1

Color Segmentation using GMM

Authors

Deepti Bisht (dbisht)

Shreya Suresh (ssuresh2)

Wichayaporn Wongkamjan (wwongkam)

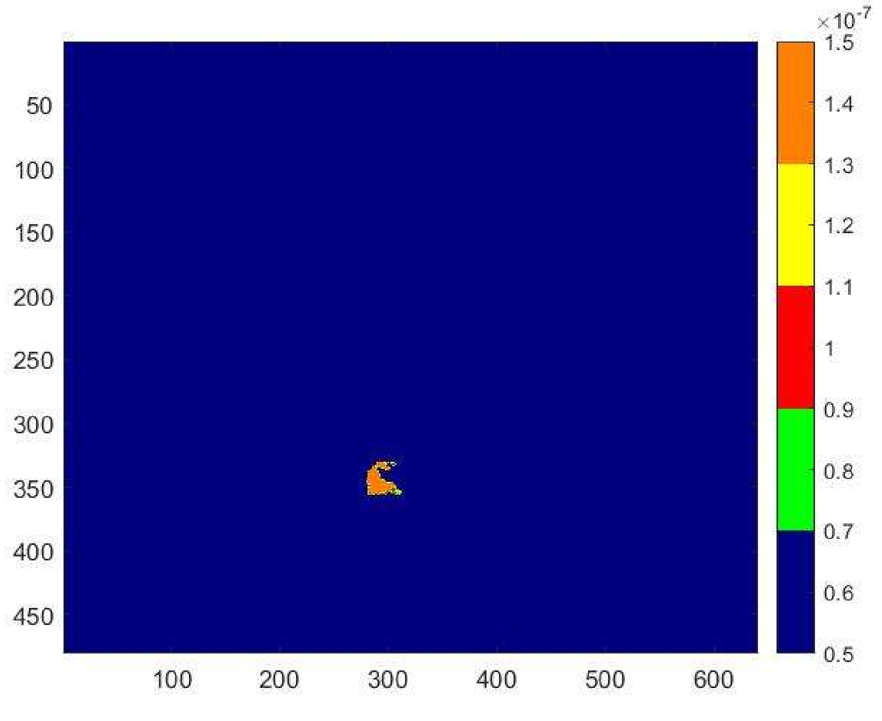
Single Gaussian

What We Did

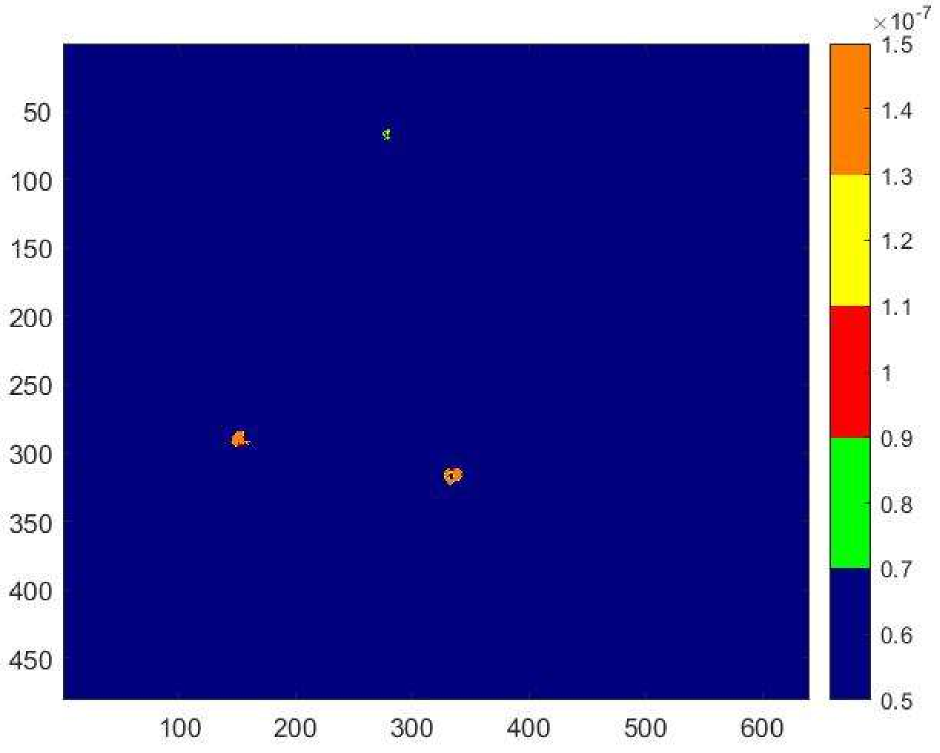
- Split the train set into two - 80% images were used for training and 20% images were used for cross validation (adjusting parameters like k, threshold, max_iterations).
- Obtained binary mask of the image using roipoly() function by manually marking the outline of the orange ball.
- Multiplied this with the train image (RGB) and obtained the RGB values of the *orange* pixels (RGBM).
- Calculated the mean (meanR, meanG, meanB) and covariance matrix (covRGB) for the orange pixels. This is the "training" phase of the model.
- For "testing", we calculated the likelihood (modelled by a gaussian distribution) of pixels belonging to class *orange*.
- This was then multiplied by the prior (taken to be 0.5) to obtain the posterior probability (from Bayes Theorem). The denominator, which would be common for all pixels, was not calculated since the probabilities would be mapped to fixed values using thresholds.
- We chose a threshold of $1.3\text{x}10^{-7}$ by observing how the mapping works if we vary the threshold from the minimum to the maximum value of orange (from RGBM).
- Finally, we used imagesc() to segment the ball and display the plot using a threshold.

Results

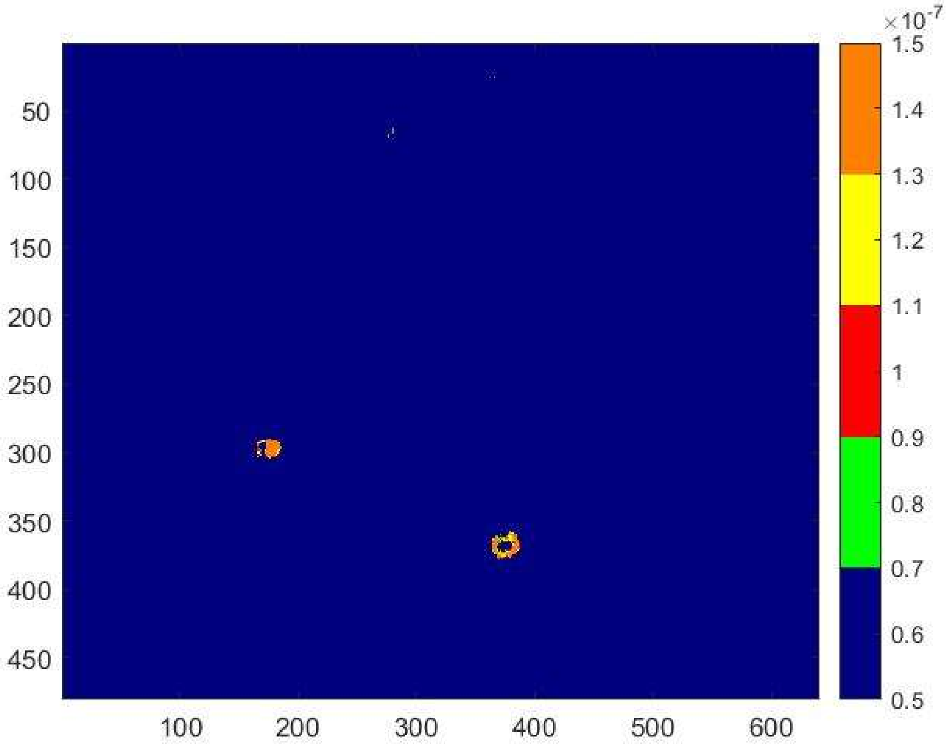
1. Test Image 1



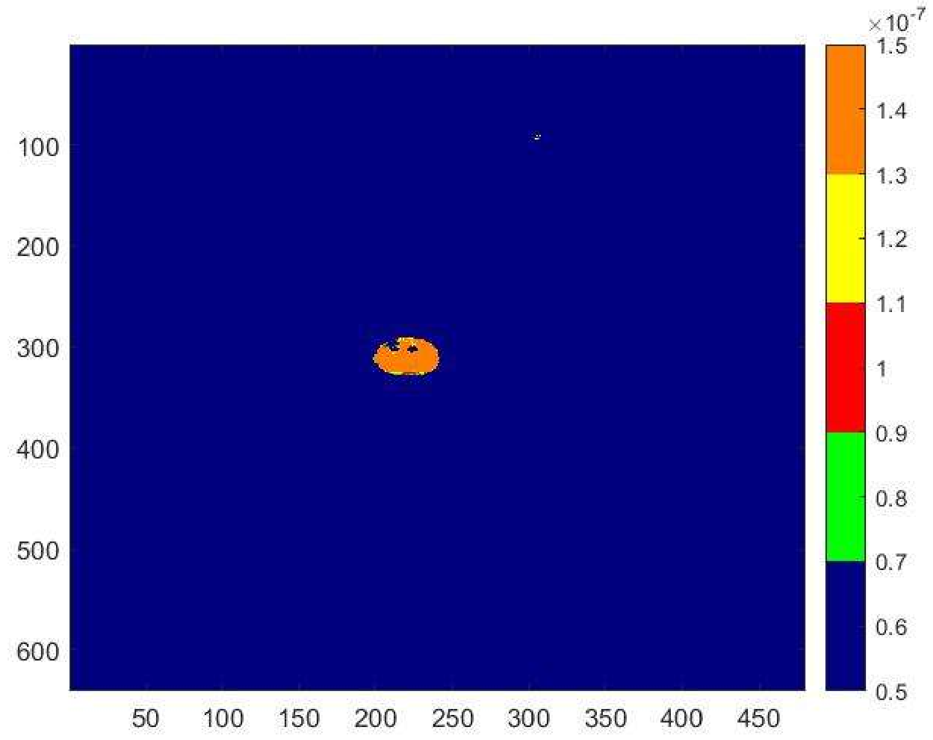
2. Test Image 2



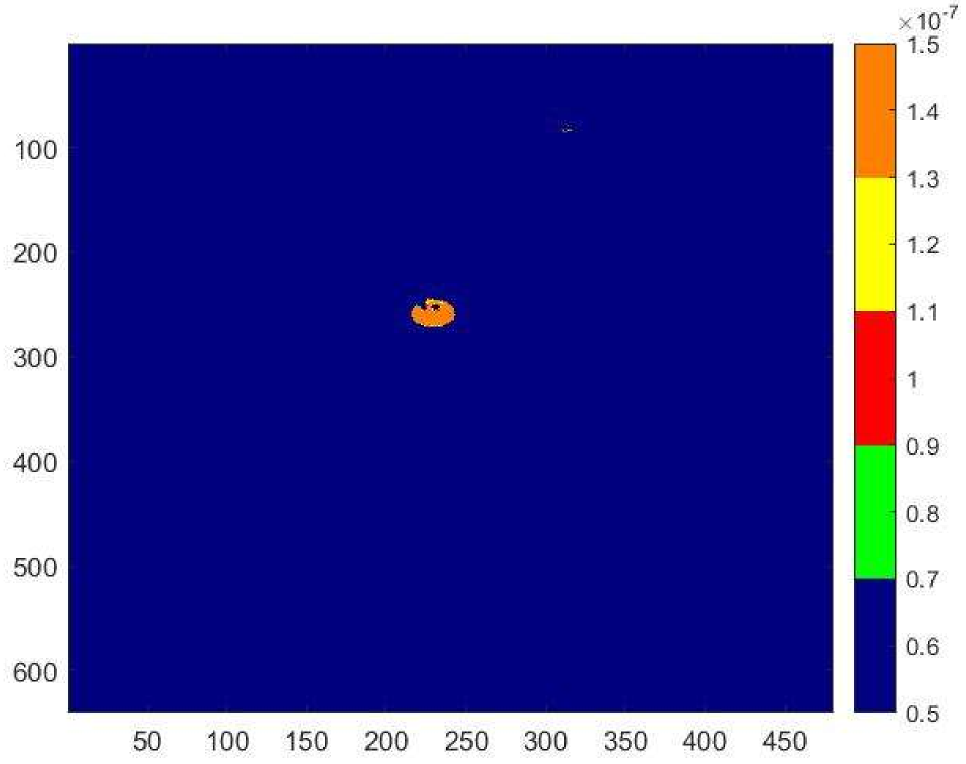
3. Test Image 3



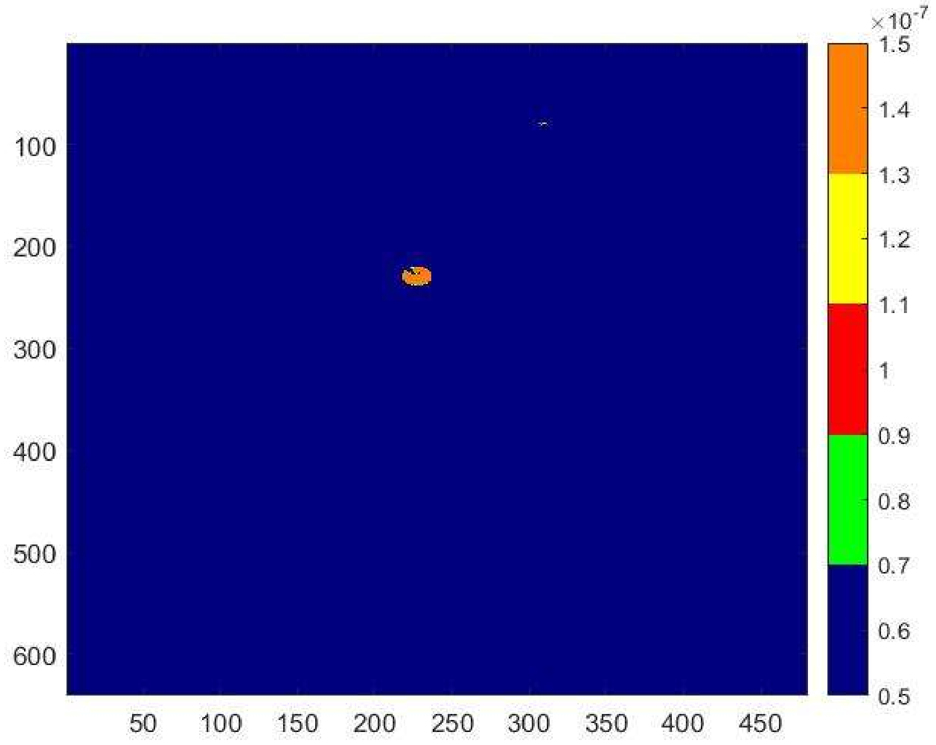
4. Test Image 4



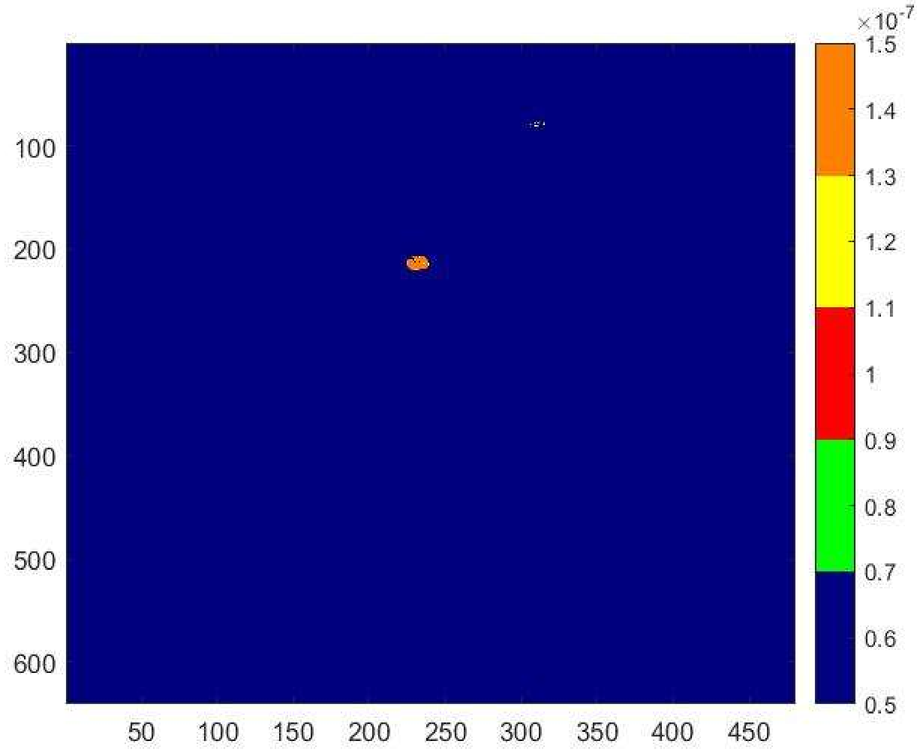
5. Test Image 5



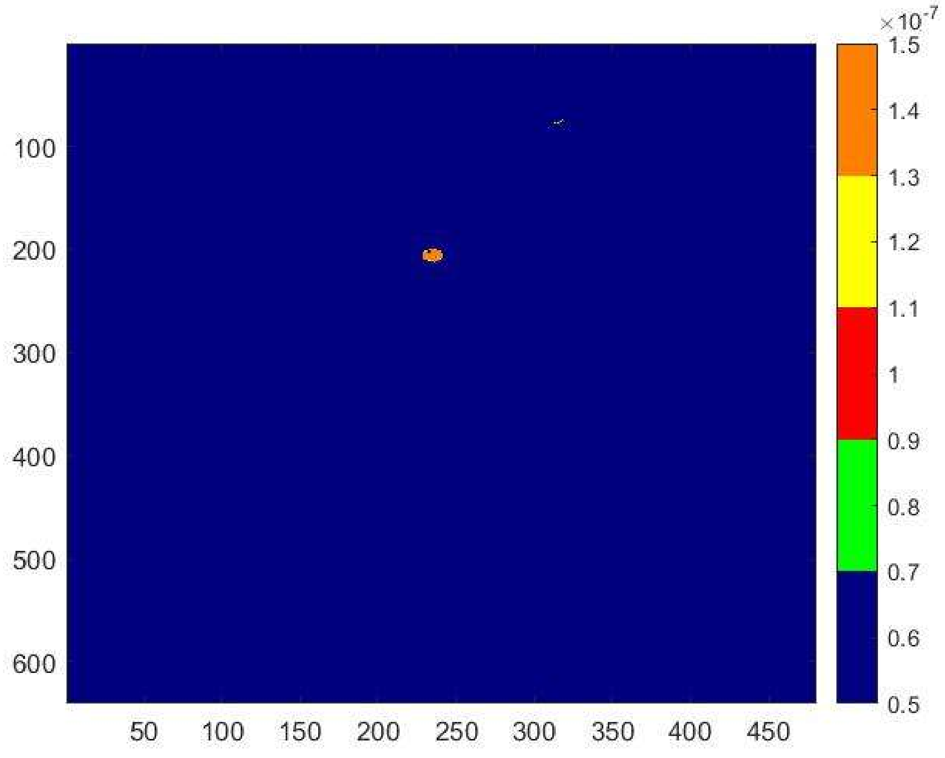
6. Test Image 6



7. Test Image 7



8. Test Image 8



Problems Faced

- Selecting the optimal threshold for segmentation.

How We Solved The Problem

- Varied the threshold between the minimum value of an orange pixel (as observed in RGBM) and the maximum value. Chose and approximate value from trial and error.
- Adjusted the value by cross validation - used the “test” (20% of training data) for this.

Limitations

- Human error induced while manually clicking on the outline of the ball while using roipoly().
- Algorithm cannot handle noise both internally (reflection, shadows on the ball) and externally (identifies pixels as orange in the background even if they have *similar* RGB values as that of orange.
- Algorithm performance heavily depends on the threshold chosen by the user.

Strengths

- Computationally economical.

Gaussian Mixture Models

What We Did

- The number of gaussian models was chosen to be 3 (k = 3).
- In the training phase, we initialized scaling factor, gaussian mean and covariance values for the models using kmeans. Performed the expectation and maximization steps. The convergence criteria (for the expectation step) chosen was 1.
- During testing, we calculated the posterior probability and clustered the *orange* ball by using the chosen threshold.

Choice of Color Space

We chose RGB model for the following reasons.

- Widely used
- Easier to understand since we have been exposed to it

Initialization Method

Kmeans was used to initialize the scaling factor, gaussian mean and covariance values. We observed that with random initializations, EM algorithm took a larger amount of time to converge.

GMM vs Single Gaussian

- Single Gaussian is not able to handle the change in lighting, reflection and shadow casting from the orange ball whereas GMM performs well in different lighting conditions.

Number Of Gaussians

- Shown below are the graphs for k = 3 (left) and k = 5 (right). The model is not performing significantly better for k = 5. Since selecting a higher value of k is computationally more expensive and the performance improvement is not huge, we chose the number of gaussian mixture models as 3.
- Same is the case with distance calculation. The performance was directly proportional to the number of models (until saturation) but the run time increased significantly.

Distance Calculation

- Training Phase
  - Calculated the area of the orange ball by summing up the pixel values (will be binary) in the output of roipoly().
  - We then fit a polynomial curve of degree 4 from the area calculated and the input distance.

- Testing Phase :
  - Used regionprops() to isolate the pixels that belong to the color orange and calculate the area of the ball.
  - Using the parameters of the curve that was obtained in the training phase and the area extracted, we calculated the distance.

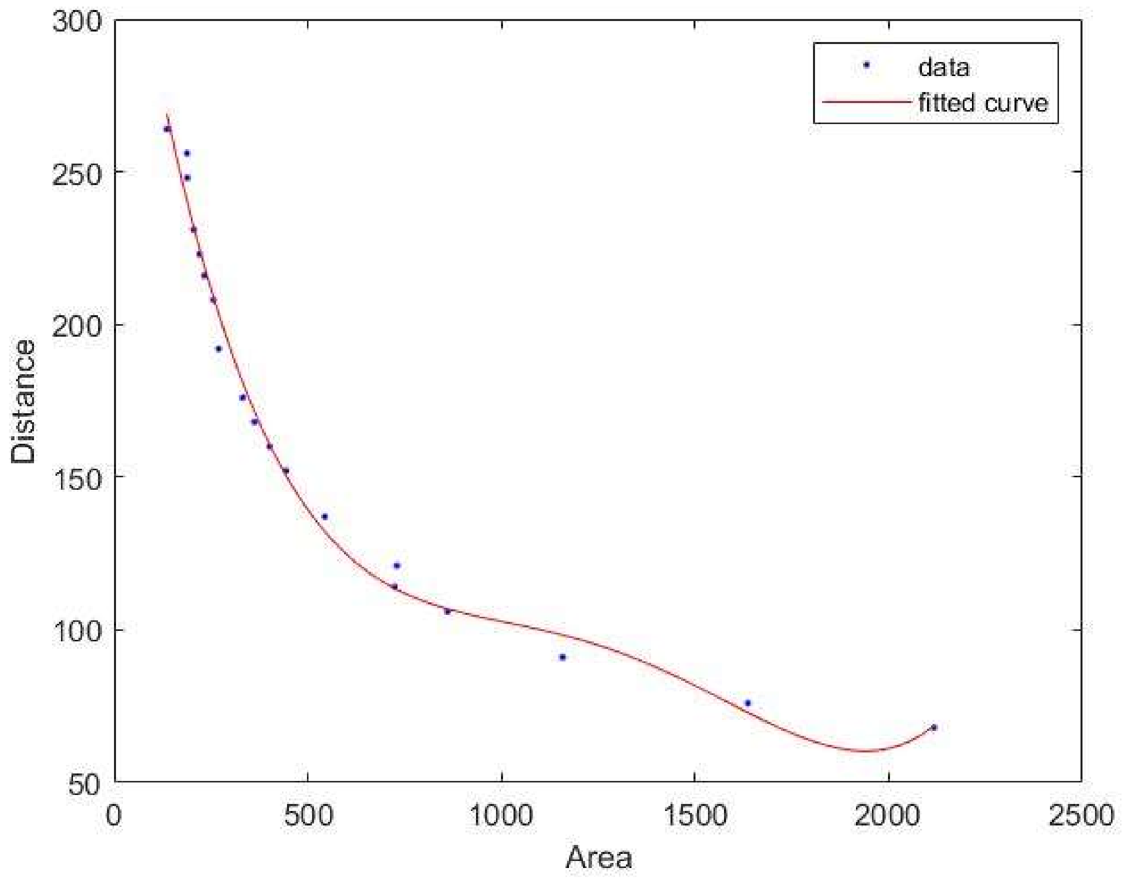
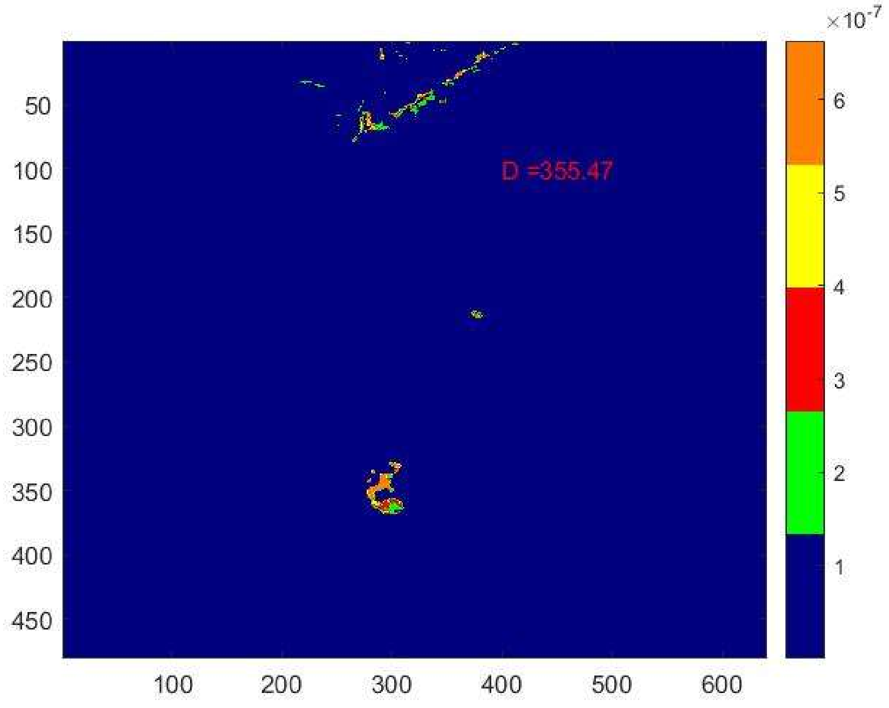


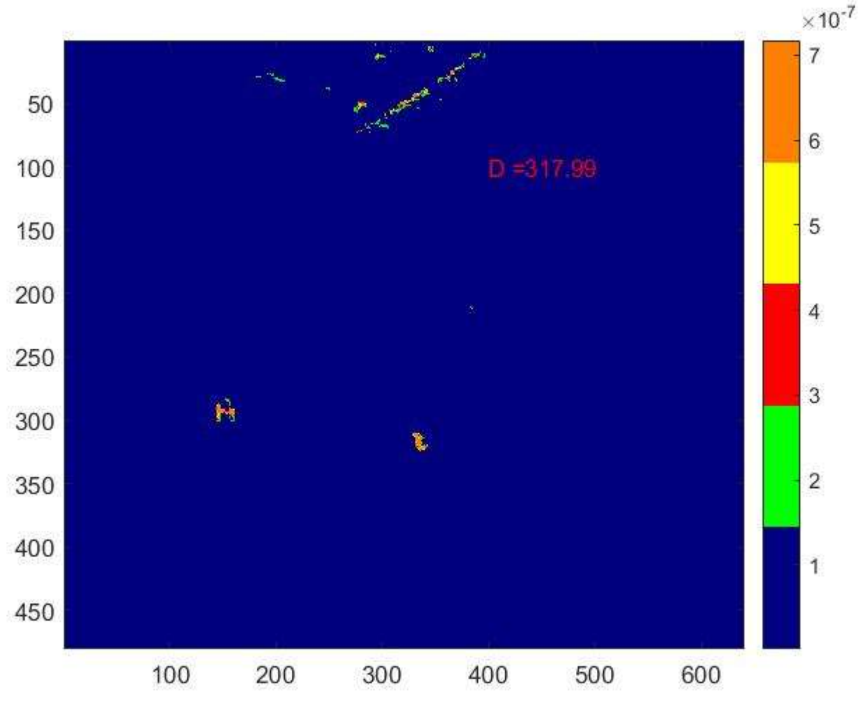
Figure: Distance vs Area Curve

Distance Estimate And Cluster Segmentation Results

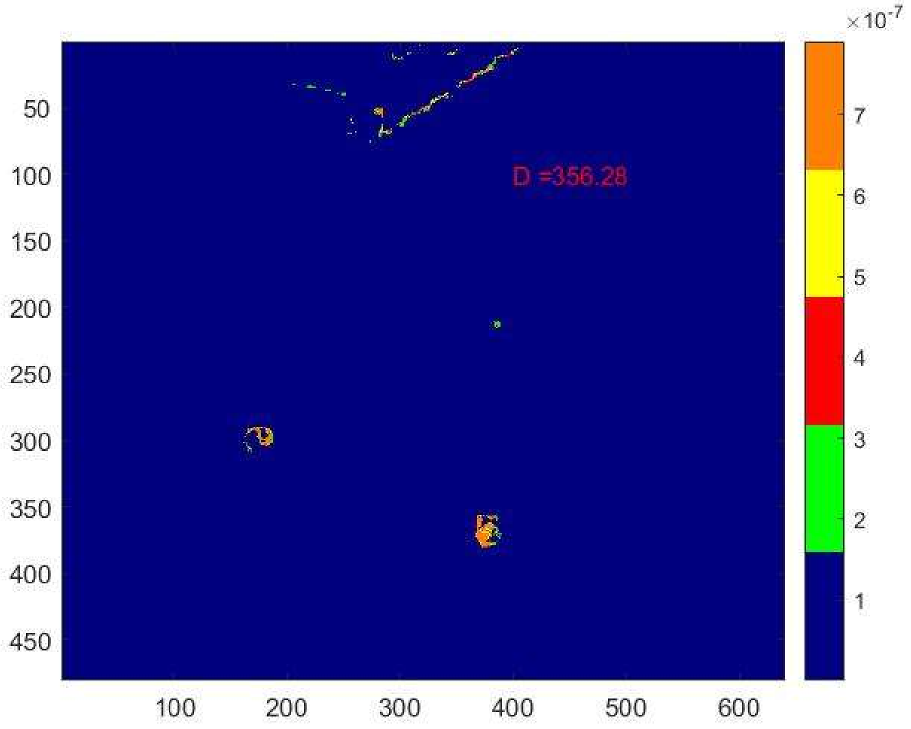
1. Test Image 1 - Both apple and ball was of color orange so we can observe the noise in the image below.



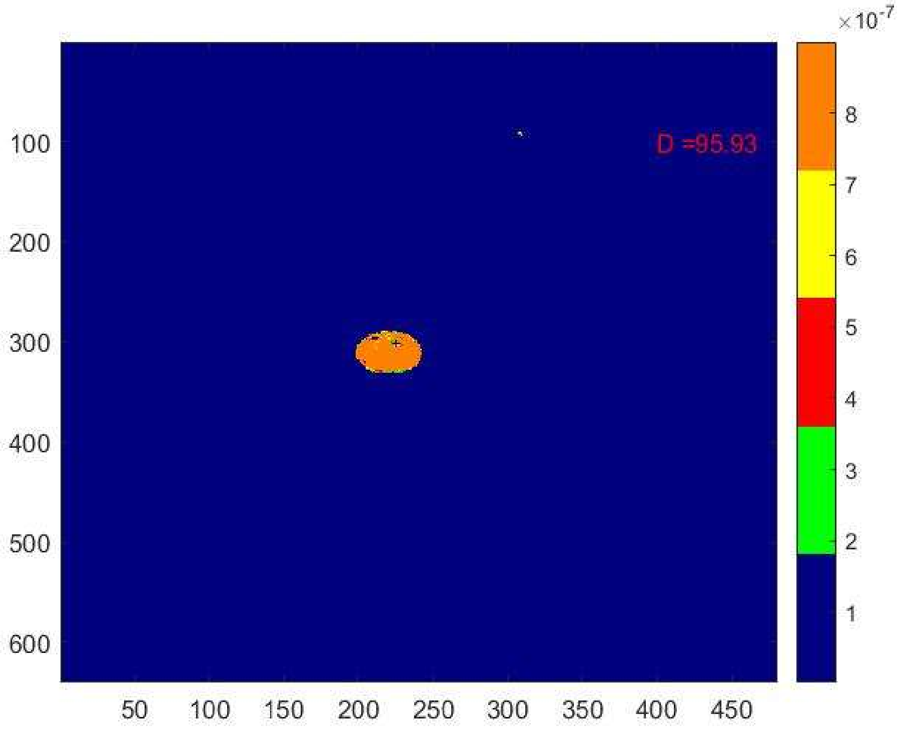
2. Test Image 2



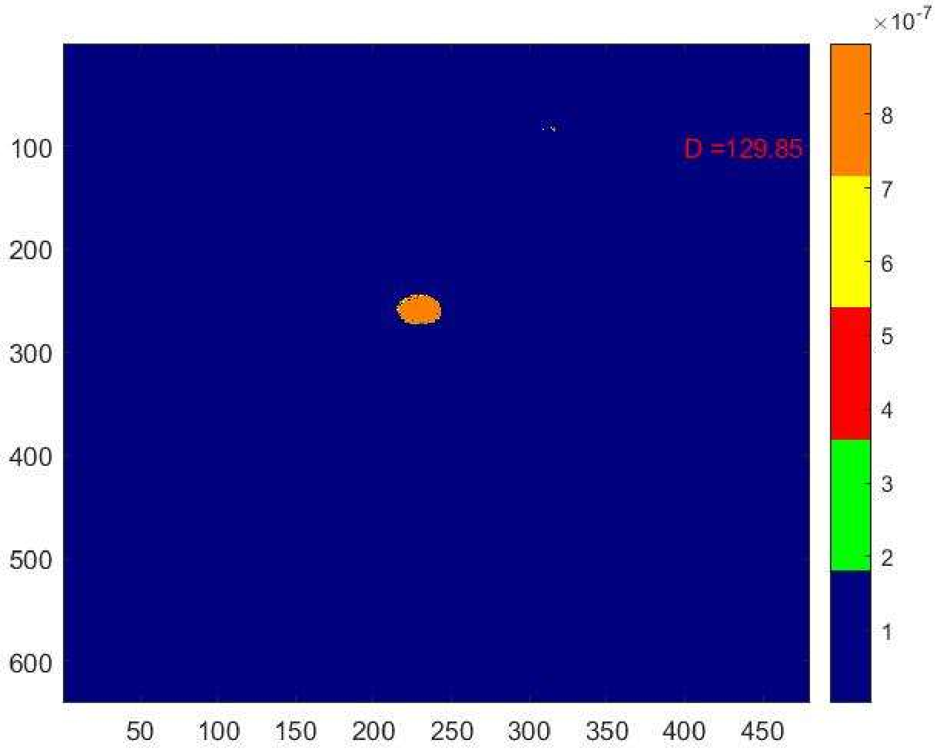
3. Test Image 3



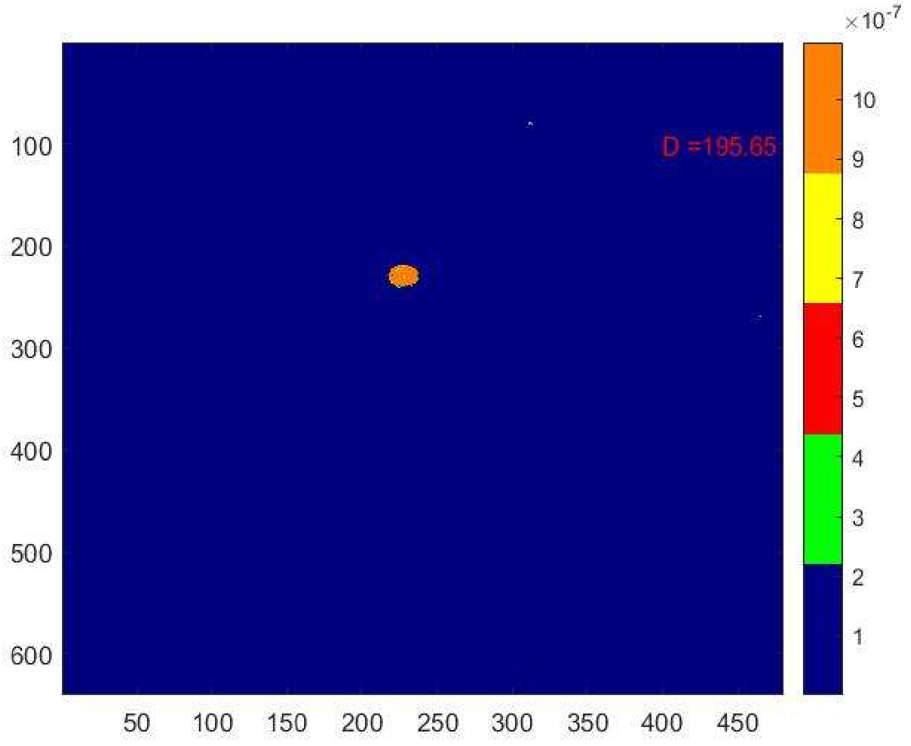
4. Test Image 4



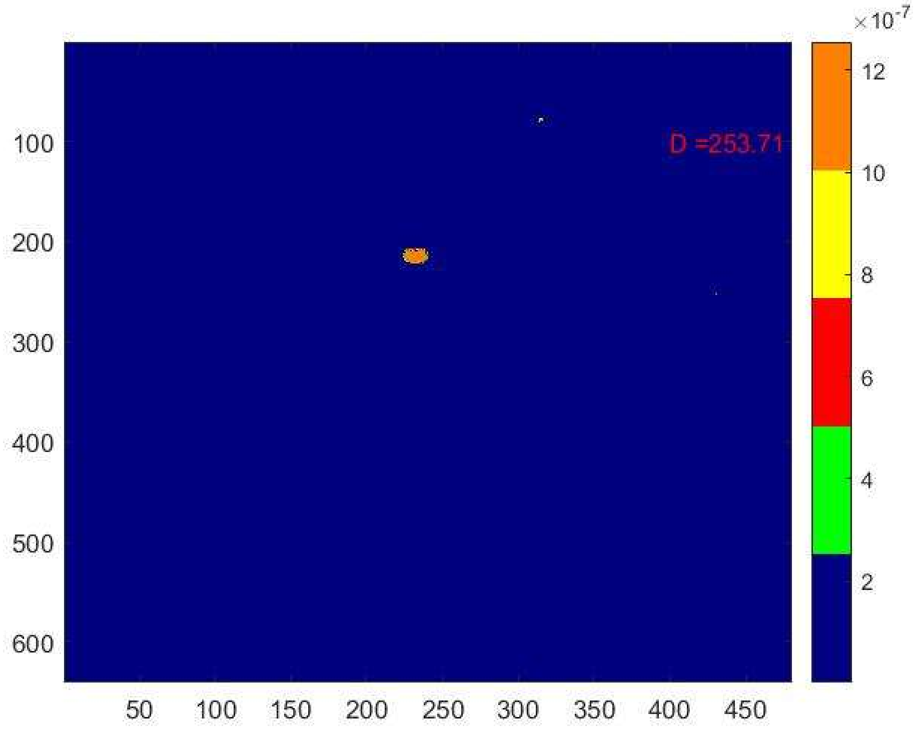
5. Test Image 5



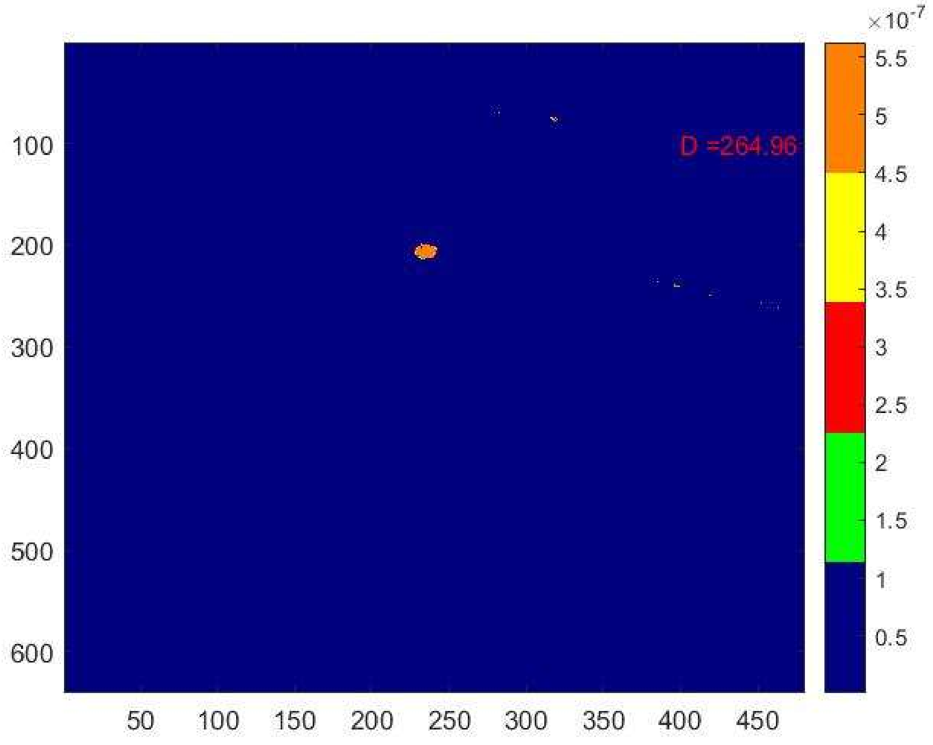
6. Test Image 6



7. Test Image 7



8. Test Image 8



Problems Faced

- Similar to that we faced in single gaussian, choosing the right threshold was a **major** problem.
- We got NaN values for the likelihood ($p(x_j|C_i)$) while trying to compute the cluster weight.
- Our plots which showed the clustering of the orange ball was performing well, while our distance estimates were off by a large margin.

How We Solved The Problem

- We set the threshold as a function of the maximum value of an *orange* pixel.
- We imposed a condition while initialization that the covariance matrix should be symmetric.
- For the last problem, we realized that we were using two different thresholds for the two tasks. We then set a global threshold which improved our distance estimates.

Limitations

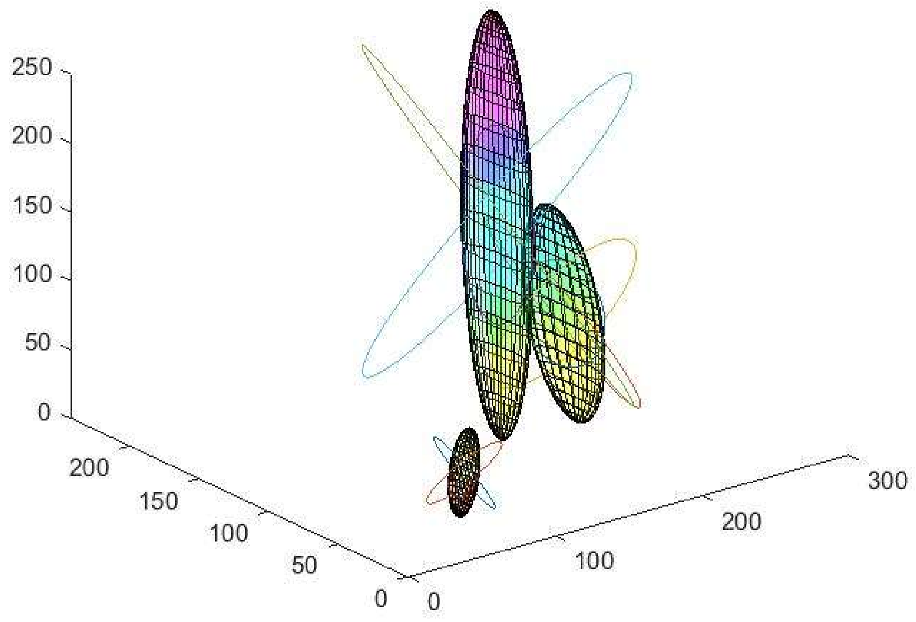
- Human error induced while manually clicking on the outline of the ball while using roipoly().
- Algorithm performance heavily depends on the threshold chosen by the user.
- The model failed on the test images which contained an apple along with the orange ball that was supposed to be identified. Since the RGB values of the apple and the orange ball were similar, GMM was not sensitive to it and classified the apple too as an orange ball.

Strengths

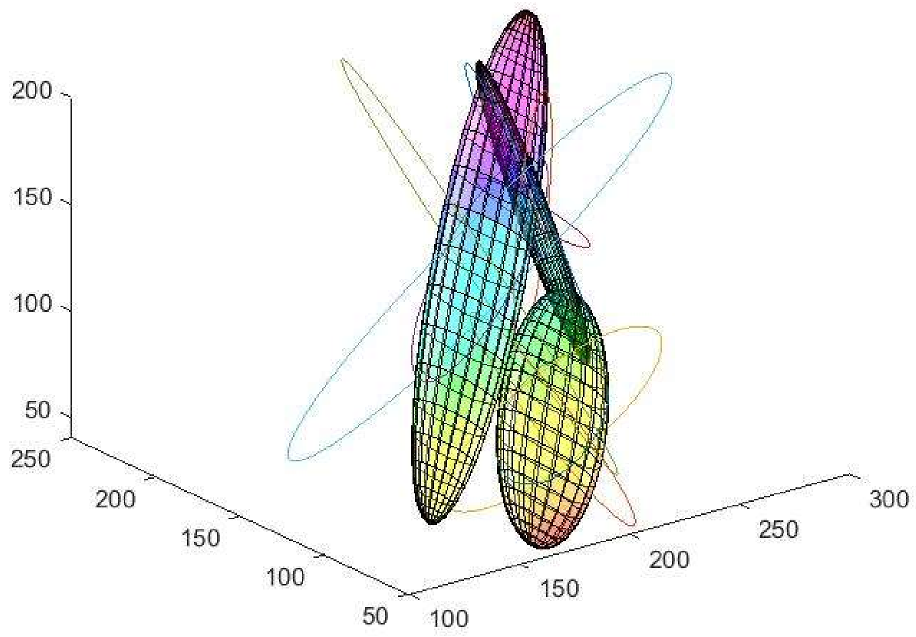
- Not very sensitive to lighting conditions.
- By using multiple gaussians to model an object (especially when there are features that are not observable) we can capture *all aspects* of the object.

Ellipsoid (used error_ellipse by AJ Johnson)

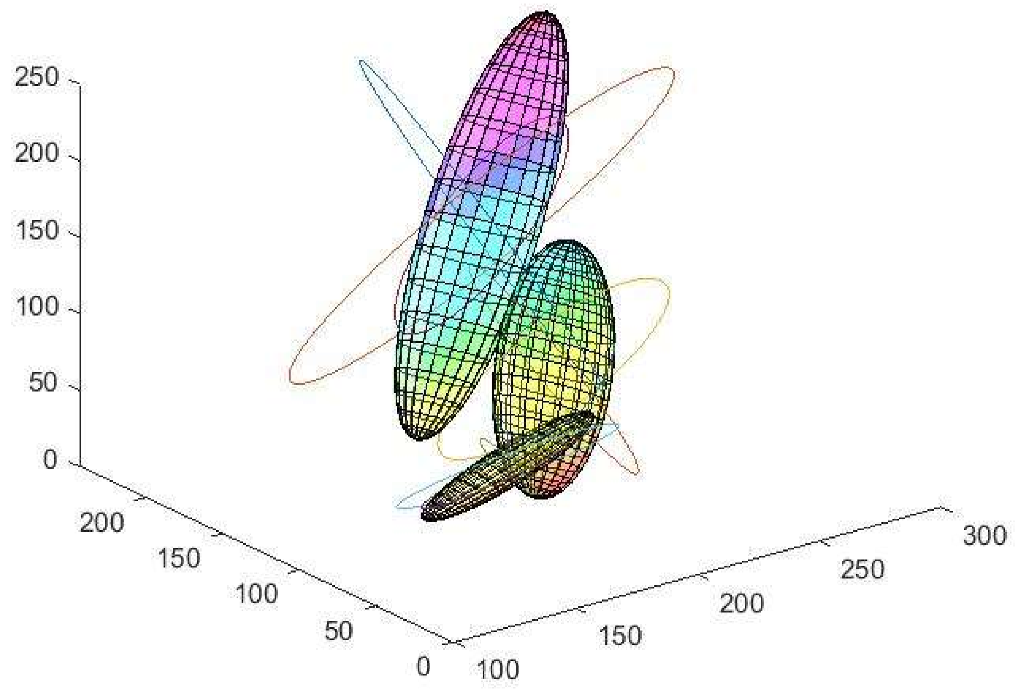
1. Test Image 1



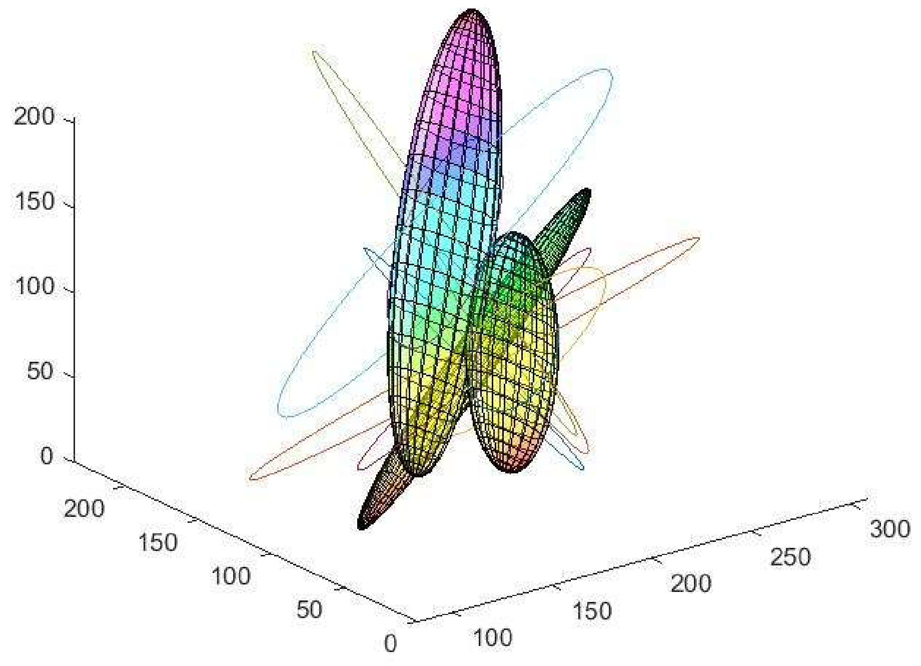
2. Test Image 2



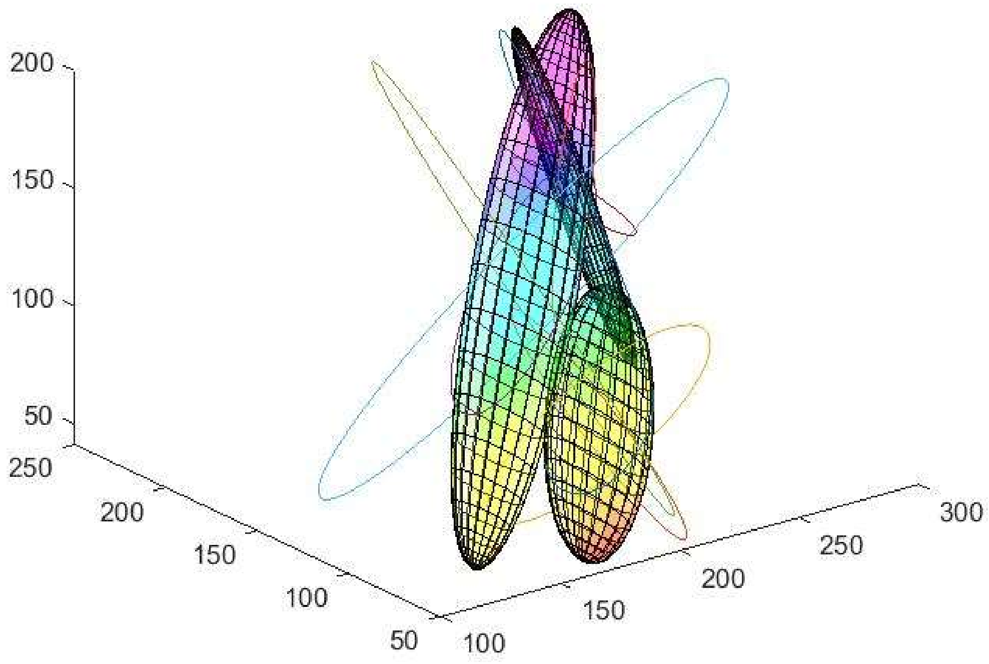
3. Test Image 3



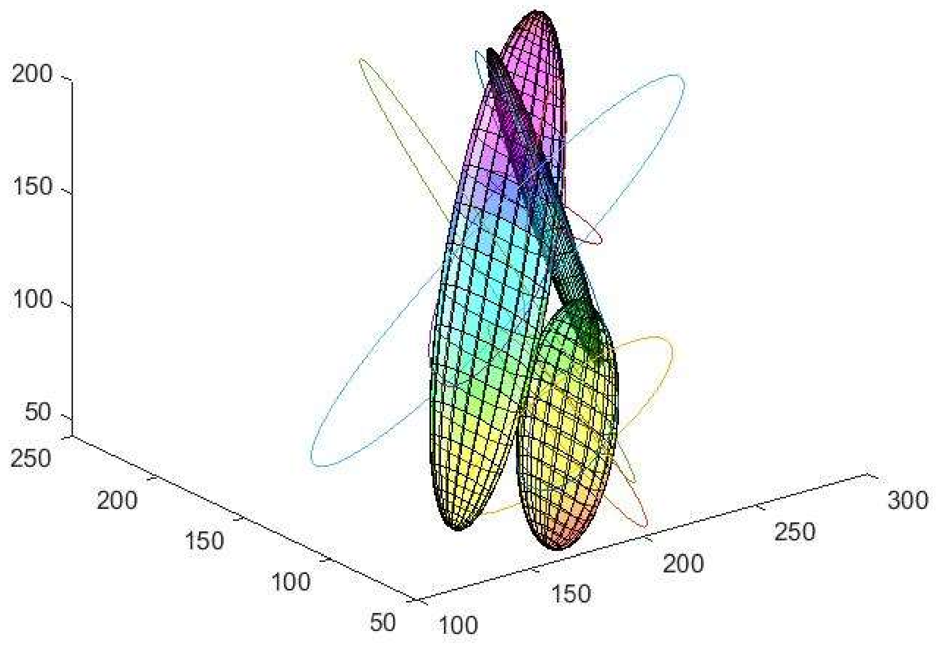
4. Test Image 4



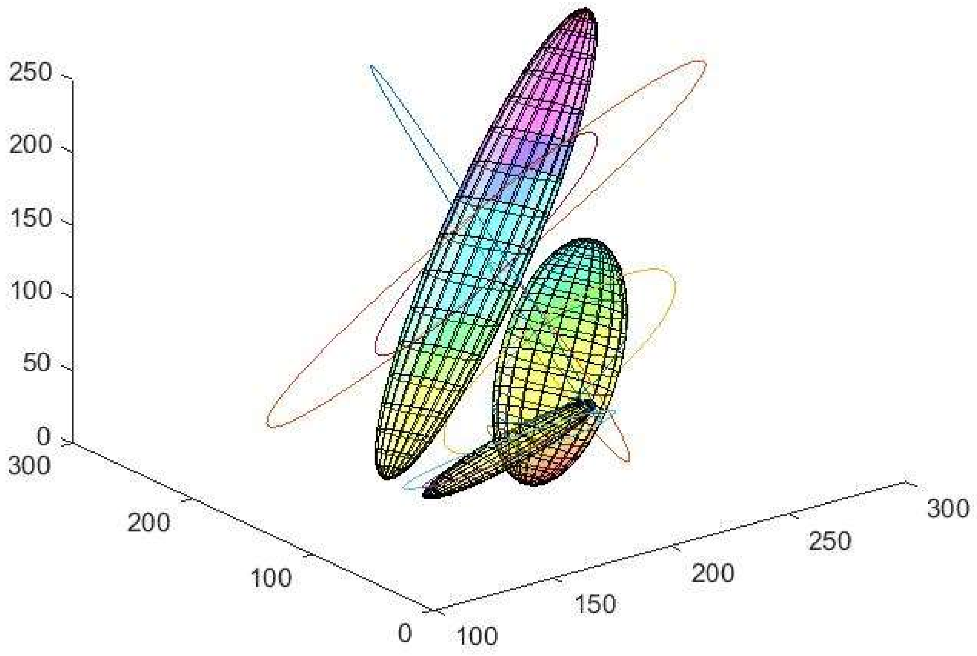
5. Test Image 5



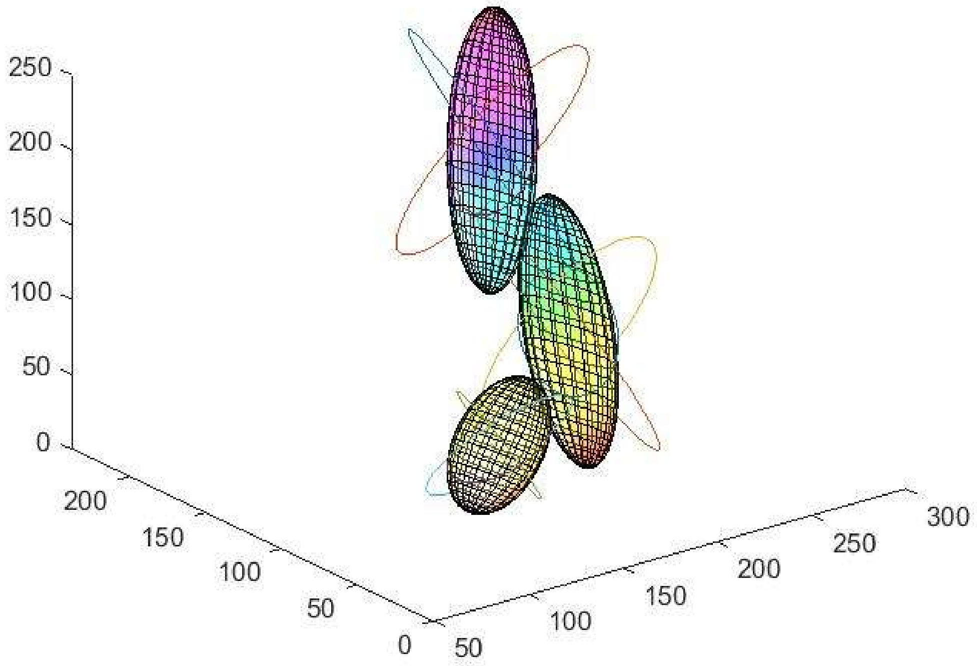
6. Test Image 6



7. Test Image 7



8. Test Image 8


250
200
150
100
50
0
200
150
100
50
0
50
100
150
200
250
300