Author: Name- Shivang Soni SID:915623718

1
Short answer problems [20 points]

1. When using the Hough Transform, we often discretized the parameter space to collect votes in an accumulator array. Alternatively, suppose we maintain a continuous vote space. Which grouping algorithm (among k-means, mean-shift, or graph-cuts) would be appropriate to recover the model parameter hypotheses from the continuous vote space? Briefly describe and explain.

Solution 1: Mean Shift

K Means: While using K means algorithm as we take the number of clusters and data points as our input and output will be the cluster number or the label, where those data points are bound to. But as in Hough transform we need to maximize the votes for a particular window in order for algorithm to detect peaks. So K means is not suitable for Hough Transform.

Mean Shift: This algorithm takes data points as input and will keep on shifting its window position until it does not find maximum element in its window. So as this algorithm converges till the center of mass of all data points. And for continuous vote space in Hough transform as this algorithm converges at its center of mass so this will be suitable.

Graph Cuts: As it takes feature space as input and will break the weakest edges connecting the two feature spaces. This does not work on accumulating votes so will not be a good choice.

2. Consider Figure 2 below. Each small dot denotes an edge point extracted from an image. Say we are going to use k-means to cluster these points' positions into k=2 groups. That is, we will run k-means where the feature inputs are the (x,y) coordinates of all the small dots. What is a likely clustering assignment that would result? Briefly explain your answer.
Solution 2:

K means algorithm will calculates the center of the cluster by analyzing the data points provided. In this case as the data points provided are in spherical shape and as K means does not divide these data points into two spherical cluster this leads to be a flaw in K means algorithm. The given dataset will be divided into two half circles as there clusters and have there cluster centers lying between the inner and the outer circle.

Author: Name- Shivang Soni SID:915623718

3. Suppose we have run the connected components algorithm on a binary image, and now have access to the multiple foreground 'blobs' within it. Write pseudo code showing how to group the blobs according to the similarity of their area (# of pixels) and aspect ratio (width/height), into some specified number of groups.
Define clearly any variables you introduce.
Sol 3:

```
function group_number = to_group_blobs_into_specific_group()
 group_number = 0
 for each blob  in the binary image:
     a = find(aspect_Ratio(blob))
     b = find(area(blob))
     if(blob.containedInAnyGroup() == null)
       group_number + = 1;
       createGroup(group_number)
       groupadd(group_number,blob)%add blob to that group
       for each blob1 in the binary image other than blob in consideration:
          c = find(aspect_Ratio(blob1))
          d = find(area(blob1))
          if(|a – c|<threshold1 && |b – d| < threshold2 && blob1.containedInAnyGroup() == null)
            groupadd(group_number,blob1); %add blob1 to blobs group
           end
        end
     else
        continue;
     end
 end
end
```

In the above Algorithm as we have access to foreground blobs we can group them together by running the above algorithm

We assume the terminology used in above algorithm like:
find(aspect_Ratio(blob)) = size(blob,2)/size(blob,1);
find(area(blob)) = #no. of pixel enclosed inside the blob.
blob.containedInAnyGroup() : check to see if that blob is already been group
createGroup : To create new group or cluster.
Threshold1: a particular value by which aspect_ratio of two blobs may vary.
Threshold2: a particular value by which area of two blobs may vary.
group_number : Total number of groups created till end.

Author: Name- Shivang Soni SID:915623718

2
Programming [80 points]
1. Color quantization with k-means [40 points]
For this problem you will write code to quantize a color space by applying k-means clustering to the pixels in a given input image, and experiment with two different color spaces—RGB and HSV. Write Matlab functions as defined below. Save each function in a file called <function-name>.m and submit all of them.

(a) [5 points] Given an RGB image, quantize the 3-dimensional RGB space, and map each pixel in the input image to its nearest k-means center. That is, replace the RGB value at each pixel with its nearest cluster's average RGB value. Use the following form:Function [outputImg, meanColors] = quantize_RGB(origImg, k) where origImg and outputImg are MxNx3 matrices of type uint8, k specifies the number of colors to quantize to, and meanColors is a k x 3 array of the k centers. Matlab tip: if the variable origImg is a 3d matrix containing a color image with numpixels pixels, X = reshape(origImg, numpixels, 3); will yield a matrix with the RGB features as its rows.

Figure1 : Original Fish Image

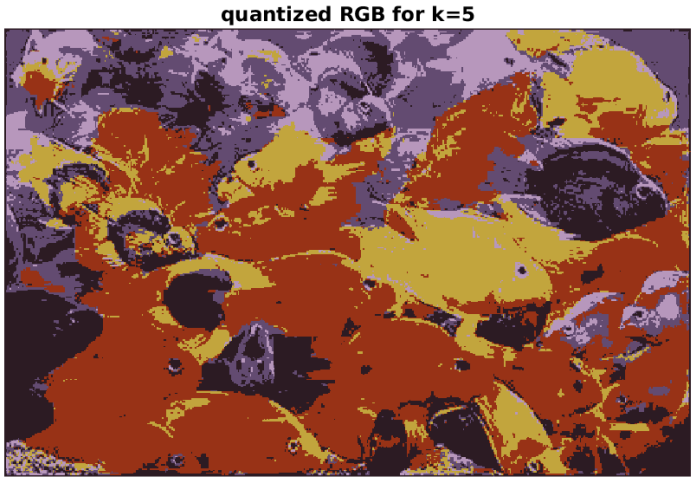


Figure2 : Quantized RGB K=5

Author: Name- Shivang Soni SID:915623718

Figure3 : Quantized RGB k=25

Author: Name- Shivang Soni SID:915623718

(b) [5 points] Given an RGB image, convert to HSV, and quantize the 1-dimensional Hue space. Map each pixel in the input image to its nearest quantized Hue value, while keeping its Saturation and Value channels the same as the input. Convert the quantized output back to RGB color space. Use the following form:
Function [outputImg, meanHues] = quantize_HSV(origImg, k)
where origImg and outputImg are MxNx3 matrices of type uint8, k specifies the number of clusters, and meanHues is a k x 1 vector of the hue centers.

Figure4 : Original Image

Author: Name- Shivang Soni SID:915623718

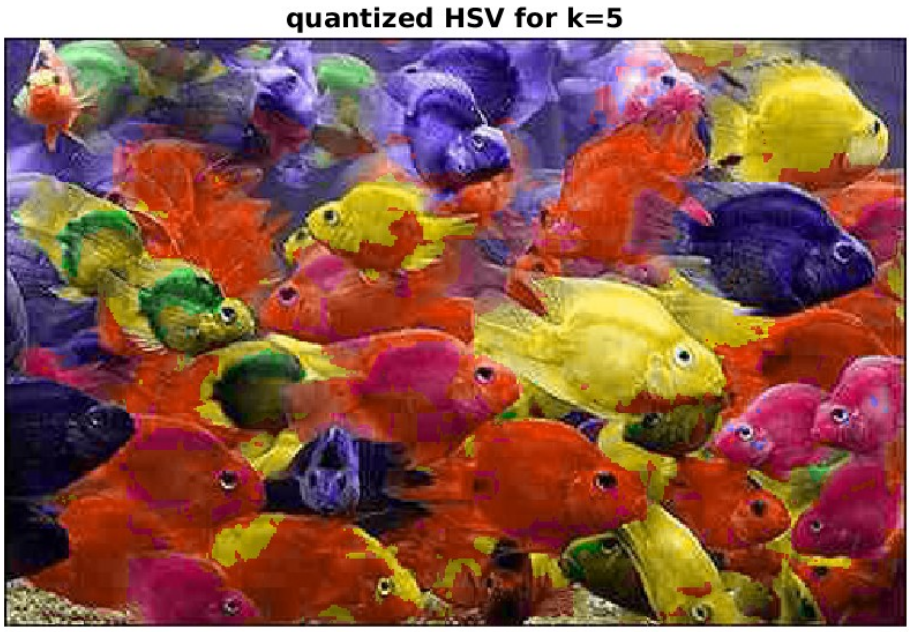


Figure5 : Quantized HSV image with k=5 clusters

Author: Name- Shivang Soni SID:915623718

Figure6 : Quantized HSV for k=25

(c) [5 points] Write a function to compute the SSD error (sum of squared distances error) between the original RGB pixel values and the quantized values, with the following form:
function [error] = compute_quantization_error(origImg,quantizedImg)
where origImg and quantizedImg are both RGB images, and error is a scalar giving the total SSD error across all pixels in the image.

Error of quantization with k=25 clusters **2.0606e+03**
Error of quantization with k=5 clusters **7.1091e+03**

(d) [5 points] Given an image, compute and display two histograms of its hue values. Let the first histogram use k equally-spaced bins (uniformly dividing up the hue values), and let the second histogram use bins defined by the k cluster center memberships (i.e., all pixels belonging to hue cluster i go to the i-th bin, for i=1,...k). Use the following form:
function [histEqual, histClustered] = get_hue_hists(im, k) where im is an MxNx3 matrix representing an RGB image, and histEqual and histClustered are the two output histograms.

Author: Name- Shivang Soni SID:915623718

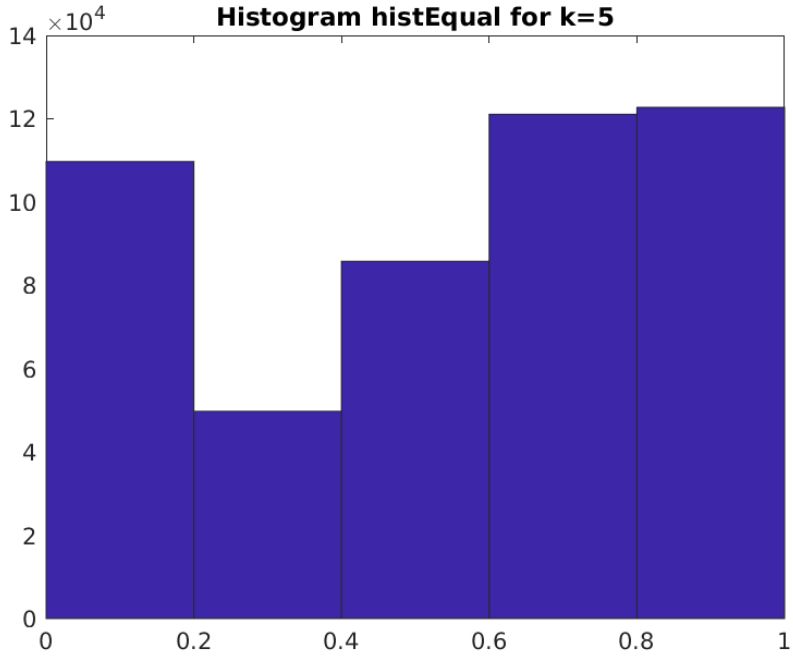


Figure7 : Histogram Hist Equal For k=5 clusters

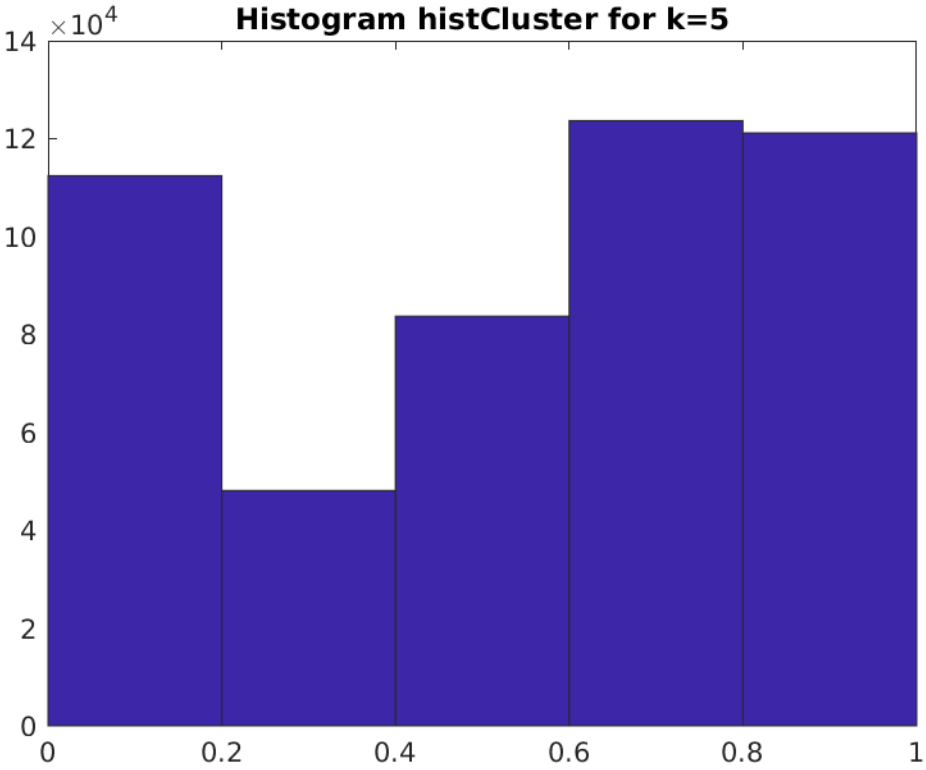


Figure8 : Histogram HistClustered For K=5 clusters

Author: Name- Shivang Soni SID:915623718

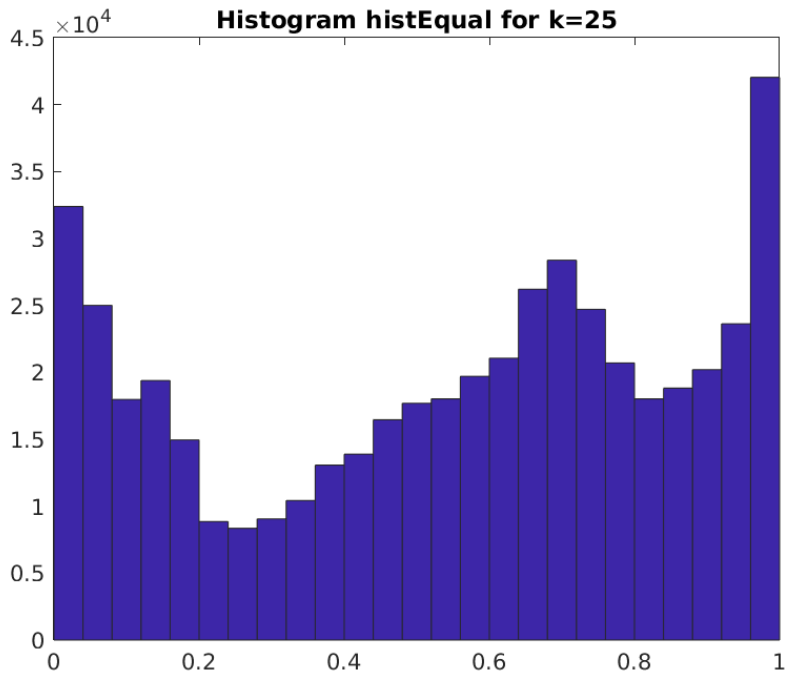


Figure9 : Hist Equal for k=25

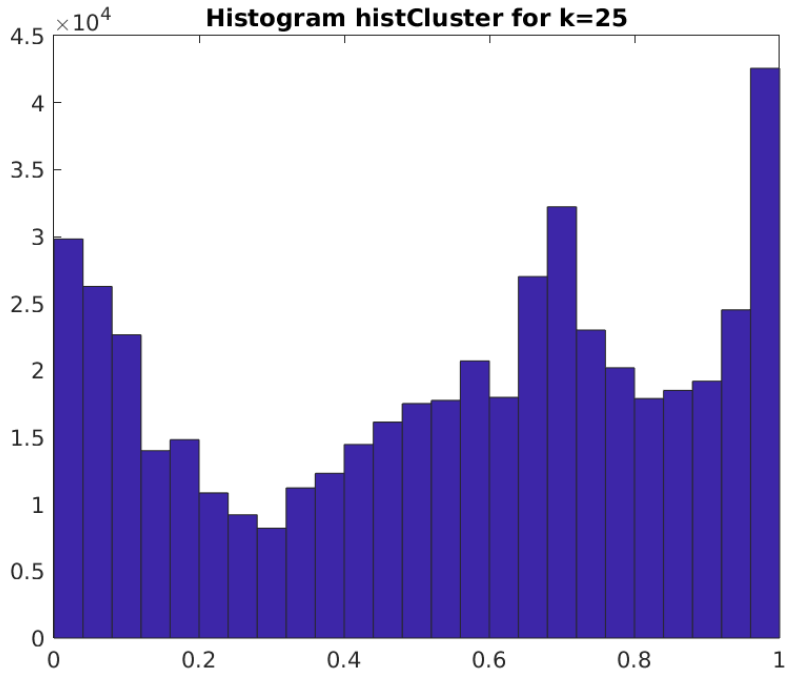


Figure9 : Hist Clustered for k=25

Author: Name- Shivang Soni SID:915623718

(e) [5 points] Write a script color_quantize_main.m that calls all the above functions appropriately using the provided image fish.jpg, and displays the results. Calculate the SSD error for the image quantized in both RGB and HSV space. Write down the SSD errors in your answer sheet. Illustrate the quantization with a lower (k=5) and higher (k=25) value of k. Be sure to convert an HSV image back to RGB before displaying with imshow. Label all plots clearly with titles.

Quantizing RGB:
Error of quantization with k=25 clusters **2.0606e+03**
Error of quantization with k=5 clusters **7.1091e+03**

Quantizing HSV:(Hue)
Error of quantization with k=25 clusters **58.7028**
Error of quantization with k=5 clusters **1.0876e+03**

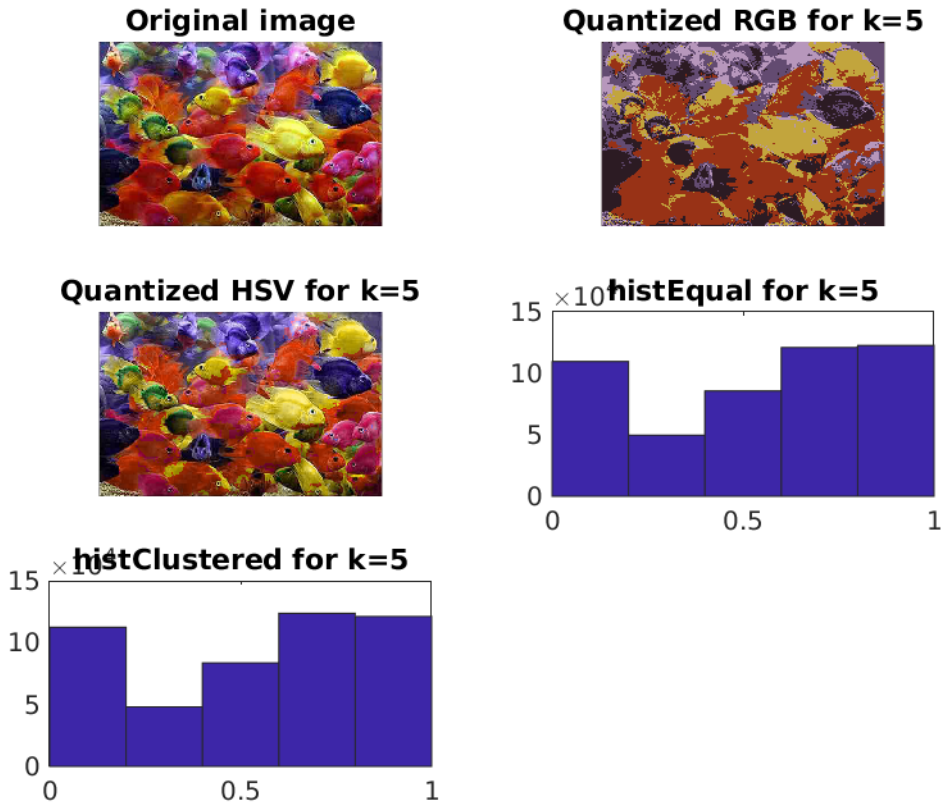


Figure10 : Subplot For K=5

Author: Name- Shivang Soni SID:915623718

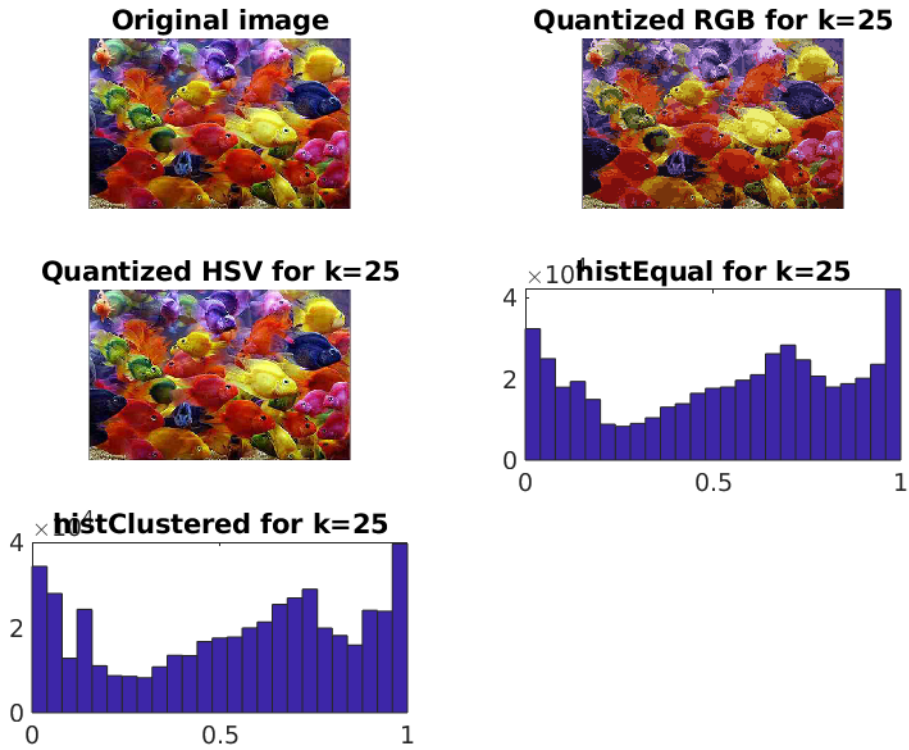


Figure11 : Subplots for k=25

Author: Name- Shivang Soni SID:915623718

(f) [15 points] In your write-up, explain all the results. How do the two forms of histogram differ? How and why do results vary depending on the color space? The value of k? Across different runs of k-means (with the same value of k)?

K means algorithm working:
Firstly based on the k value inputted from the user k random points will be selected by the k means algorithm and those are considered as the center . Then for all the rest of the incoming points we will find the euclidean distance to the cluster center and add that particular point to the given cluster by updating the centroid of the given cluster. This process we repeat for all the points in the provided matrix and up to x iterations (here x denotes the point till we get convergence).

1)**Analysis of Figure 10 and 11 based on k values:**
As the points are divided into k clusters so we can only see k number of cluster on the screen in which all the intensity of the image is classified into. This is for the small k value whereas if we keep on increasing the k value we can able to see more or like the same image as the input image as the max number of cluster an image can have is the number of different intensities present for that particular image.
So when the k value is small the points our forced out to be in k clusters so we obtained destroted image as compared to the original image but as we keep on increasing the k value we will come closer to get the image which looks almost same as the original image.
2.)**Color quantization:**
As the image consist of 2^8 different values of one pixel (In HSV suppose for Hue) and as by the color quantization as the image to be forced to decrease from this wide range of value to n(bin value) so this caused a good amount of error on analyzing the SSD error produced above as for image fro which we only changed Hue pixel intensity based on the n cluster there is very less variation in error produced but whereas for RGB as we changed the pixel intensity for R,G and B we get high error value due to that change.
3.) **Two forms of histogram differ**
As the hist Equal bins denote the just separating the output obtained from the k means into k different bins. And hist Clustered specifies to form bins based on the k clusters. So in Figure 10 the 5 bins denote the points having same intensity between 0 to 1/(number of bins) are clubbed together where as in case of k means clustering as the points are rounded out to the nearest cluster center so there is a little bit of variation between Hist Clustered output and Hist Equal output.

Author: Name- Shivang Soni SID:915623718

2. Circle detection with the Hough Transform [40 points]
Implement a Hough Transform circle detector that takes an input image and a fixed radius, and returns the centers of any detected circles of about that size. Include a function with the following form:

Function [centers] = detect_circles(im, radius, useGradient)
where im is the input image, radius specifies the size of circle we are looking for, and useGradient is a flag that allows the user to optionally exploit the gradient direction measured at the edge points. The
output centers is an N x 2 matrix in which each row lists the (x,y) position of a detected circle's center. Save this function in a file called detect_circles.m and submit it.
Then experiment with the basic framework, and in your write-up analyze the following:
(a) [10 points] Explain your implementation in concise steps (English, not code).

For each pixel in the image find the edge map by using canny edge detector and by setting some particular threshold to it. This will detect all the edges which lies above the threshold and color those pixels white.

Now to detect the circles we need to move from our image space to the Hough Space.Hough Space is a place where we will accumulate votes from the nearby pixel to find where exactly the circle center is:

Moving from X,Y plane i.e. Cartesian Coordinate system to Hough Space Coordinate System where a,b denotes the axis as well as X,Y are constant

$(x-a)^2 + (y-b)^2 = r^2$
To compute value of a,b we need the value of 3 variables x,y,r;

So Let H(a,b) is the accumulator array defined to accumulate all votes from the nearby detected edges.

1.)We move through all the pixels detected as an edge by canny edge detector:
2.)For each pixels detected with the fixed radius we compute the votes accumulated by those at every point in Hough Space. This can be computed by changing X,Y coordinates to polar form and compute a,b by trying different theta values.
3.) Then we sort out the Hough Space in descending order and take the first point which are detected as circle center.

Returned the obtained center for a circle with fixed radius.

Author: Name- Shivang Soni SID:915623718

(b) [10 points] Demonstrate the function applied to the provided images jupiter.jpg and egg.jpg (and an image of your choosing if you like). Display the accumulator arrays obtained by setting useGradient to 0 and 1. In each case, display the images with detected circle(s), labeling the figure with the radius. You can use impixelinfo to estimate the radius of interest manually.

Figure12 : Original Egg Image

For use Gradient = 0 for egg

Author: Name- Shivang Soni SID:915623718

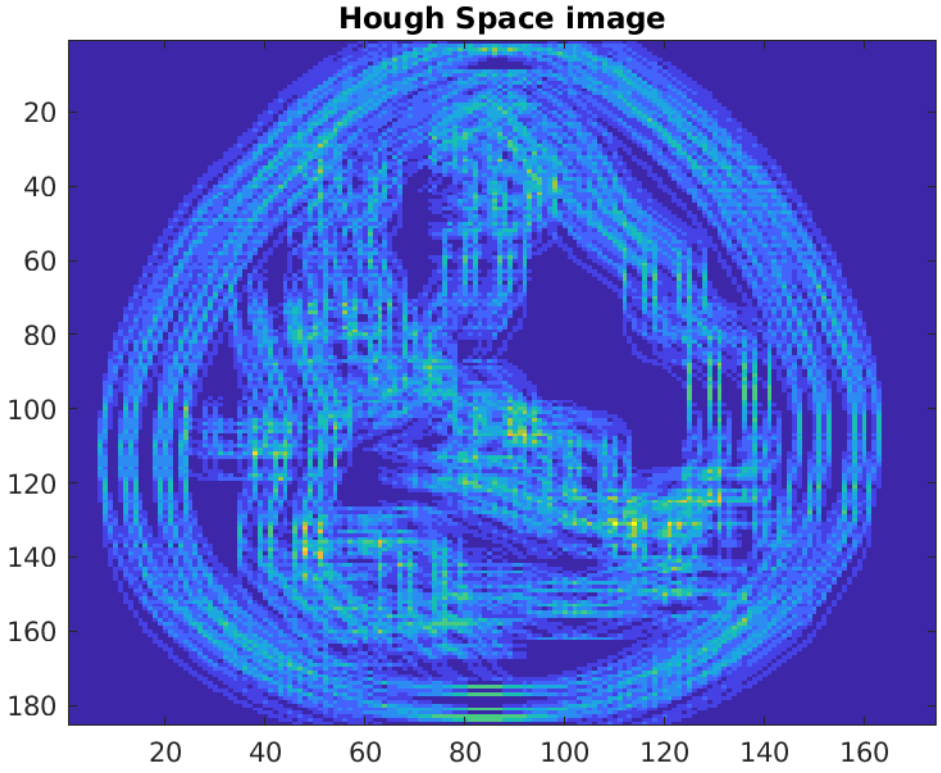


Figure13 : Hough Space Image where circle radius will be at high intensity value

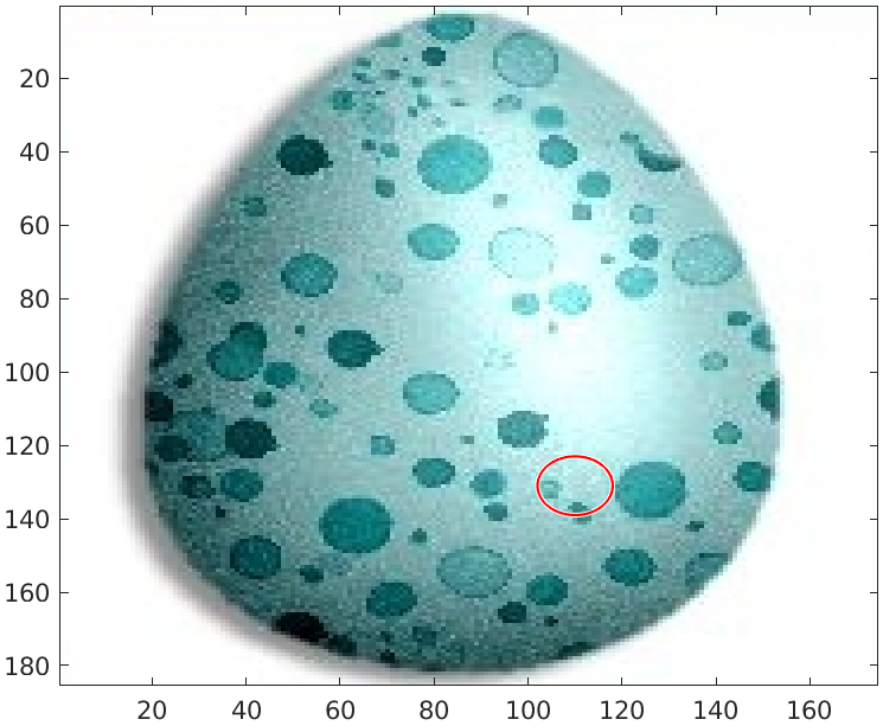


Figure14 : Circle Detected At high Intensity Value

Author: Name- Shivang Soni SID:915623718

useGradient = 1

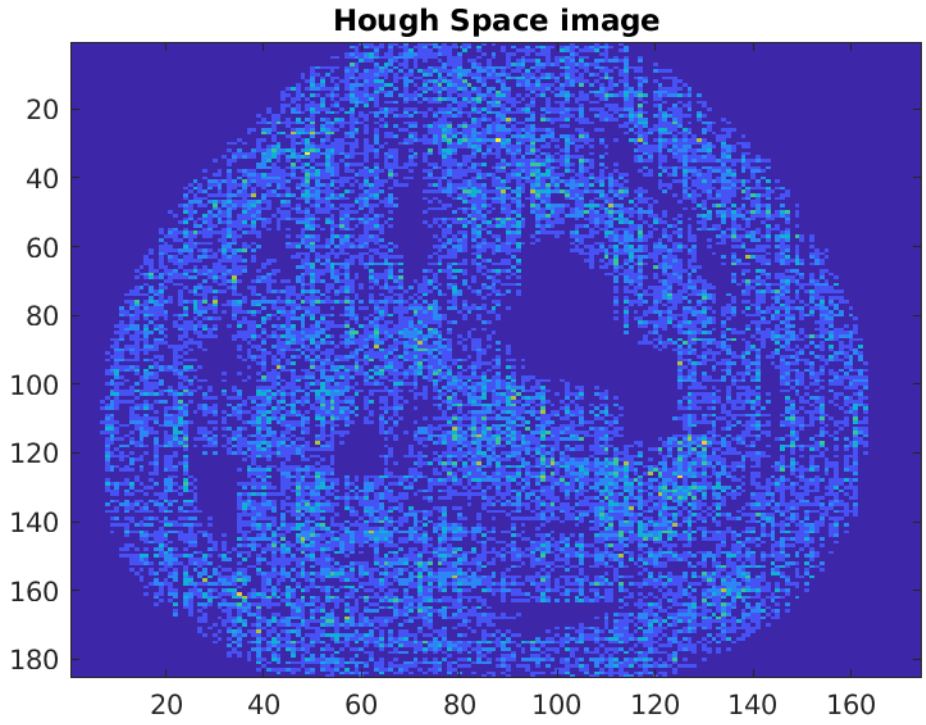


Figure15 : Image When Use Gradient = 1

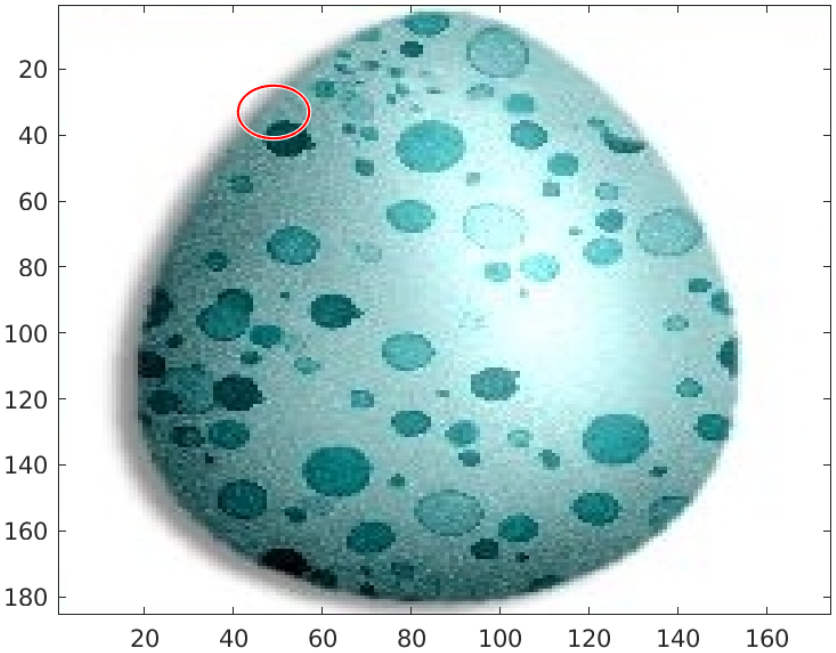


Figure16 : Detected Circle For Use Gradient = 1

Author: Name- Shivang Soni SID:915623718

For Jupiter:

For useGradient = 0 Radius = 30

Figure17 : Original Image of Jupiter

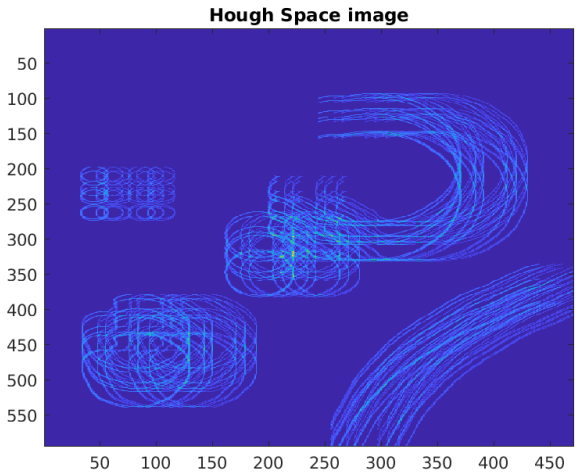


Figure18 : Hough Space image useGradient = 0 Radius - 30

Author: Name- Shivang Soni SID:915623718

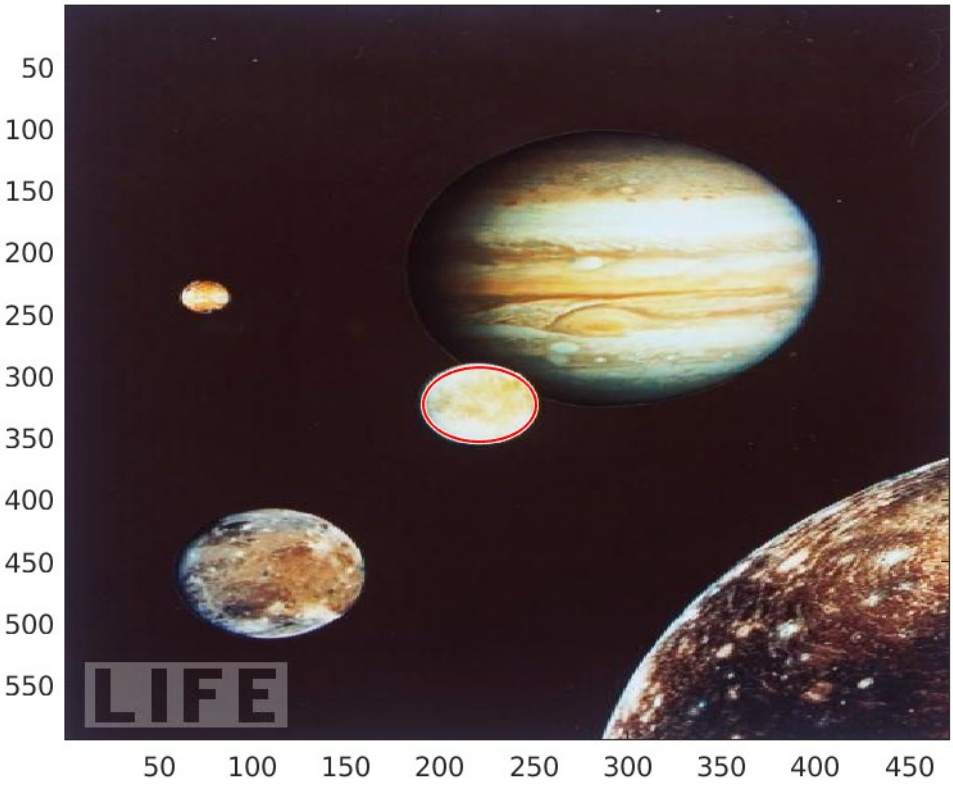


Figure19 : Detected Circle For Radius = 30 by useGradient = 0

useGradient = 1
Radius = 30

Author: Name- Shivang Soni SID:915623718

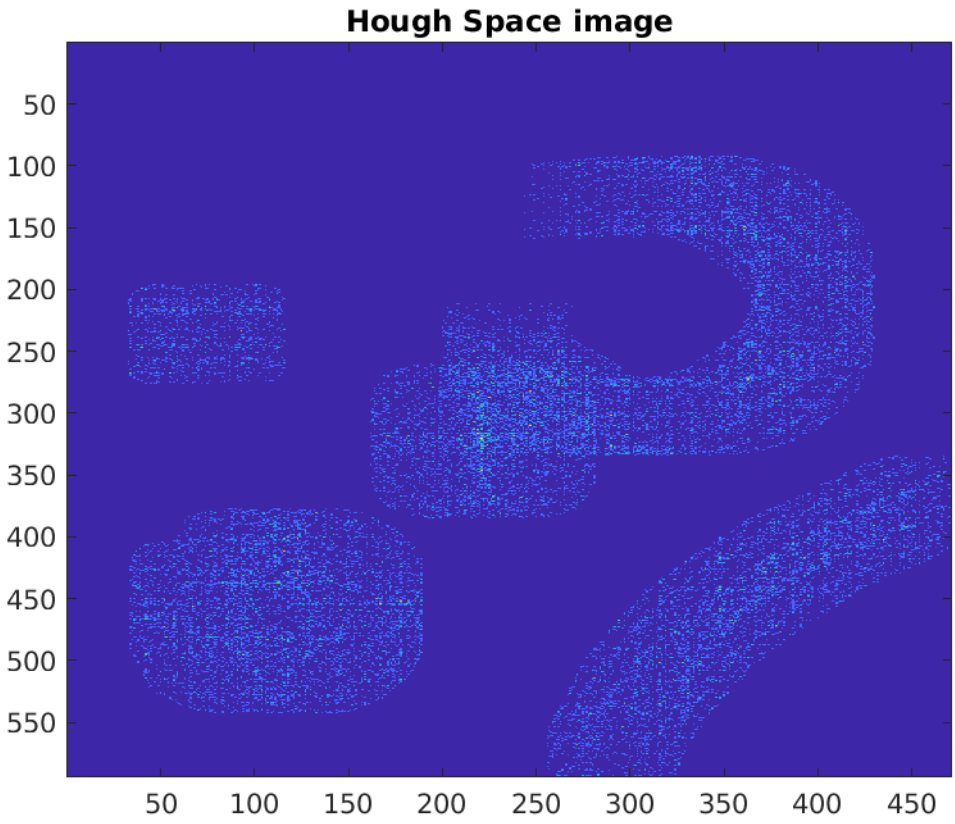


Figure20 : Hough Space image for R=30 & useGradient=1

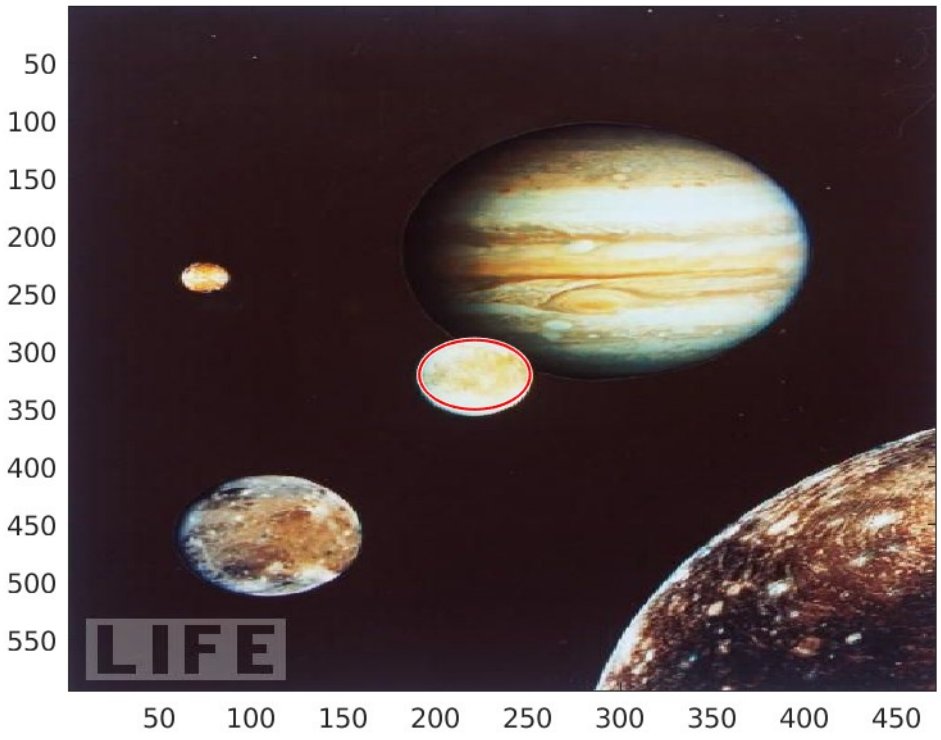
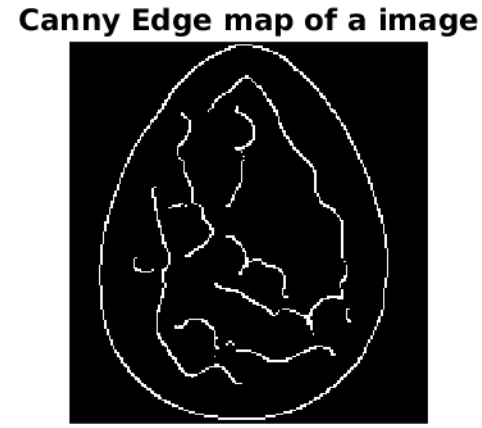


Figure21: Circle Detected using UseGradient = 1

Here above in the hough space images the points which high center intensity are darker in color as compared to other points. And the brightest point among those will be picked up for center.

c) [10 points] For one of the images, display and briefly comment on the appearance of the Hough space accumulator array.
Solution.)
If we take above figures 17,20,21(useGradient = 1) into consideration we can tell as the Algorithm to detect circle runs for radius = 30 so the accumulator array will be formed by taking into consideration the radius of the given circle so the votes in the accumulated array will be casted based on that as the circle with center (x,y) and radius = 30 will accumulate the most votes as there is a strong edge detected at the circumference of a circle which will in turn result in detection of the circle. Below are the edge maps for 'egg.jpg' and 'jupiter.jpg'



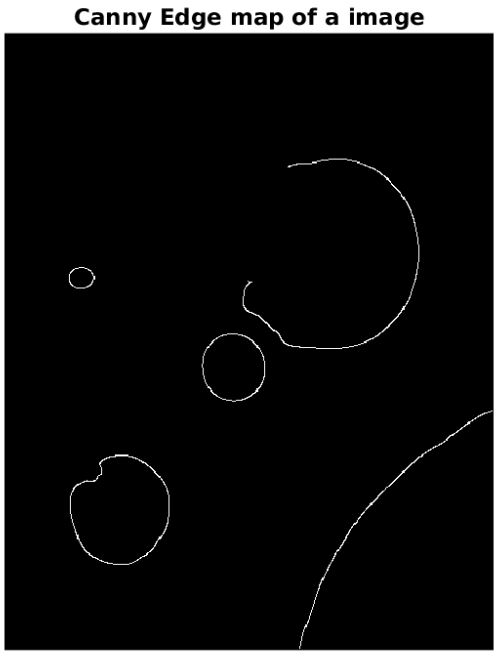


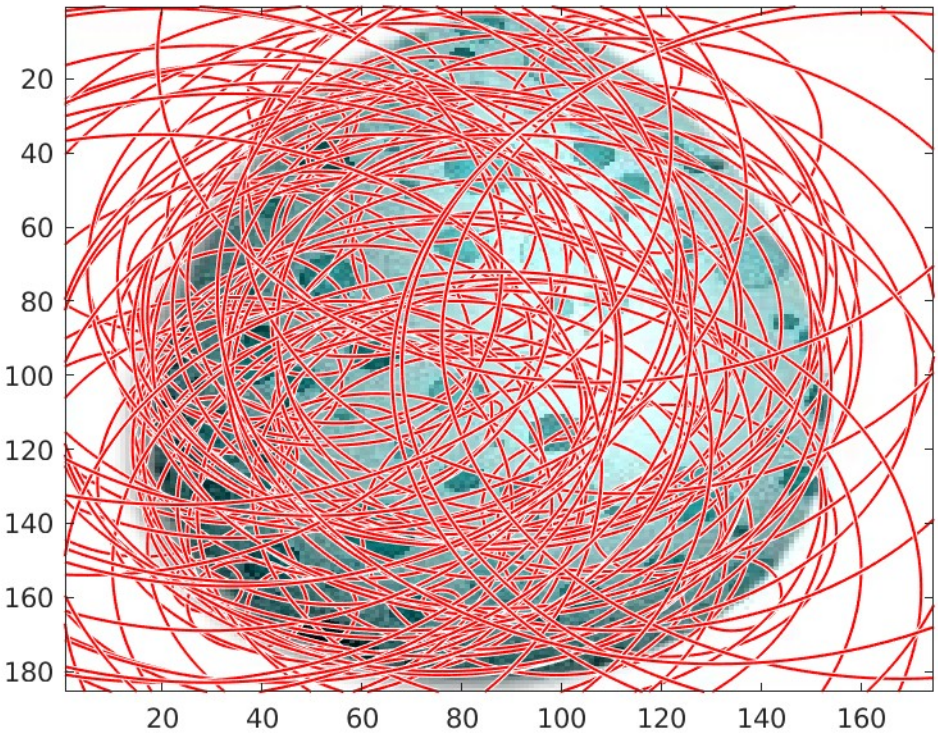
Figure23 : Edge Map For Jupiter Image

Author: Name- Shivang Soni SID:915623718

(d) [5 points] Experiment with ways to determine how many circles are present by post-processing the accumulator array.
By using find() command in Matlab and we can setup the threshold to detect that all the discretized bins having votes higher than threshold set which will be considered for a specific radius.
So for the circle with radius 110 only the bigger 1 circle is detected in the Jupiter image. So we can calculate the circles present in the accumulator array of same radius just by outputting the accumulator array..

(e) [5 points] For one of the images, demonstrate the impact of the vote space quantization (bin size).
Solution)
Less will be the bin size less accurate the result will be as I am considering bin size as 1 so the detected circles are not the exact match sometimes to the exactly matching circle. This might be caused due presence of some noise in the image

Extra Credit:
Extend your Hough circle detector implementation to detect circles of any radius. Demonstrate the method applied to the test images.
Script name: ExtraCredit.m
Testing Script: ExtraCreditTest.m



. *Figure 24: Egg detecting circles upto 100*

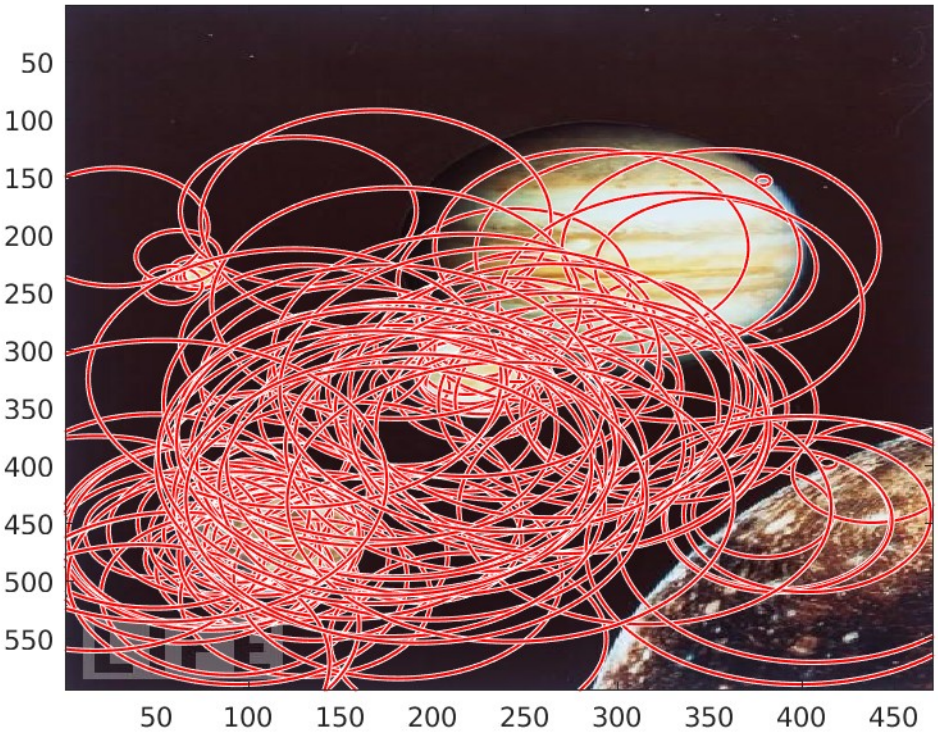


Figure25 : Jupiter Detecting circle upto 125 radius