ECE-5554 Computer Vision: Problem Set 2

Murat Ambarkutuk
murata@vt.edu

Mechanical Engineering Department,
Virginia Polytechnic Institute and State University

October 6, 2015

Answer Sheet

1 Short Answer Problems

1. Since filter bank contains rotated variants of the texture model, it is expected to get orientation insensitive response. In other words, we should be able to find the texture no matter its direction.

2. Since kmeans algorithm utilizes Euclidian distance as distinguishing criteria, it may lead unexpected clustering. If the given data set clustered by a group of people, the resulting would be the curves connecting the data points to circles (due to Gestalt Principles). On the other hand, the resulting clusters of kmeans clustering will have two half circles for each cluster, unlike the human clustering result.

3. kmeans: K-means algorithm is not suitable for the task, due to the fact that its inputs are data points and desired cluster number and outputs are the labeled set. However, for Hough transform the outcome should be the place where maximum vote (area, bin or point) is located.
   graph-cuts: Graph-cut algorithm takes the feature space as input and divides that feature space into 2 by trying to break the weakest bounds connecting features in the feature space.
   Mean-shift: This algorithm takes a data set as input and converges to the center of mass. This algorithm can be useful when employed to find the maxima of hough space.

4. Functions:

   -
     ```
     point findCenterOfMass(blob){
     –for each pixel in blob
     —-sumPos += position(pixel)
     –centerOfMass = sumPos/size(blob)
     – return centerOfMass
     }
     ```

   -
     ```
     double circularity(blob){
     –radiusMean = size(blob)
     –for each pixel in boundary(blob)
     ```

```
—-sumSquaredDistance += (centerOfMass - position(pixel))$^2$
–circularity = sumSquaredDistance / size(boundry(blob))
–return circularity
}
```

```
• clusters kMeans(circularity[ ],k){
– return kmeans = circularity[ ]
}
```

Variables:

- sumPos: $\sum_{pixels} Position(Pixel)$
- centerOfMass: Center of mass point of the blob.
- sumSquaredDistance: $\sum_{pixels} Position(pixel) - Position(centerOfMass)$
- circularity: Radius invarient circularity feature. The feature is expected to invarient of radius of the the shape, because sum of the squared distance is normalized with the total number of the pixels bounding the blob.

Algorithm:

- Find the center of mass point for each blob.
- Extract radius invarient circularity feature.
- Cluster the blobs according to their circularity feature.
- This algorithm tries to explain blobs by assuming (fitting) circle around it. The lowest score means the best fit for circular shapes.

2 Programming

2.1 Color Quantization Results

2.1.1 RGB Space Quantization with K-Means

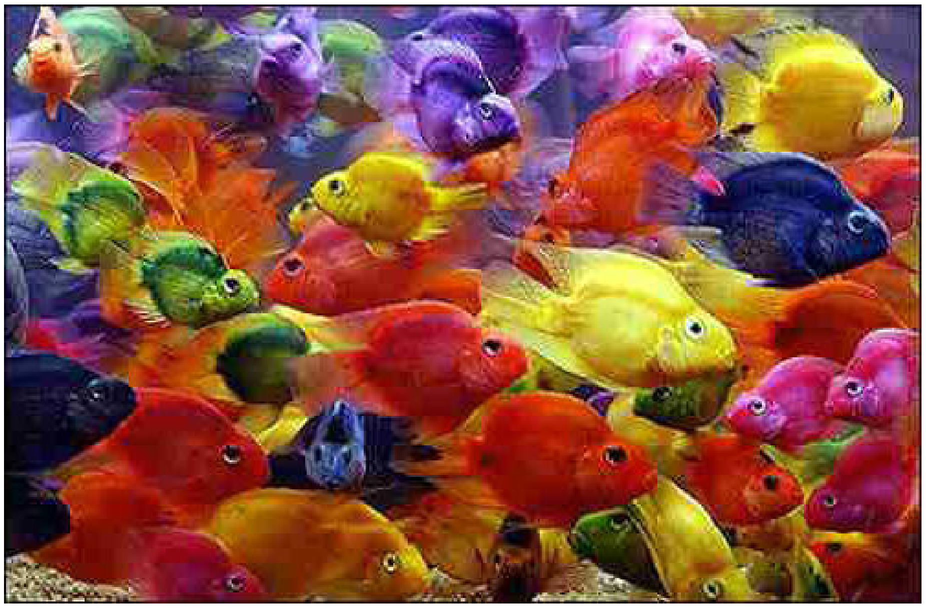

Figure 1: Input Image

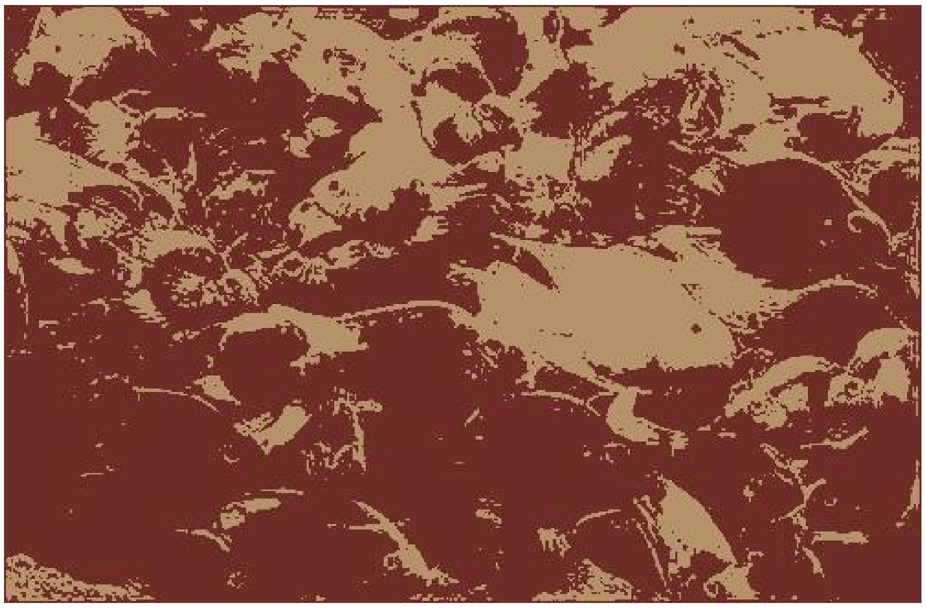

Figure 2: RGB Color Quantization Output Image (n=2)

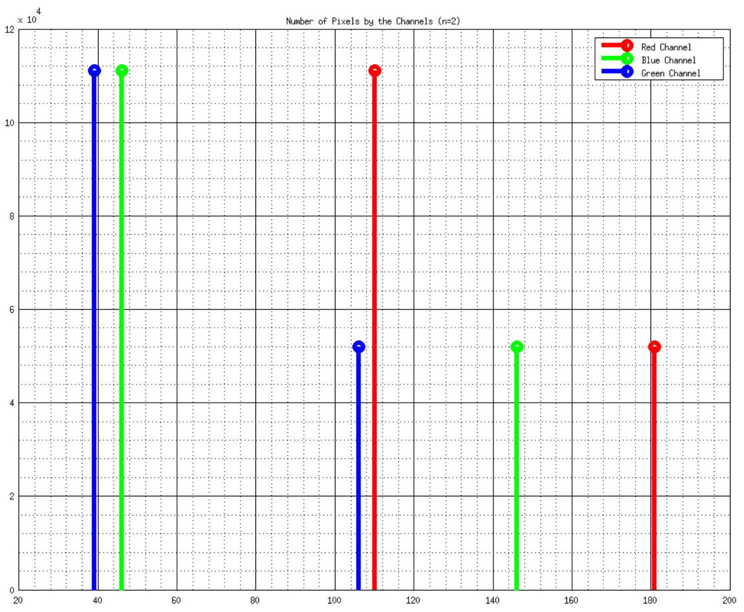

Figure 3: Histogram (n=2), Each color channel is represented by its own color

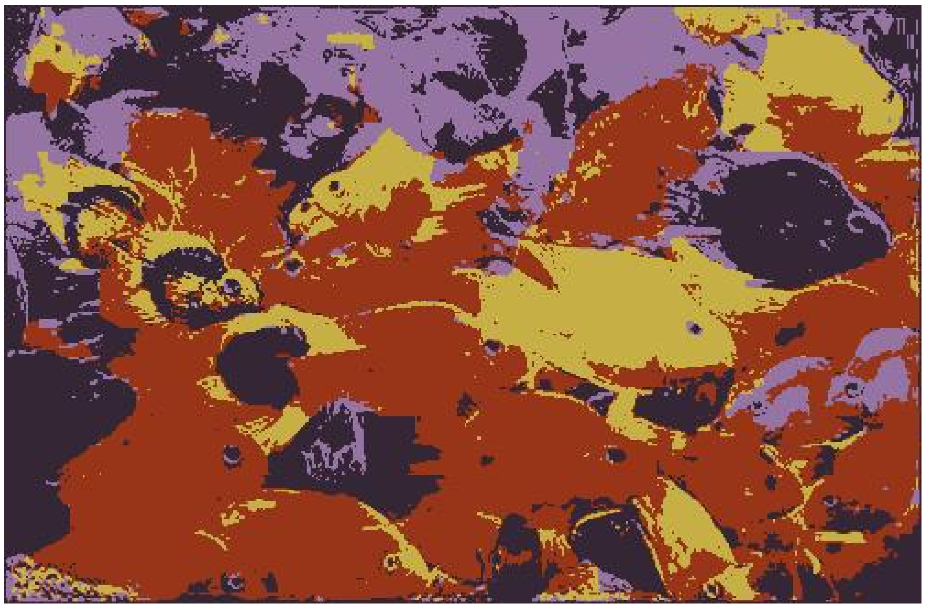

Figure 4: RGB Color Quantization Output Image (n=4)

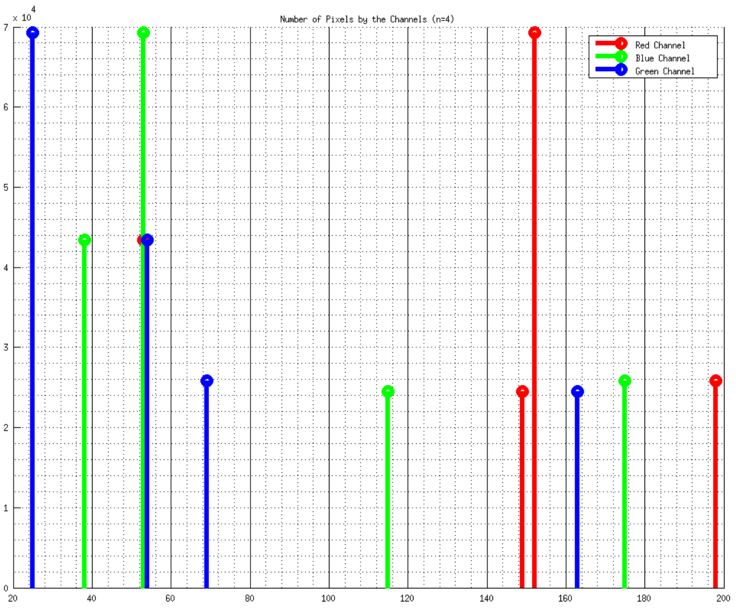

Figure 5: Histogram (n=4), Each color channel is represented by its own color

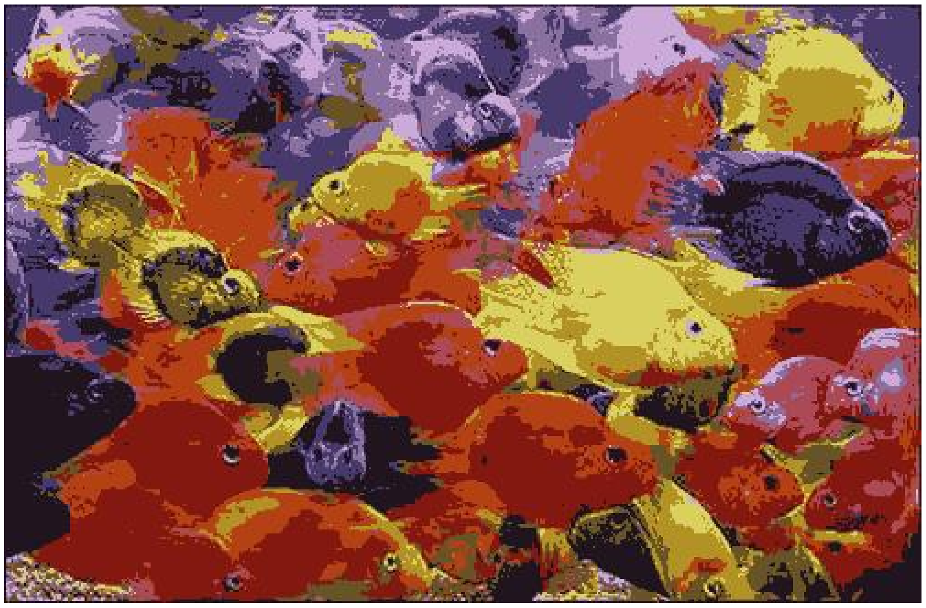

Figure 6: RGB Color Quantization Output Image (n=10)

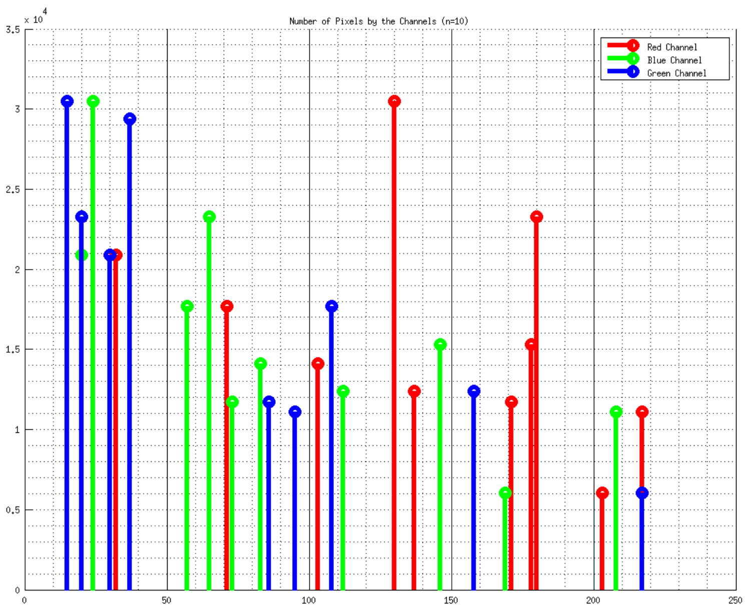

Figure 7: Histogram (n=10), Each color channel is represented by its own color

2.1.2 HSV Quantization with K-Means

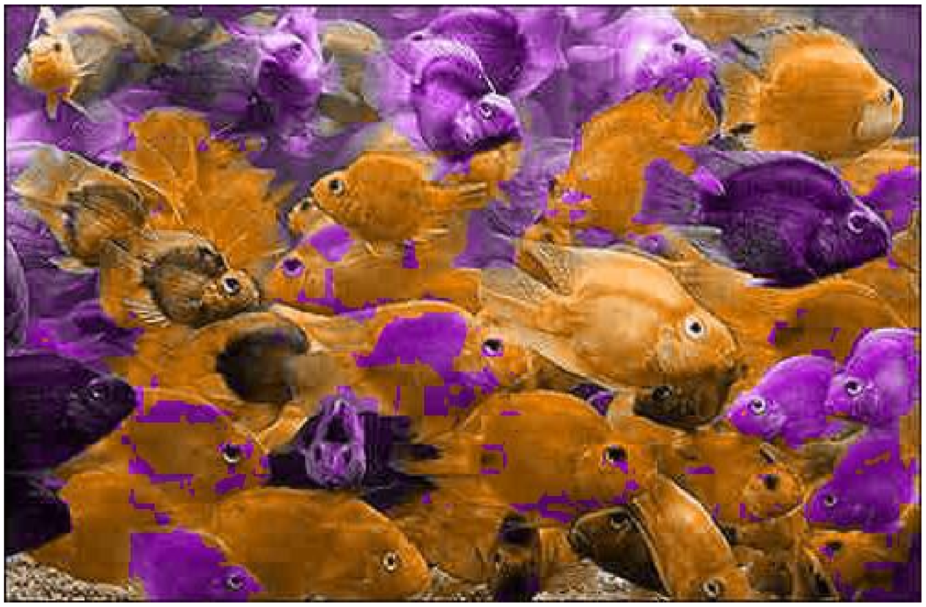

Figure 8: HSV Color Quantization Output Image (n=2)

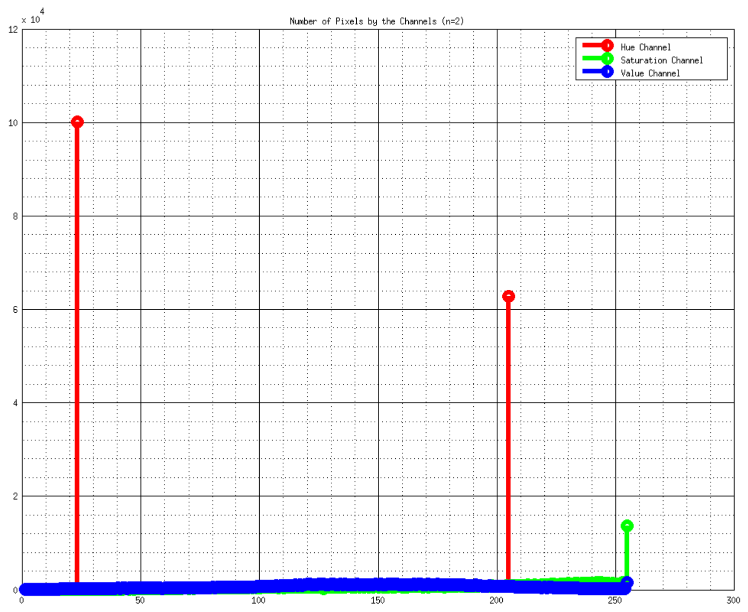

Figure 9: Histogram (n=2), Each color channel is represented by its own color

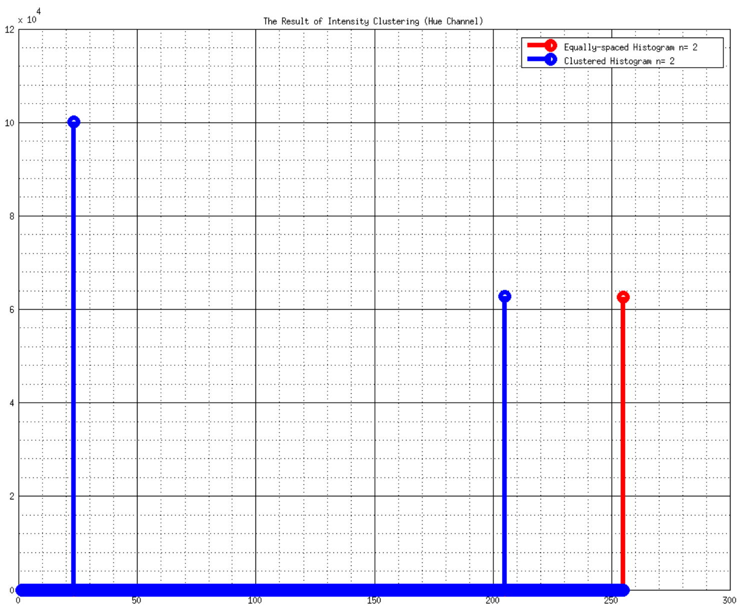

Figure 10: Histogram (n=2), Equally spaced bins vs. Quantized Histogram

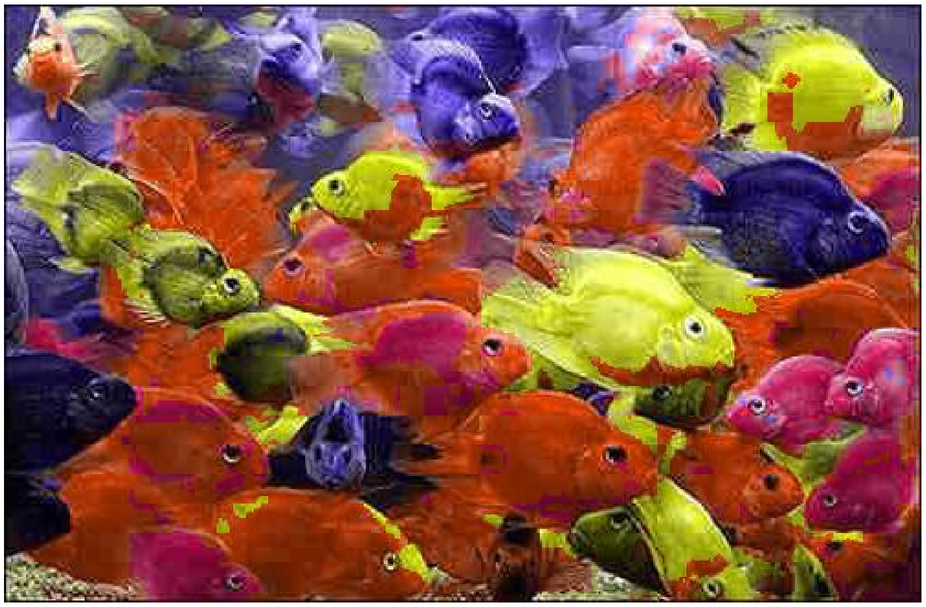

Figure 11: HSV Color Quantization Output Image (n=4)

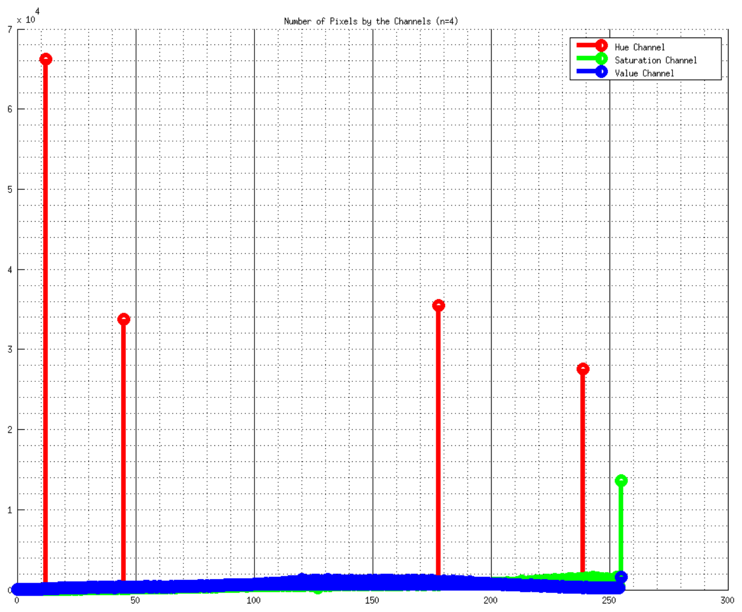

Figure 12: Histogram (n=4), Each color channel is represented by its own color

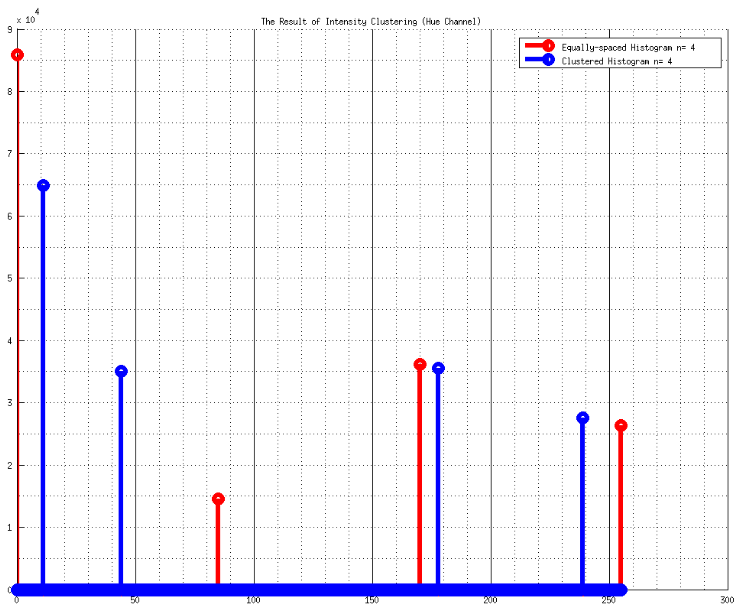

Figure 13: Histogram (n=4), Equally spaced bins vs. Quantized Histogram

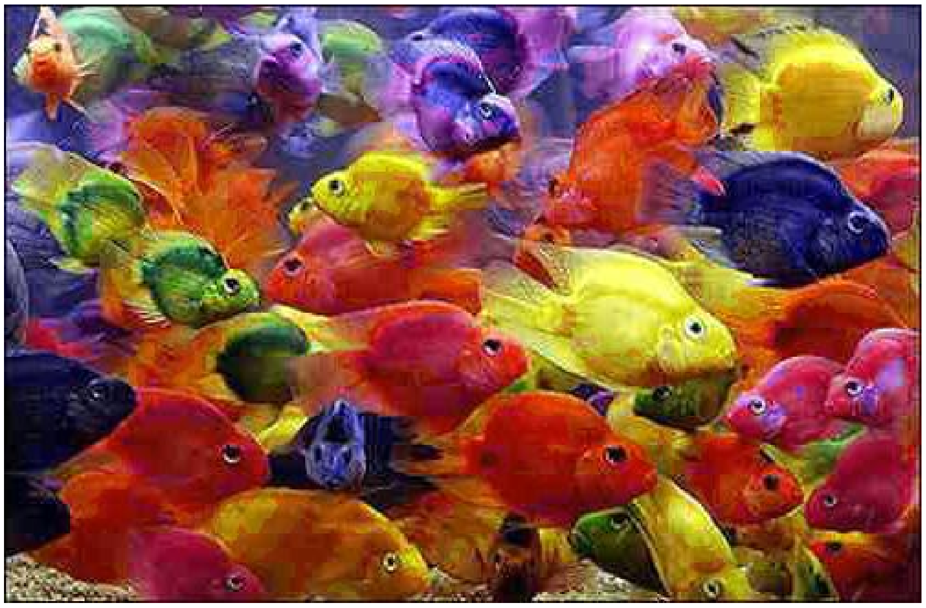

Figure 14: HSV Color Quantization Output Image (n=10)

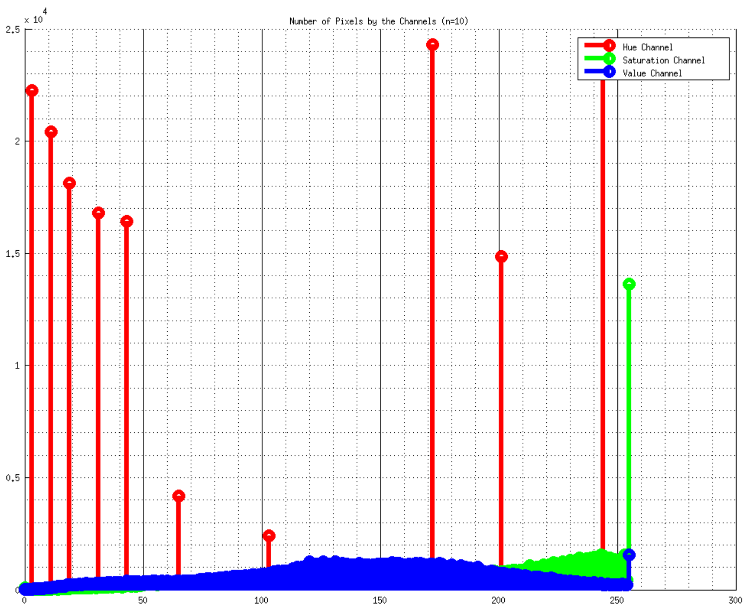

Figure 15: Histogram (n=10), Each color channel is represented by its own color

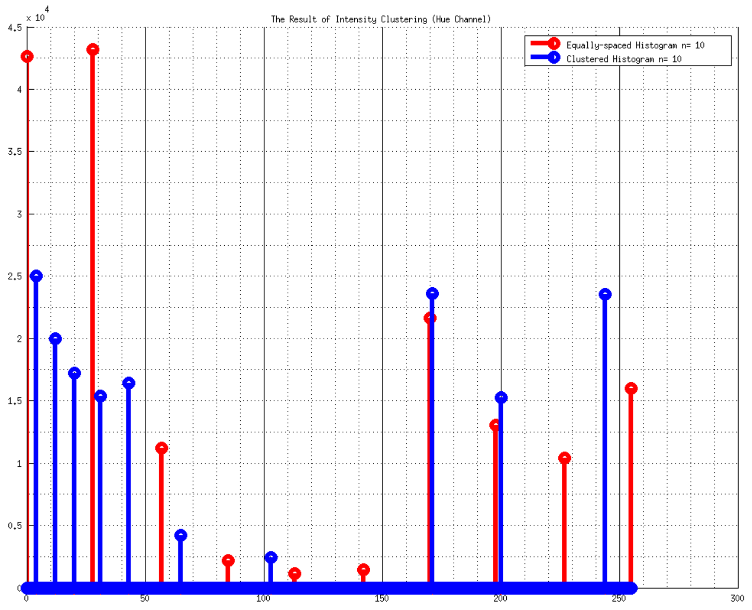

Figure 16: Histogram (n=10), Equally spaced bins vs. Quantized Histogram

2.1.3 Discussion

Color quantization is a technique that enables us to represent the pixels with fewer bits to store images in smaller memory maps. For instance, 2^8 different colors can be mapped to n different colors. In this way each pixel can be represented with $log_2 n$ bits, instead of 8.

Color space	n=2	n=4	n=10
RGB Clustering:	99.3278	91.7612	80.3925
HSV Clustering:	27.0659	16.1680	7.9427

Although quantization saves memory space, it also introduces color reproduction error since each pixel will be represented by the centeroid color of the cluster which it is assigned. Table-2.1.3 shows mean of color reproduction error (number of runs= 15). As it can be assessed from the table, the error reduces as $n \rightarrow 255$. Thus, the trade-off between memory and the error should be tuned for each run by choosing the right number of clusters. The artifact and the effects of n can be seen in figures-{2, 8, 4, 11, 6, 14}

Color space	Run=1	Run=2	Run=3
RGB Clustering:	80.0289	80.5505	80.3925
HSV Clustering:	5.4682	5.1711	8.8450

Along with k, the initialization has significant effect on the overall performance of kmeans algorithm. The initial step of kmeans is to choose n random points inside the data set and start assigning the data to the closest cluster. For that reason, starting point both has effect on the total number of iteration and the points that clusters converge. The table-2.1.3 below shows the different color reproduction error across arbitrary runs. (n=10)
The difference between color spaces also changes the overall performance of the algorithm. For instance, as it can be seen on the figures-{6, 14}, the results of quantization of HSV and RGB color spaces display great difference. Even though n is the same ($n = 10$) for the both instances, the results greatly vary. The reasons why that difference occurs is the algorithm is not used for each channel in the HSV image, the scales of mappings are different.
$RGB : 16,777,216 \rightarrow n^3$, while $HSV : 16,777,216 \rightarrow n * 65536$

This difference leads better image quality on the HSV quantization and fewer artifacts). Moreoveer, since less computation is done in the HSV quantization process, it is expected to run faster than RGB clustering.

2.2 Circular Hough Transform

2.2.1 Implementation:

- Data Reduction: The image is converted to grayscale to reduce the amount of data processed.
- Gradient calculation: For later use, calculate gradient image with Sobel Kernel.
- Matlab provides gradient direction in angles in between $[-180, 180]$ degrees, convert it to to radians $[0, 2\pi]$.
- Calculate the vector of cosine and sine in theta range. (In case of not using gradient.)

- Initialize Hough space ($H_{(m*n)}$) with same dimension of the input image. (a and b resolution is 1)
- Initialize the center and vote arrays in which the most voted centers and their votes will be stored.
- Detect edges with Canny algorithm. I used Canny because of the fact that it gurantees that edges will be 1 pixel wide and I can tune the output by optimizating σ and thresholds. By doing so, I reduce the total number of data will be processed.
- Edge points are stored in the vector to eleminate one for loop in the Hough transform.
- Calculate a and b, delete the results out of the image plane. Since cosine and sine values are stored in the vector, the formulate turns into vector operation. The resulting a and b are also vector.
  Side not: If the useGradient flag is set, a and b are calculated with the gradient value acquired from imgradient() function. To take the complementary angle into account gradient angle and its complementary value is stored into a vector. Then, a and b are calculated (not a full list, just two points for each)
- Increment cells in the hough space; when gradient angle is used, 2 cells are incremented, otherwise the number of affected cells in the hough space equals to cosine and sine vector length.
- Sort the houghspace with correlating indices.
- Sort the n-strongest circle centers.

2.2.2 Results

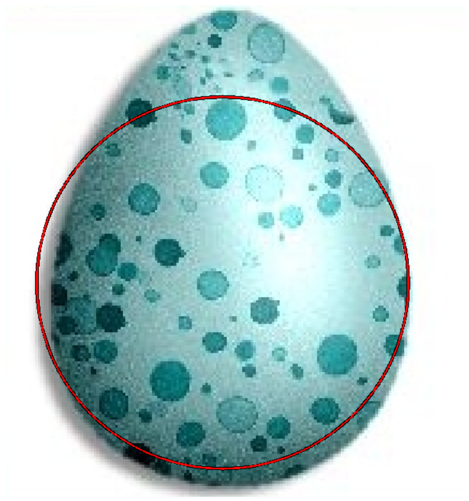

Figure 17: The most strongest circle with radius 70. (useGradient=0)

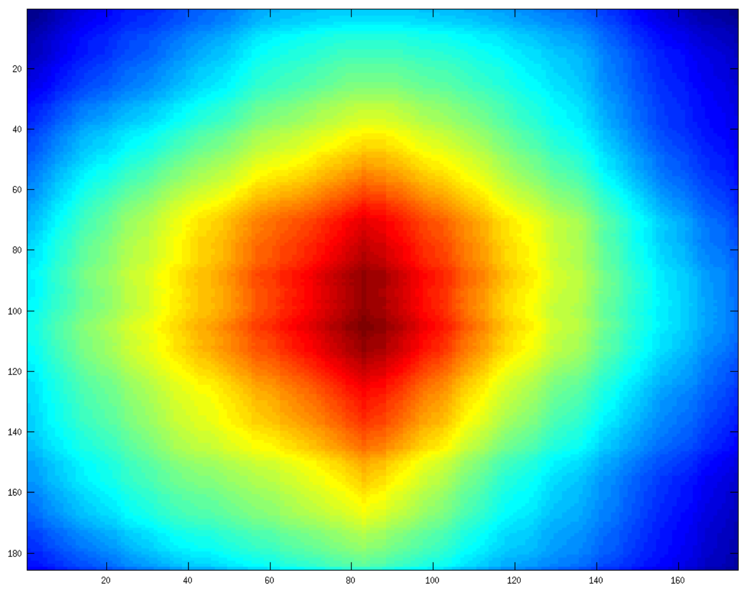

Figure 18: The hough space with radius 70. (useGradient=0)

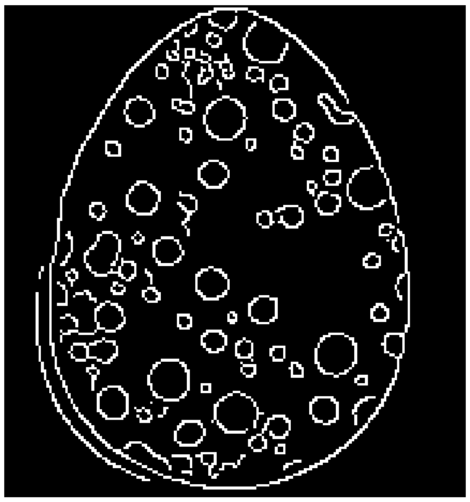

Figure 19: The Edge image resulting to the hough space above

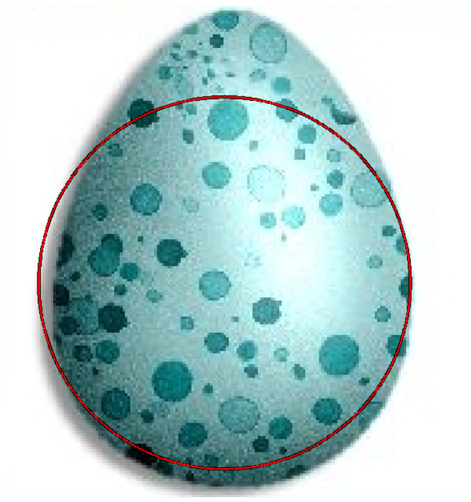

Figure 20: The most strongest circle with radius 70. (useGradient=1)

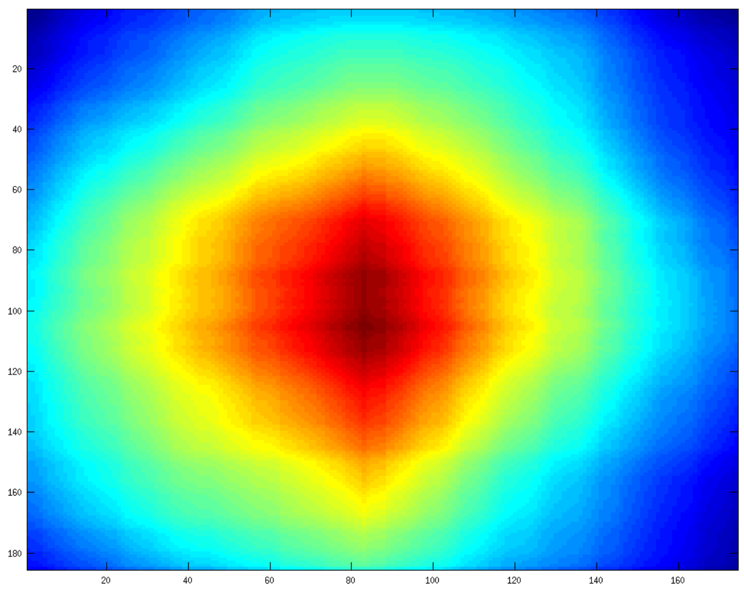

Figure 21: The hough space with radius 70. (useGradient=1)

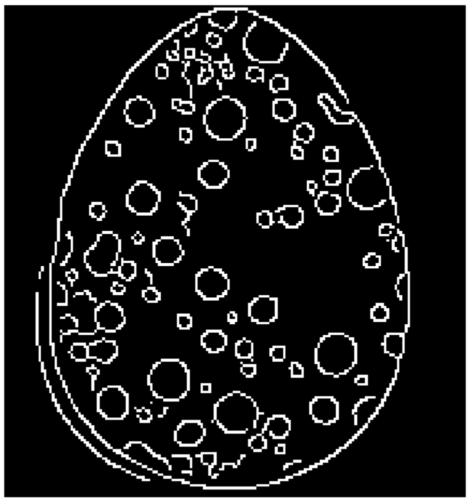

Figure 22: The Edge image resulting to the hough space above

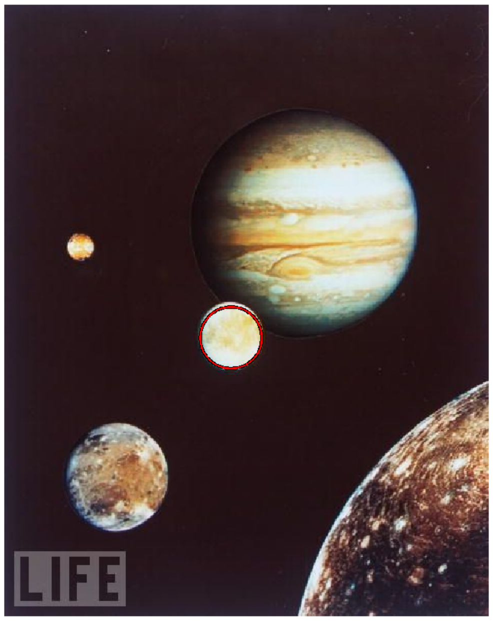

Figure 23: The most strongest circle with radius 30. (useGradient=0)

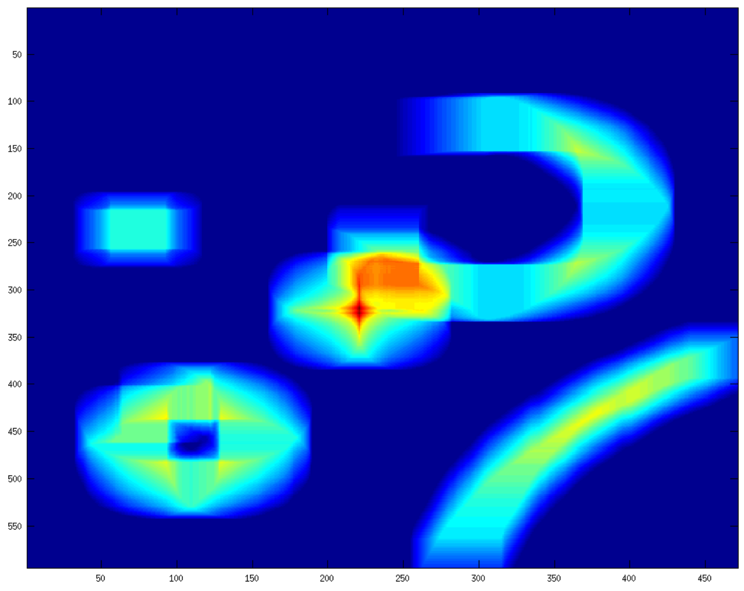

Figure 24: The hough space with radius 30. (useGradient=0)

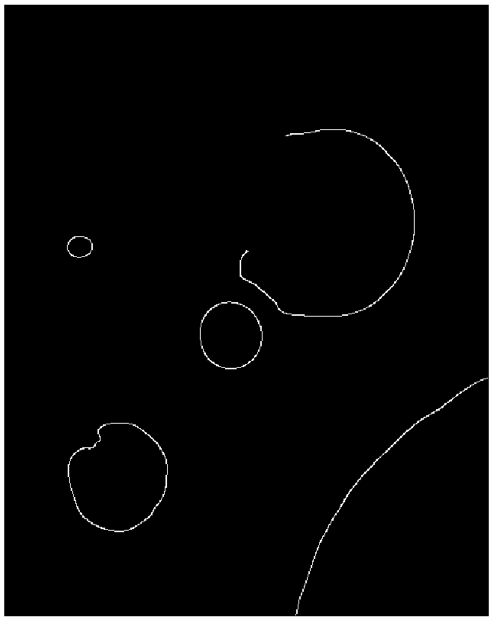

Figure 25: The Edge image resulting to the hough space above

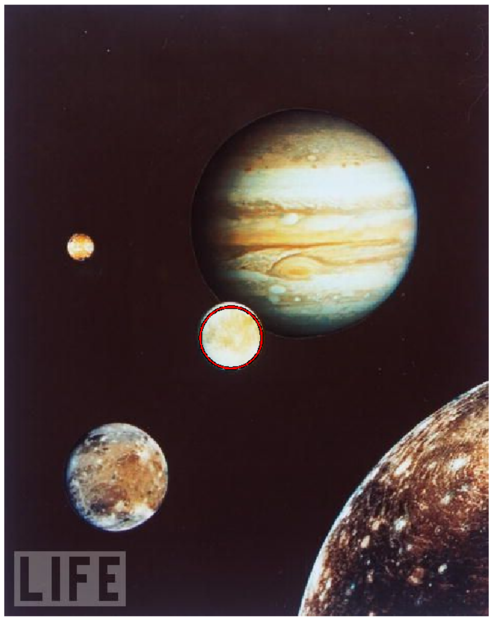

Figure 26: The most strongest circle with radius 30. (useGradient=1)

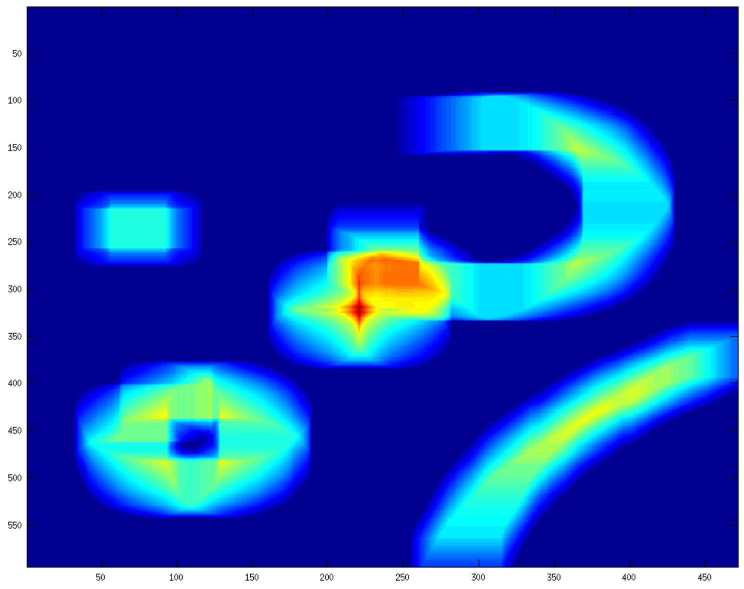

Figure 27: The hough space with radius 30. (useGradient=1)

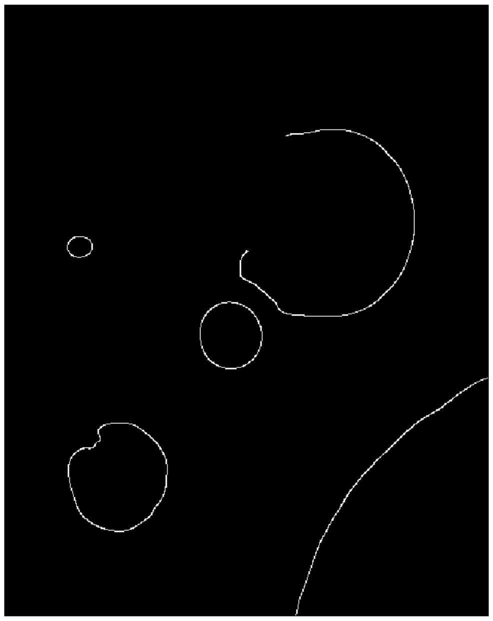

Figure 28: The Edge image resulting to the hough space above

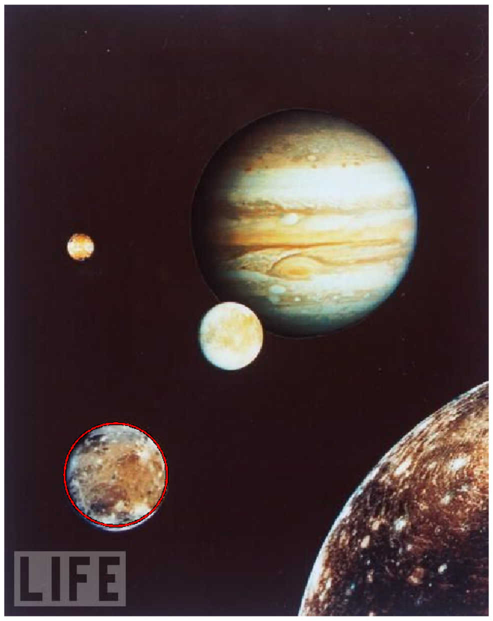

Figure 29: The most strongest circle with radius 50. (useGradient=0)

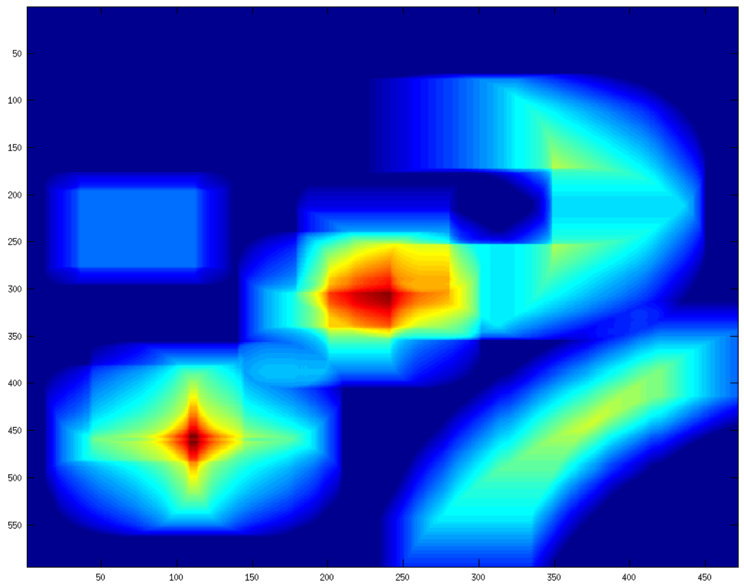

Figure 30: The hough space with radius 50. (useGradient=0)

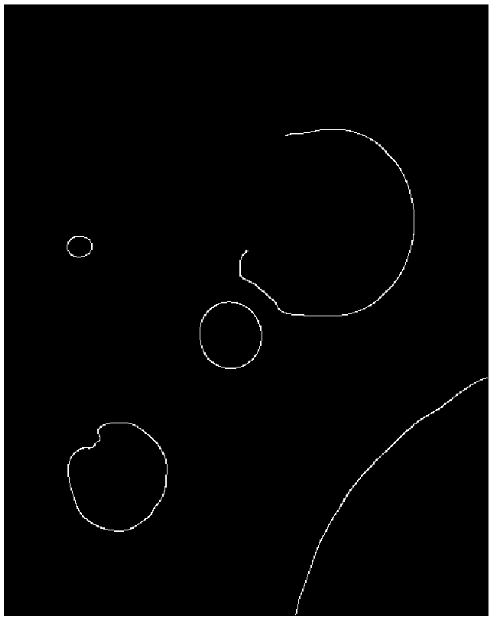

Figure 31: The Edge image resulting to the hough space above

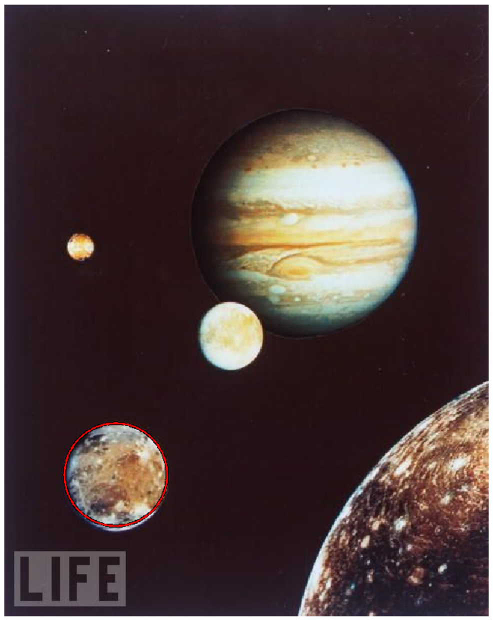

Figure 32: The most strongest circle with radius 50. (useGradient=1)

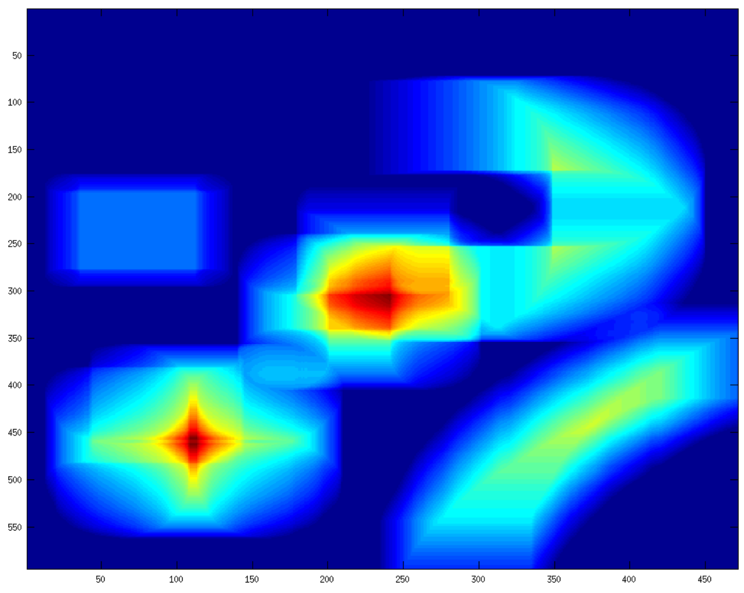

Figure 33: The hough space with radius 50. (useGradient=1)

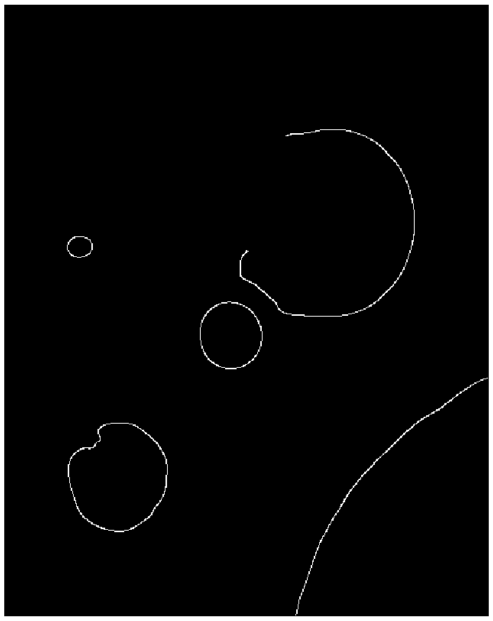

Figure 34: The Edge image resulting to the hough space above

2.2.3 Discussion

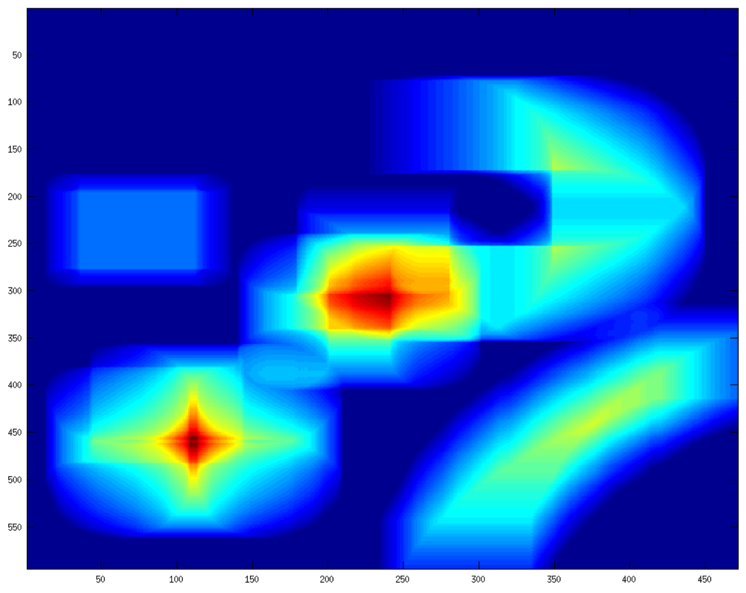

Figure 35: The hough space with radius 50. (useGradient=0)

Hough Space: The success of hough transform heavily depends on successfully constructing the hough space. On the other hand, as figure-35 depicts, the hough space closely correlates to edge image. Thus, the overall success of the transform requires precise edge detection steps.
Given that, tuning the edge detector is one of the most important part of the transform. Otherwise, the noise in the image may lead algorithm to fit circles to inaccurate results.
Another point is worth to mention is that, hough transfor is robust for missing and misleading information thanks to the accumulated voting system. For instance, despite figure-34 has missing and misleading (inaccurate edges due to the lightning conditions and shades), the circle fitting has been successively done, as it can be seen on figure-32.
Figure-30 shows the accumulated votes for radius of 50 pixel circles

with figure-34. To successfully fit a circle with the given radius, each edge point votes all the possible circles to which it is belong with given radius. As voting process continues, it can be inferable that the the peaks of hough space relates to the circle centers. Thus, the bright areas in the figure-30 relates to the circle centers with 50 pixel radii.
The Impact of Bin Size: The bin size is the another parameter which has deep impact on the overall performance of the fit. Reducing the bin size would lead to have less precise results for the centers points, albeit fast.
Post-processing Experimentation: The result image was extremely noisy due to the edge detection algorithm I used. For that reason, before starting tuning edge detection algorithm, I implemented a morphological edge detection process which makes sure that all the edges are 1 pixel wide and no clutters inside the circles.
Attached *morphologicalEdges.m* file contains series of operations accounts and tries to eleminate the noise inside the planet bottom-right in the image.

3 Optional

For radii 16,30 and 50 the marked circles, the edge image and the hough spaces (respectively) can be seen in the figure below. Attached *houghRadii.m and detectCirclesRadii.m* are the source files for multi-radii Circular Hough Transform.

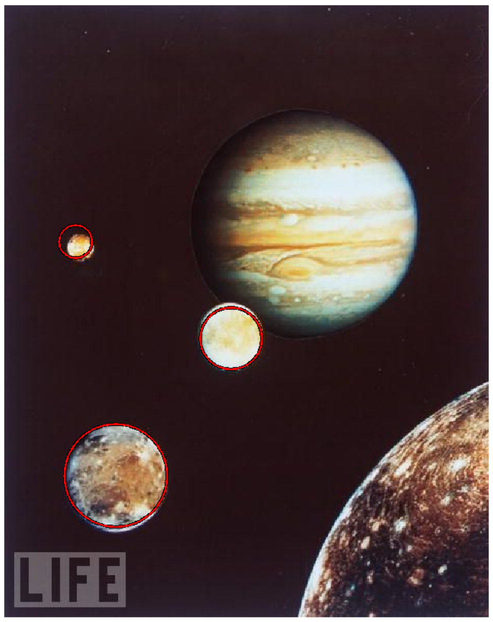

Figure 36: Multiple radii circles is fit with Circular Hough Transform

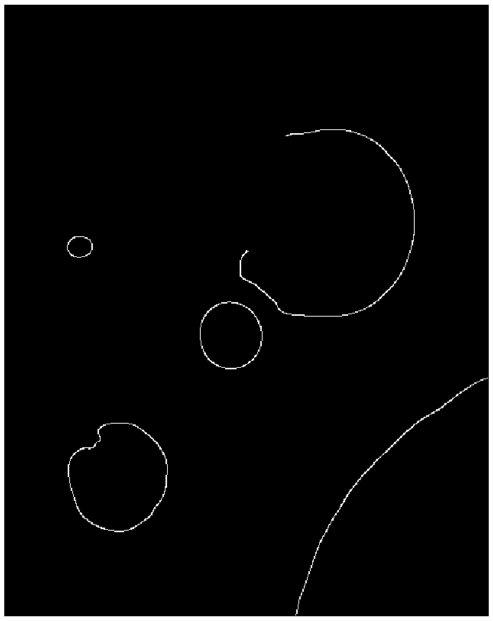

Figure 37: The Edge Image

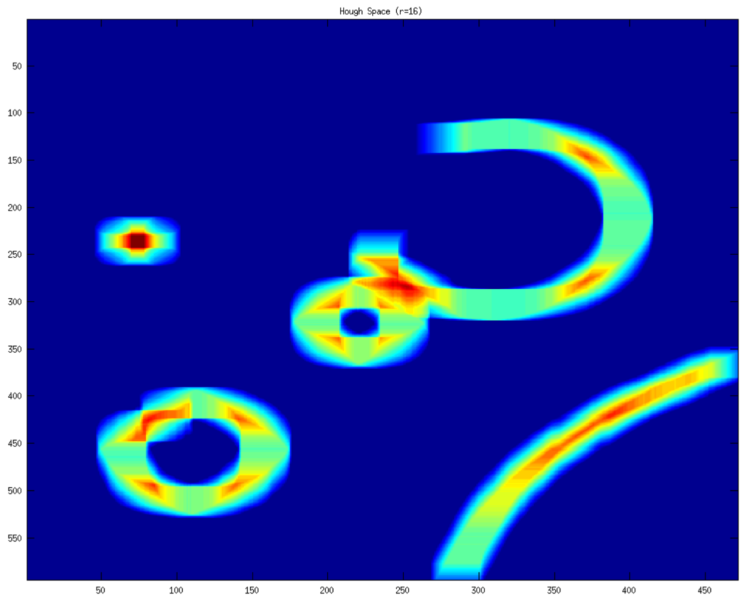

Figure 38: The Hough Space (r=16)

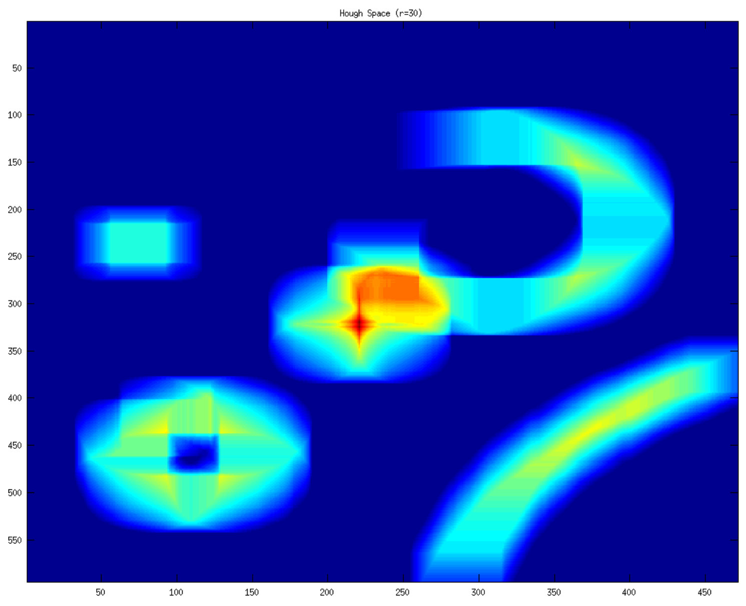

Figure 39: The Hough Space (r=30)

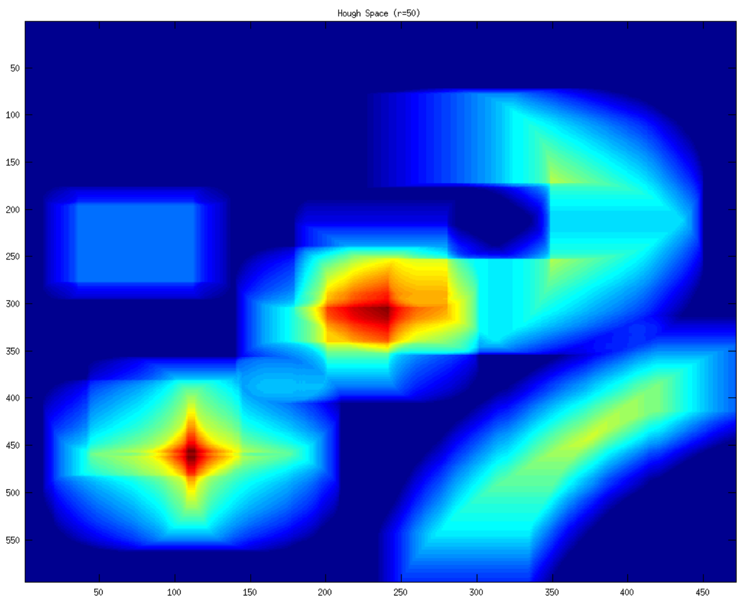

Figure 40: The Hough Space (r=50)