PS2 - Siddharth Bhal

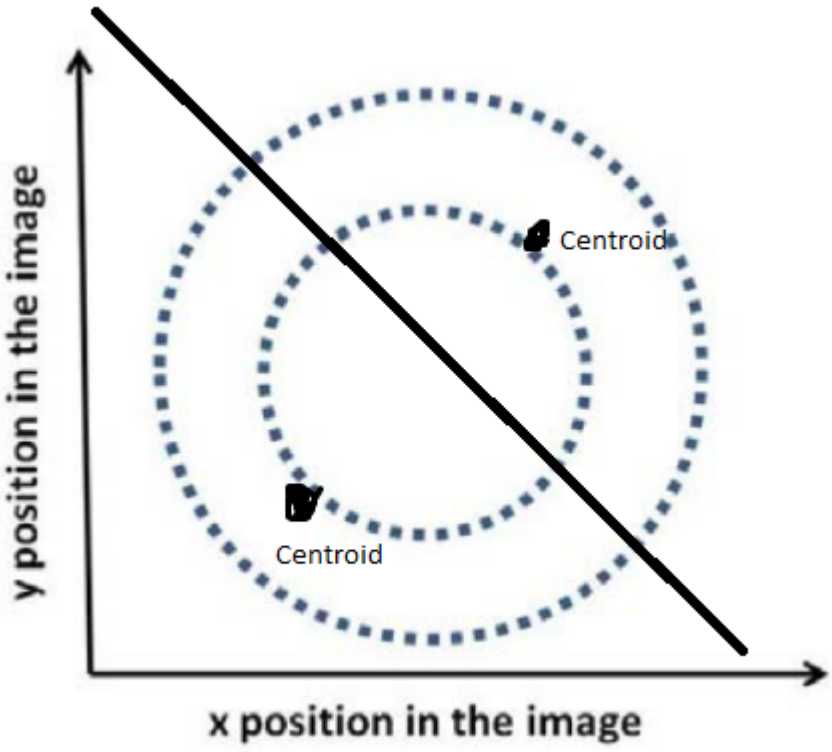
Short Answer Solutions

1

Resulting representation is sensitive to orientation. Representation is sensitive to orientation since gradient in all direction are present in image.

2

Kmeans is spherical clustering. Final centroid of cluster will depend on starting position. One of the final clustering could be :The resulting clustering will be



Other configuration are also possible pertaining to different staring points. But line distributing both clusters will always cut both circular figure in half.

3

Mean Shift algorithm will be useful for a continuous voting space. Main reason is we don’t know number of clusters to group the image in beforehand which is necessary of k-means algorithm.

4

Here we can use generalized Hough transform.
Algorithm is :

1. Take a blob and have a referenced point inside the blob .
2. At each boundry point compute displacement vector wrt the reference point.
3. Store the displacement vector in a table indexed by gradient orientation.
4. Detection Procedure - For each edge point:
    1. Use the gradient orientation theta to index into stored table.
    2. Use retrieve r vector to vote for referenced point in parameter space.

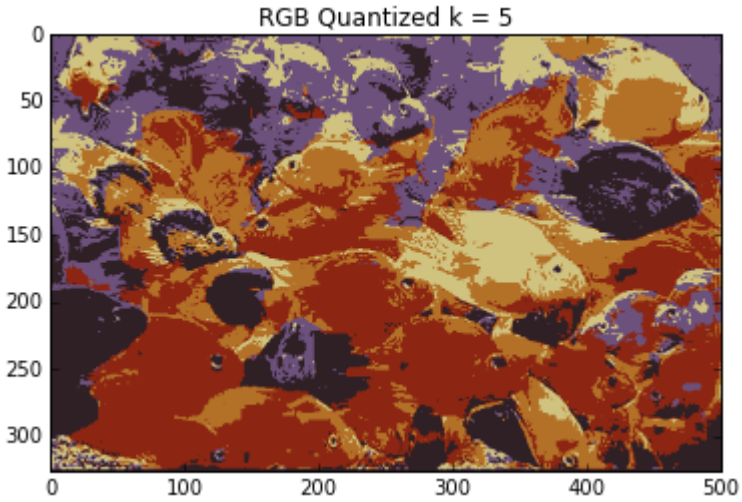
The most voted points in parameter space will tell us the group of centers.
These circles are the one which should be part of same group.

Programming Solutions

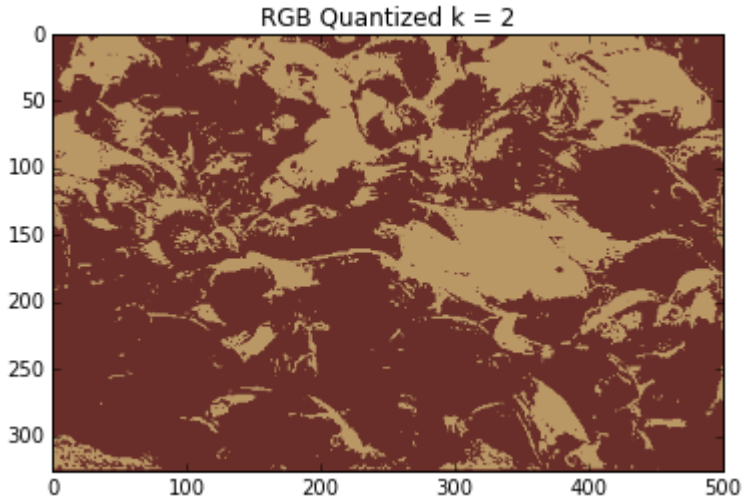
1

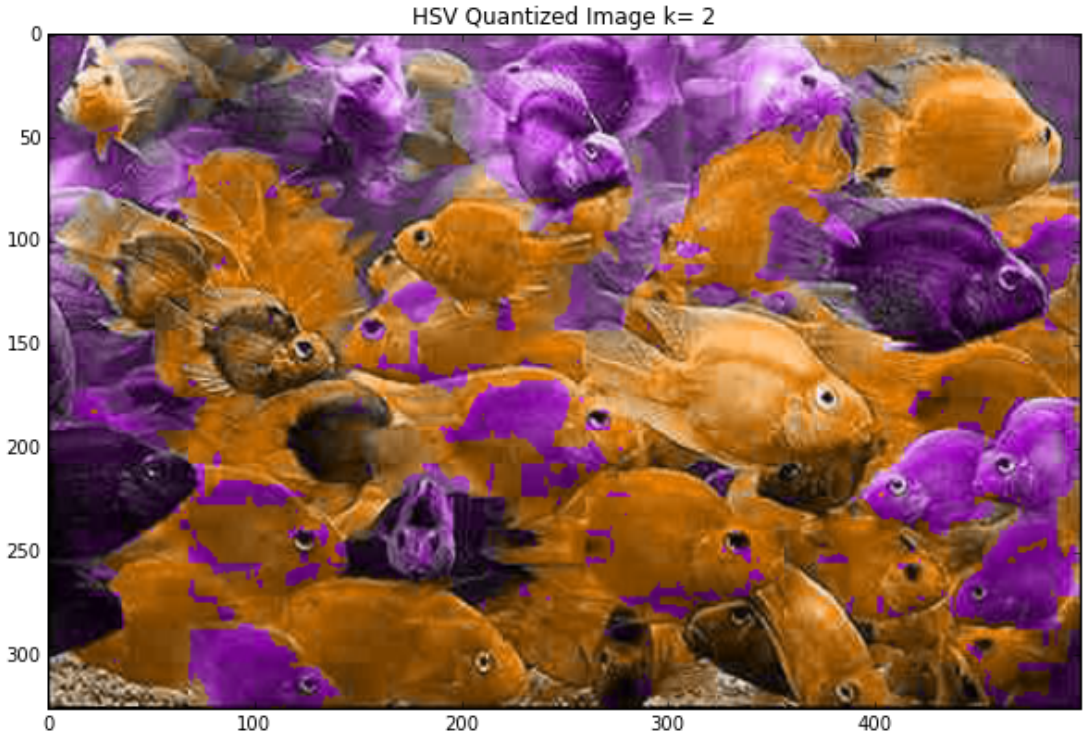
(e)

SSD = 6112029109.99 (between original image and quantizedRGB with k=5)
SSD = 70599739.603 (between original image and quantizedHSV with k=5)



SSD = 6116034978.98 (between original image and quantizedRGB with k=2)
SSD = 402471817.47 (between original image and quantizedHSV with k=2)



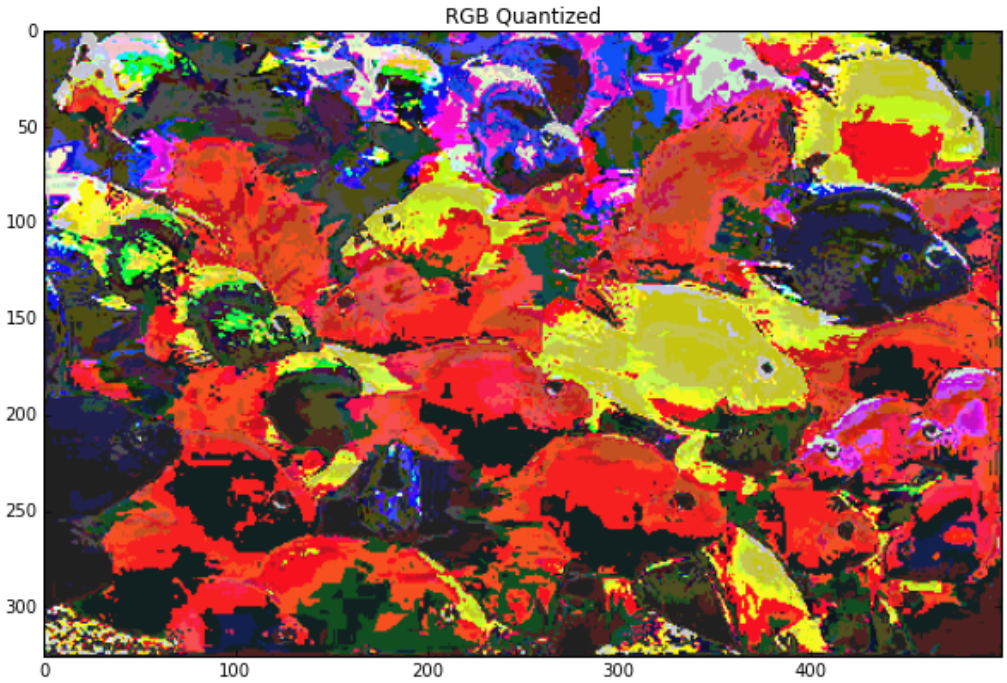


(f)

1.

Original Image:

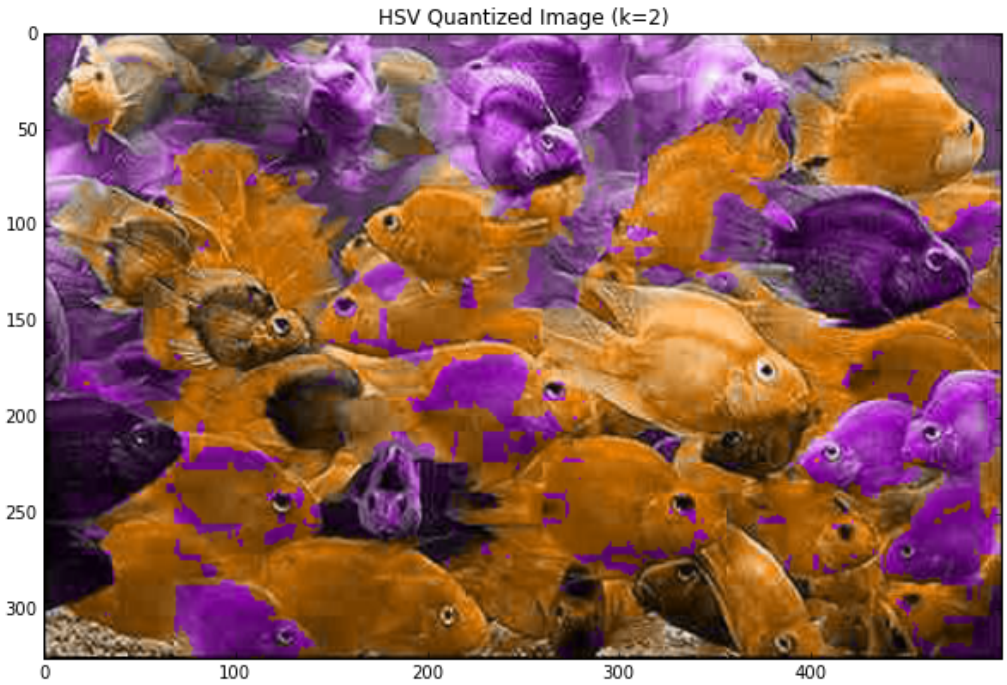
2. RGB Quantized Image (k=5)



We can observed only 5 colors in the image and image hasn’t lost its much characteristics due to equalization.

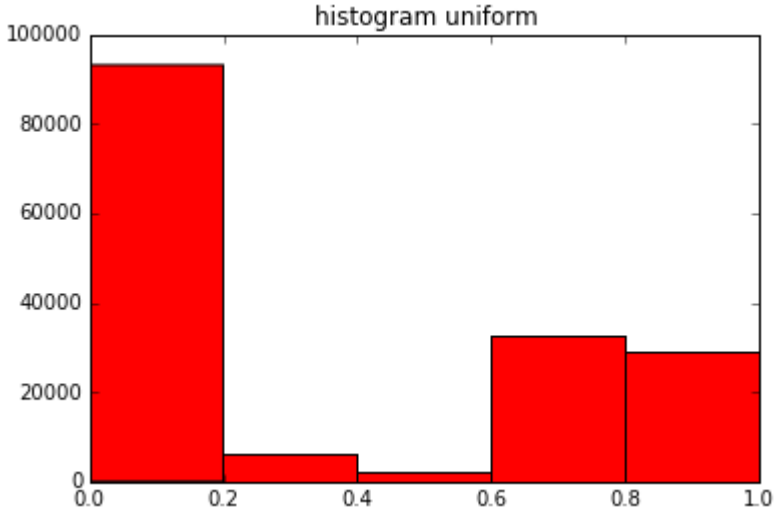
3. HSV Quantized (k = 5)

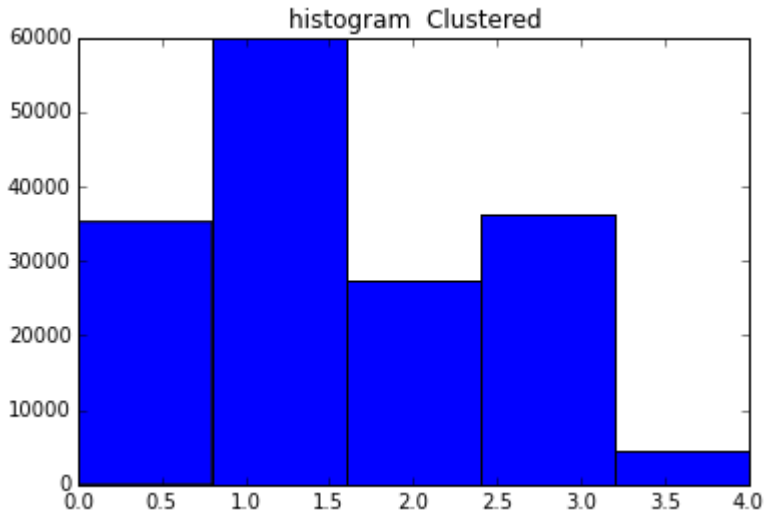

HSV Quantized Image
0
50
100
150
200
250
300
0
100
200
300
400


HSV Quantized Image (k=2)
0
50
100
150
200
250
300
0
100
200
300
400

Clearly we can see the difference between k=2 and k=5 hue quantized images. Since hue specifies the color range of image. Decreasing the number of clusters is decreasing number of colors in images.

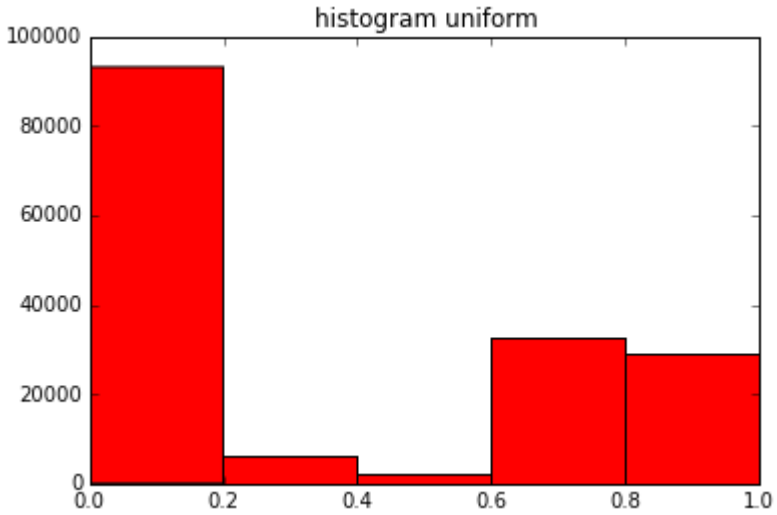
4. For k=5

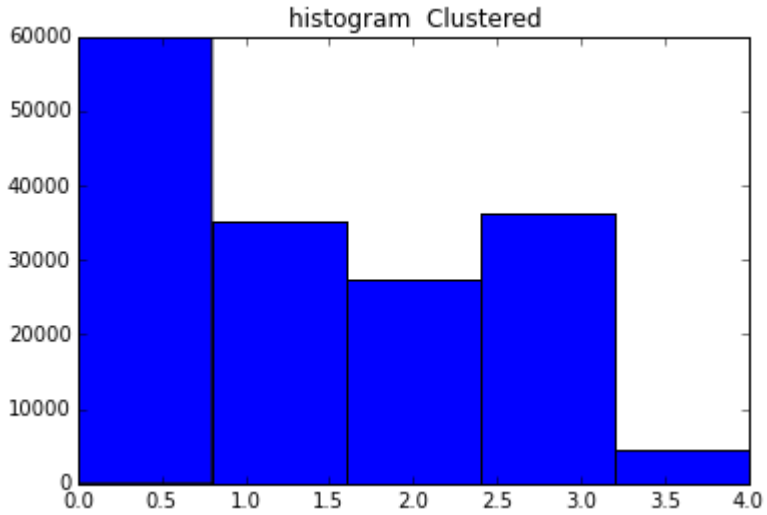




Here clustered hue histograms are always spread evenly than uniform histograms.

For k=2





Also, the height of bins is dependent on number of clusters to be formed.

Here I have just pasted bin diagram. Pictures for same can be found in above examples.

2

(a)

Below are the steps I am using to detect circles in image:

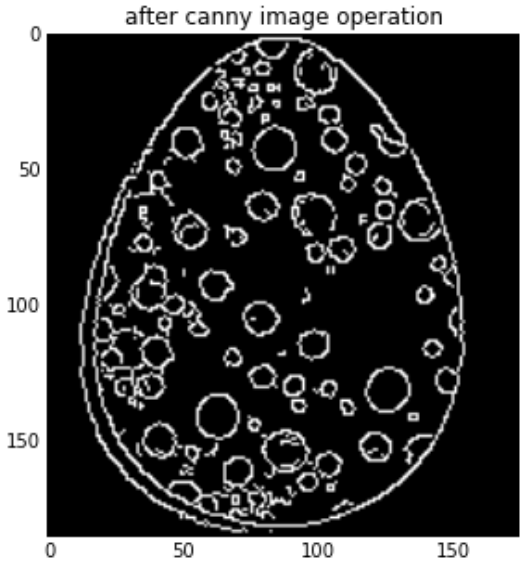
1. Use canny image detector to get edges of image.
2. Traverse all pixels of edge image
3. For each pixel traverse all angles in steps of pi/200 radians.
4. calculate points for which to vote using equations:
    1. a = r - cos(theta)
    2. b = r + sin(theta)
5. vote for above calculated votes in an accumulator array.

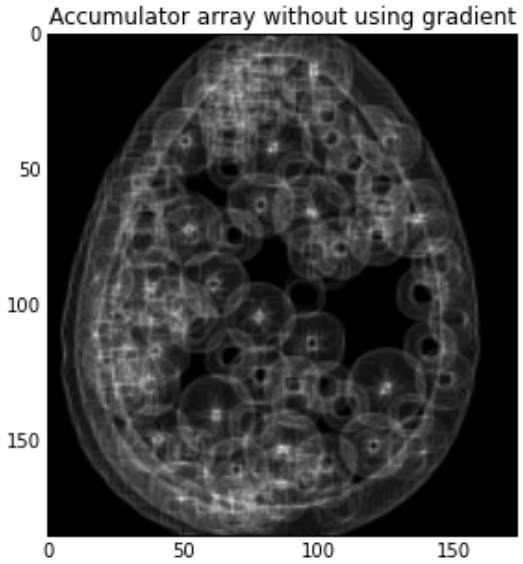
We can use gradient also to ease our implementation, in that case we don’t need to traverse through all angles since we already know the angle to vote to.

Here we are getting the edge points of image and finding corresponding points in parameter space and voting for them.
Then in parameter space, pixels which receives more number of votes is likely to be center of circle.

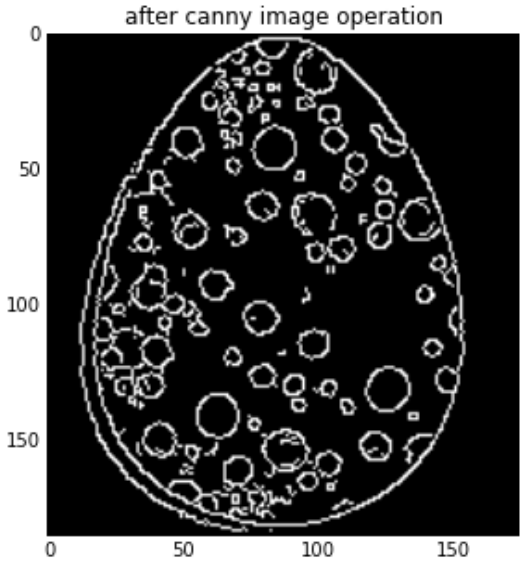
(b)

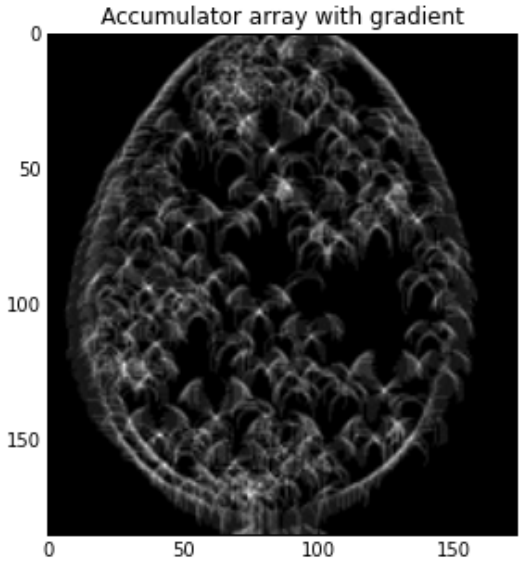
Without Using Gradient: (with r = 7)



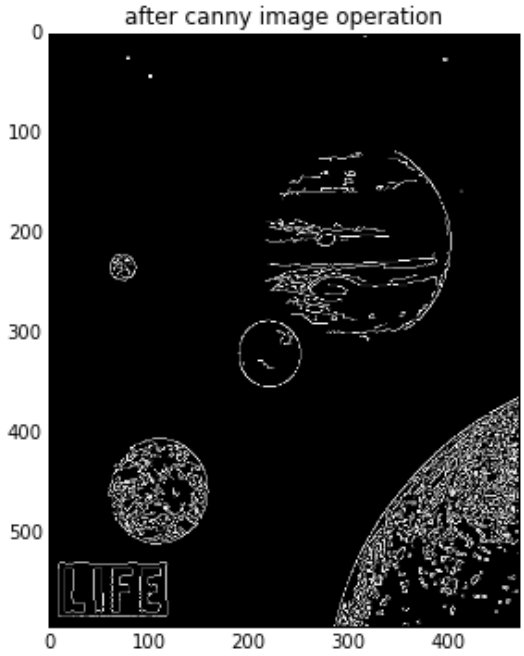


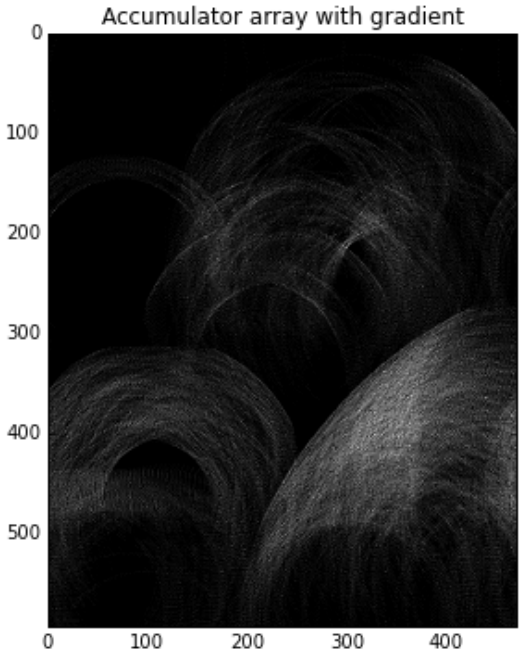
Using Gradient: (with r = 7)



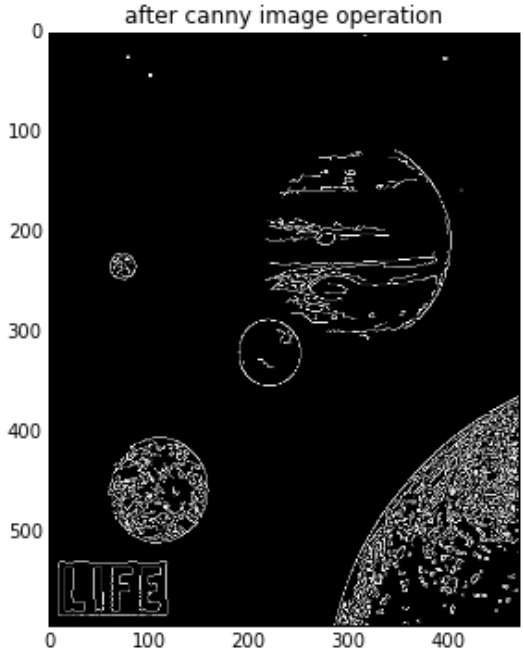


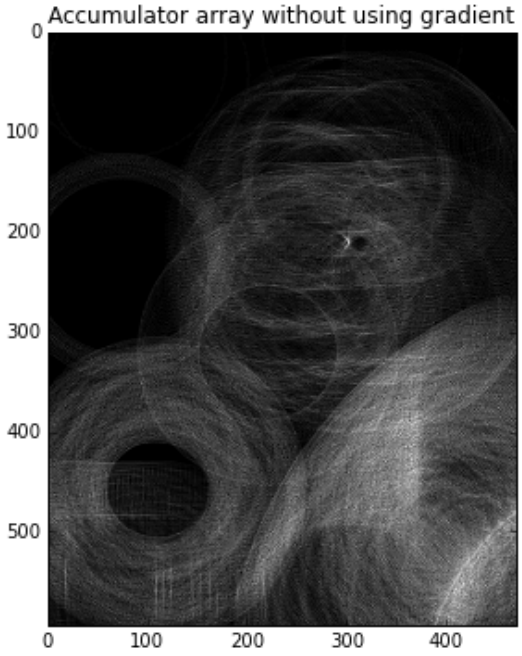
With Gradient (with r = 100)





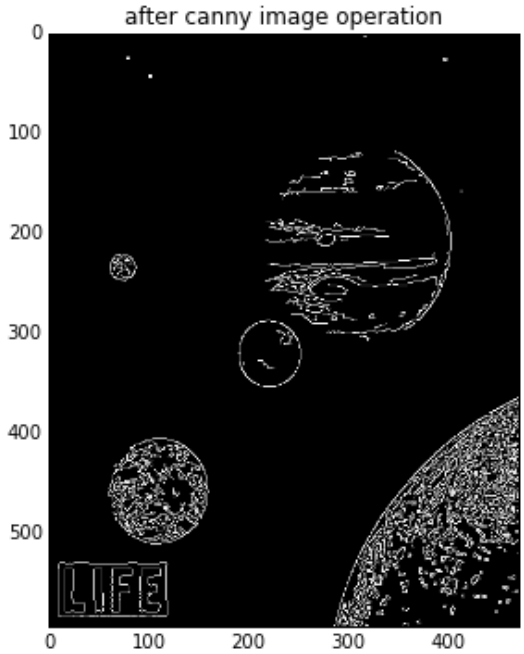
Without Gradient (with r = 100)

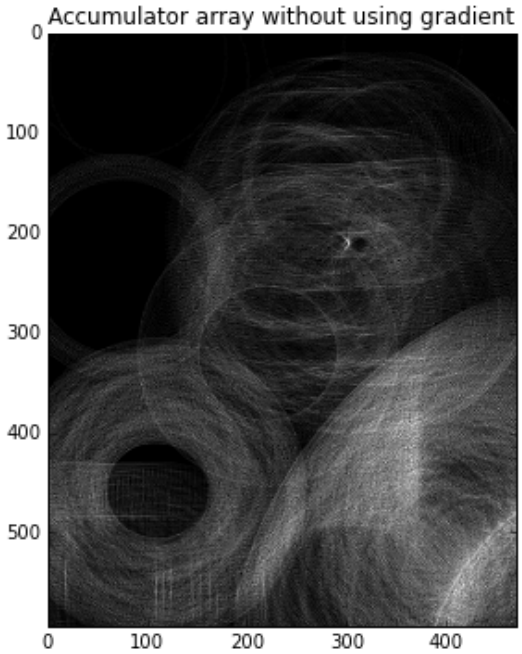




(c)

[[0. 0. 0. ..., 0. 0. 0.]
[0. 0. 0. ..., 0. 0. 0.]
[0. 0. 0. ..., 0. 0. 0.]
...,
[0. 4. 5. ..., 11. 10. 6.]
[0. 1. 1. ..., 7. 8. 6.]
[0. 3. 4. ..., 11. 11. 10.]
for below image



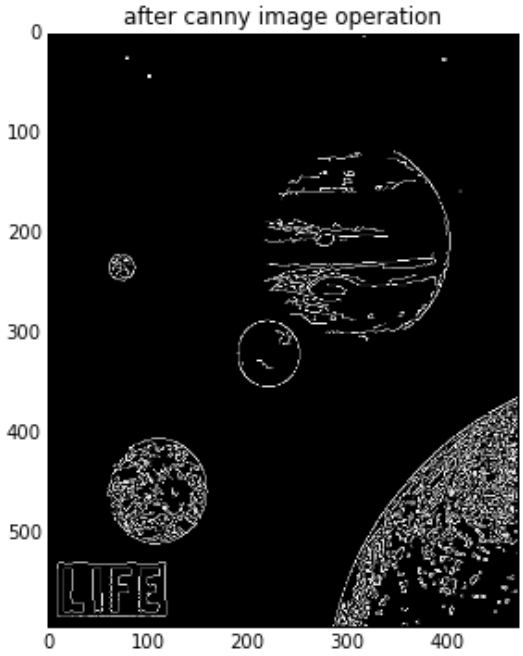


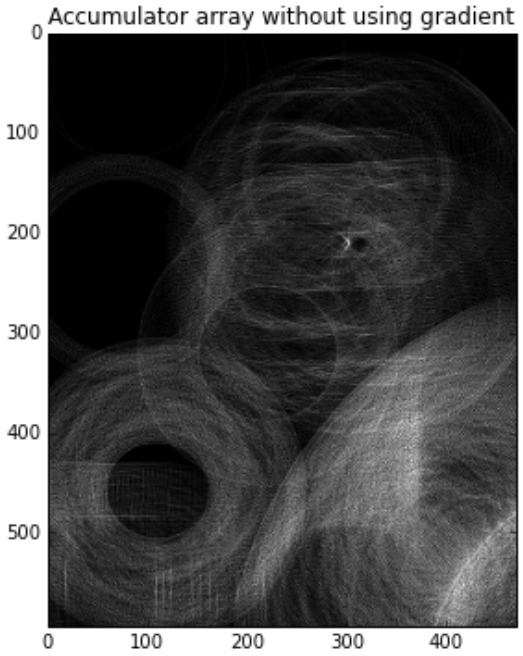
This accumulator array can easily be matched with accumulated displayed image.
I am not pasting complete accumulator array but you can easily see complete array by running code.

(d)

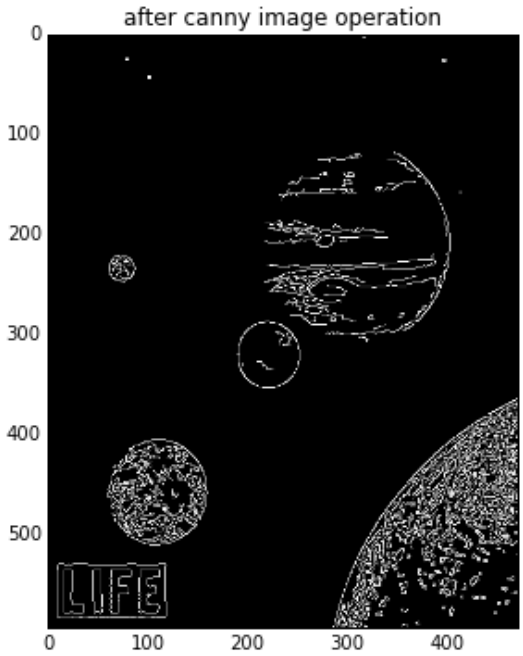
We can experiment with changing the radius and threshold to detect if the voted point in parameter space is center of circle or not.

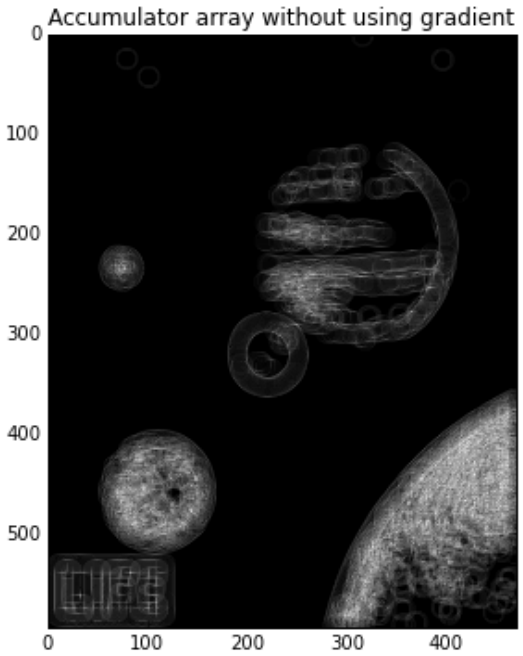
With r=100





with r=10

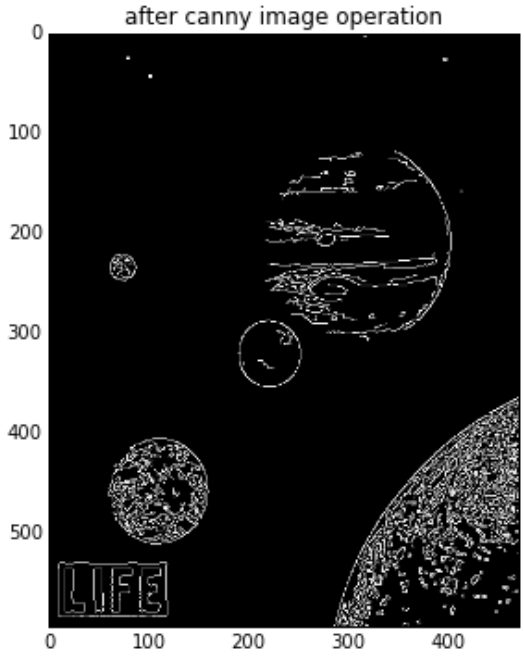


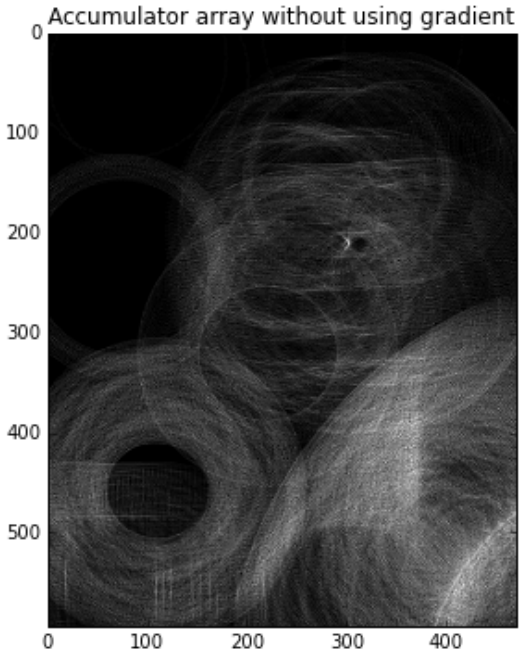


Its clearly visible that with r=10, more votes are going to smaller circles than with r=100 case.

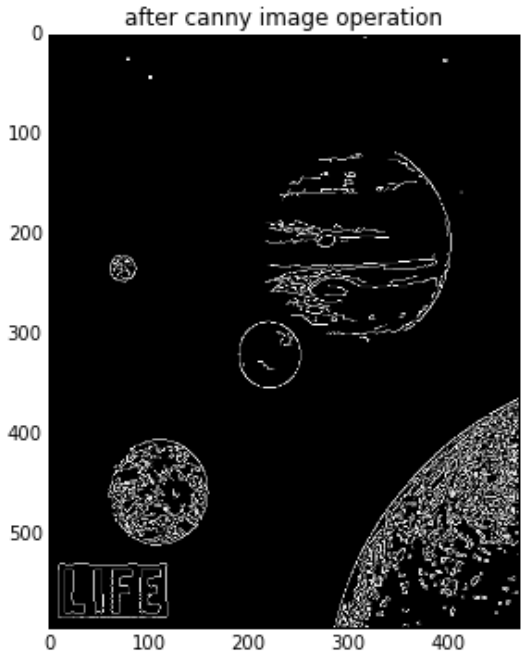
(e)

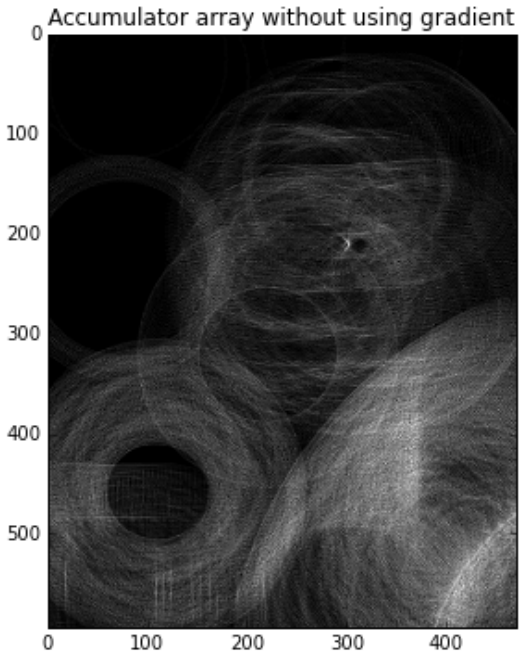
with voting only 1 point





with voting for 10 points





Accumulator array with more votes gives us more sharper image.