# 1 Short answer problems

1. The result representation is sensitive to orientation. Because different filters of different orientations can capture different kinds of textures, resulting in different representation.

2. The result will be two semicircle clusters. And where we draw the diameter to divide the points into two semicircle clusters depends on the two initial points. Because the distance between a point in a semicircle and the center of that semicircle is always less than the distance between that point and the center of another semicircle.

3. Mean shift is the most appropriate. Because mean shift seeks local maxima of density in the feature space. And in vote space, the point with highest density is most likely to represent a specific shape in the original image.

4. (1) Create a table of reference vectors indexed by gradient orientation θ

(2) For each one of the blobs:

For each point in the outer boundary of one blob:

Retrieve the reference vector according to gradient orientation

Determine a vote for the reference point according to the reference vector

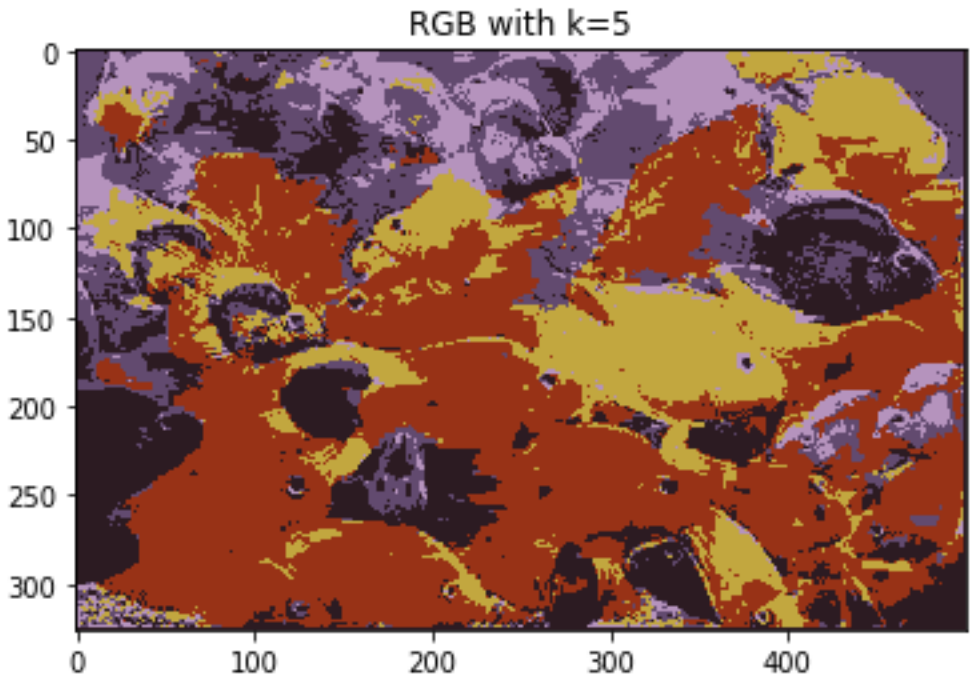
According to the votes of all the boundary points, determine a reference point for the boundary of that blob

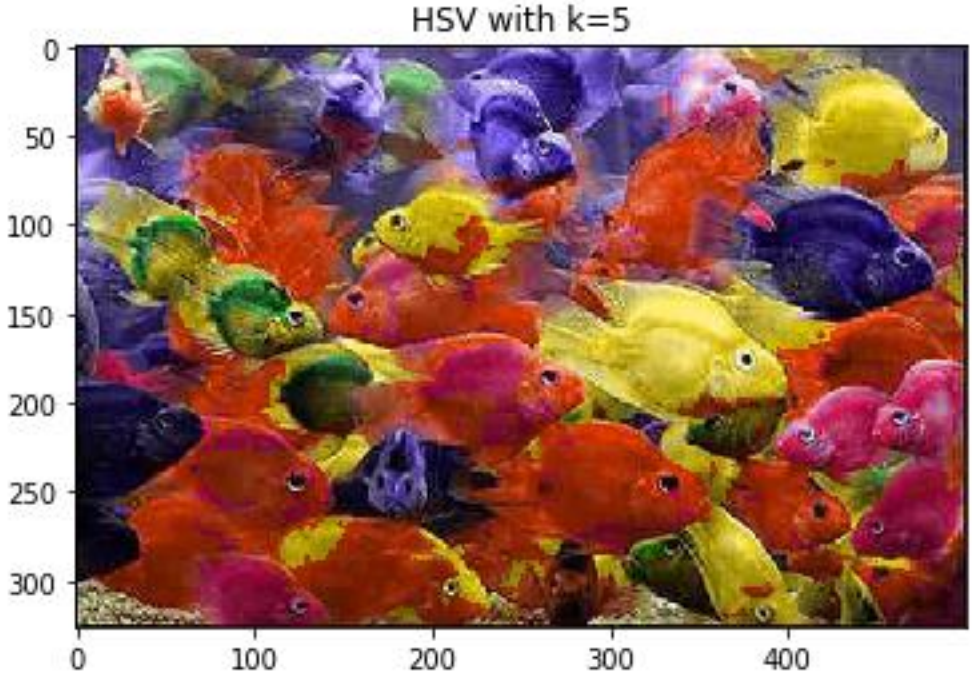
(3) Do K-Means clustering for all the blobs using the reference point in (2) as the feature.

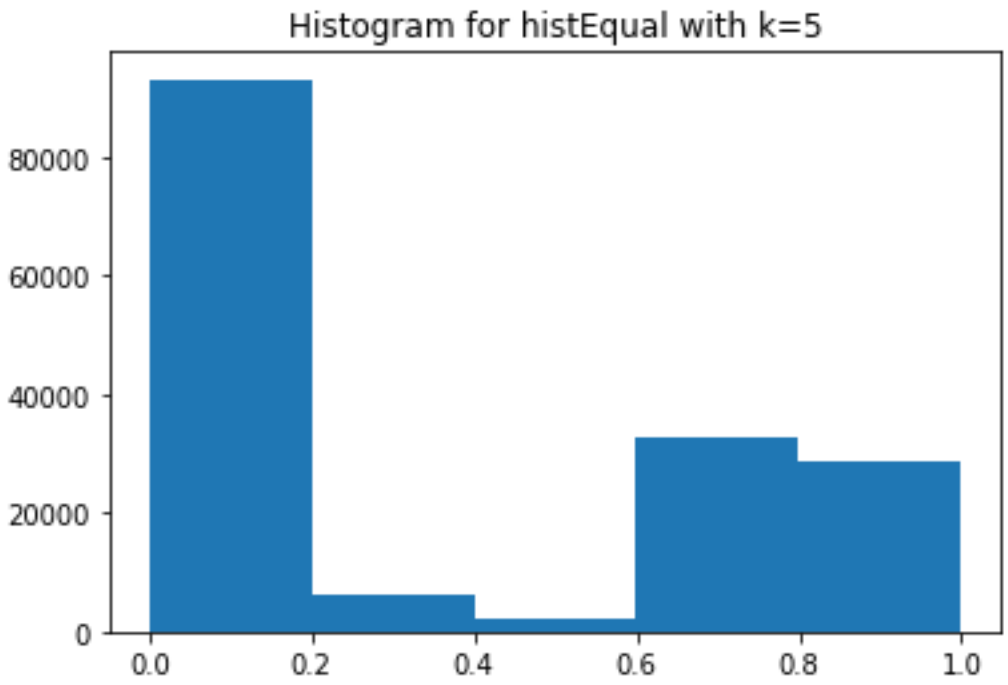
# 2 Programming problem:

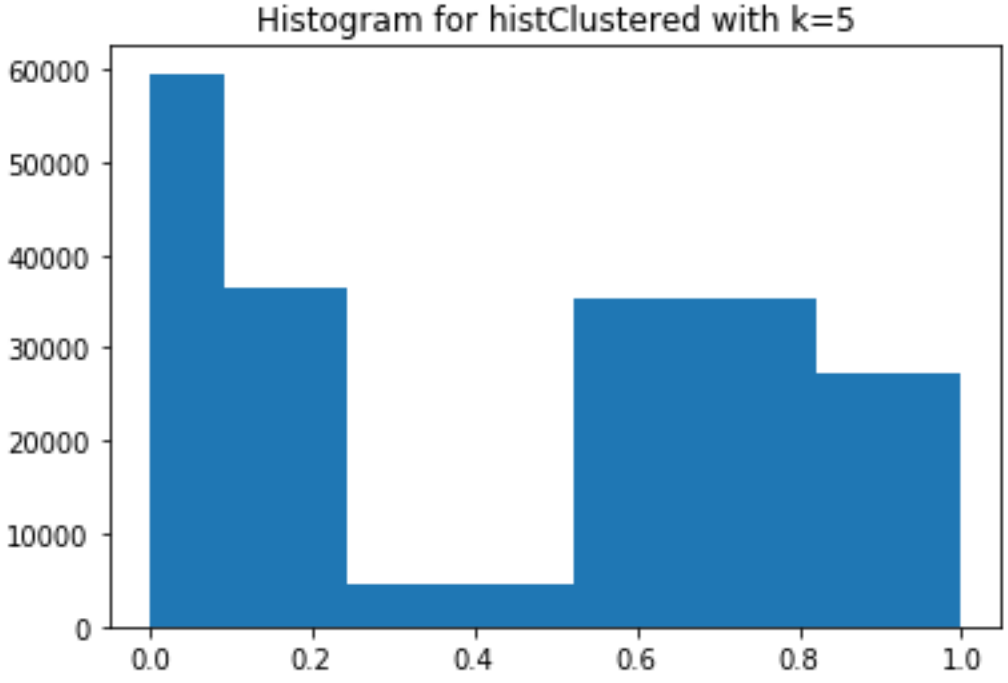
1(e)

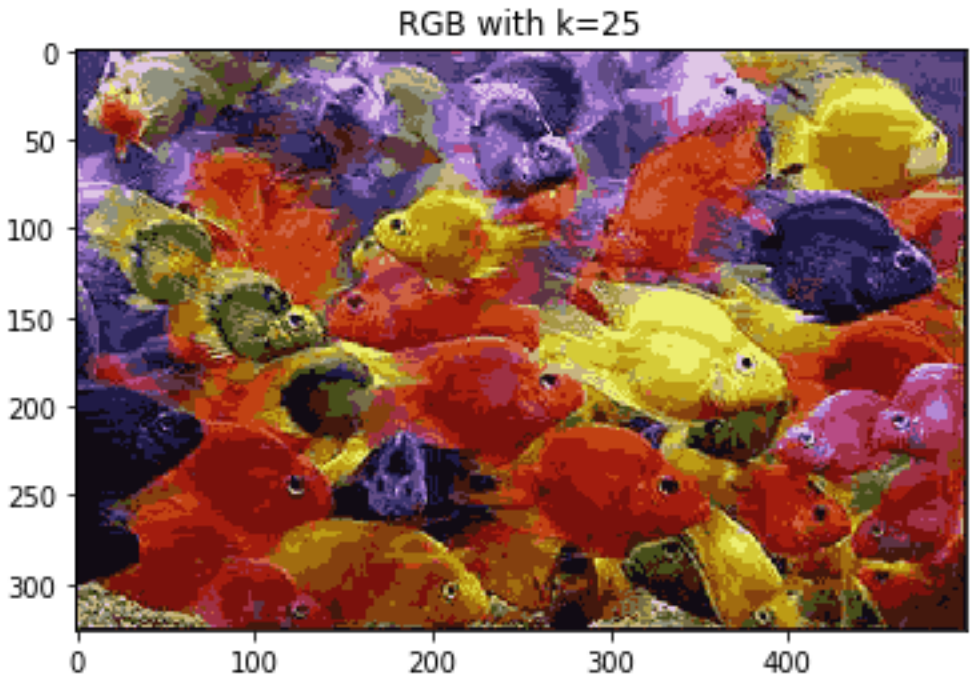
| SSD Error | RGB | HSV |
|---|---|---|
| K=5 | 50069528 | 15408428 |
| K=25 | 40730802 | 3952993 |

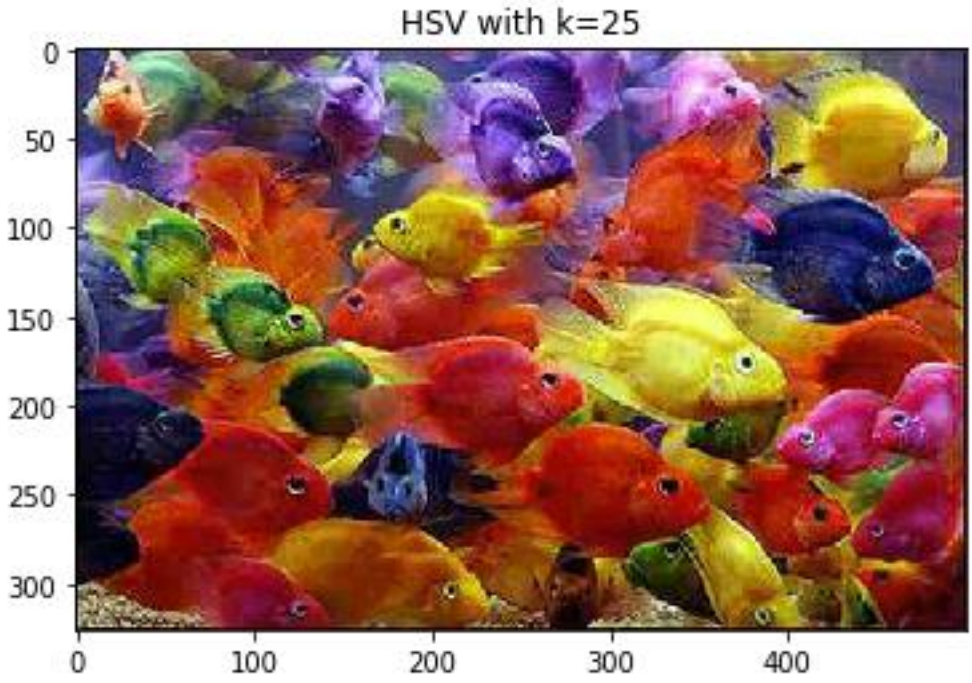

RGB with k=5
0
50
100
150
200
250
300
0
100
200
300
400

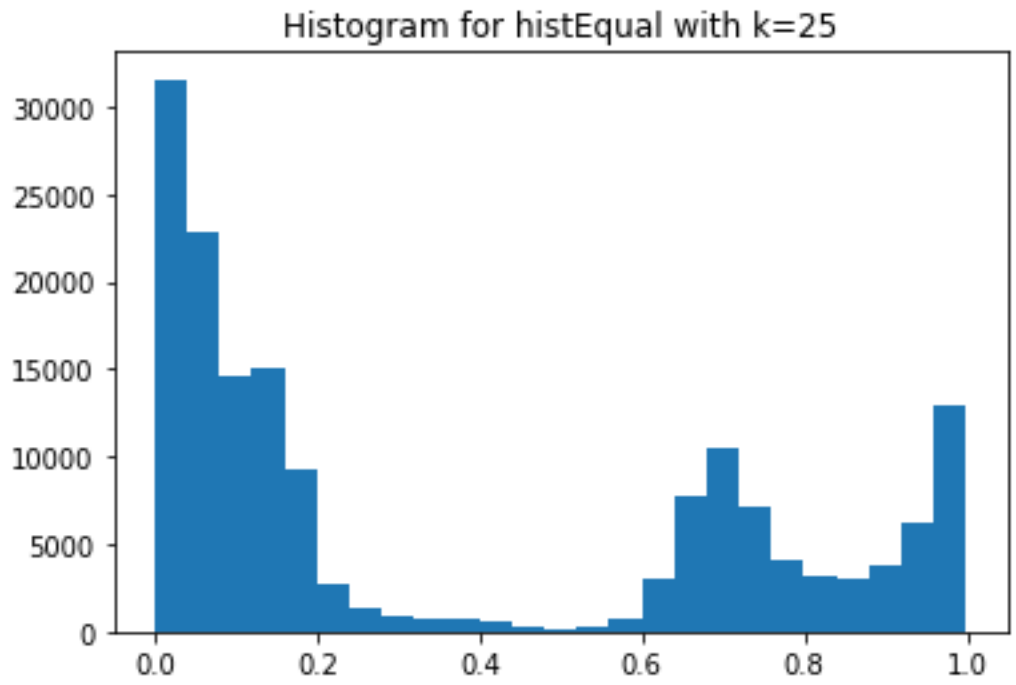

HSV with k=5
0
50
100
150
200
250
300
0
100
200
300
400

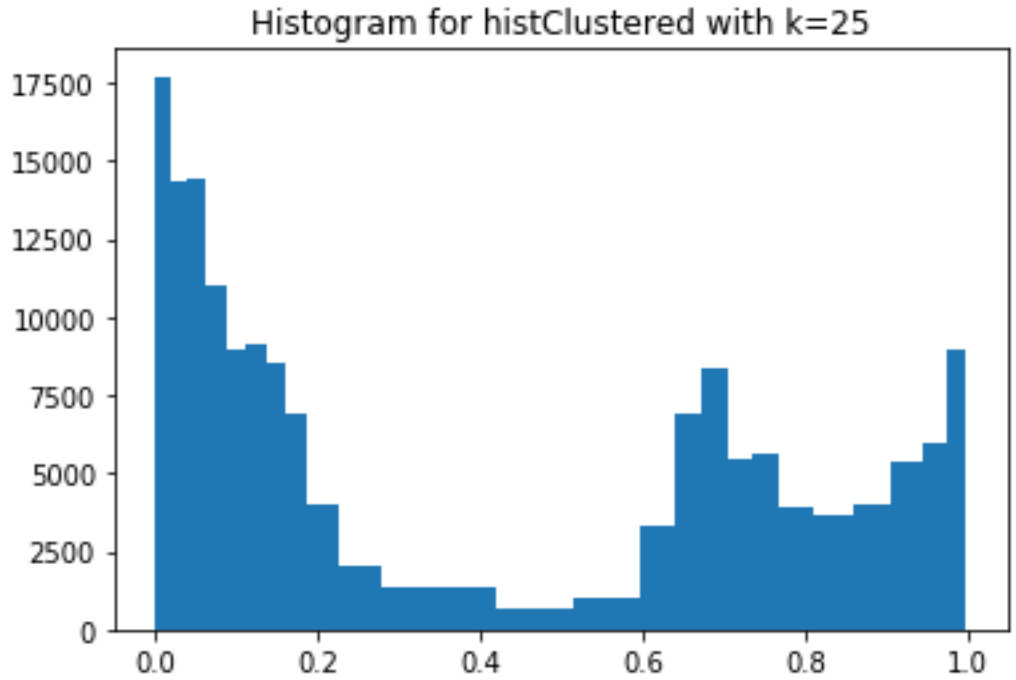

Histogram for histEqual with k=5
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


Histogram for histClustered with k=5
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0


RGB with k=25
0
50
100
150
200
250
300
0
100
200
300
400


HSV with k=25
0
50
100
150
200
250
300
0
100
200
300
400




(f)

histEqual represents actual distribution of Hue value of the original image. histClustered represents how we cluster the Hue value into different groups

In RGB space, we will have k colors in the result image after we do the quantization. In HSV space, we will have more than k colors because we only quantize the Hue value and keep Saturation and Value channel the same as the input

For larger value of k, we have more quantization levels, and therefore the result image is more colorful and more like the original image

We may have different initialization across different runs of K-Means algorithms and therefore the algorithm could converge to different local maxima. In such cases we will have different result images.

2

(a) (1) First convert the original image into a binary edge image. We can use canny edge detector.

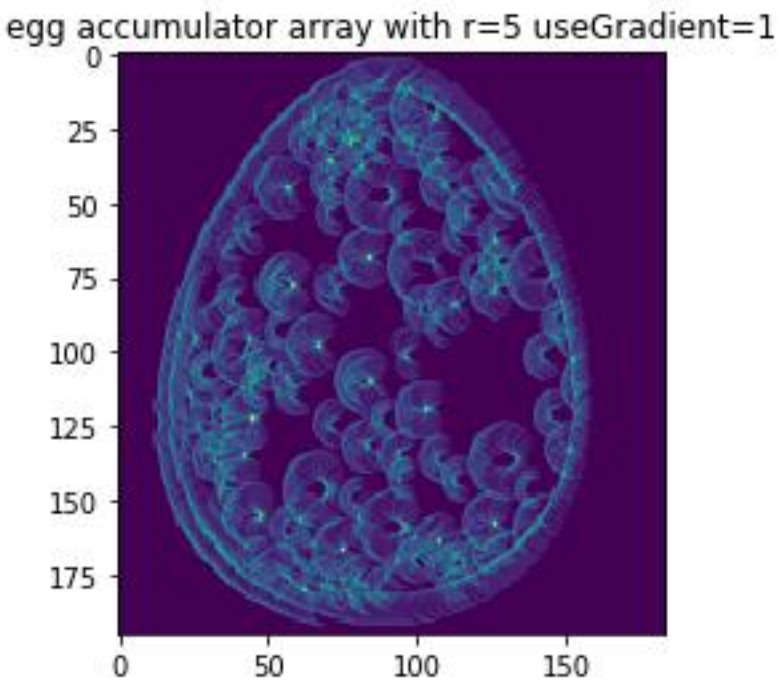
(2) Take the votes from each edge pixel and build the accumulator array.

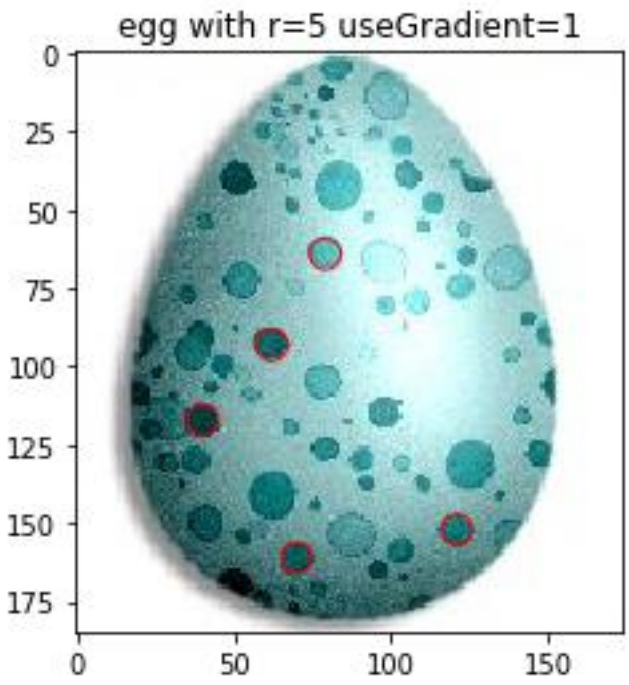
If useGradient=0, one edge pixel votes for a circle in Hough space.

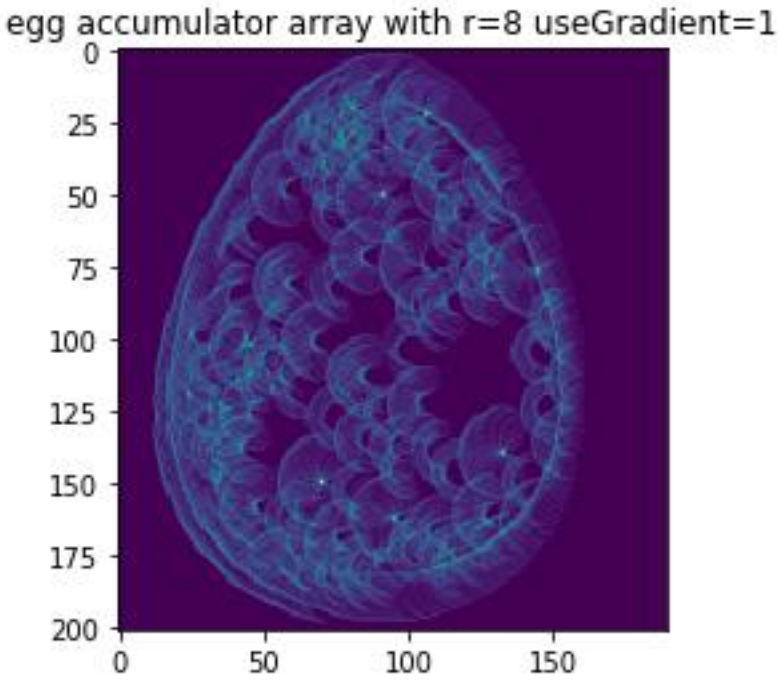
If useGradient=1, one edge pixel votes for a point decided by the polar form of the circle equation

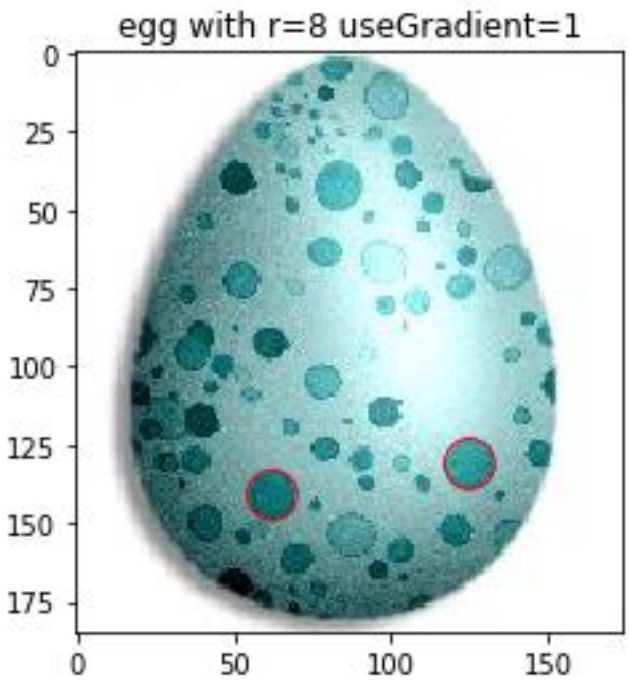
(3) Choose the centers with the number of votes greater than a preset threshold

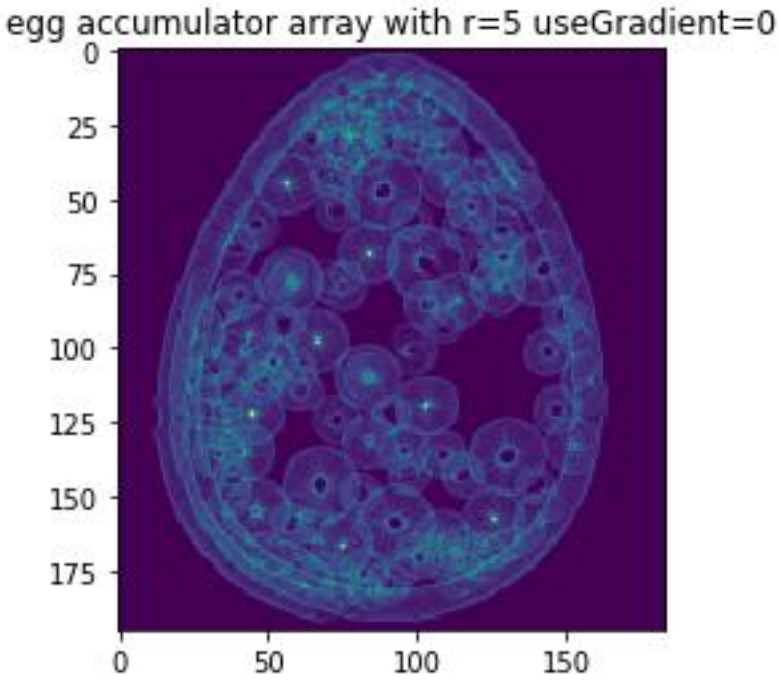
(b)

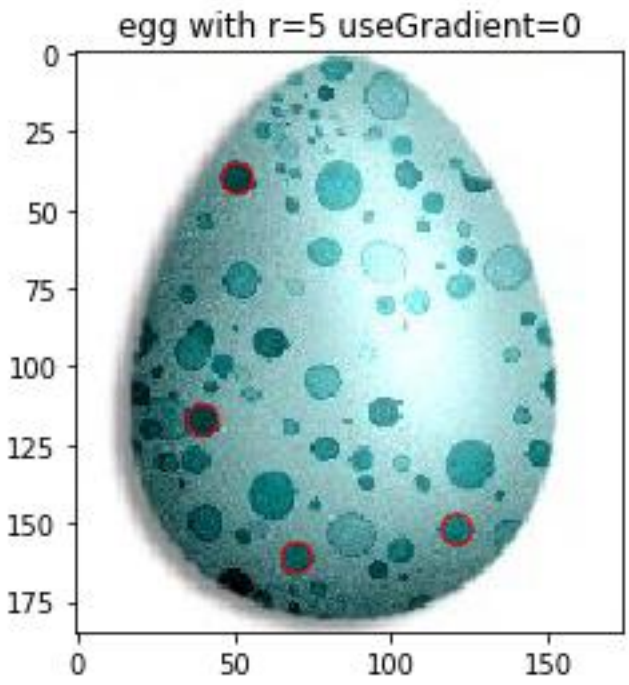


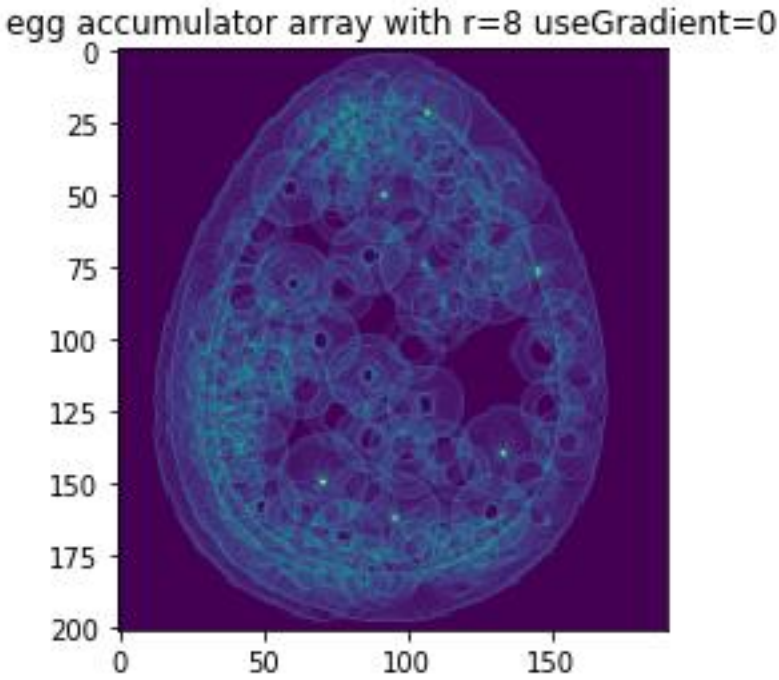

egg with r=5 useGradient=1
0
25
50
75
100
125
150
175
0
50
100
150

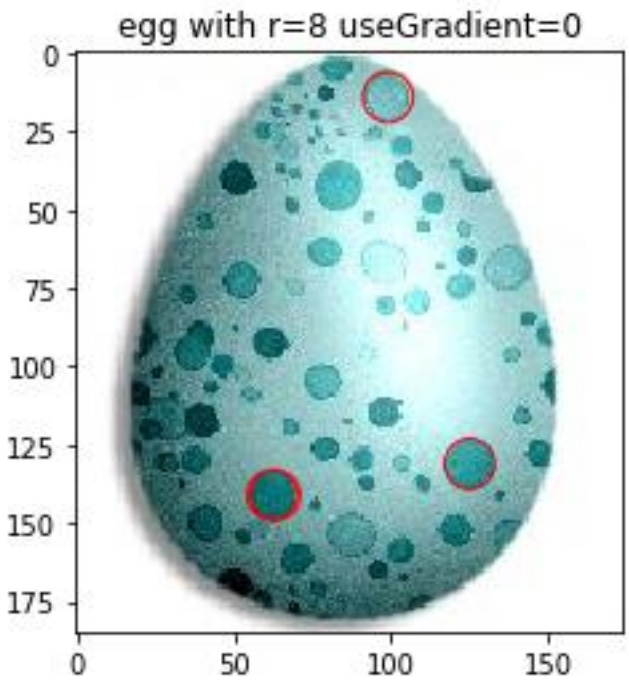

egg accumulator array with r=8 useGradient=1
0
25
50
75
100
125
150
175
200
0
50
100
150

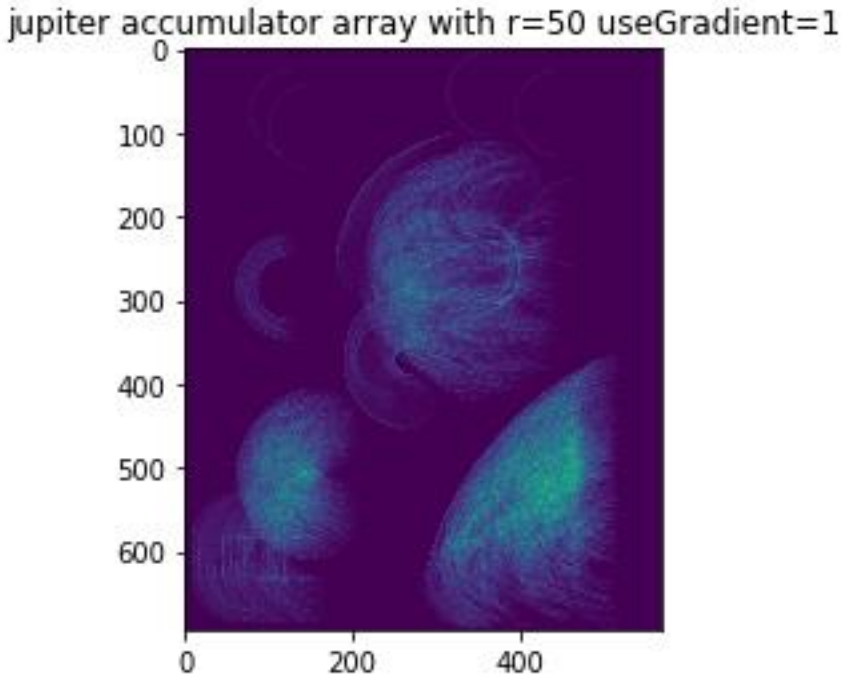

egg with r=8 useGradient=1
0
25
50
75
100
125
150
175
0
50
100
150


egg accumulator array with r=5 useGradient=0
0
25
50
75
100
125
150
175
0
50
100
150

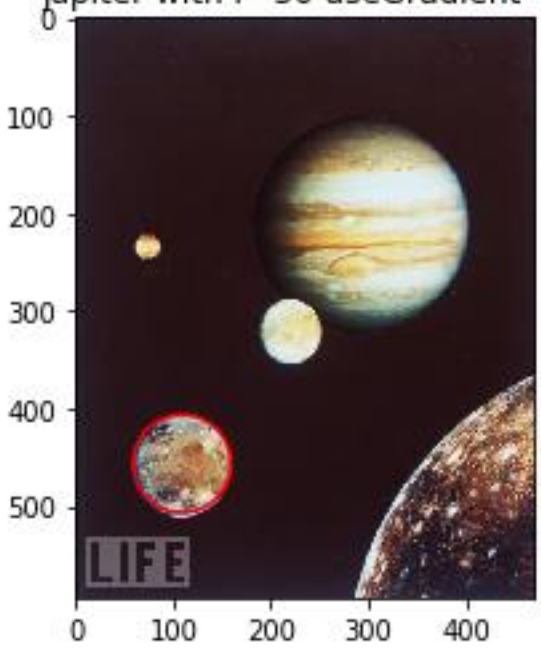

egg with r=5 useGradient=0
0
25
50
75
100
125
150
175
0
50
100
150


egg accumulator array with r=8 useGradient=0
0
25
50
75
100
125
150
175
200
0
50
100
150

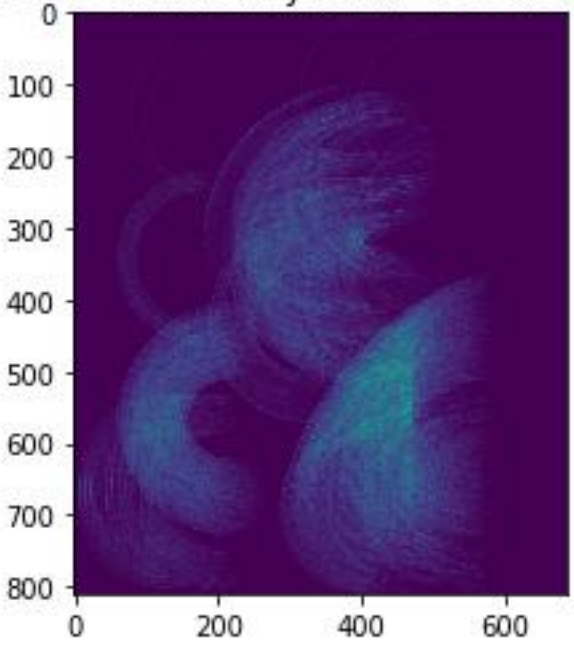

egg with r=8 useGradient=0
0
25
50
75
100
125
150
175
0
50
100
150


jupiter accumulator array with r=50 useGradient=1
0
100
200
300
400
500
600
0
200
400

jupiter with r=50 useGradient=1

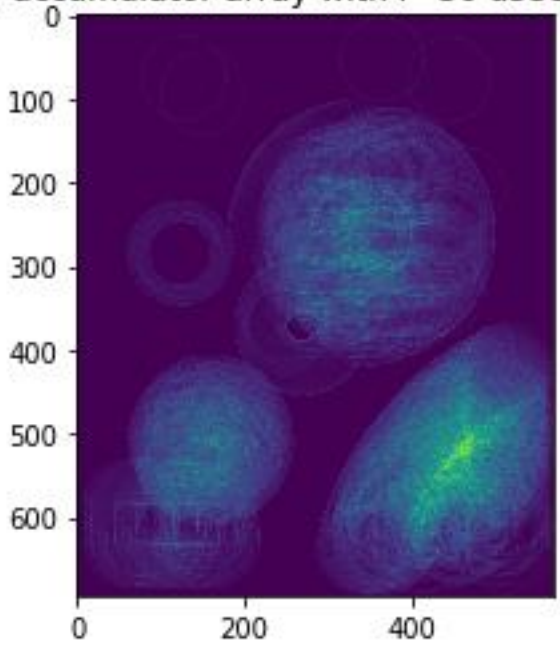



jupiter accumulator array with r=108 useGradient=1



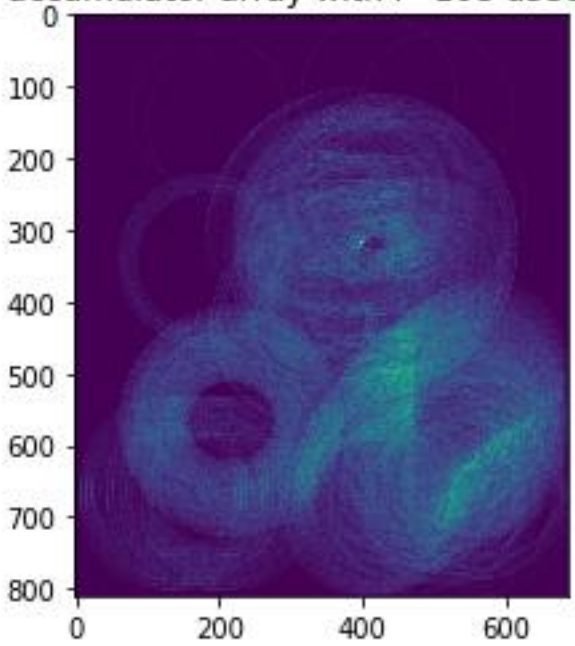
jupiter with r=108 useGradient=1

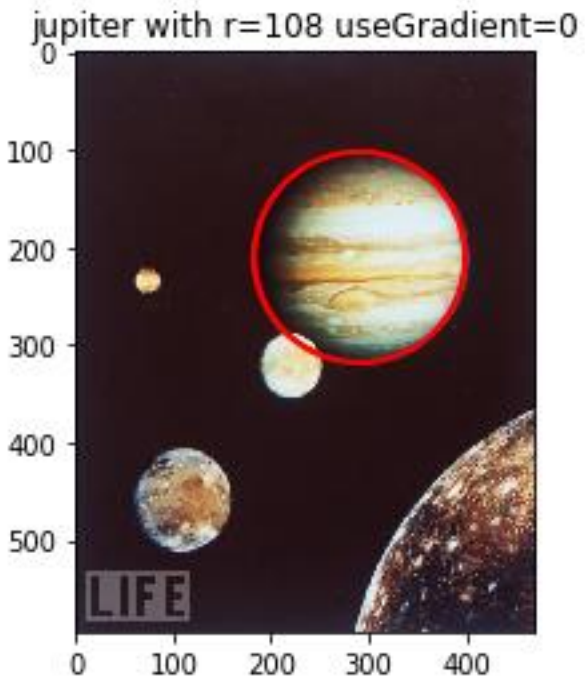

jupiter accumulator array with r=50 useGradient=0



jupiter with r=50 useGradient=0

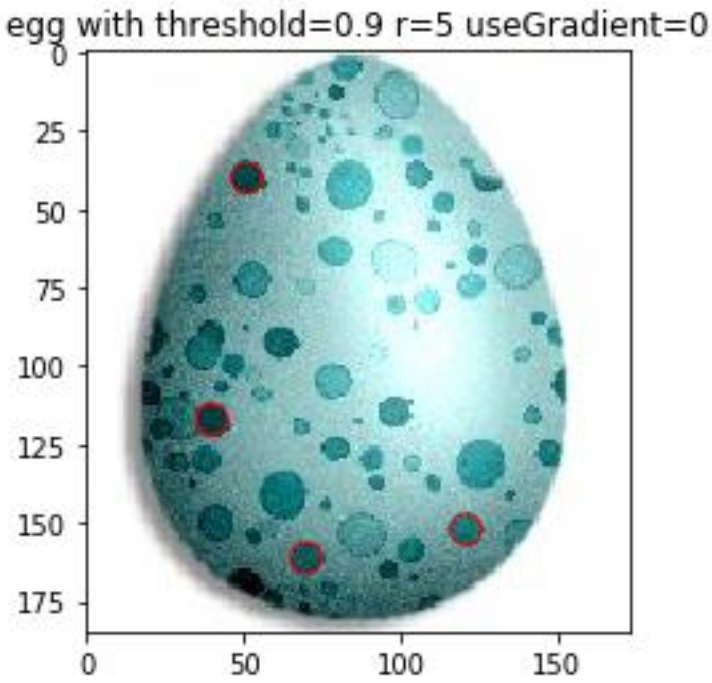

jupiter accumulator array with r=108 useGradient=0

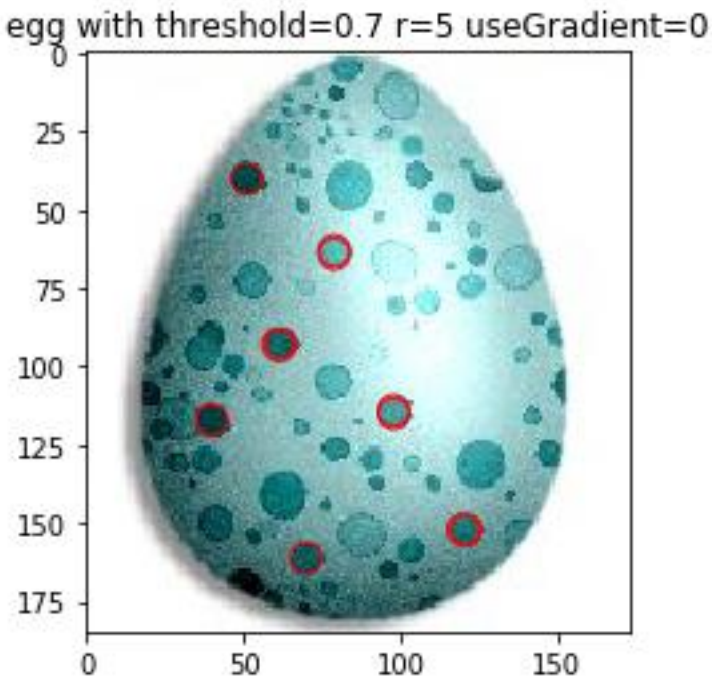


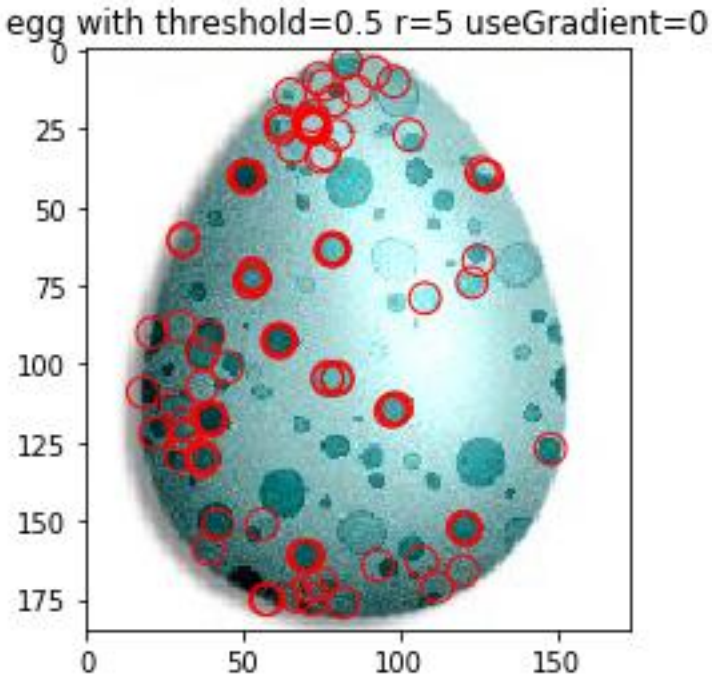



(c) The accumulator array captures all the votes in the Hough space. Here, lighter color represents more votes while darker color means less votes. And we can determine there is a circle in the original image when we find a peak in the accumulator array

(d) We can change the threshold value when we decide whether there is a peak in the accumulator array to control the number of circles to be present. When we decrease the threshold value, the number of circles will increase

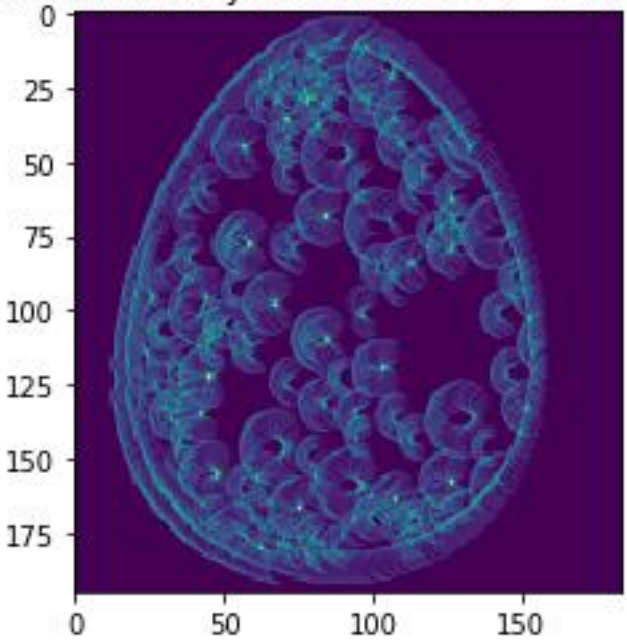




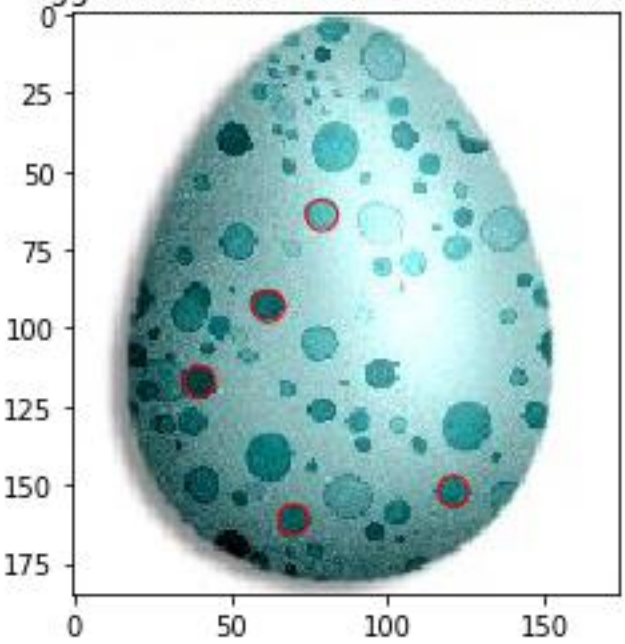




(e) When we increase the bin size, we are increasing the granularity and therefore the result will be less accurate

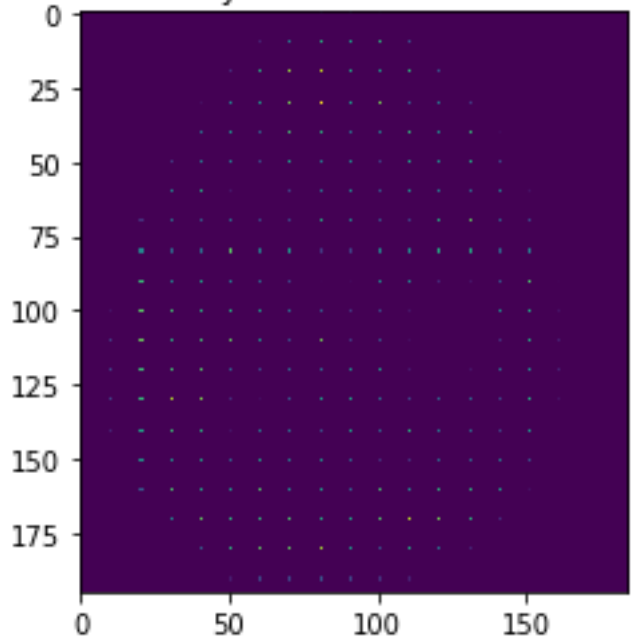
egg accumulator array with binsize=1 r=5 useGradient=1




egg with binsize=1 r=5 useGradient=1



egg accumulator array with binsize=10 r=5 useGradient=1




egg with binsize=10 r=5 useGradient=1