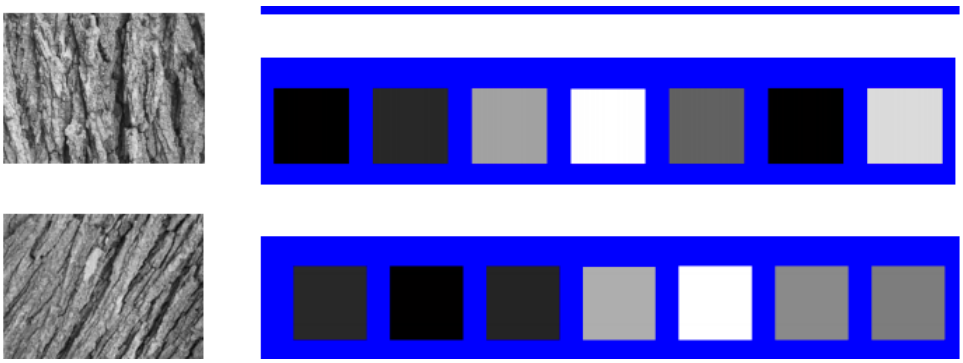
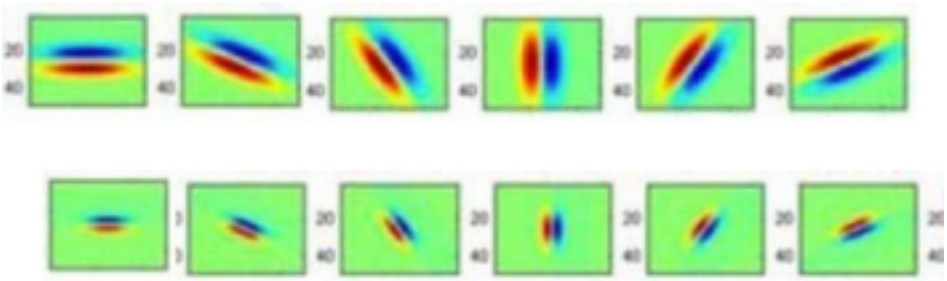
Taylor Yang
998055387

1 if we treat the whole bank as one vector array. Then Yes, the result will look different.



However if we just compare the difference then its rotationally invariant, because this just a scale different.



2You will make two semi-circle. So the result will cut these two circles in half and the left ones will generate one group and the right ones will generate another group.

3 We will use mean-shift, to recover. For *k*-means it will difficult to recover because we need the number of clusters to be pre-determined. It has two significant limitations. First, the *k*-means algorithm requires that the number of clusters to be pre-determined. For a continuous vote space, which means a vote, space where any bin can take on any possible vote value. It is difficult to acclimate an appropriate cluster number, so it's difficult to use k means. Moreover, k-means in general, incapable of identifying non-convex clusters.
For graph cuts, Because it wants balanced partitions and time complexity can be high so we don't want use it.

4Because we already know the blobs and have access to them, we will use k-means which
First get the
Width=columns
Height=rows
ratio= width/height
for i=1: (# of pixels
  `[idx,C]=kmeans(X,blobs);`
find the distance that close to the k means value C

Group these values by the Cluster Center C
end

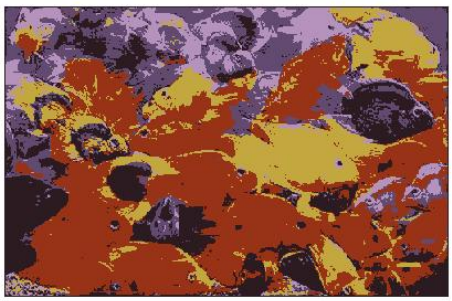
2 Programming

1. Color quantization with k-means

For lower value k=5
RGB error1 =462330026
HSV error2=6.108184744956442e+09

For higher (k=25) value
RGB error1 = 134857035
HSV error2=6.108222267724505e+09

RGB1,k=5

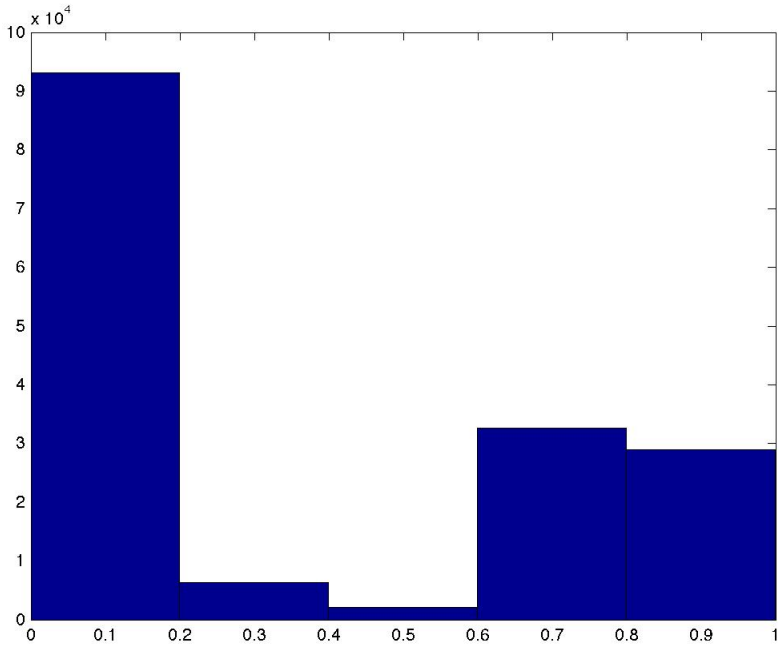


HSV1,k=5

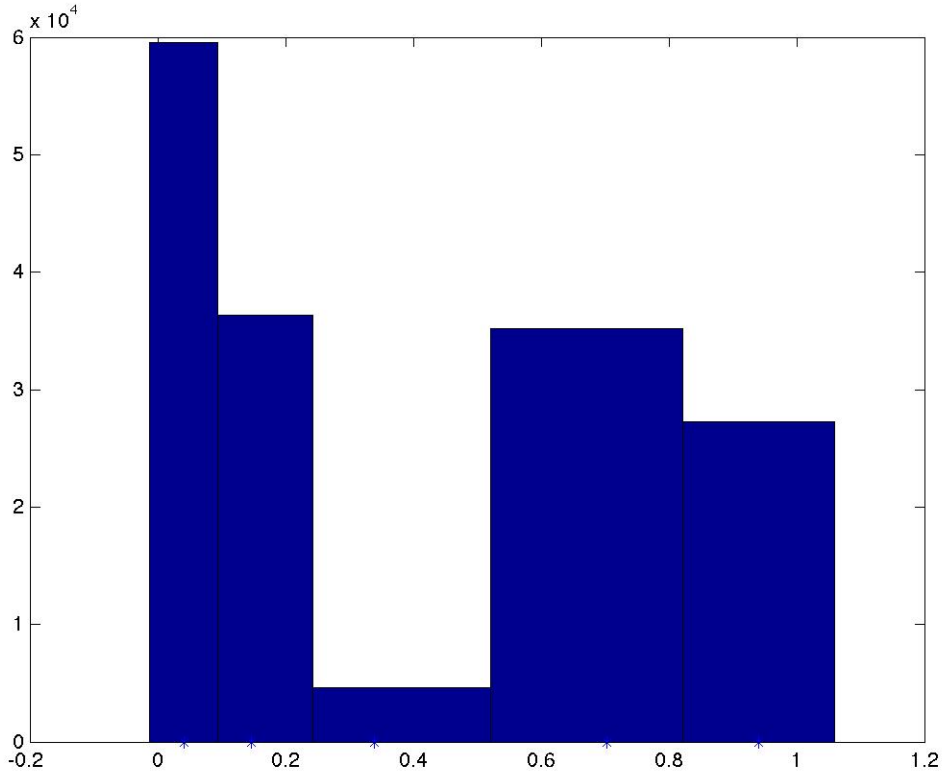
RGB2,k=25

HSV2,k=25

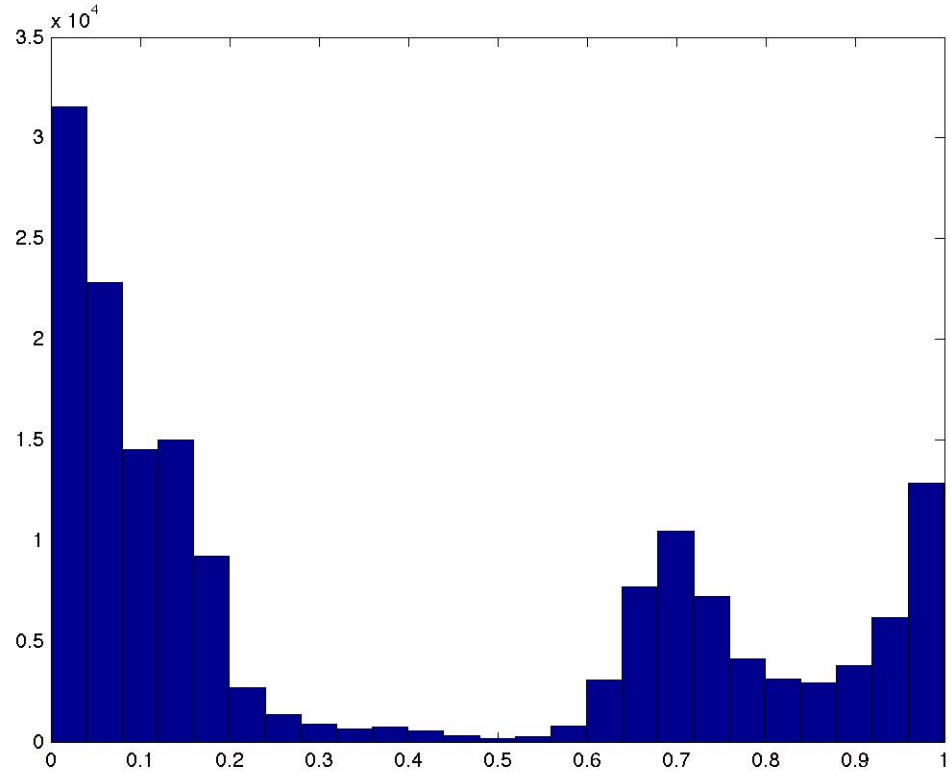
(f)
For lower value k=5
use k equally-spaced bins (uniformly dividing up the hue values),



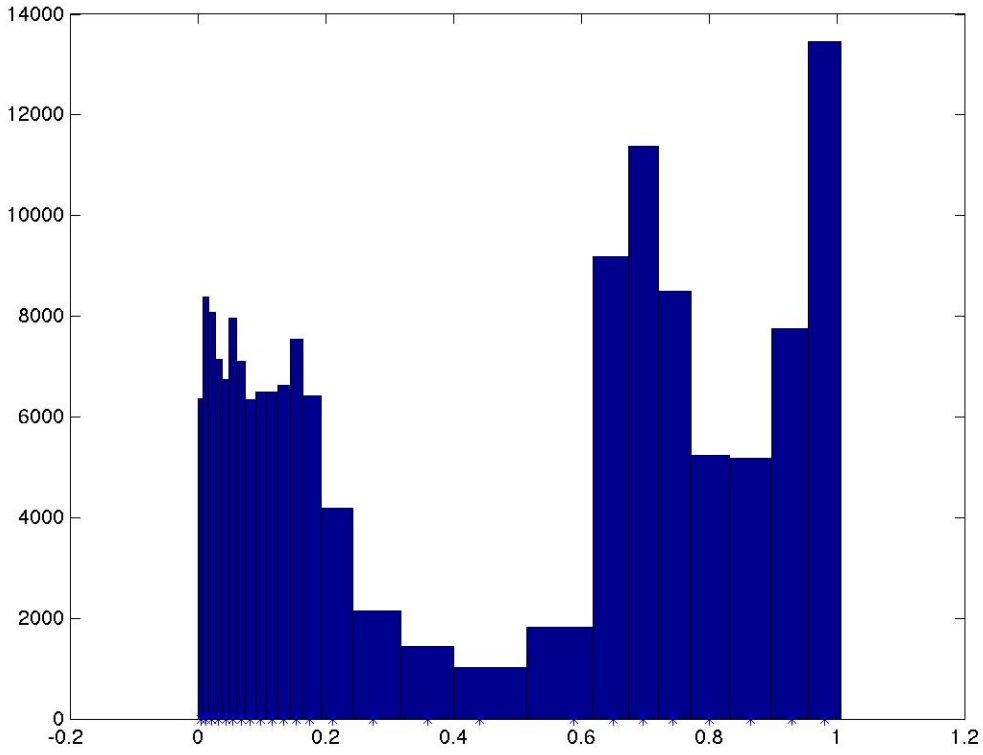
use bins defined by the k cluster center memberships



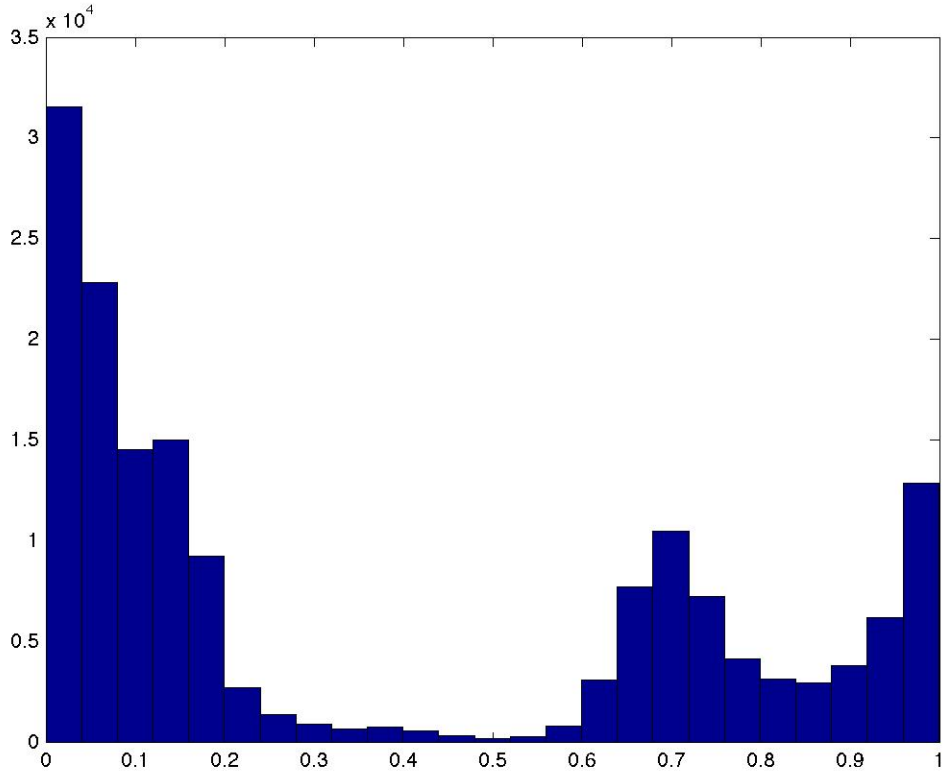
For higher (k=25) value
use k equally-spaced bins (uniformly dividing up the hue values),

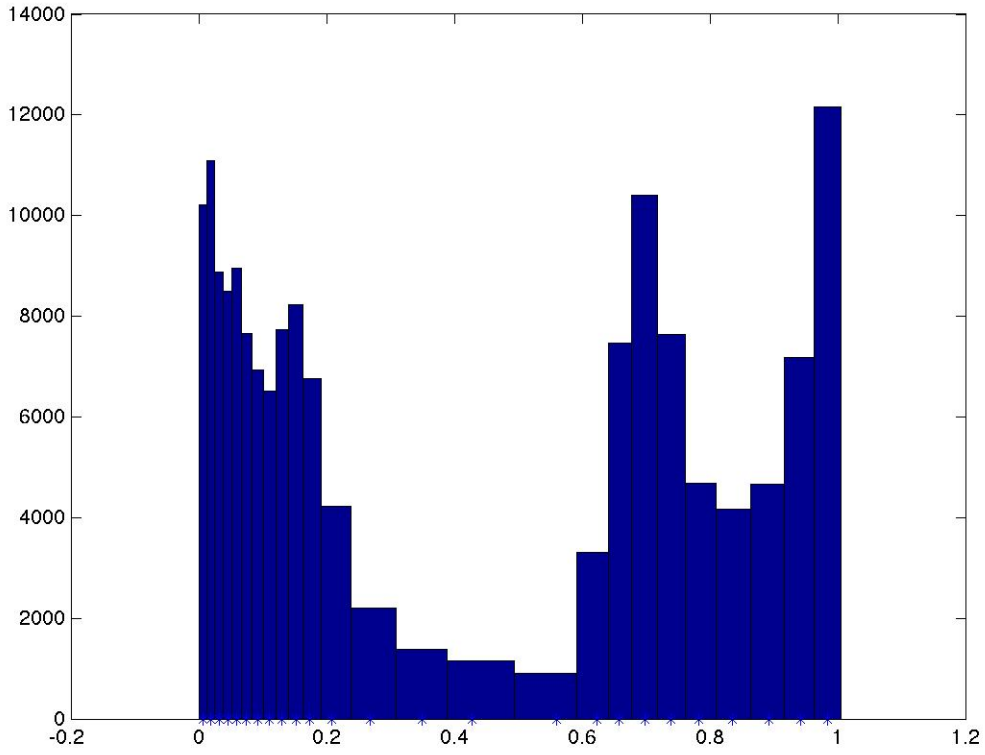


use bins defined by the k cluster center memberships



Second Run using k=25



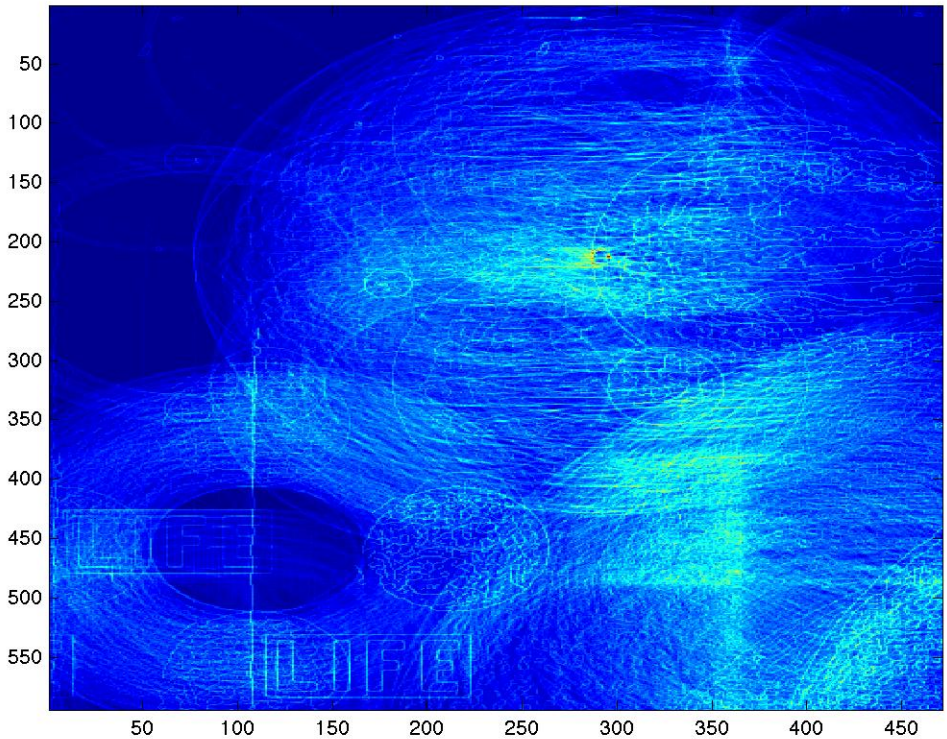


f)
From the above histograms we can see that use k equally spaced bins (uniformly dividing up the hue values) histogram looks a litter similar to the histogram use bins defined by the k cluster center memberships. But because we use bins defined by the k cluster membership so the histogram will look different because k is slightly different each time and members group to that k will different.
We can also find out from the plot.jpg that when k value is small the picture is less colorful because we limit the color space.
The K values for each run with same k=25, is slightly different, because we randomly choose the cluster at the beginning.

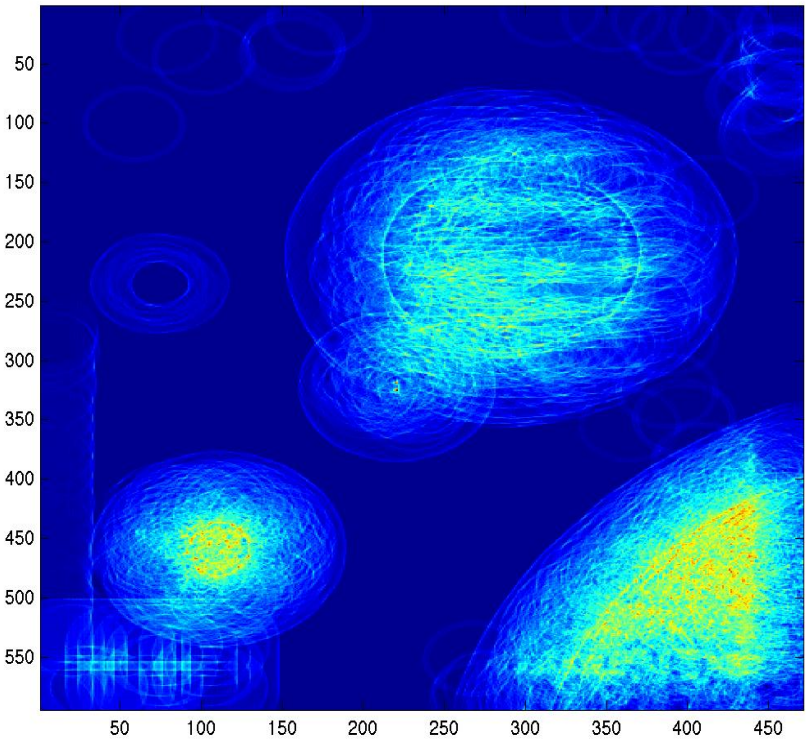
2. Circle detection with the Hough Transform [

a)
For my detection with the Hough transform function, I first use grayscale function to change me RGB image to a Black and white binary image. Then I use canny edge function to find out the edges. I then set up an accumulator space which is initialized to be 0 and set as the same size of the image. When the useGradient flag is off, for every edges and for theta from 0 to 360 degrees , I will use the formula a=round(x+radius*cos(theta));b=round(y+radius*sin(theta)); This is generate from the equation of a circle (x-a)^2+(x-b)^2=radius^2. Base on the position of b and a, I will increment the value of my accumulator array at that position a and b. I will then normalize my array so I can more easily see the differences and find the maximum value. Finally, I will search for the local maxima cells. Instead of just find the maximum, I will try to find the values which are above 0.9, this will give me around top 10 maximum value of my accumulator array. At last, all these values that I find are the ones with the highest probability of my circles' centers. When the useGradient flag is set to one, Instead of trying all 360 degrees, I will use gradient function for all my pixels, and use atan2 to get the correspond theta. I will then use these thetas for my a and b formula.

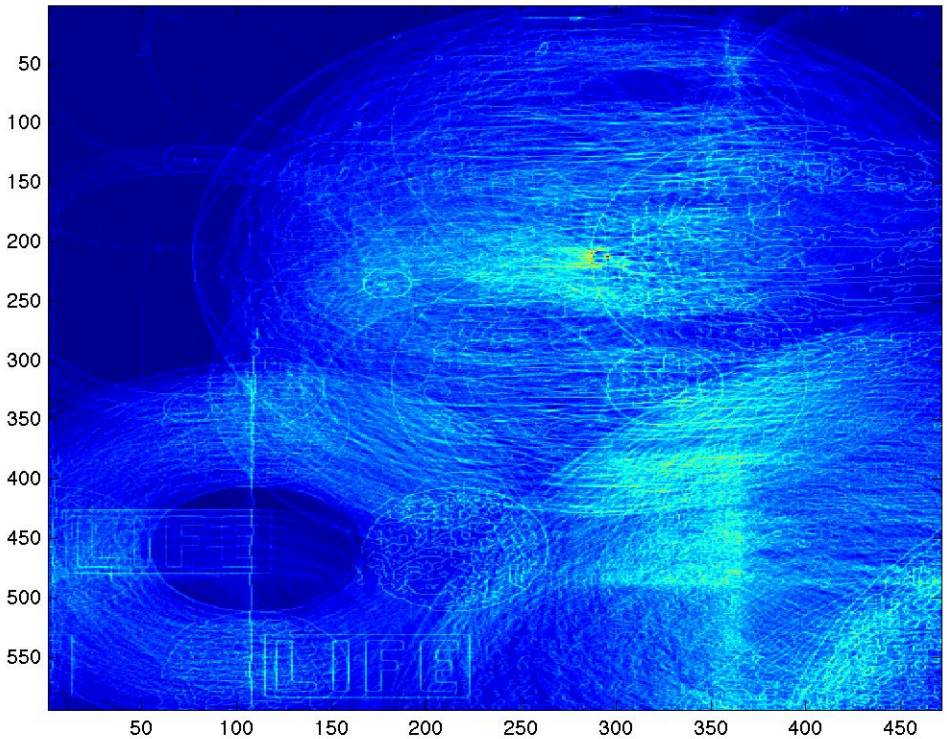
Gradient =0 With radius = 105



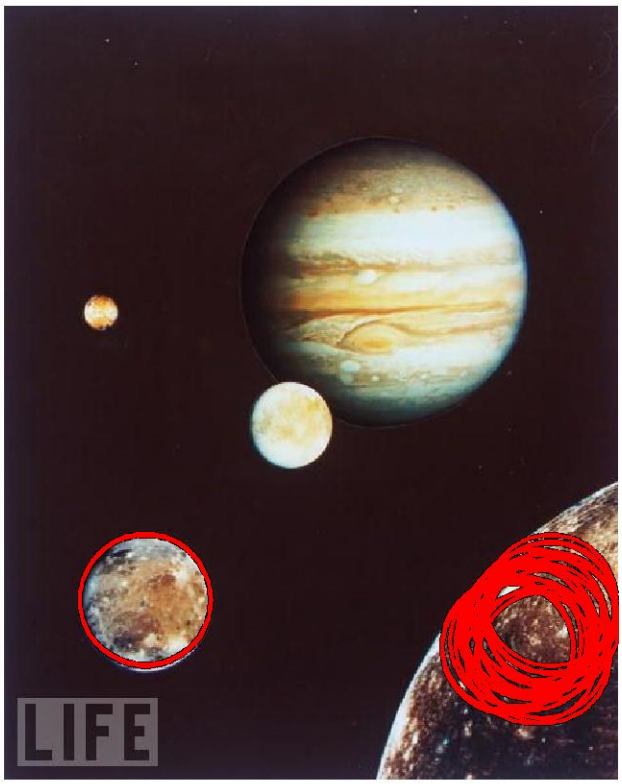
Gradient =0 Radius =30

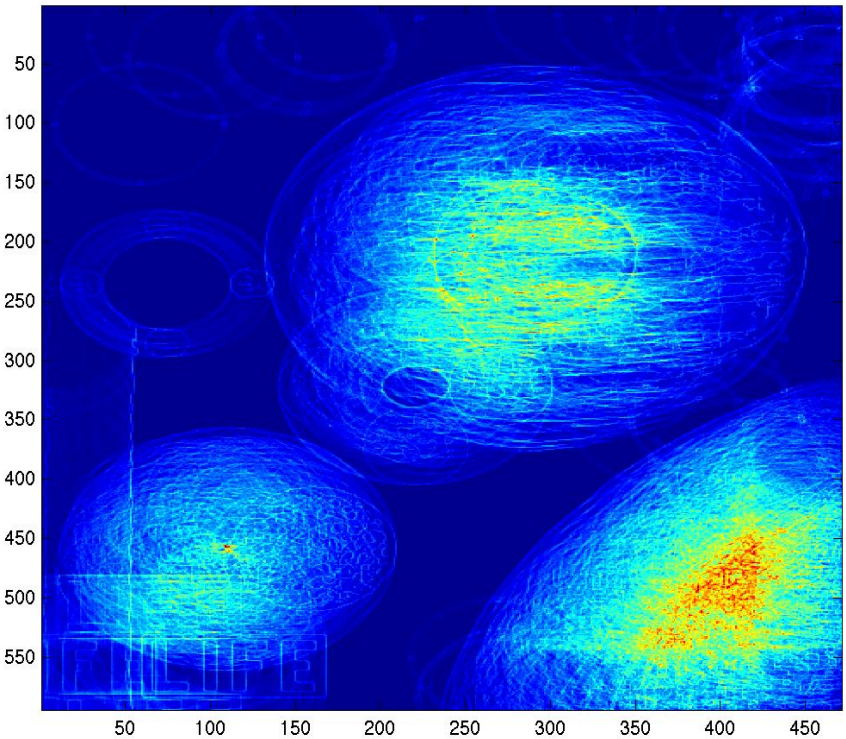


Gradient =1 Radius =105

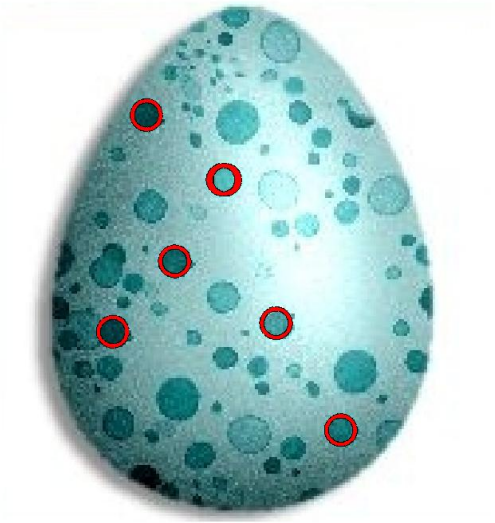


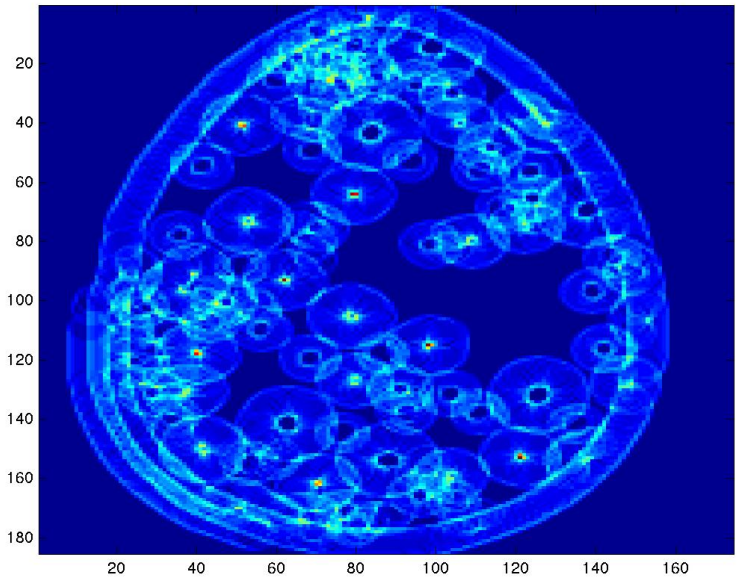
Gradient =1 with radius =50



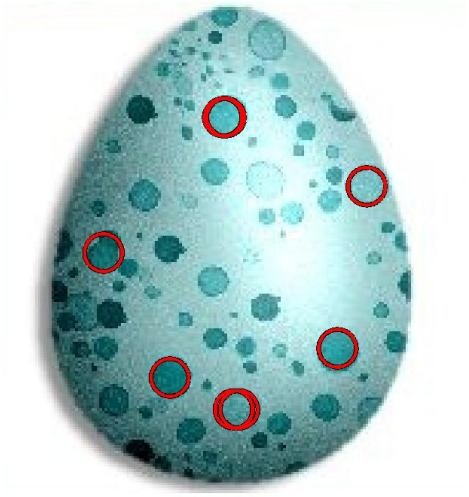


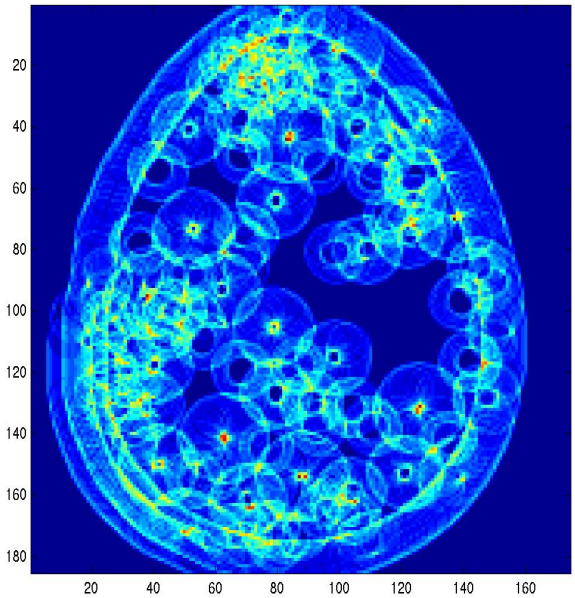
Gradient =0 with radius =5





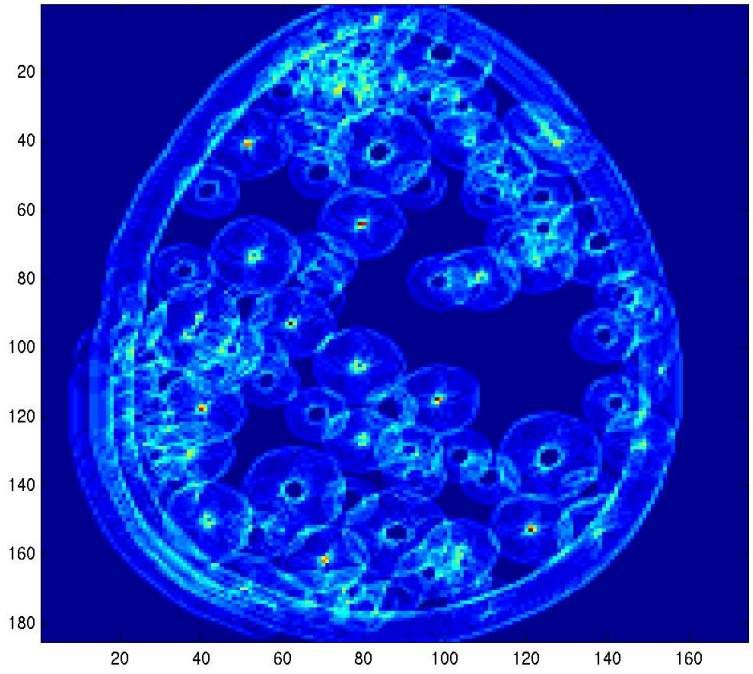
Gradient =0 with radius =7



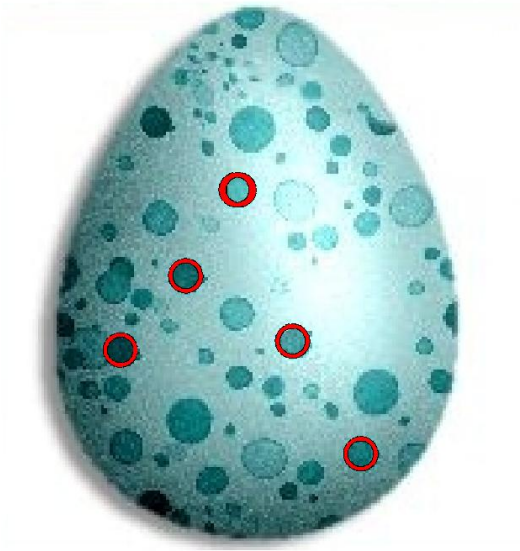


c) We can that the accumulator array use radius 7 drew many circles, and the high value(red dot) are exactly the centers of the radius 7 circle from the original image. However, there are some high values at the upper left because that place we have many curves that also fits radius 7 so the machine misunderstood them as some circles with radius 7. Eventually we didn't choose them because they are not as high as the actual centers.

d)Gradient =1 with radius =5

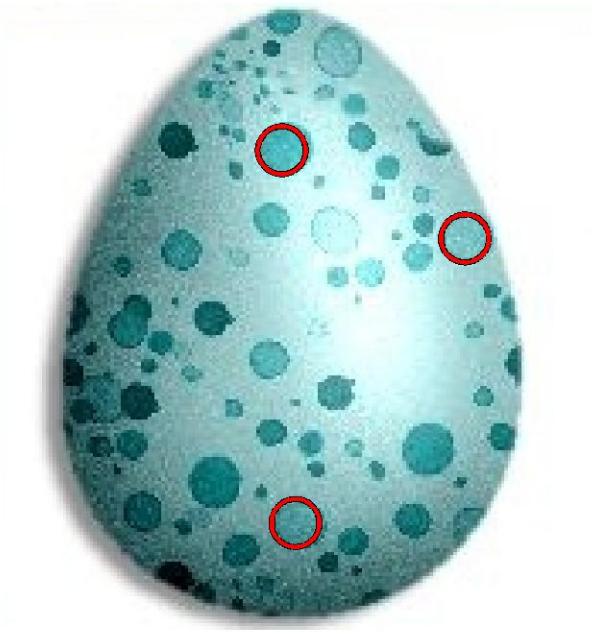


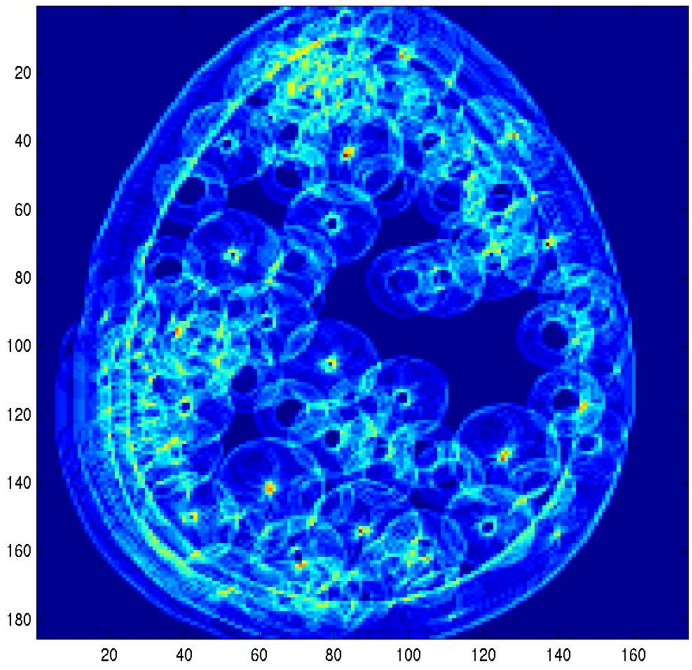
From the above Hough accumulator, we can guess there should probably have 5- 6



circles

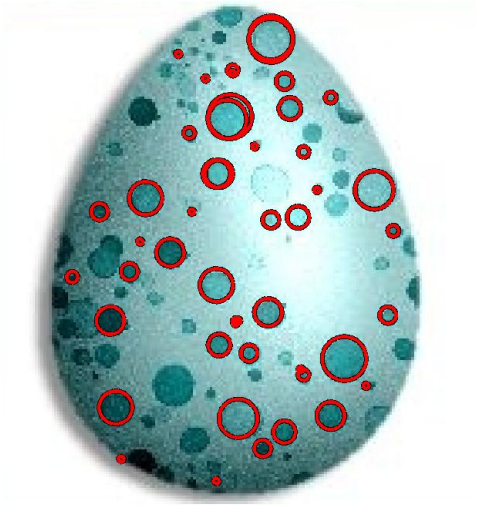
Gradient =1 with radius =7





e)

You can have a very fine-grained accumulator array or a coarse one. This question asks you to play around with amount of discretization/quantization/fine-graining(same thing said in different ways, in this context) that you provide your accumulator array with.



3