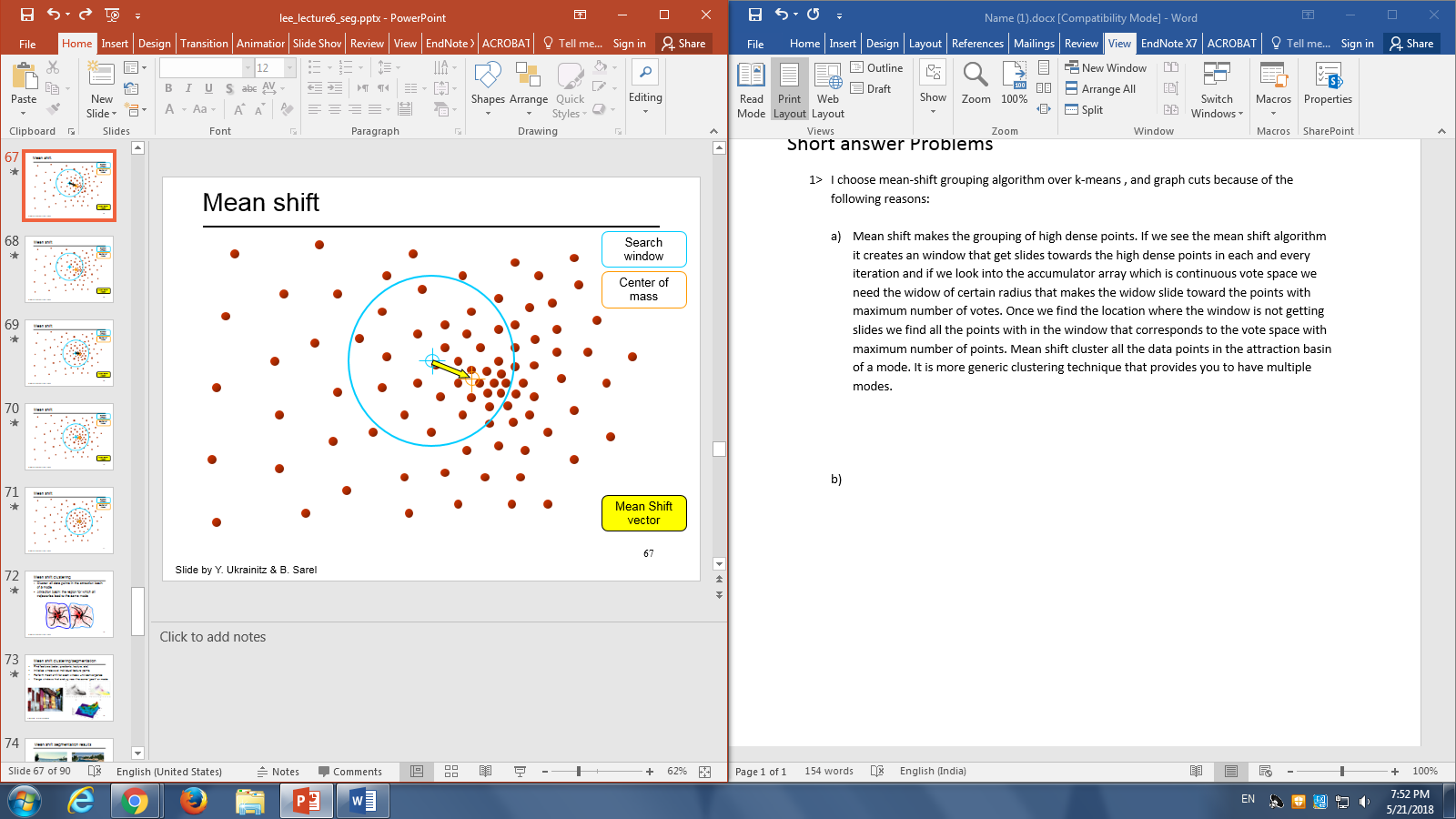
Name : Sai Kopparthi

Student ID: 915623695

Short answer Problems

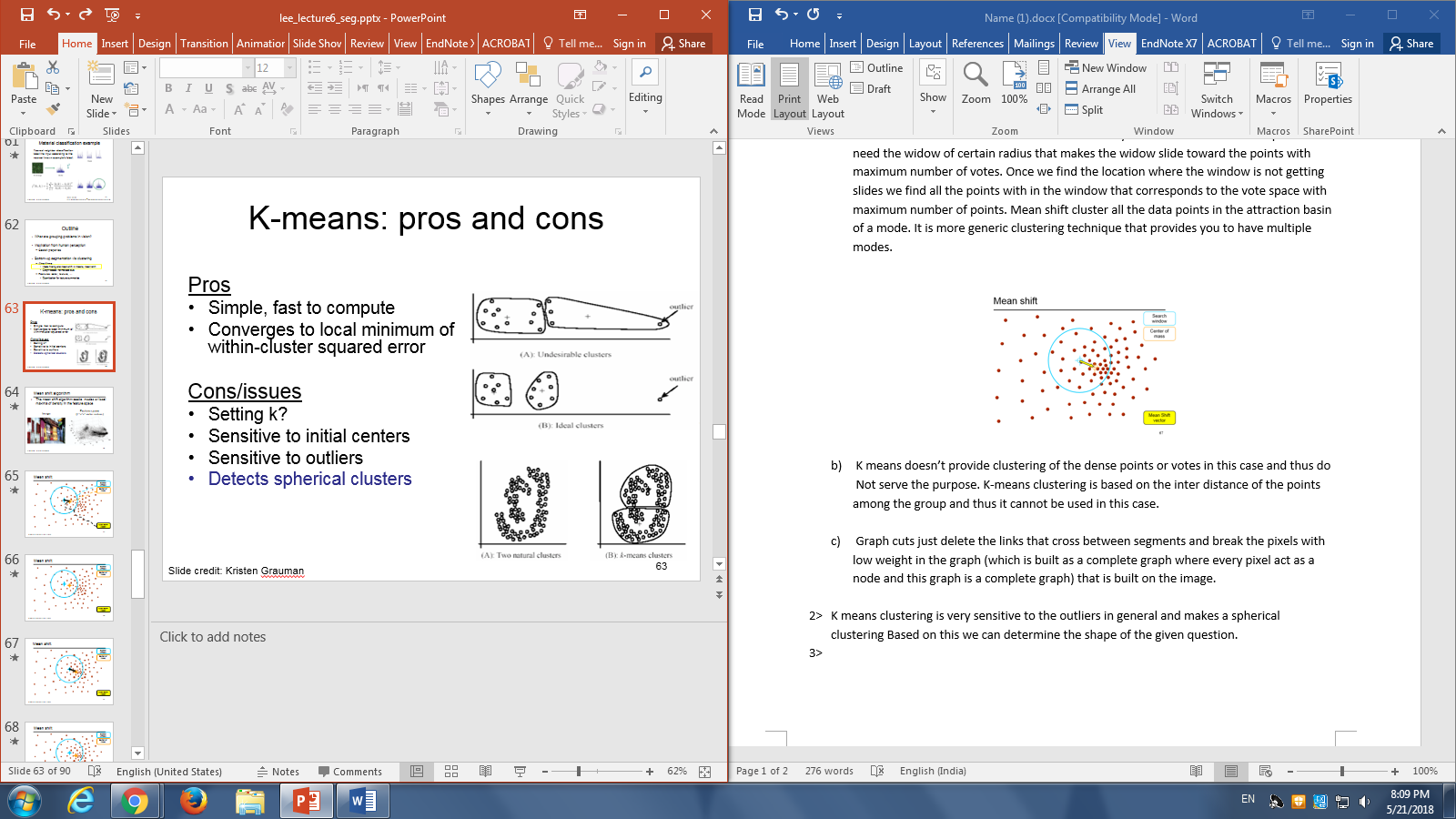
1. I choose mean-shift grouping algorithm over k-means , and graph cuts because of the following reasons:
2. Mean shift makes the grouping of high dense points. If we see the mean shift algorithm it creates an window that get slides towards the high dense points in each and every iteration and if we look into the accumulator array which is continuous vote space we need the widow of certain radius that makes the widow slide toward the points with maximum number of votes. Once we find the location where the window is not getting slides we find all the points with in the window that corresponds to the vote space with maximum number of points. Mean shift cluster all the data points in the attraction basin of a mode. It is more generic clustering technique that provides you to have multiple modes.



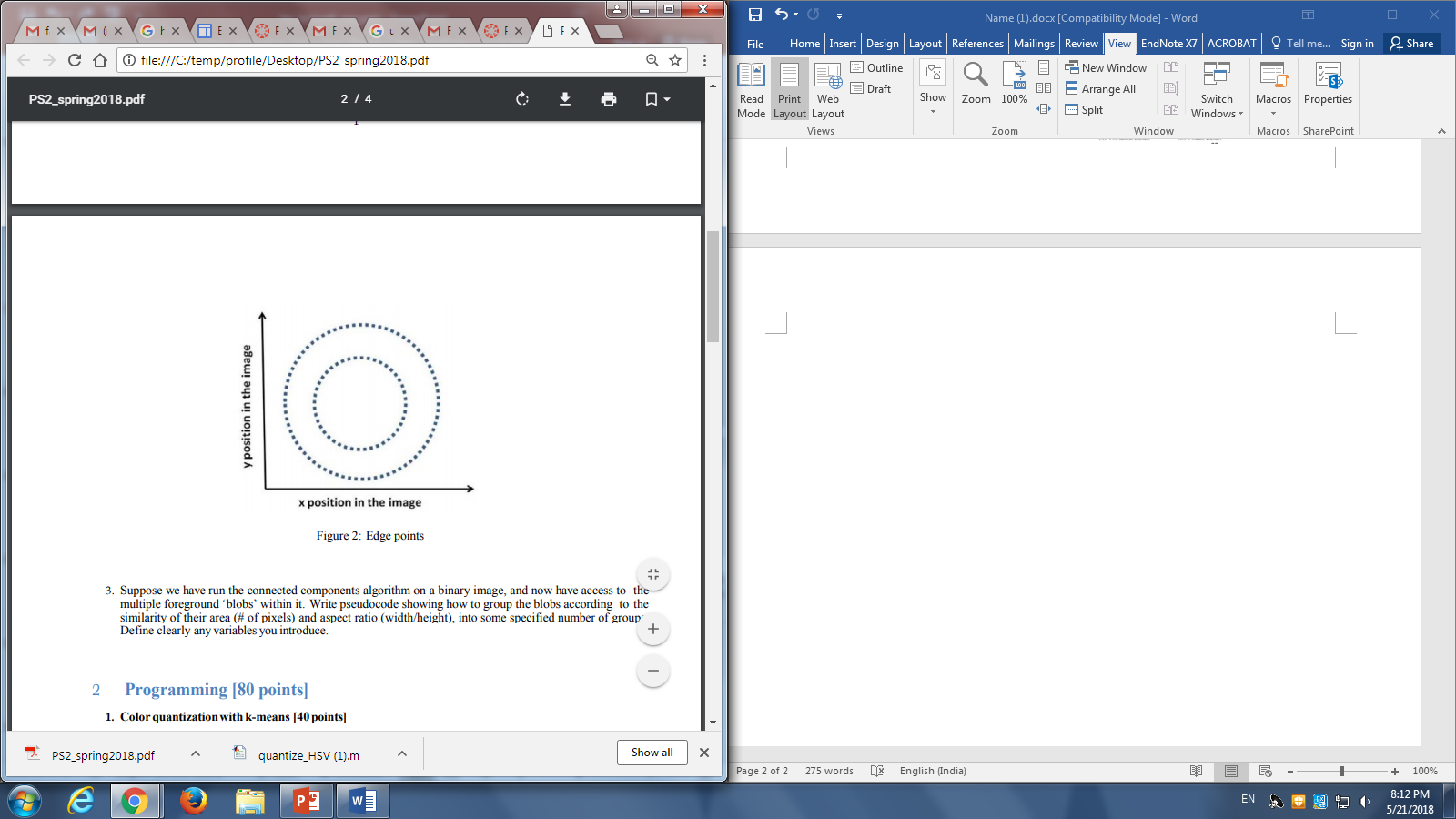
1. K means doesn’t provide clustering of the dense points or votes in this case and thus do

Not serve the purpose. K-means clustering is based on the inter distance of the points among the group and thus it cannot be used in this case.

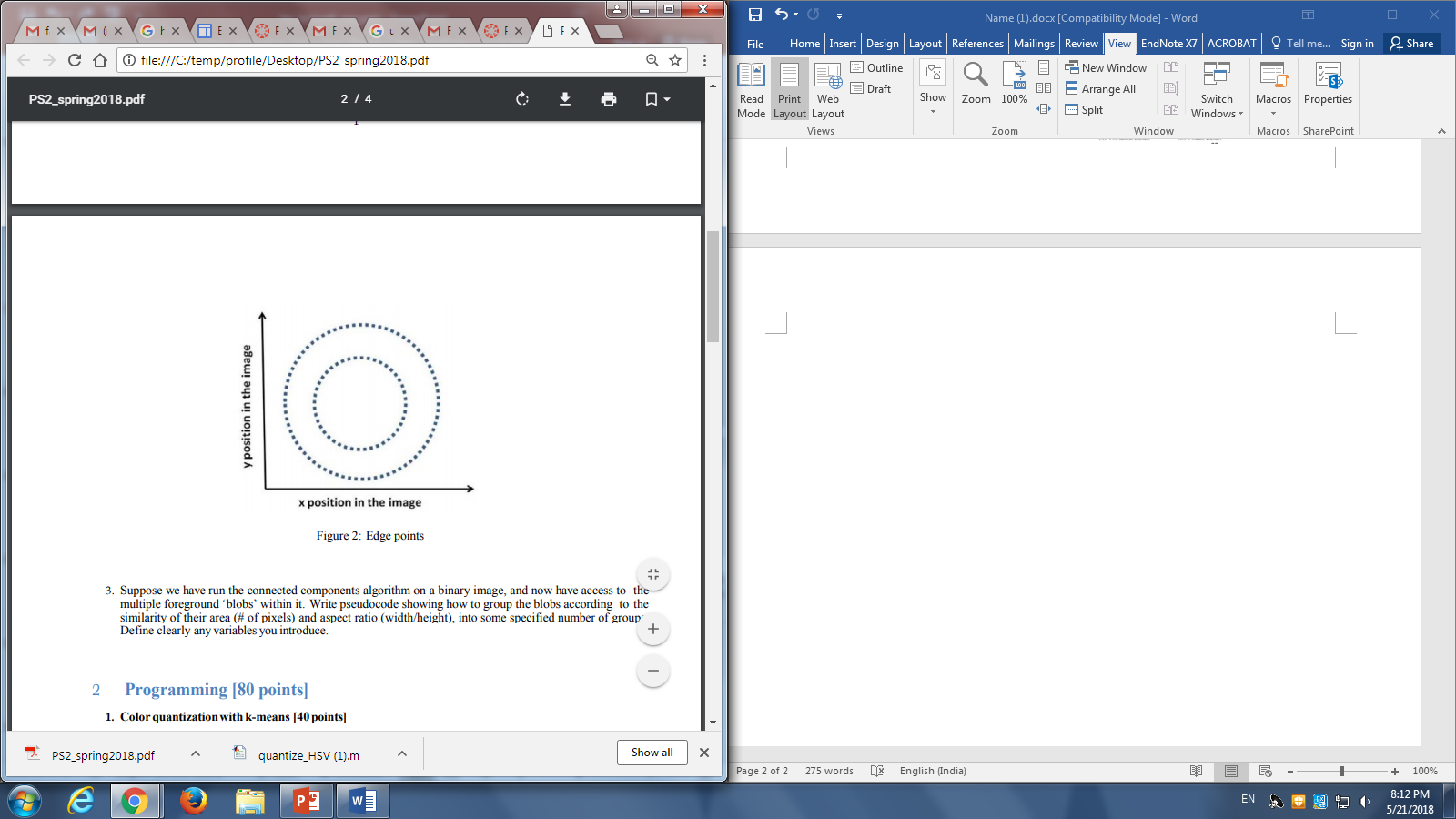
1. Graph cuts just delete the links that cross between segments and break the pixels with low weight in the graph (which is built as a complete graph where every pixel act as a node and this graph is a complete graph) that is built on the image.
2. K means clustering is very sensitive to the outliers in general and makes a spherical clustering Based on this we can determine the shape of the given question.



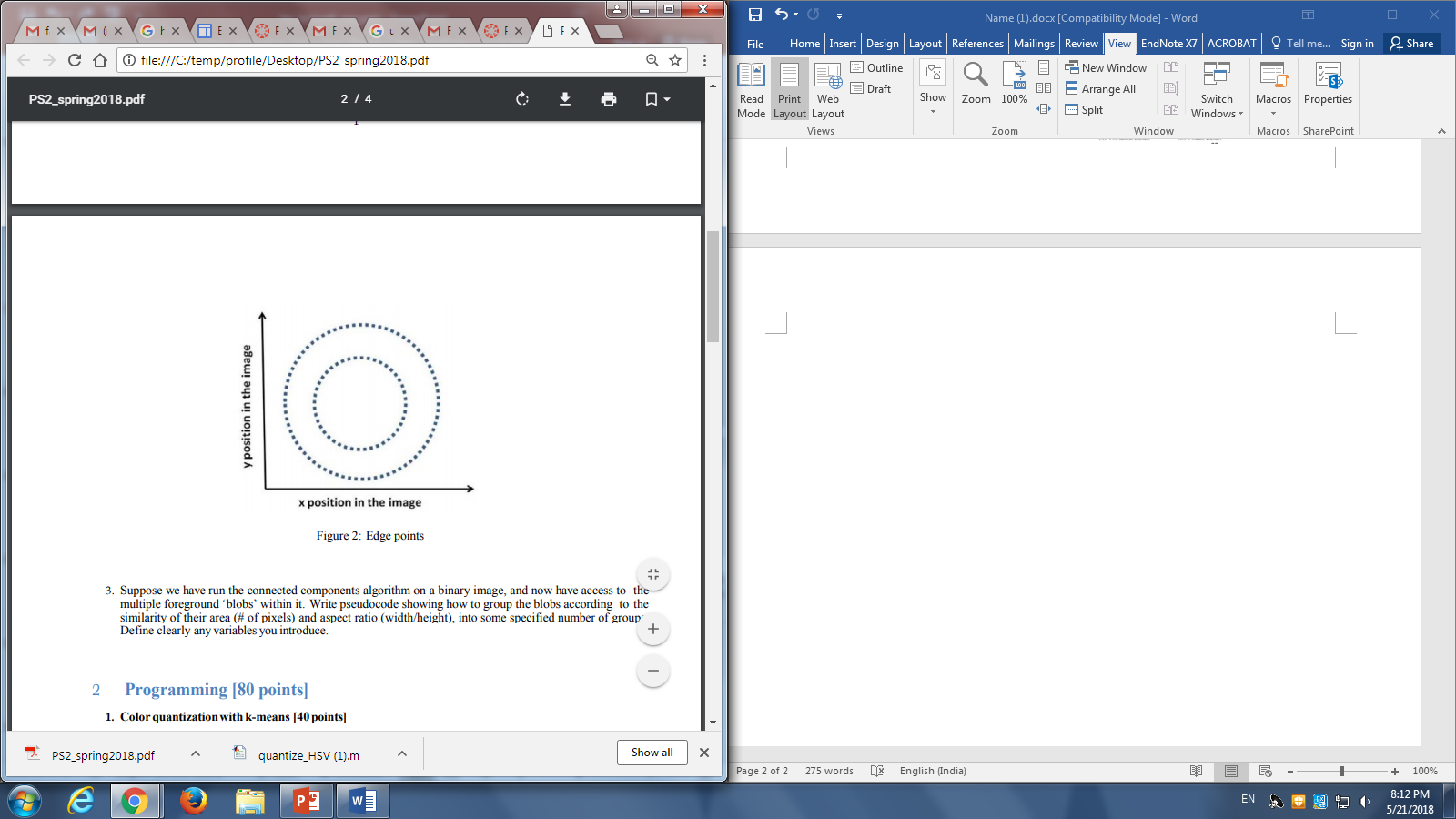
So basically the image get clustered into two sphere based on the initially selected random points



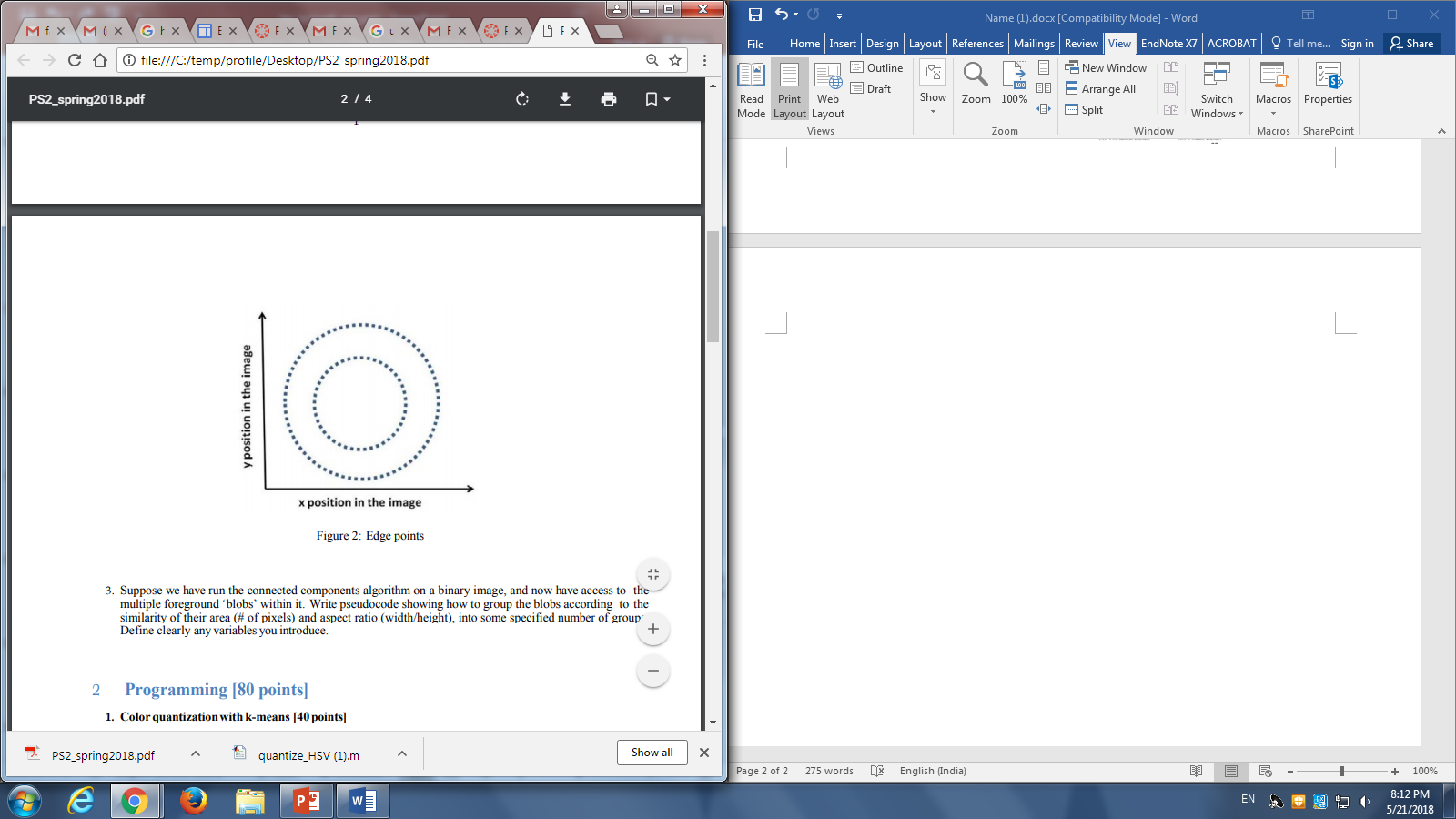
The output after some iterarations look like



And end up getteinfg something like this

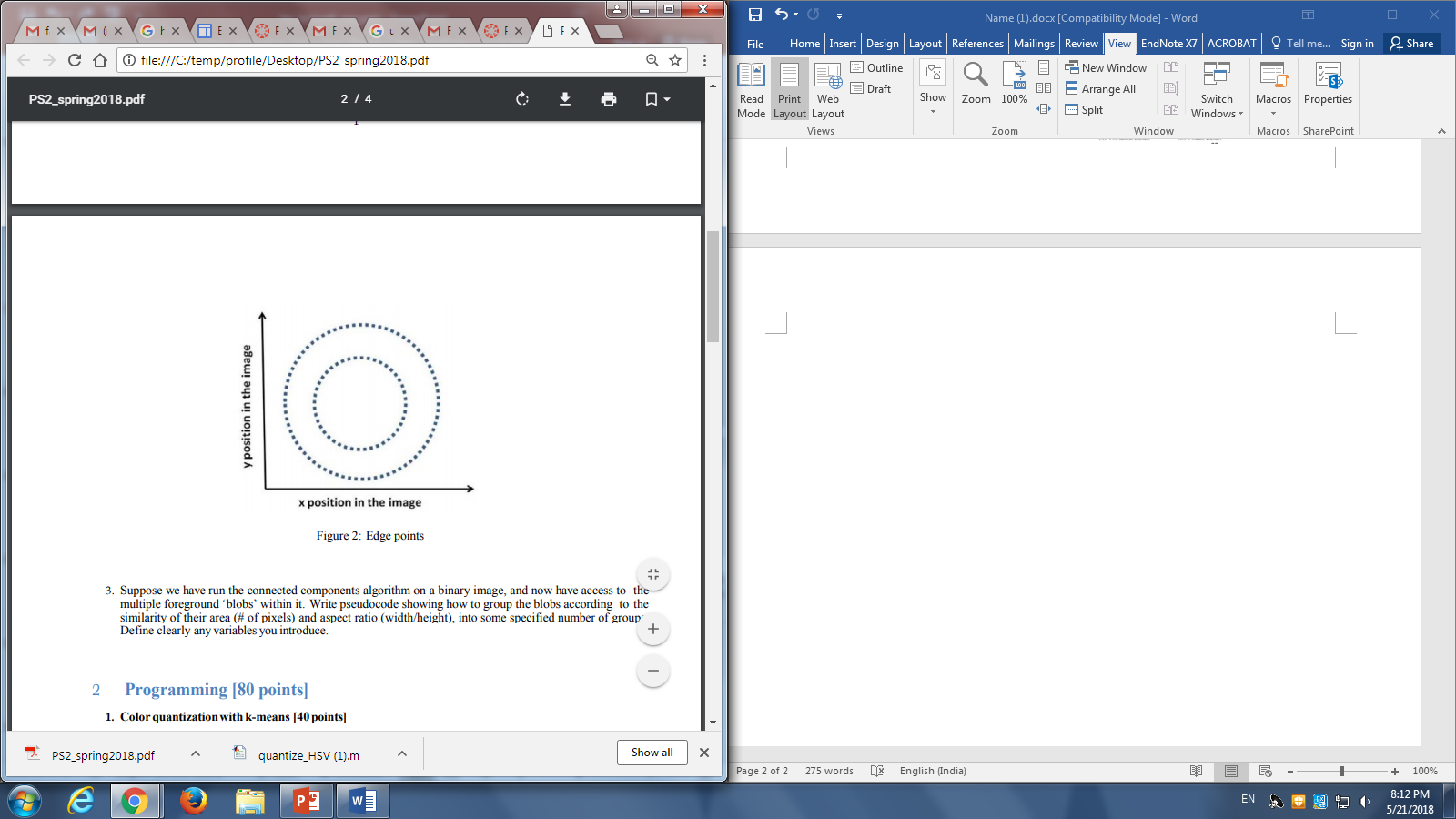


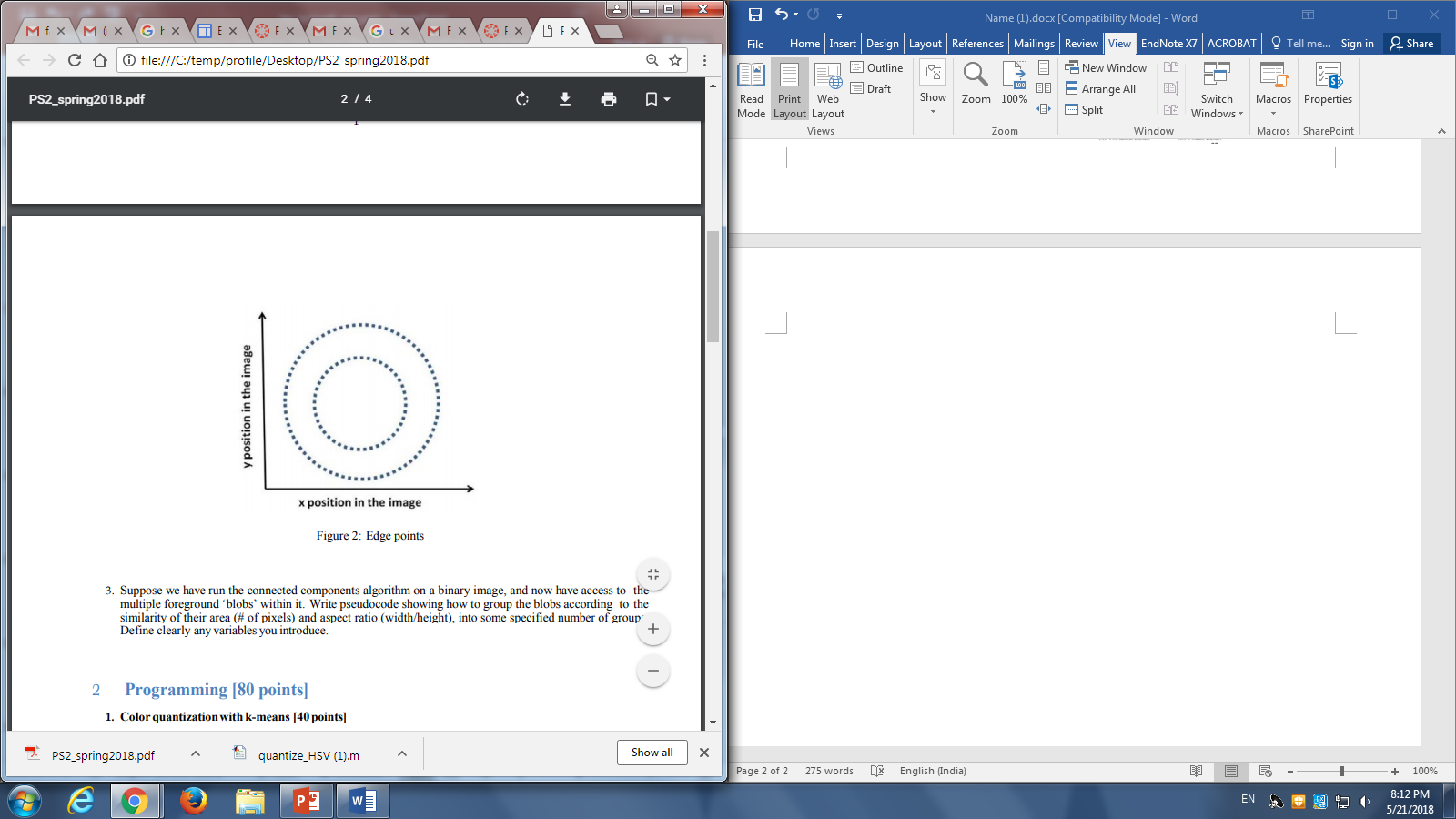
Here is an example for one case



For the given randomly selected points in the above image the grouping is done the following way then the clustering is started doing in the following way.

Step1





Cluster 2 gets updated as the center of the cluster 2 shifts down and now the groups are updated

3>First we need to find the 2d feature space that represents the number of the pixels and the aspect ratio (width/height).

So we need to store all the pixel that are represented in the blob.

For each pixel in the blob:

sumposition = position(pixel);

COM = sumposition/size(blob);

Now calculate the boundaries by the following

For each pixel in boundary :

SSD = (COM – Position(pixel)^2) +SSD

Return (SSD.size(boundary(blob)))

Return circularity

So now once you are able to find the circularity apply Kmeans

[output,center] = kmeans(boundary[],k)

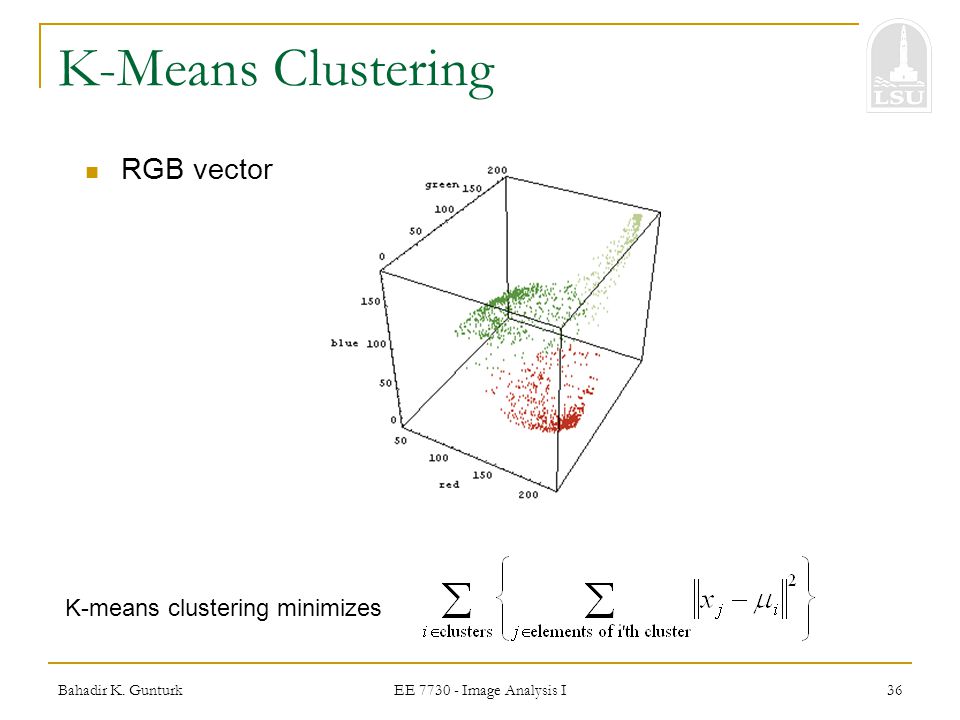
Where k are the number of clusters that are needed to be grouped into.

**Part 2**

e> SSD error when k=3 for quantize\_RGB = 1468734, Quantize\_hue: 48432144

f>K means algorithm on Quantize\_RGB here what we are doing is that we are trying to quantize the values of the given image based on its RGB feature shape. I am reshaping it into 2-d array an passing it to the k-means function. The value of K represents the number of clusters you want to group your given feature space. Since K means randomly selects initial points you vary to get different histclustered histogram. And this make sense because of k-means running with always a different center value and cluster all the points that are near to it and recomputed the mean weight of the whole cluster and we proceed till there is no variation in the moment of the groups or will there is no change in the mean values of the group.

Here the feature space is 3-d RGB values



Quantize\_HSV cluster the values based on the hue value space and in the 1st histogram I am returning the histogram with equal bins i.e., each bar represent count of a range of the hue values

The first histogram that returns the histEquale gives the histogram with values on x axis as range of hue values that are equally splited in 10 bins i.e., if hue is ranging from 1-255 the first bin hue value ranges from 1-25 and next bin from 25- 50 and their corresponding count values on the y-axis.

The second histogram is the Histclustered represents the x axis with center values of each cluster and keep the track of the count value for each bin with the particular bin value.

**Part 3**

a>

step1>First I am trying to find to convert the edges that can vote or contribute to my vote space for this I used Canny edge detecter and I threshold the values of the canny edge detector to a certain value threshold to remove all the noise that removes the edges that don’t contribute to mmy vote space.

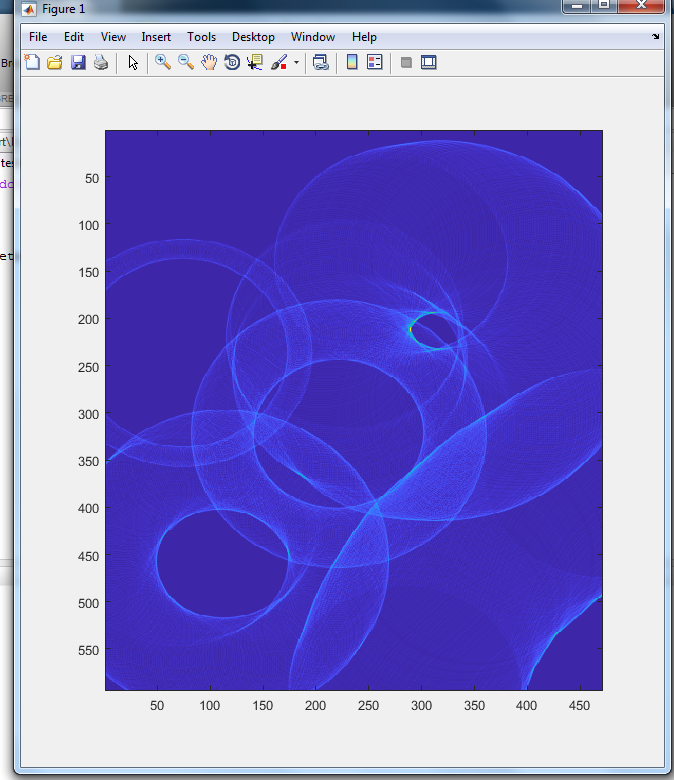
Step2>Then I am finding finding the edges from the edge function that return a logical array and get the location in which the value is equale to 1 that contributes tho the edge.

Step3>Then I am converting the given edge into the hough transform space and update my accmualator array(Here my accumulator array is ca) which basically keeps track of all the votes in hough space.And usegradient = 0 I am looping across all the possible theta value and use gradient = 1 I am use the imgradientxy find the gradient direction by applying tan inverse function follow the above steps.

Since we don’t know the number of centers that are available that return the exact fit circle some time we return top 5 centers that can even gives us back the false positive values I am trying to first find the maximum value first and then return the return number points that are available and contribute to the circles. And this is how I did the post processing of my accumulator array.

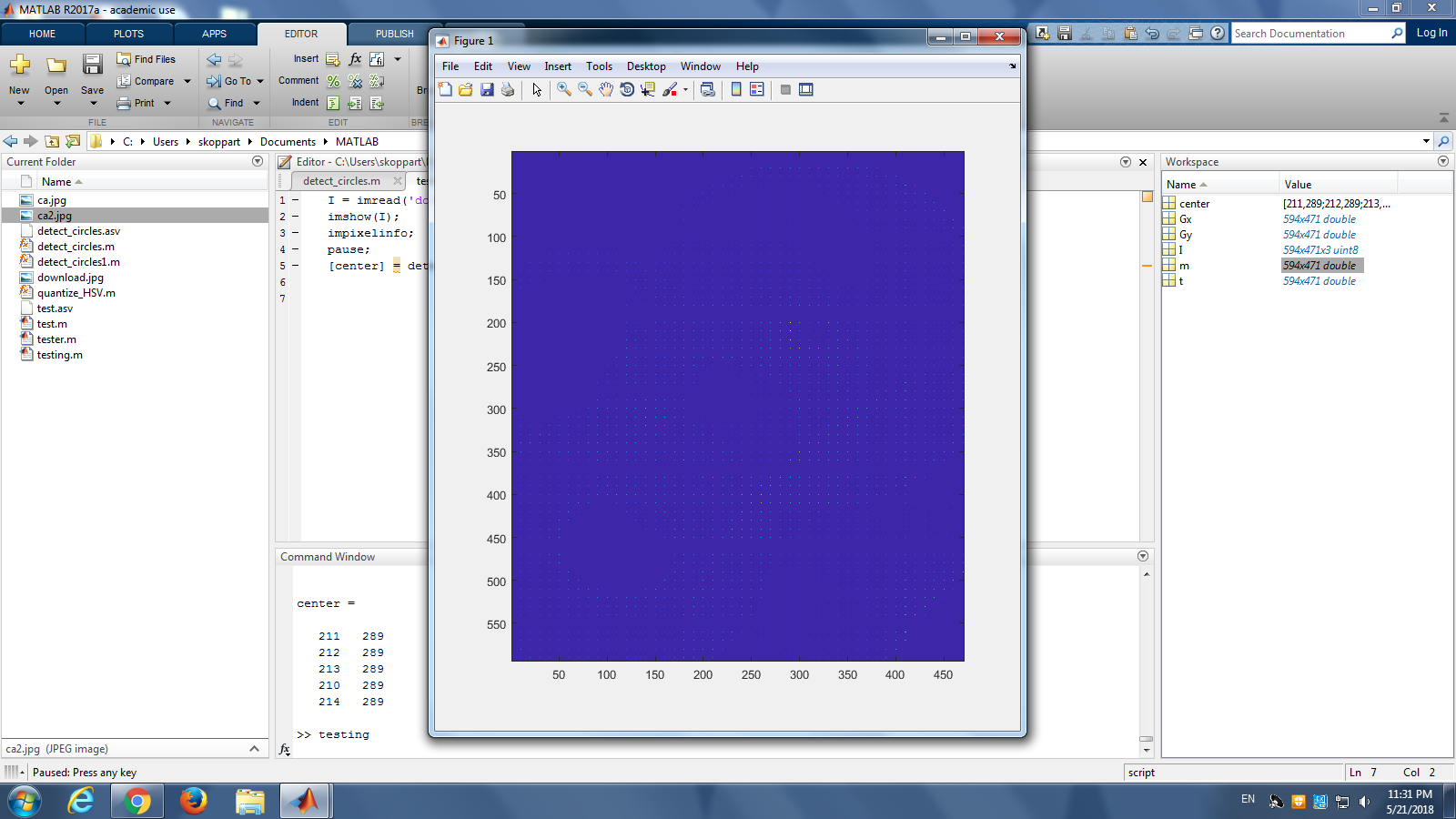
Here the counter variable gives me the number of circles that actually contribute the circle of the hough transforms circles.

c>

The

The accumulator array represent the descretized vote space of The brighter position represents the center of required circle. This accumulator array is represented by count of every vote that is contributed by every edge point of my give image. We can even extend this by giving more votes for stronger edges and change the sampling to give less/more resolution.

e>In the last I did rounding of the of my bins to nearest 10 to check how would be impact continuous image space and it looks like this



Extra Credit

I tried to implement it but got stuck somewhere. Here I looped across another paramenter that is R through out all possibiletie from 0 to max(size(image,1),size(image,2)) and follow the same procedure of finding the hough transform points and make a vote of it which is 3 dimentional where another dimention represents the r that provides you a cone.

This is the code I used for the unknown circle radius

%%

function [centers] = detect\_circles(I,radius)

PI = 3.1420;

Ia = I;

I = rgb2gray(I);

e = edge(I,'Canny',0.6,8);

imshow(e)

pause

[r,c] = find(e==1);

ca = zeros(size(I,1),size(I,2),max(size(I,1),size(I,2)));

a = zeros(size(r,1),360);

b = zeros(size(c,1),360);

centers = zeros(5,3);

for i = 1:size(r,1)

for radius = 1:max(size(I,1),size(I,2))

for t = 1:1:360

a(i,t) = round(r(i) - (radius \* cos(t\*PI/180)));

b(i,t) = round(c(i) - (radius \* sin(t\*PI/180)));

x = a(i,t);

y = b(i,t);

if(x>0 & y>0 & x<size(I,1) & y<size(I,2))

ca(x,y,radius) = ca(x,y,radius) + 1;

end

end

end

end

cm = ca;

for i = 1:5

[Value,Ind]=max(cm(:));

[M,N,r]=ind2sub(size(cm),Ind);

centers(i,1) = M;

centers(i,2) = N;

centers(i,3) = r;

cm(M,N,r) = 0;

end

for i = 1:size(centers,1)

for radius = 1:max(size(I,1),size(I,2))

Ia = circle(centers(i,1),centers(i,2),radius,Ia);

end

end

imshow(Ia);

pause;

end

function I = circle(x,y,r,I)

PI = 3.1420;

for th = 1:1:360

xunit =(round(r \* cos(th\*PI/180) + x));

yunit = (round (r \* sin(th\*PI/180) + y));

I(xunit,yunit,1) = 255;

I(xunit,yunit,2) = 0;

I(xunit,yunit,3) = 0;

end

end

%%