HW4

95318168 HONGRU LI

1 Hough Transform

(a) The unhouge start from Hough transform histogram and end with the voted lines.

First it caps the values histogram

Then dilate and do connect components labeling

Then find the center of each clusters for the line’s ro and theta.

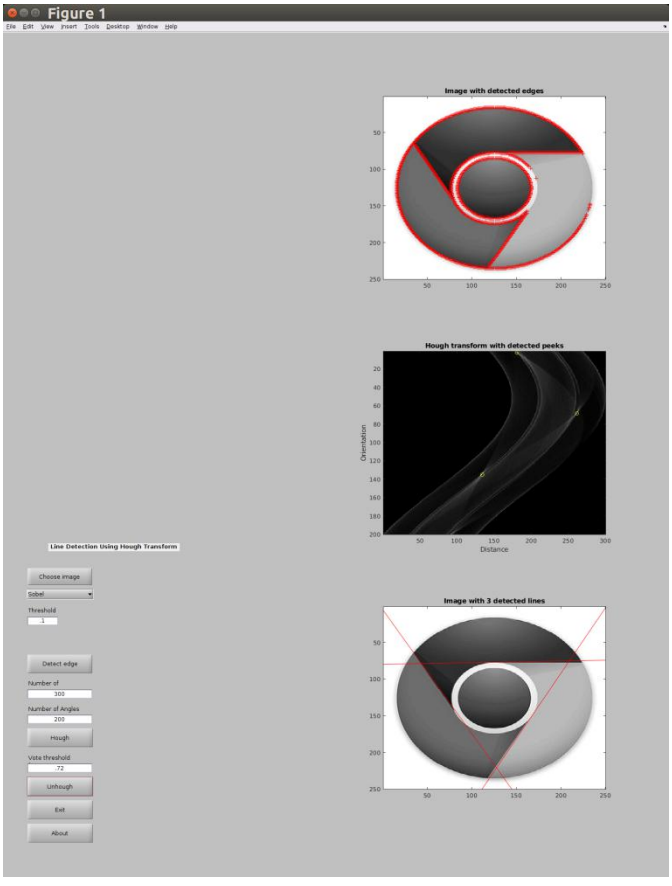
(b) Threshold is the ‘threshold’ in matlab function edge(I, method, threshold), It is the threshold to extract the edge points after applying edge detection kernels such like sobel.

Number of distance and Number of angles is the granularity of (Ro, Theta) Hough transform axis. The axis are divided to the number of you specified pieces when finding the line.

Vote threshold fraction is the percentage threshold used to filter the Hough histogram:

```
TH=double(H>P*max(H(:)));
```

(c)



(d)

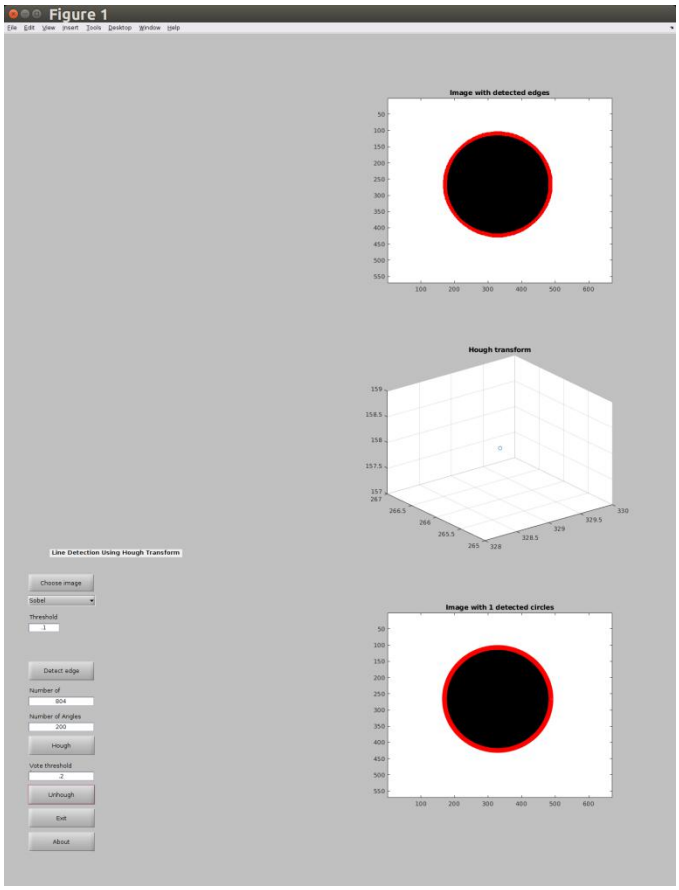


Figure 1
File Edit View Insert Tools Desktop Window Help
Image with detected edges
Hough transform
Line Detection Using Hough Transform
Choose image
Sobel
Threshold
-1
Detect edge
Number of
804
Number of Angles
200
Hough
Vote threshold
-2
Unhough
Exit
About
Image with 1 detected circles

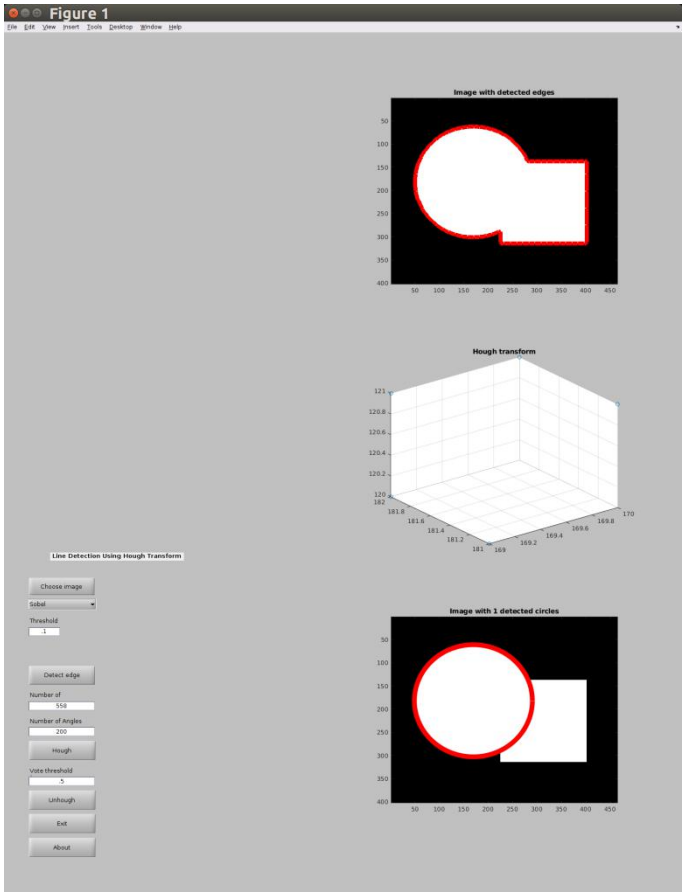


Figure 1
File Edit View Insert Tools Desktop Window Help
Image with detected edges
Hough transform
Line Detection Using Hough Transform
Choose image
Sobel
Threshold
-1
Detect edge
Number of
550
Number of Angles
200
Hough
Vote threshold
-5
Unhough
Exit
About
Image with 1 detected circles

(e)

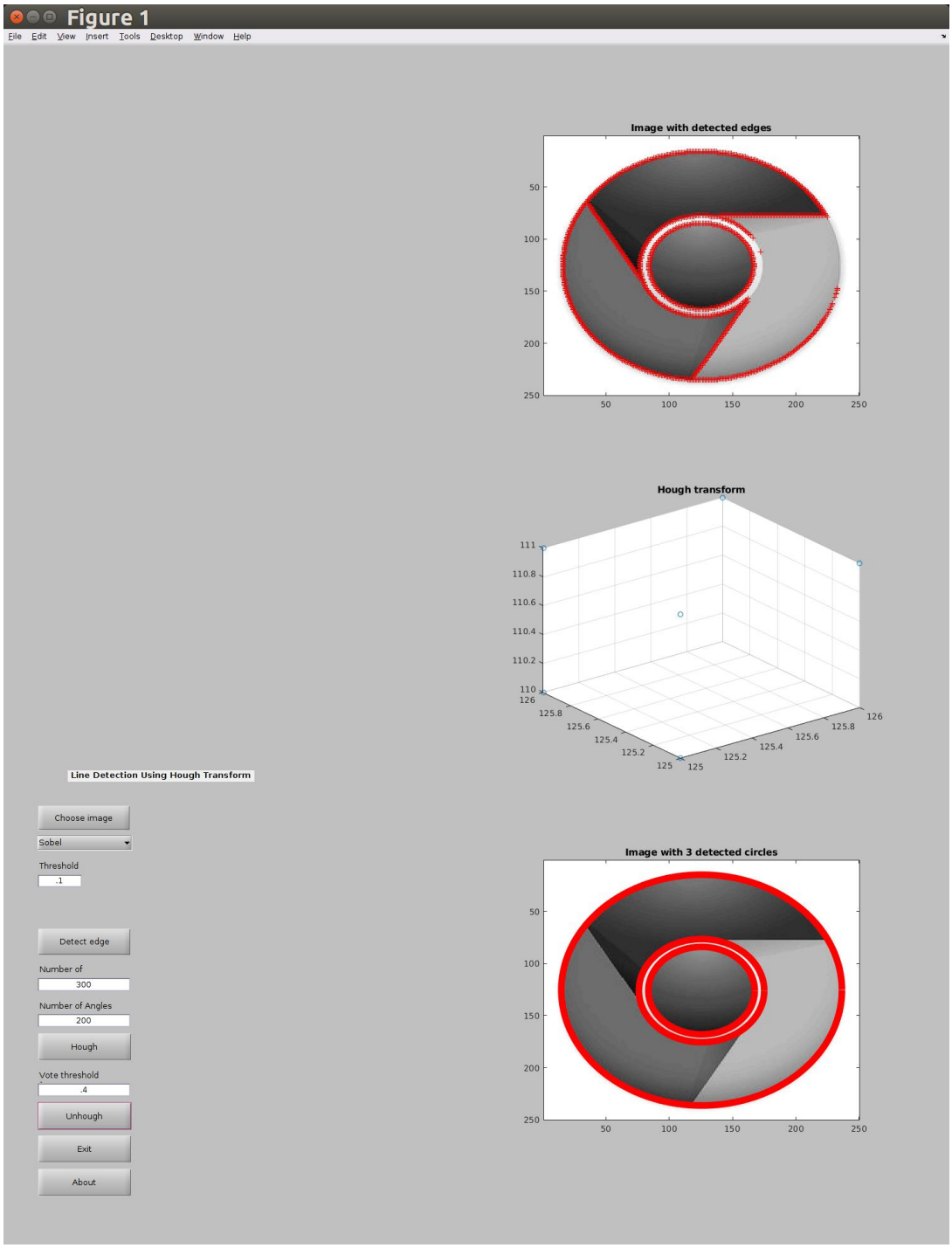


Figure 1
File Edit View Insert Tools Desktop Window Help
Image with detected edges
Hough transform
Line Detection Using Hough Transform
Choose image
Sobel
Threshold
.1
Detect edge
Number of
300
Number of Angles
200
Hough
Vote threshold
.4
Unhough
Exit
About
Image with 3 detected circles

Code:

CVhough.m

```
function [H, m, b] =CVhough(I, edgedata,nT,nS)
%CVhough Hough transform of a binary matrix
%
%function [H,m,b]=CVhough(edgedata,nT,nS)
%    edgedata a 2-row matrix, with the x and y coordinates of the edges
%    H votes histogram
%    m and b: picture width and height

MAXDIST=1.2;
if nargin<1
    error('require at least one input argument: binary image')
elseif nargin<2
    warning('defualt value of 200 assigned to number of orientations nT')
    nT=200;
    warning(['defualt value of', max(edgedata(:))*MAXDIST, 'assigned to number of orientations nS'])
    nS=max(edgedata(:))*MAXDIST;
elseif nargin<3
    warning(['defualt value of', max(edgedata(:))*MAXDIST, 'assigned to number of orientations nS'])
    nS=max(edgedata(:))*MAXDIST;
end

row=edgedata(2,:);
col=edgedata(1,:);

n = size(edgedata,2 );
[w, h] = size(I);

H = zeros(w,h,1+round(sqrt(w^2 + h^2)));%store the temp result, avoid atomic operation

hw=waitbar(0,'Performing Hough Transform...');
for a = 1:w
    for b = 1:h
        for i= 1:n
            r = 1+round(sqrt((row(i)-b)^2+(col(i)-a)^2));
            H(a,b,r) = H(a,b,r) + 1;
        end
    end
```

```
    waitbar(a/w,hw);
end
close(hw);

m = w;
b = h;
```

CBhough.m

```
watchon;
H=findobj(gcf,'Tag','EditTextNDist');
nS=str2num(get(H,'String'));
H=findobj(gcf,'Tag','EditTextNAng');
nT=str2num(get(H,'String'));
[HF,m,b]=CVhough(I, edgedata,nT,nS);
subplot(3,2,4);
c = find(HF>0.7*max(HF(:)));
[X,Y,Z] = ind2sub(size(HF), c);
scatter3(X,Y,Z);
title('Hough transform')
subplot(3,2,6);
cla
watchoff;
H=findobj(gcf,'Tag','PushbuttonUnhough');
set(H,'Enable','on');
```

Cvunhough.m

```
function Circles=CVunhough(H,m,b,P)
%CVunhough finds lines from a Hough histogram
%function Circles=CVunhough(H,m,b,P)
%    H votes histogram of size [nT,nS]
%    P percentage threshold
%    Circles circles found

DILATEFRAC=.02;


if nargin<3
    error('require at least 3 input arguments: histogram matrix H, 2 distance mapping parameters m & b');
elseif nargin<4
    warning('defualt value of 0.7 assigned to percentage threshold P');
    P=0.7;
```

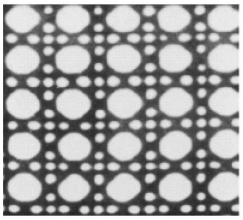
```
end

[w, h, r]=size(H);

TH=H>P*max(H(:));

%Morphological
se = strel('sphere', 1)
H1=imdilate(TH, se);

%Labeling
Circles = regionprops3(H1, 'centroid');
end
```

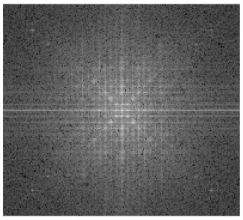
CBunhough.m

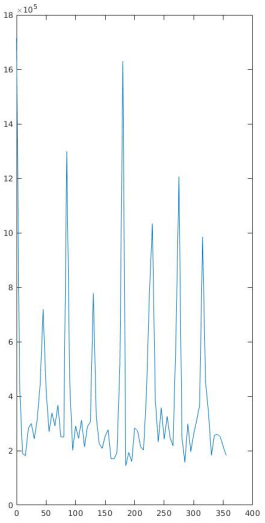
```
H=findobj(gcf,'Tag','EditTextUnhough');
P=str2num(get(H,'String'));
Circles=CVunhough(HF,m,b,P);
Circles = table2array(Circles);
subplot(3,2,6)
imagesc(I);
hold on
colormap(gray);
ang=0:0.01:2*pi;
for i = 1:size(Circles)
    xp=Circles(i,3)*cos(ang);
    yp=Circles(i,3)*sin(ang);
    plot(Circles(i,2)+xp,Circles(i,1)+yp, 'r','linewidth',8);
end
title(['Image with ', num2str(size(Circles,1)),' detected circles'])
hold off
```

2 Texture Analysis

(b)



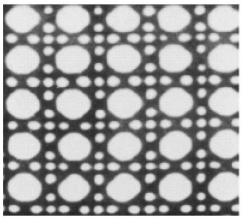


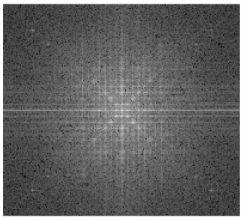


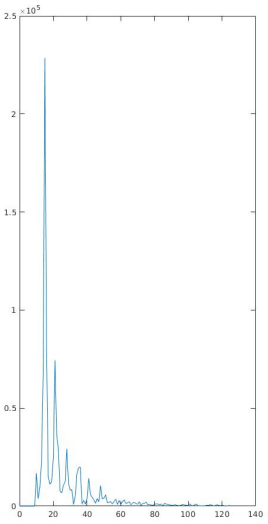
This reflect a circular behavior.

```
I = imread('grid.JPG');
I = rgb2gray(I);
F = fftshift(fft2(I));
figure;
subplot(1,3,1);imshow(I);
subplot(1,3,2);imshow(log(1+abs(F)), []);
ang_degree = 0:5:355;
S_theta = zeros(1,72);
center_x = size(F,1)/2;
center_y = size(F,2)/2;
subplot(1,3,3);
for i = 1:72
    for r = 10:30
            x_pos = round(center_x + r*sind(ang_degree(1,i)));
            y_pos = round(center_y + r*cosd(ang_degree(1,i)));
            S_theta(1, i) = S_theta(1, i) + abs(F(x_pos, y_pos));
    end
end
plot(ang_degree, S_theta);
```

(c)



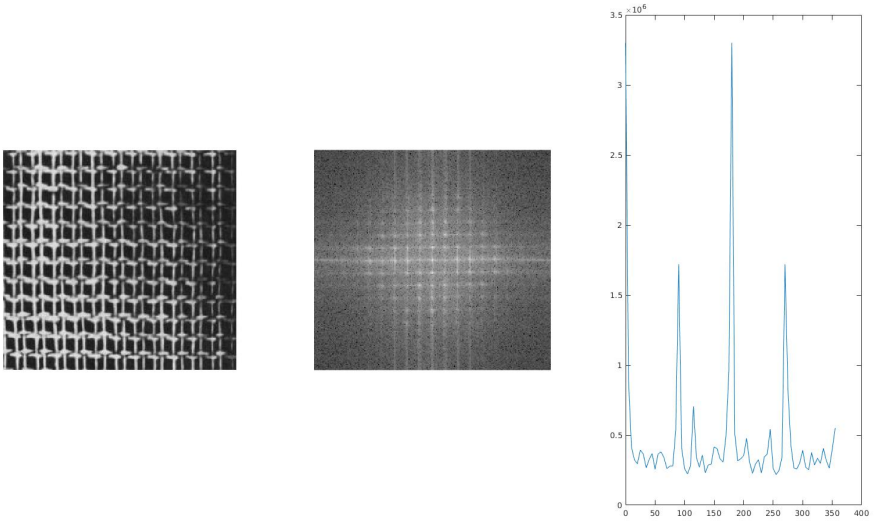


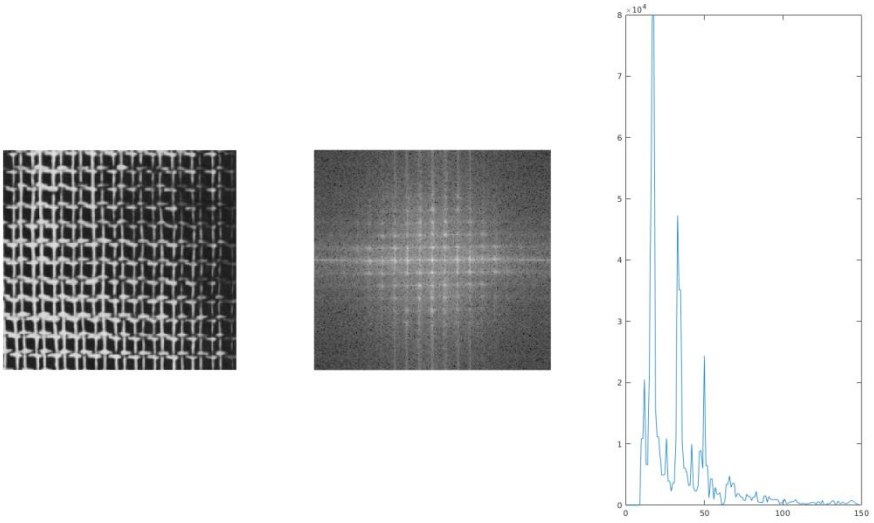


I did not plot the value near zero radius since the values are usually big and need scaling. This reflect radial direction behavior in Fourier spectrum. The peaks at 45 degree suggest repeating structures in spatial domain. And this repeating pattern happens in the direction vertical to the 45 degree direction.

```
I = imread('grid.JPG');
I = rgb2gray(I);
F = fftshift(fft2(I));
figure;
subplot(1,3,1);imshow(I);
subplot(1,3,2);imshow(log(1+abs(F)), []);
center_x = size(F,1)/2;
center_y = size(F,2)/2;
[w,h] = size(F);
subplot(1,3,3);
r = 10;
S = zeros(1,999);
while true
    x_pos = round(center_x + r*sin(pi/4));
    y_pos = round(center_y + r*cos(pi/4));
    if x_pos>w||y_pos>h
        break;
    end
    S(1, r+1) =  abs(F(x_pos, y_pos));
    r = r+1;
end
S = S(1,1:r);
i = 0:1:size(S,2)-1;
plot(i, S);
```

(d)





I think it is hard to distinguish the difference using S(r) which indicate the two image has similar pattern in radial direction behavior in their Fourier spectrum. But in S(thehta) plot the periodic.jpg has more peaks compared to grid.jpg, which means different circular behavior in Fourier spectrum.

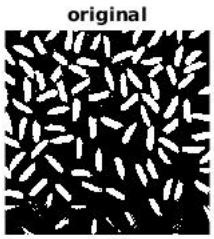
3 Morphological Operations

(a)

The problem is there are noise and some rices are not correctly segmented using this gray level threshold method. The gray level of rice at upper half of the image is different from those at the bottom of the image.

Code:

```
I = imread('rice.png');
level = graythresh(I);
BW = imbinarize(I,level);
imshow(BW);
```



(e)

The opening operation generally has two effect:

It remove noises, and it smooth(since the dilate let the grain area always using the

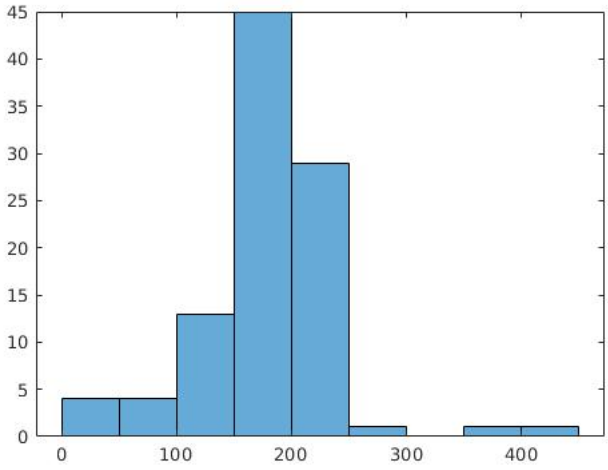
largest value in its neighbor while erosion reduce the invasion of gray value from rice to the background area) the rice so that the gray level for the rice becomes similar and than is easier to split from the background using gray level threshold. So as is shown above, the rices are segmented correctly and the noise got reduced a lot.

```
I = imread('rice.png');
level = graythresh(I);
BW = imbinarize(I,level);
%b
se = strel('disk',15);
Io = imopen(I,se);
%c
Io = I-Io;
%d
Io = imadjust(Io);
%e
level = graythresh(Io);
BW_e = imbinarize(Io,level);
subplot(2,1,1);imshow(BW);title('original');
subplot(2,1,2);imshow(BW_e);title('substract opening');
```

(f) I apply morphological opening in binary image. The noise disappeared at the cost of smoothed corners.

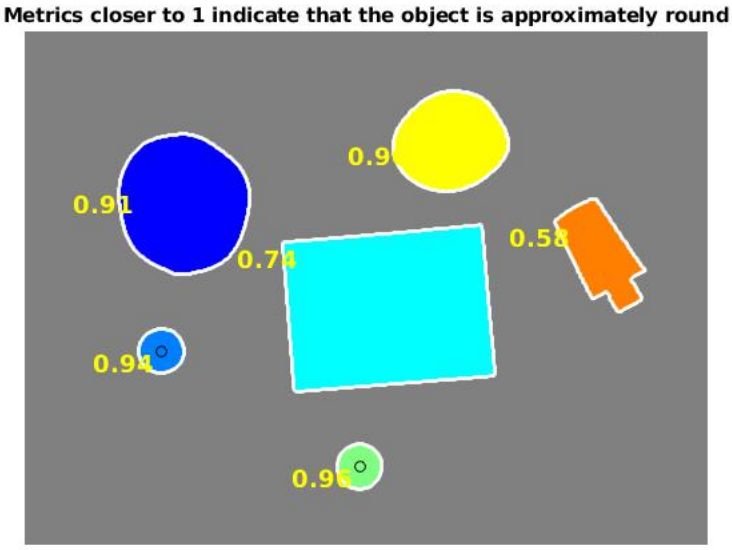
(g) I only get 98 connected regions for 100 rices. That is because two pair of rices have overlap in their region and thus they are count as only two connect regions.

(h) The maximum are is 401 and the minimum is 4.



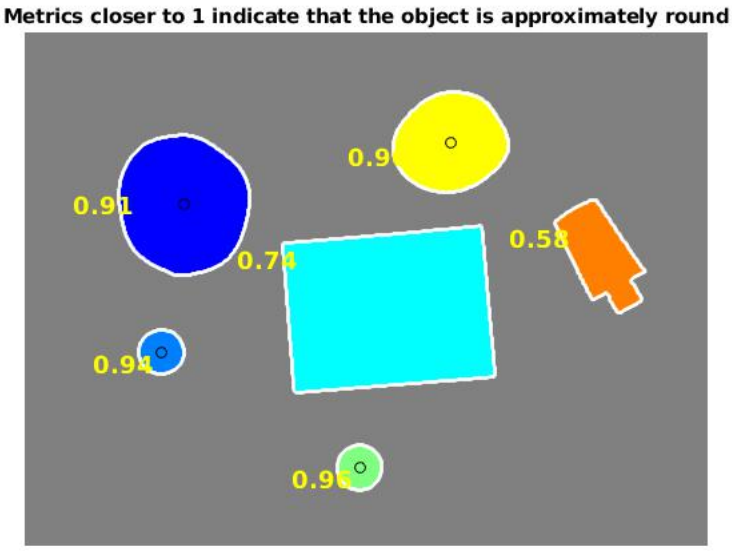
But I didn't quite understand what is the point of this question. The area is how many pixels are there in each connect components. And the numbers are different since the size of rice are different. The particular small area is from rices not completely shown in the image. The two large sample are from two adjacent pairs mentioned in last question.

4 Identifying geometrical shapes from images



(a) At first the code convert the image to binary image, do smoothing and reduce noise, then fill the holes in regions. It colors different different connect regions and tracking the boundary of each region. Then after doing connect component labeling, the code compute the perimeter of each contour and calculate the roundness metric given by 4*pi*area/perimeter^2. If this metric is shown in the image and if the metric is over 0.93, the code will label the center of this region with a small circle.

(b) The threshold is telling you how good a circle should be labeled a circle. A perfect circle will have roundness metric of 1. So if I set the threshold to 0.9 then the two rubber band will also be labeled as circles.

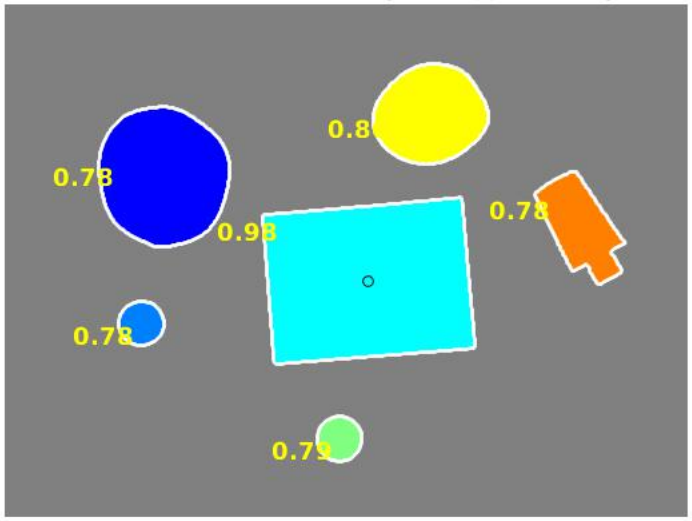


(c) The minBoundingbox function in my code is from https://www.mathworks.com/matlabcentral/fileexchange/31126-2d-minimal-bounding-box

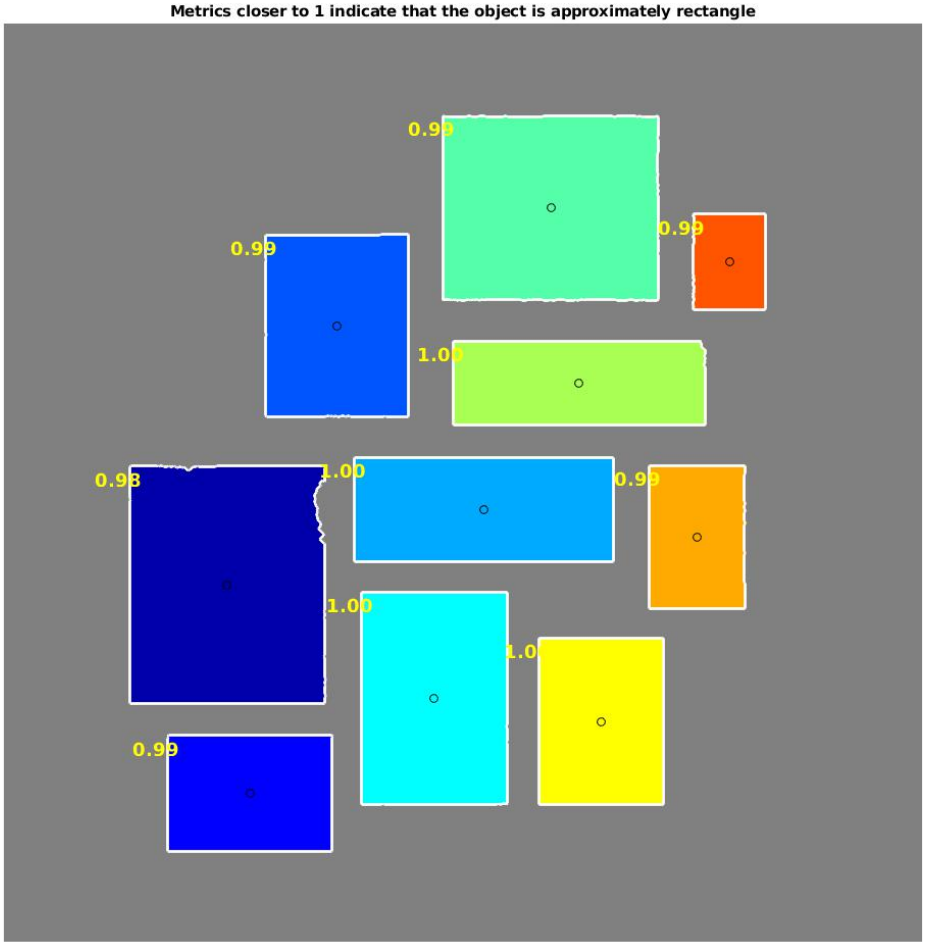
In general, you need first find the area of minimum bounding box and the rec metric is that area over area of the region selected as is shown in lecture notes. So I find a open source implementation of find the minimum bounding box for a set of points online(allow bounding box from all directions)

The result:

Metrics closer to 1 indicate that the object is approximately rectangle

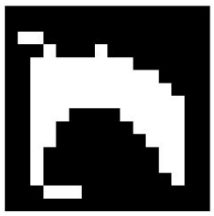


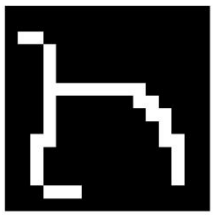
Metrics closer to 1 indicate that the object is approximately rectangle



5 Skeletonization of Objects

(c)





Code:

```
I = load('input.mat');
I = I.input;
subplot(2,1,1);imshow(I);
attacker = zeros(4,3,3,3);
attacker(1,1,:,:) = [-1 0 -1; 0 1 1;-1 1 1];
attacker(1,2,:,:) = [-1 1 1; 0 1 1;-1 0 -1];
attacker(1,3,:,:) = [-1 -1 1; 0 1 1;-1 -1 1];
attacker(2,1,:,:) = [-1 0 -1; 1 1 0;1 1 -1];
attacker(2,2,:,:) = [-1 0 -1; 0 1 1;-1 1 1];
attacker(2,3,:,:) = [-1 0 -1; -1 1 -1;1 1 1];
attacker(3,1,:,:) = [1 1 -1; 1 1 0;-1 0 -1];
attacker(3,2,:,:) = [-1 0 -1; 1 1 0;1 1 -1];
attacker(3,3,:,:) = [1 -1 -1; 1 1 0;1 -1 -1];
attacker(4,1,:,:) = [-1 1 1; 0 1 1;-1 0 -1];
attacker(4,2,:,:) = [1 1 -1; 1 1 0;-1 0 -1];
attacker(4,3,:,:) = [1 1 1; -1 1 -1;-1 0 -1];
[w, h] = size(I);
while true
    finish = false;
    for n=1:4
        maskmat = zeros(w,h);
        for i=2:w-1
            for j=2:h-1
```

```
                result = mask(I, maskmat, attacker(n,:,:,:),i, j);
                maskmat(i,j) = result;
                finish = result || finish;
            end
        end
        I = I & ~maskmat;
    end
    if ~finish
        break;
    end
end
subplot(2,1,2);imshow(I);

function result = mask(I, mask, attackers, x, y)
%MASK Summary of this function goes here
%   Detailed explanation goes here
attackers = squeeze(attackers);
result = false;
for n=1:size(attackers,1)
    match = true;
    for i=1:size(attackers, 2)
        for j=1:size(attackers, 3)
            if attackers(n, i, j) == -1
                continue;
            else
                match = match & (I(x+i-2,y+j-2)==attackers(n, i, j));
            end
        end
    end
    result=result||match;
end
end
```

(d)

I don’t know whether the above shape is what you mean by perturbed side. The skeleton is different from that of a perfect rectangle. And I don’t know whether this skeleton should be called correct. Maybe the protrusion is not what you want.