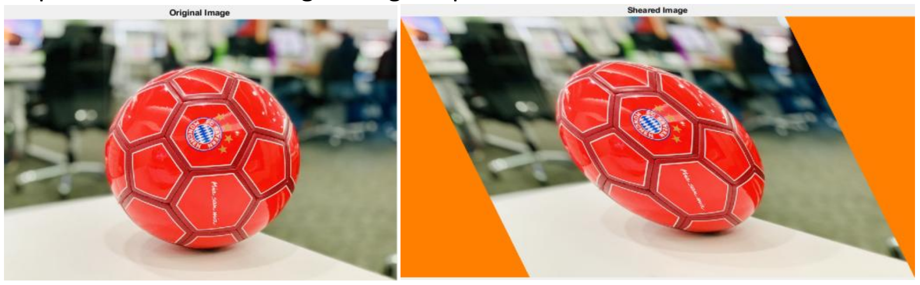
CPS621 Winter2022
Lab02 Report
Name: Tusaif Azmat, Student#: 500660278.

Part 1: Image transformation and padding.

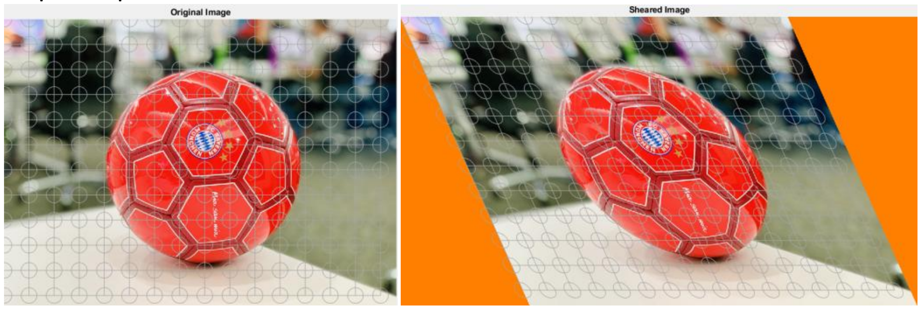
1.1: original image

1.2: The four steps and perform the transformation and padding.

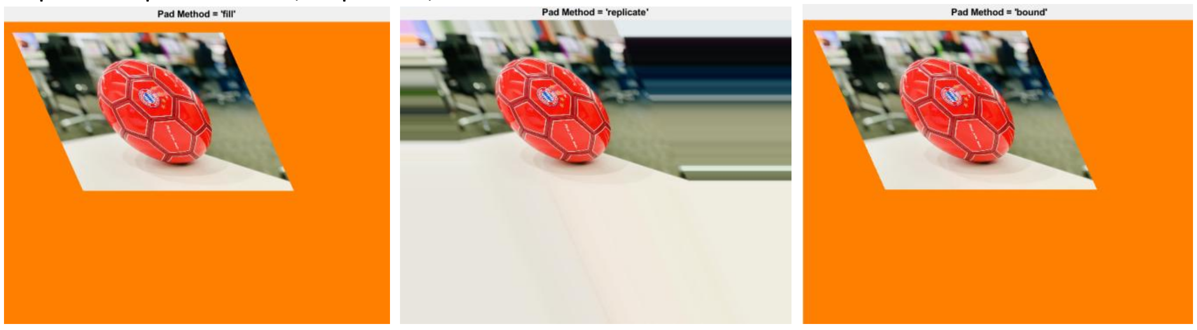
Step 1: Transform an Image Using Simple Shear

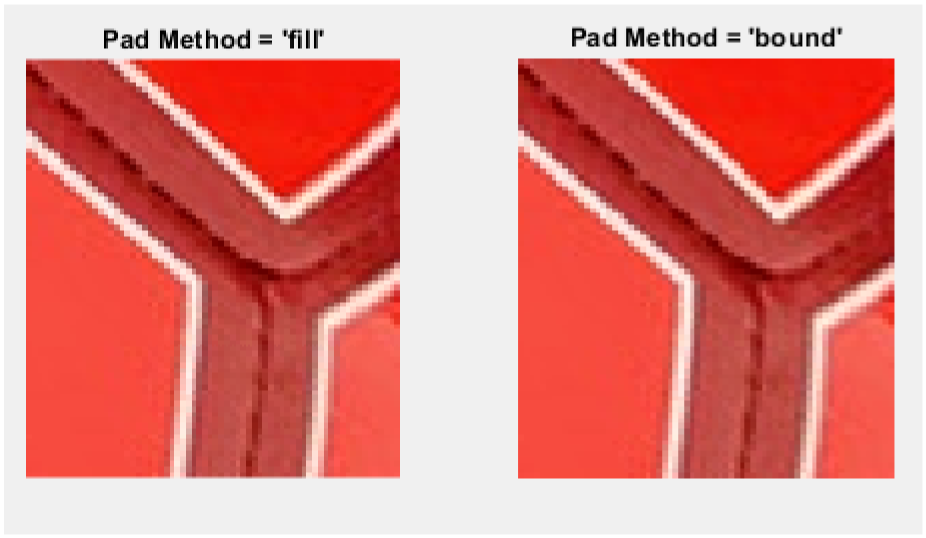


Step 2: Explore the Transformation

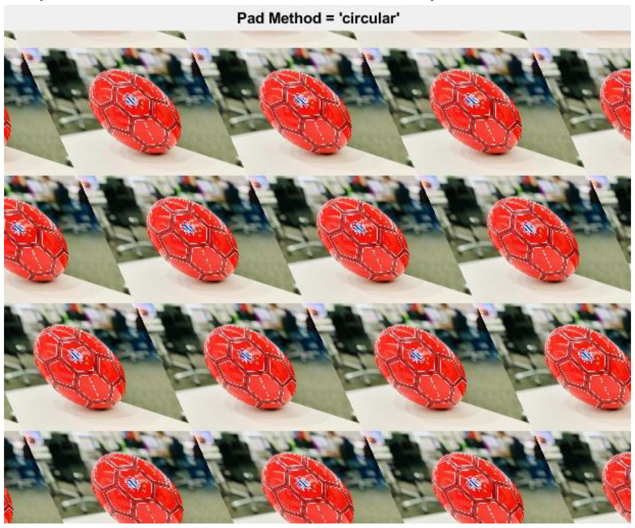


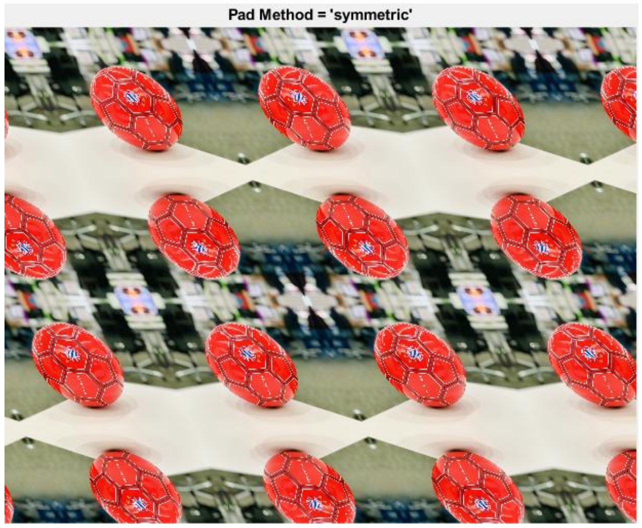
Step 3: Compare the ‘fill’, ‘replicate’, and ‘bound’ Pad Methods





Step 4: Exercise the 'circular;' and 'symmetric' Pad Methods





1.6 Report:

Padding and Shearing of an image is done at the same time by following the four steps.

Step1: Transformed the image using a simple shear. Shearing is a form of image transformation. It is an affine transformation, where lines and parallelism is preserved for the original image but the empty image region with no data will be filled with an orange colour. Imtransform function of matlab does that for us with setting its parameters. We got the shear image as can be seen above in step1 image.

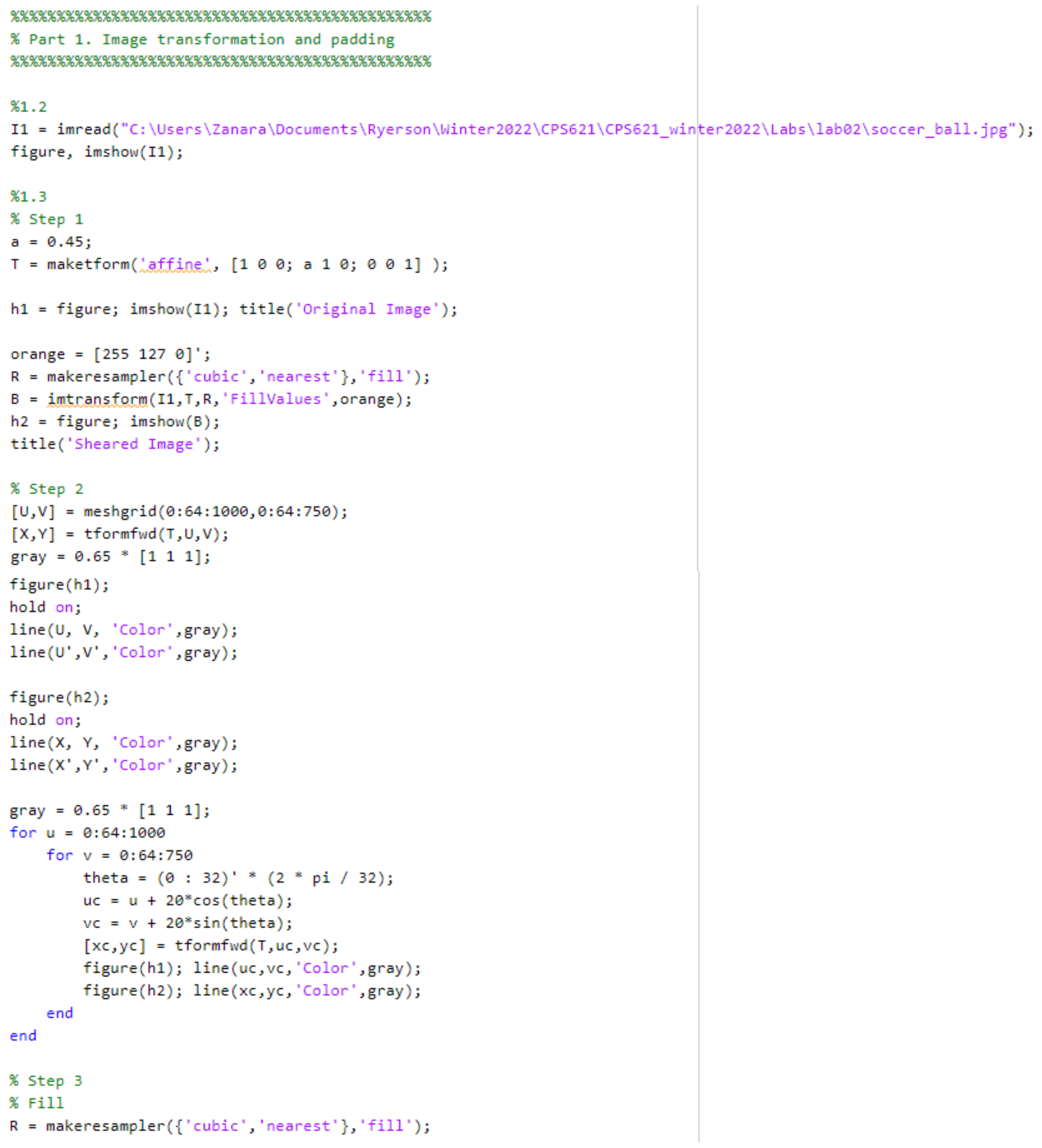
Step2: Transformation exploring. I transformed the image with a grid of straight lines and circles with 'tformfwd' function of matlab. I use both the original and sheared image for this exploration.

Step3: Padding of image by using 'fill', 'replicate' and 'bound' methods first to see the different resulting images. In matlab padding can be done with 'makeresampler' function, so first I use the 'fill' padding by calling 'fill' in 'makeresampler' that filled the empty region with a previously orange colour, applied the cubic interpolation across the edges. Then I use the 'replicate' parameter of 'makeresampler' function to fill the empty region by copying the colour at the boundaries of the image. Similarly use the 'bound' parameter of 'makeresampler' function. The results of 'fill' are similar but 'bound' uses strict boundaries of the image.

Step4: use the remaining padding method 'circular' and 'symmetric'. As can be seem above screen shots that 'circular' repeats the image in a tiled fashion. It's like circular repetition in each dimension of the image. But with 'symmetric' it repeats the image in a tiled fashion also but the image is symmetrically mirrored.

In the part1 of this lab I have explored the image transformation and how it affects straight line and circles, and then I explore the various options for image padding with the help of matlab function 'imtransform'.

MatLab Code:

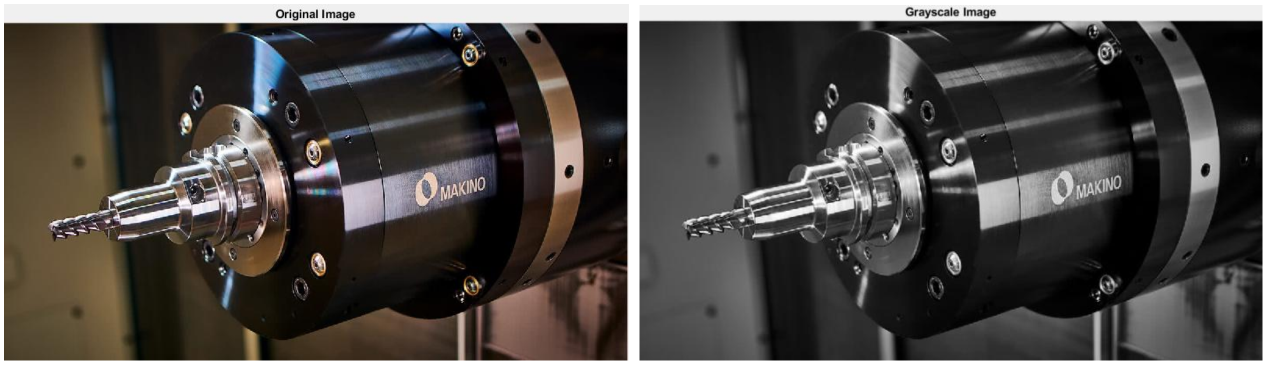


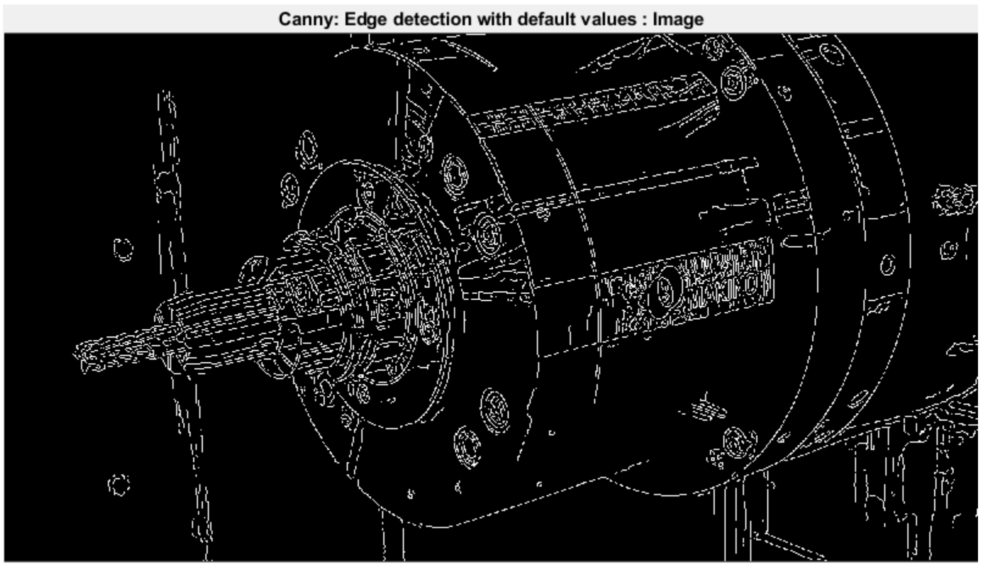
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 1. Image transformation and padding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1.2
I1 = imread("C:\Users\Zanara\Documents\Ryerson\Winter2022\CPS621\CPS621_winter2022\Labs\lab02\soccer_ball.jpg");
figure, imshow(I1);

%1.3
% Step 1
a = 0.45;
T = maketform('affine', [1 0 0; a 1 0; 0 0 1] );

h1 = figure; imshow(I1); title('Original Image');

orange = [255 127 0]';
R = makeresampler({'cubic','nearest'},'fill');
B = imtransform(I1,T,R,'FillValues',orange);
h2 = figure; imshow(B);
title('Sheared Image');

% Step 2
[U,V] = meshgrid(0:64:1000,0:64:750);
[X,Y] = tformfwd(T,U,V);
gray = 0.65 * [1 1 1];
figure(h1);
hold on;
line(U, V, 'Color',gray);
line(U',V','Color',gray);

figure(h2);
hold on;
line(X, Y, 'Color',gray);
line(X',Y','Color',gray);

gray = 0.65 * [1 1 1];
for u = 0:64:1000
    for v = 0:64:750
        theta = (0 : 32)' * (2 * pi / 32);
        uc = u + 20*cos(theta);
        vc = v + 20*sin(theta);
        [xc,yc] = tformfwd(T,uc,vc);
        figure(h1); line(uc,vc,'Color',gray);
        figure(h2); line(xc,yc,'Color',gray);
    end
end

% Step 3
% Fill
R = makeresampler({'cubic','nearest'},'fill');
```

```
Bf = imtransform(I1,T,R,'XData',[-49 1800],'YData',[-49 1400],...
                 'FillValues',orange);

figure, imshow(Bf);
title('Pad Method = ''fill''');
%Replicate
R = makeresampler({'cubic','nearest'},'replicate');
Br = imtransform(I1,T,R,'XData',[-49 1800],'YData', [-49 1400]);

figure, imshow(Br);
title('Pad Method = ''replicate''');
%Bound
R = makeresampler({'cubic','nearest'}, 'bound');
Bb = imtransform(I1,T,R,'XData',[-49 1800],'YData',[-49 1400],...
                 'FillValues',orange);
figure, imshow(Bb);
title('Pad Method = ''bound''');
%padding differnce
R = makeresampler({'cubic','nearest'},'fill');
Cf = imtransform(I1,T,R,'XData',[823 890],'YData',[445 520],...
                 'FillValues',orange);

R = makeresampler({'cubic','nearest'},'bound');
Cb = imtransform(I1,T,R,'XData',[823 890],'YData',[445 520],...
                 'FillValues',orange);
Cf = imresize(Cf,12,'nearest');
Cb = imresize(Cb,12,'nearest');

figure;
subplot(1,2,1); imshow(Cf); title('Pad Method = ''fill''');
subplot(1,2,2); imshow(Cb); title('Pad Method = ''bound''');

% Step 4
%Circular
Thalf = maketform('affine',[1 0; a 1; 0 0]/2);

R = makeresampler({'cubic','nearest'},'circular');
Bc = imtransform(I1,Thalf,R,'XData',[-49 1800],'YData',[-49 1400],...
                 'FillValues',orange);
figure, imshow(Bc);
title('Pad Method = ''circular''');
%Symmetric
R = makeresampler({'cubic','nearest'},'symmetric');
Bs = imtransform(I1,Thalf,R,'XData',[-49 1800],'YData',[-49 1400],...
                 'FillValues',orange);
figure, imshow(Bs);
title('Pad Method = ''symmetric''');
```

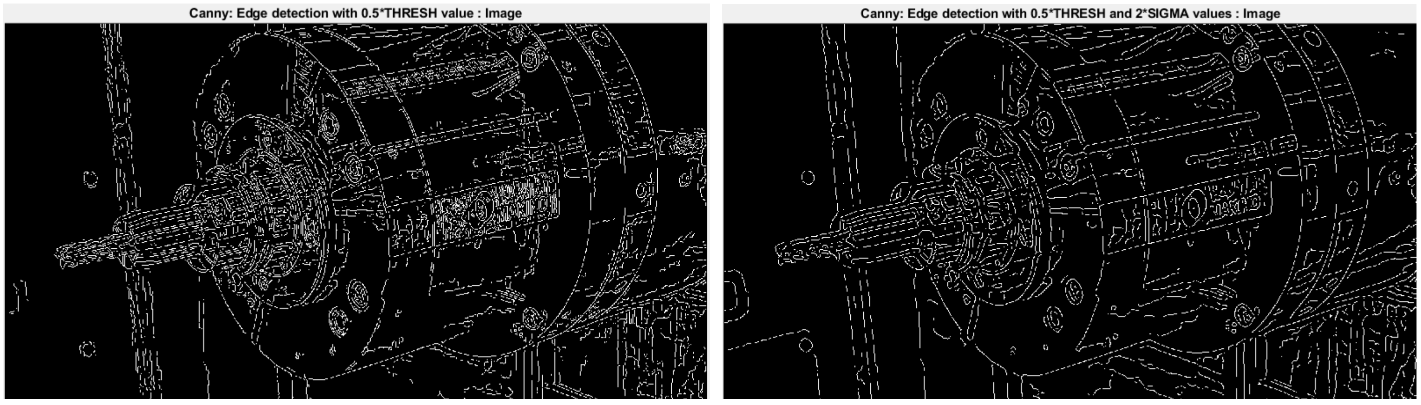
Part 2. Canny edge detection

2.2:





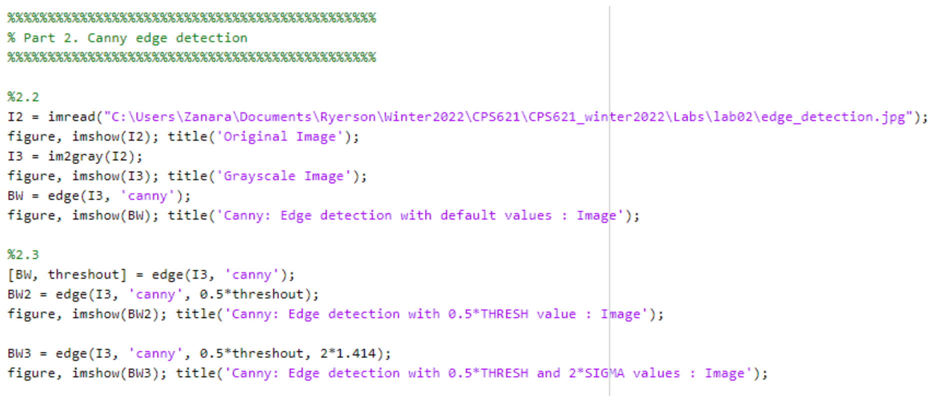
2.3:


Canny: Edge detection with 0.5*THRESH value : Image
Canny: Edge detection with 0.5*THRESH and 2*SIGMA values : Image

2.4 Report:

I have performed the edge detection on the grayscale image using Matlab function with the default parameters which can be seen in the figure above in 2.2 sections. As you can see it detected all the edges in the image with a bit messy structures where lines are not clearly separated. By changing the values of threshold, it removed the some of the lines previously detected. It can be seen above in the image with the threshold value of 0.5*threshold. Then again providing the function with Sigma value times 2 and got a bit better edge detection as in the figure above it provided more details on the edges and removed the unclear details. I believe that by providing different values to the function could improve the edge detection and enhance the image further with better results.

MatLab Code:



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2. Canny edge detection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%2.2
I2 = imread("C:\Users\Zanara\Documents\Ryerson\Winter2022\CPS621\CPS621_winter2022\Labs\lab02\edge_detection.jpg");
figure, imshow(I2); title('Original Image');
I3 = im2gray(I2);
figure, imshow(I3); title('Grayscale Image');
BW = edge(I3, 'canny');
figure, imshow(BW); title('Canny: Edge detection with default values : Image');

%2.3
[BW, threshout] = edge(I3, 'canny');
BW2 = edge(I3, 'canny', 0.5*threshout);
figure, imshow(BW2); title('Canny: Edge detection with 0.5*THRESH value : Image');

BW3 = edge(I3, 'canny', 0.5*threshout, 2*1.414);
figure, imshow(BW3); title('Canny: Edge detection with 0.5*THRESH and 2*SIGMA values : Image');
```