School of Electronic Engineering - Dublin City University

Modules ee425/ee453 – Coursework Assignment

DCU School of Electronic Engineering
Assignment Submission

Student Name(s):	**Radwan Duadu**
Student Number(s):	**14342606**
Programme:	**B.Eng. in Electronic & Computer Engineering**
Project Title:	**Image processing Assignment**
Module code:	**EE425/EE453**
EMAIL	**Radwan.duadu2@mail.dcu.ie**
Lecturer:	**Paul**
Project Due Date:	

Signed:_*Radwan duadu*____________

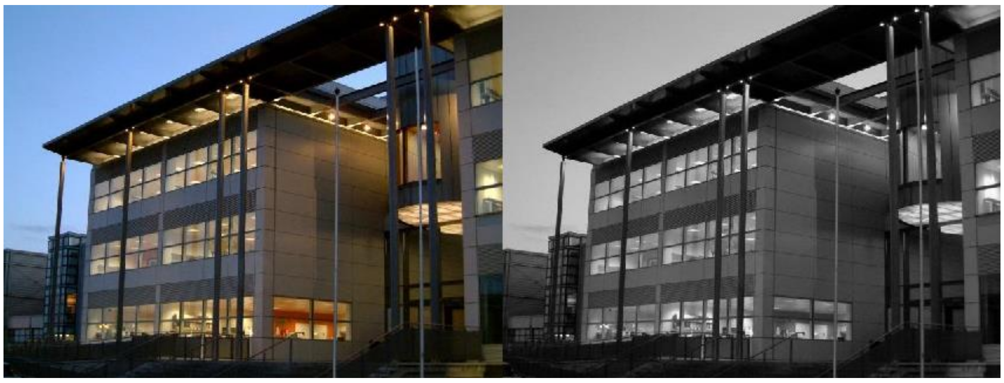
Part 1: Edge Detection

a) Develop a Matlab programme to automatically load the colour image illustrated in Figure 1 and convert it to a uint8 greyscale representation.

Pseudo-code:

```
1. First load input image.
2. Convert to colour image to grey.
3. Display input and grey image
```

Result:

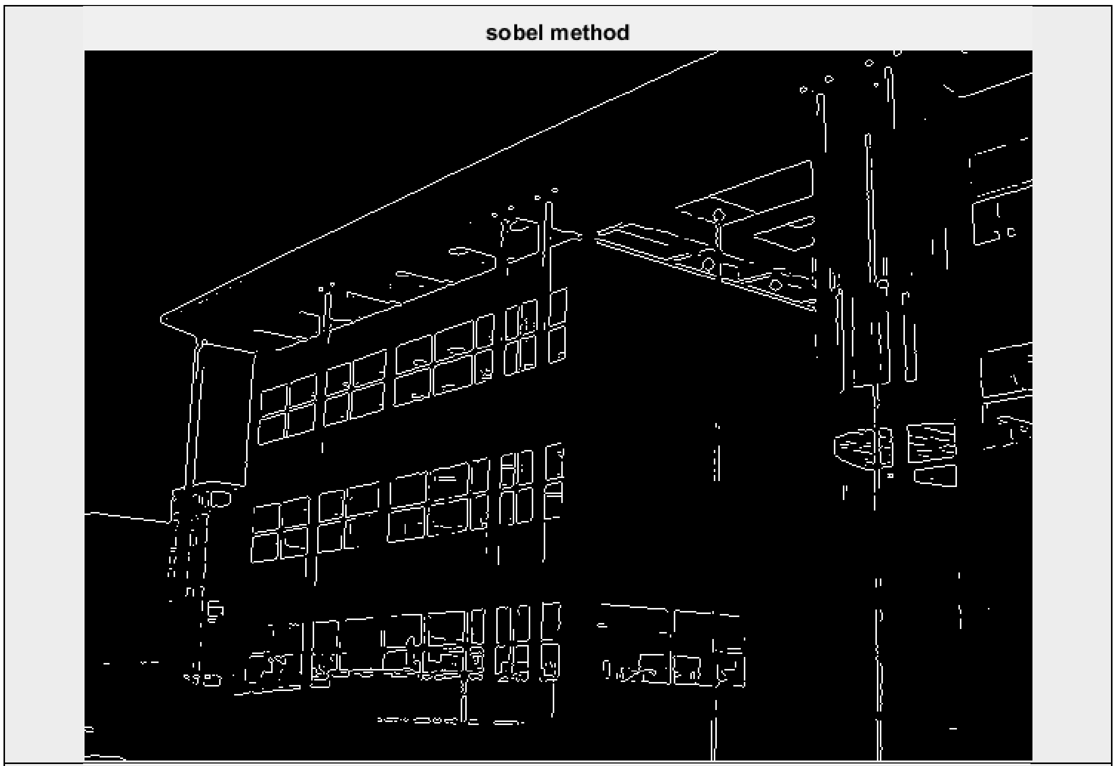


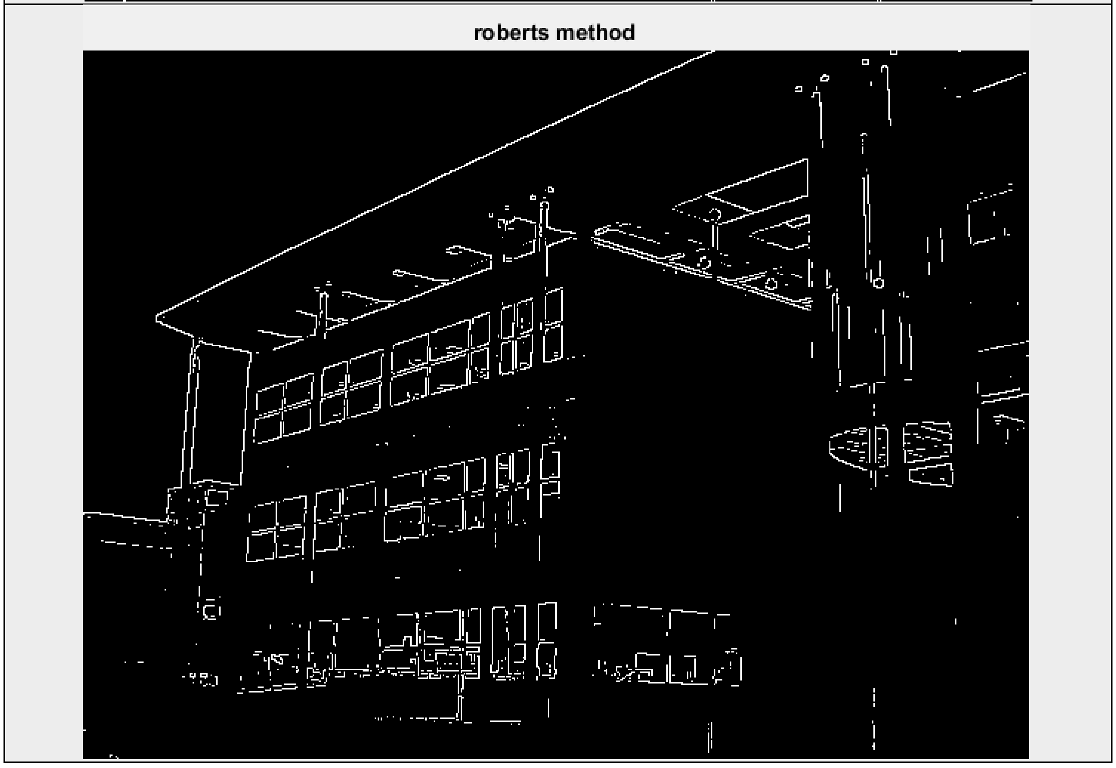
b) Using the grey scale image found in (a) apply both Roberts and Sobel edge detectors. Comment on the differences between the Sobel and Roberts edge detected images. Is this what you expected?

Pseudo-code:

```
1. Using Matlab provided edge function to convert grey scale image to Sobel
   and Robert edge detectors.
2. Display Sobel and Roberts edge detection images.
```

Result:





Comment:

As can be seen above the Sobel edge detector contained a clear representation of the edges, it contains less noise, better at curve detection and has stronger lines.

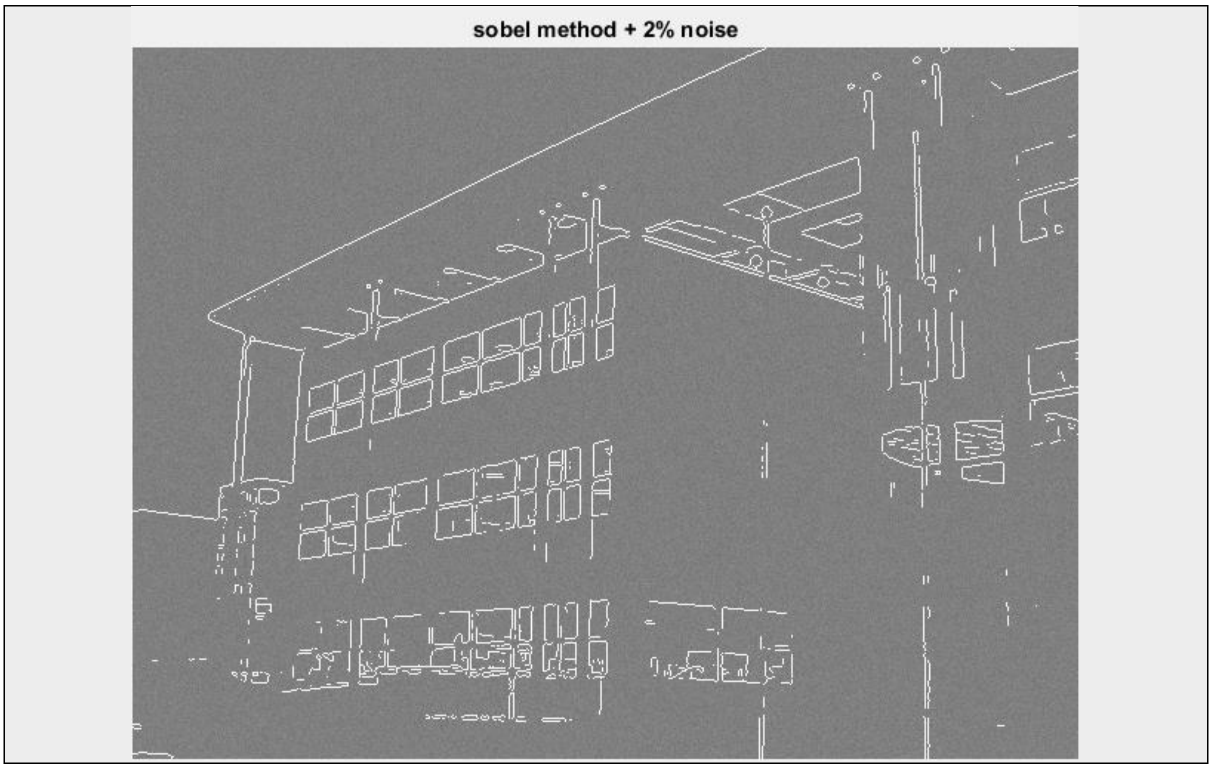
c) Experiment with different levels of Gaussian noise. Comment on the effect Gaussian noise has on each edge detector. Which edge detector is more robust to this type of noise?

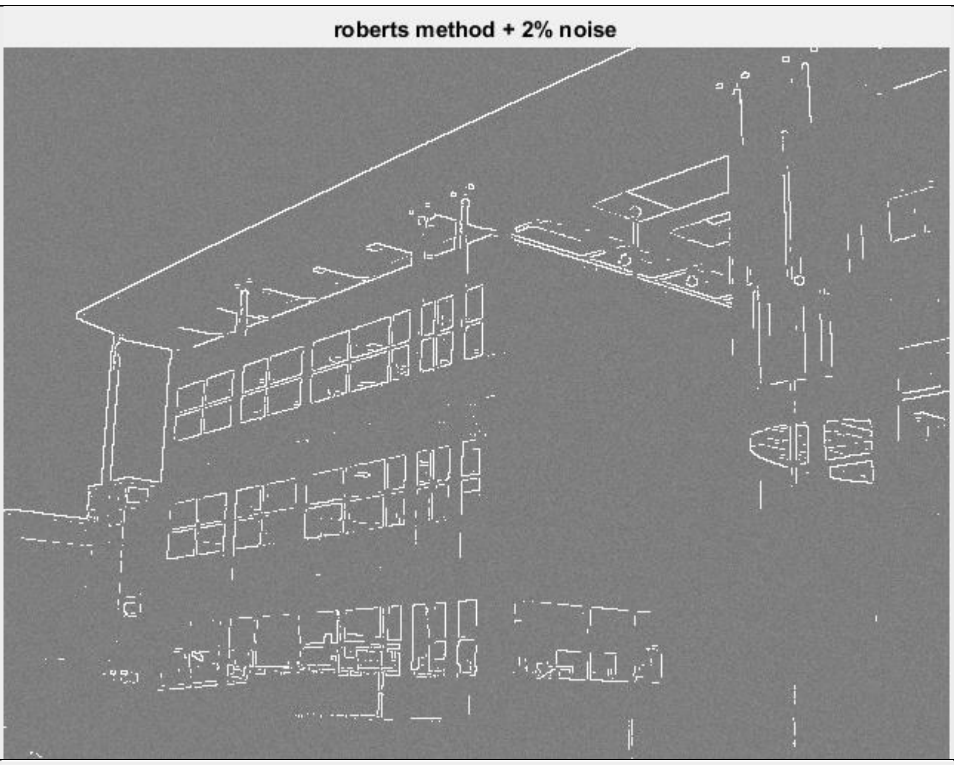
Pseudo-code:

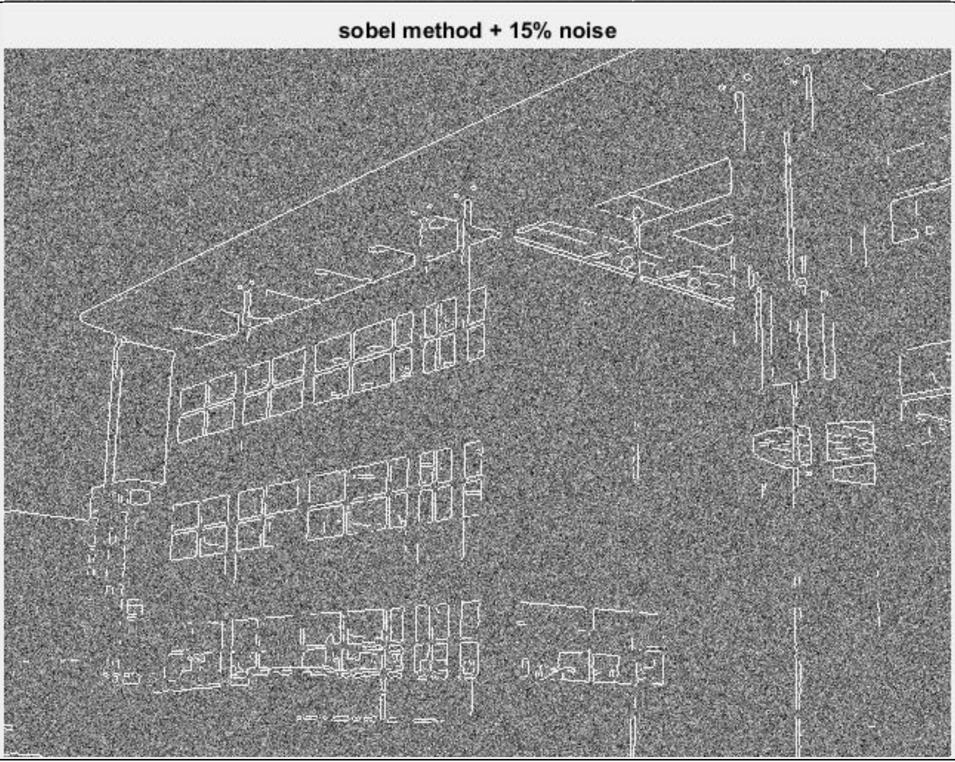
1. To experiment with the noise 4 tests were taken from 1% to 15%.
2. These experiments were done using the vsg Gaussian noise function.
3. Results are then compared by eye.

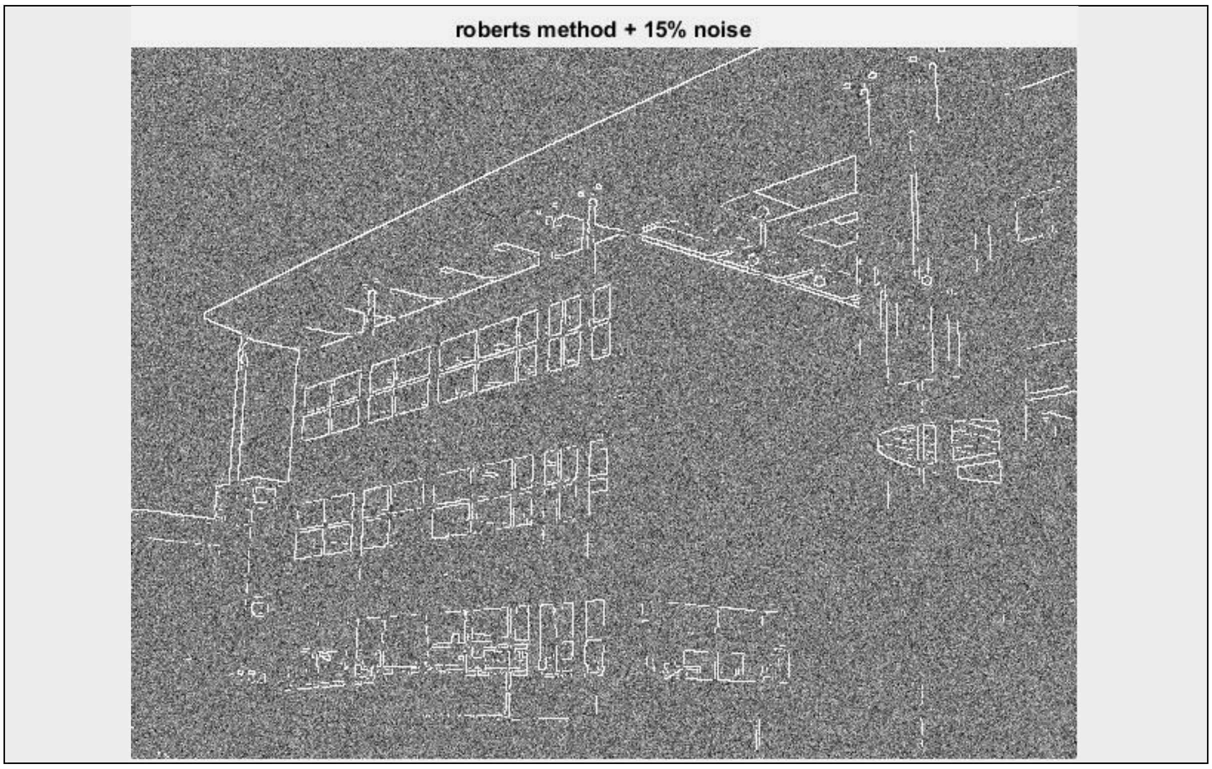
Result:

4 tests were done from 2% to 15% to compare the two edge detectors below is an example of two of the tests completed. The matlab code for the problem can be seen below which tests the two other tests seen.




roberts method + 2% noise


sobel method + 15% noise



As can be seen above at 15% noise the Sobel edge detector resulted in more accurate representation of all the edges in the original image even after the addition of noise.

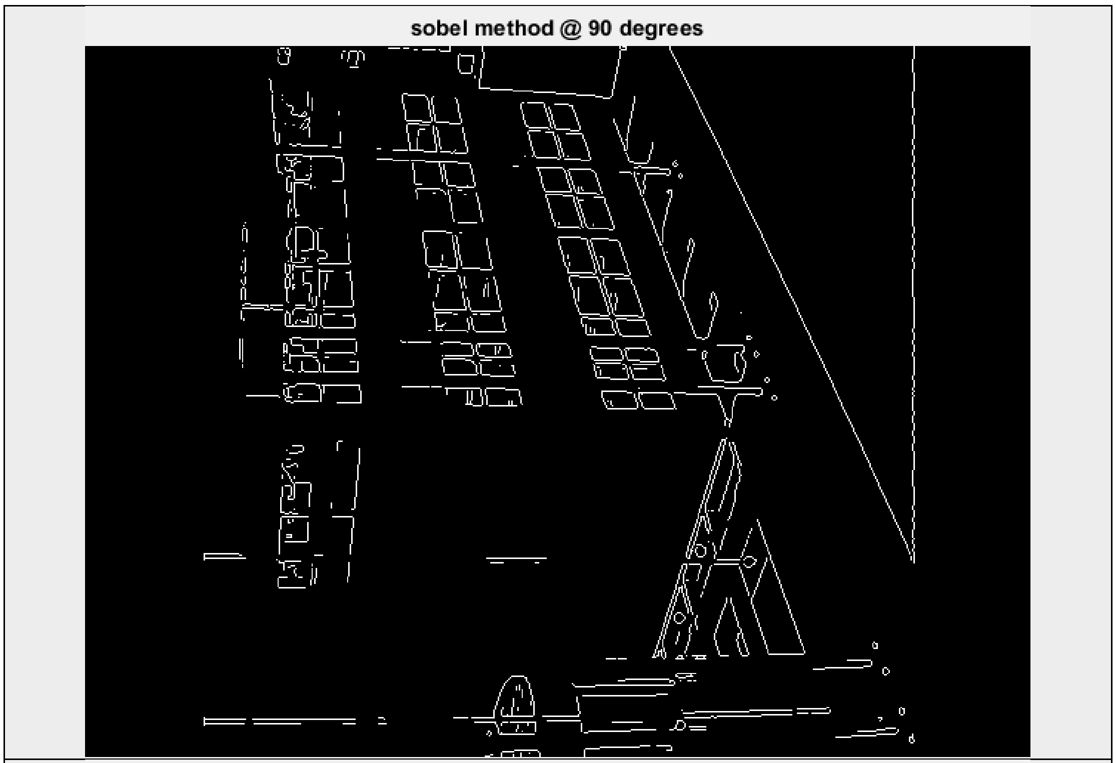
d) Experiment with different angles of rotation. Comment on the effect rotation has on each edge detector. Which edge detector is more robust to rotation?

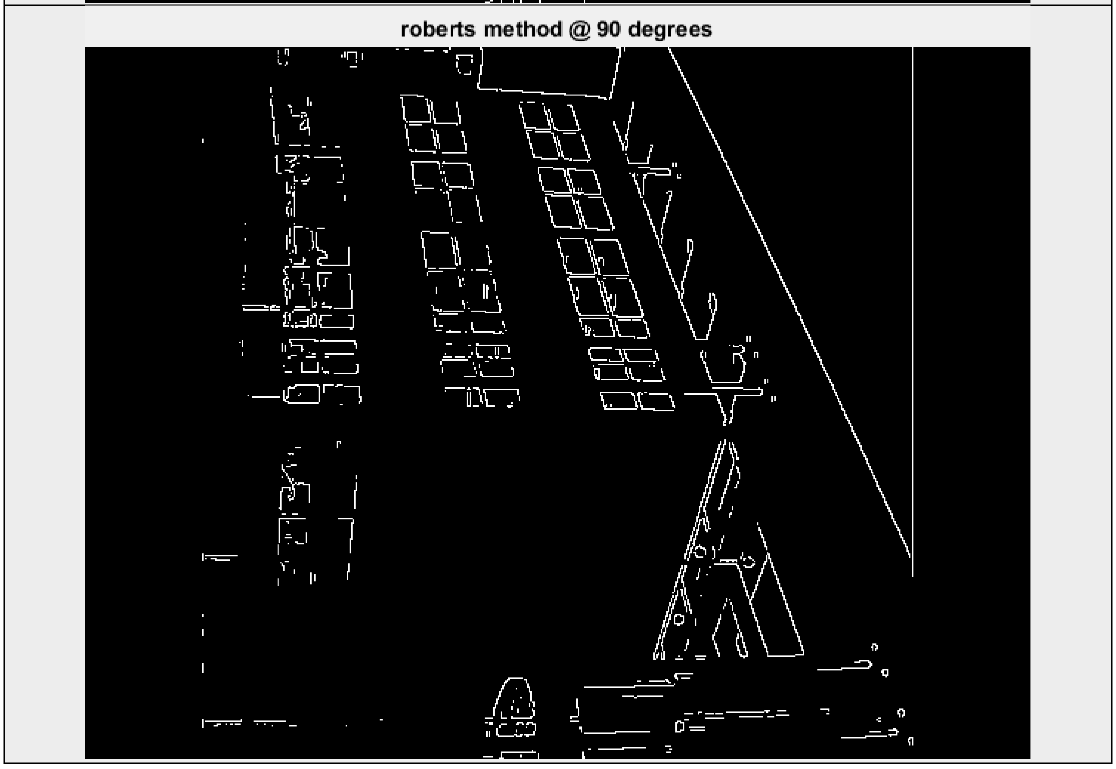
Pseudo-code:

```
1. Use vsg rotation function to rotate grey scale image
2. Use edge detection function on the rotated image
3. Compare the two rotated images.
```

Result:

To test which edge detector handles rotated images 5 tests were completed at different rotation and from these the result of one of the tests can be seen below.





Comment:

As can be seen above from the example provided the Sobel edge detector, had more information and a more accurate resolution of the original image.
Many other examples are provided in the matlab code will confirm that the Sobel edge detector handles edge detection better after rotation.

Matlab-code:

```matlab
%% EE425/EE453 Assignment Part 1
%
% Edge_Detection.m
%
%% automatically load a colour image & then convert to grey scale

addpath('C:\VSG_IPA_toolbox');

% clear workspace and free up system memory.
clc;                        % Clear command window.
clear all;                  % Clear all variables and functions from
memory

% Load an image
img1 = imread('stokes_building.jpg') ;

% convert to greyscale
img2=rgb2gray(img1);

% display the results
figure('Name','input images','NumberTitle','off'),
subplot(1,2,1); imshow(img1), title('colour image'),
subplot(1,2,2); imshow(img2), title('greyscale image');
%% Edge detection using robert & sobel methods

% finding edge's using robert's method
roberts = edge(img2,'Roberts');
% finding edge's using sobel's method
sobel = edge(img2,'Sobel');

%display results vs grey scale image
h=figure,imshow(roberts), title('roberts method'),
h=figure,imshow(sobel), title('sobel method');
%% calc images with noise

% noise @ 15
roberts15 = vsg('GaussianNoise',roberts,0.15);
sobels15 = vsg('GaussianNoise',sobel,0.15);



% noise @ 10
roberts10 = vsg('GaussianNoise',roberts,0.1);
sobels10 = vsg('GaussianNoise',sobel,0.1);

% noise @ 5
roberts5 = vsg('GaussianNoise',roberts,0.05);
sobels5 = vsg('GaussianNoise',sobel,0.05);

% noise @ 2
roberts2 = vsg('GaussianNoise',roberts,0.02);
sobels2 = vsg('GaussianNoise',sobel,0.02);

%% Guass noise @2
h=figure; imshow(roberts2), title('roberts method + 2% noise'),
h=figure; imshow(sobels2), title('sobel method + 2% noise');

%% Guass noise @5
h=figure; imshow(roberts5), title('roberts method + 5% noise'),
h=figure; imshow(sobels5), title('sobel method + 5% noise');

%% Guass noise @10
h=figure; imshow(roberts10), title('roberts method + 10% noise'),
h=figure; imshow(sobels10), title('sobel method + 10% noise');

%% Guass noise @15
h=figure; imshow(roberts15), title('roberts method + 15% noise'),
h=figure; imshow(sobels15), title('sobel method + 15% noise');
%% rotating the images

 grey30 = vsg('RotateImg',img2,30);
 grey90 = vsg('RotateImg',img2,90);
 grey180 = vsg('RotateImg',img2,180);
 grey210 = vsg('RotateImg',img2,210);
 grey330 = vsg('RotateImg',img2,330);

 robertsA30  = edge(grey30,'Roberts');
 robertsA90  = edge(grey90,'Roberts');
 robertsA180 = edge(grey180,'Roberts');
 robertsA210 = edge(grey210,'Roberts');
 robertsA330 = edge(grey330,'Roberts');

 sobelsA30  = edge(grey30,'Sobel');
 sobelsA90  = edge(grey90,'Sobel');
 sobelsA180 = edge(grey180,'Sobel');
 sobelsA210 = edge(grey210,'Sobel');
 sobelsA330 = edge(grey330,'Sobel');

 %% angle @30
h=figure; imshow(roberts), title('roberts image'),
h=figure; imshow(robertsA30), title('roberts method + 30 degrees'),
h=figure; imshow(sobel), title('sobel image'),
h=figure; imshow(sobelsA30), title('sobel method + 30 degrees');
 %% angle @90
h=figure; imshow(robertsA90), title('roberts method @ 90 degrees'),
h=figure; imshow(sobelsA90), title('sobel method @ 90 degrees');
%% angle @ 210
h=figure; imshow(robertsA210), title('roberts method @ 210 degrees'),
h=figure; imshow(sobelsA210), title('sobel method @ 210 degrees');
%% angle @330
h=figure; imshow(robertsA330), title('roberts method @ 330 degrees'),
h=figure; imshow(sobelsA330), title('sobel method @ 330 degrees');
```

Part 2: Binary Convolution based Template Matching

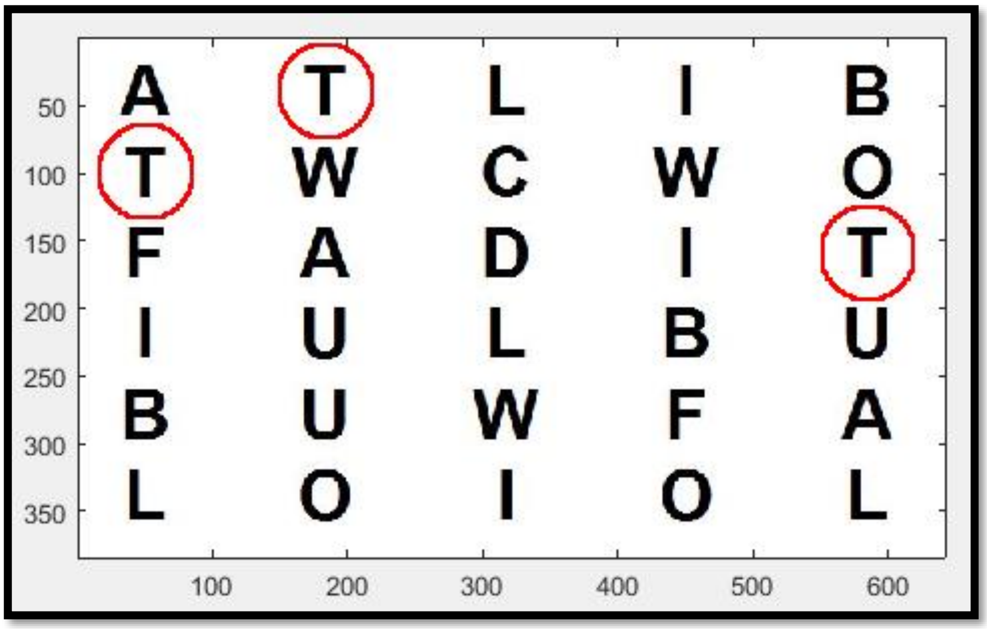
a) Using the source image in Figure 2 (letters.jpg) and the letter template image in Figures 4 (template_T.jpg) develop a fully automated10, data driven binary based convolutional approach to detecting and highlighting all the instances of the template letters “T” in the source image (Figure 2). See Figure 5 for a sample output for the template letter “T”.

Pseudo-code:

```
1. Load input images letters.jpg & template_T.jpg
2. Threshold and invert images so they are in binary form.
3. Using convolution function in the vsg toolbox preform a convolution between
   the two inverted images.
4. A high output indicates where the two images have a very high similarity.
5. Threshold is applied to isolate regions with the highest similarity between the
   two images.
6. Image is converted to binary.
7. Find function is applied to find coordinates of the regions with highest
   similarities.
8. For loop is then used to surround the regions with a circle.
9. Result is then displayed.
```

Result:

Result of the code can be seen below where the letter T was accurately found and surrounded with a circle.



b) What happens if you apply the template to a rotated version of the input image?

Expectation for this implementation is that when the input image is rotated the algorithm will still find points of high intensity after the convolution, but they will be incorrect as no perfect match is found and the second-best match will be selected instead for example the letter L.

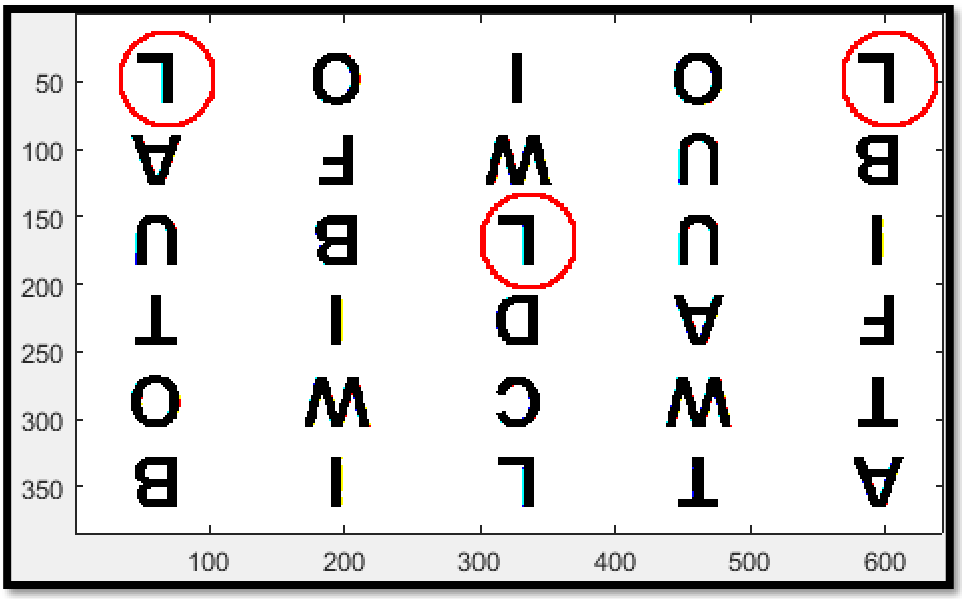
Pseudo-code:

```
1. Rotate letter.jpg after loading in the image.
```

2. Threshold and invert same as before.
3. Convolve image and display regions of highest threshold.

Result:

When image is rotated 180-degrees result is that the regions that represent the highest intensity after convolution is where the letter L is located as when it is flipped upside down it has a very close similarity to the letter T.



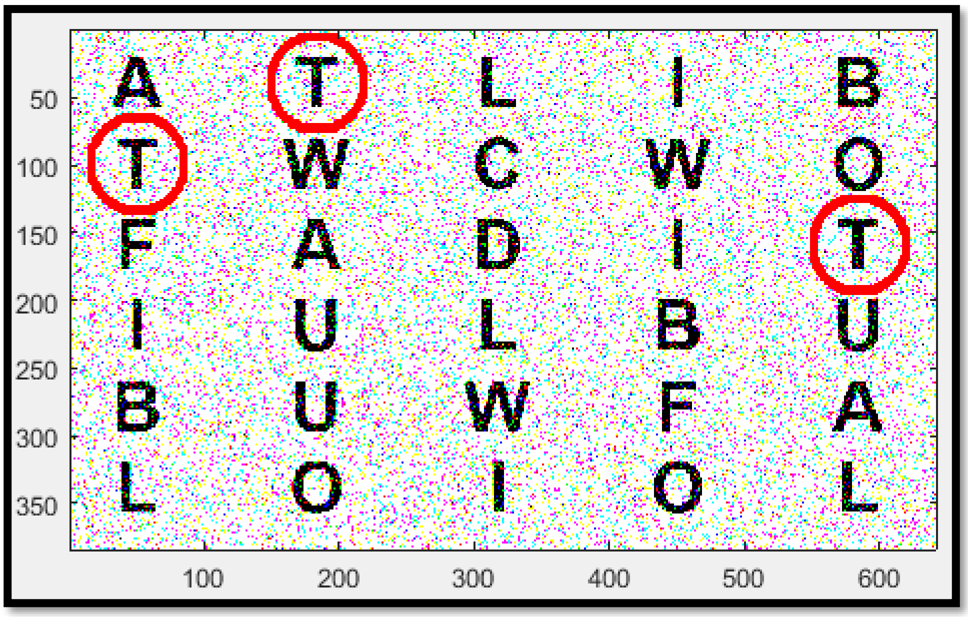
c) Experiment with different levels of Impulse (salt and pepper) noise. Comment on the effect Impulse noise has on performance of the convolutional template matching approach. At what level of Impulse noise does the programme fail?

Pseudo-code:

1. Apply noise to letter.jpg image.
2. Convolve image with template image.
3. Apply a threshold to convolved image, the threshold will consider the ratio of noise imputed into the system and will reduce the accuracy of the threshold by the percentage of noise imputed.
4. Result is then displayed.

Result:

Multiple tests were done at different noise values and the highest the noise was raised to was 15 percent, the tests were taken by increasing noise by 2 % each time until the program failed. 15% was the highest noise value in which the program operated correctly.

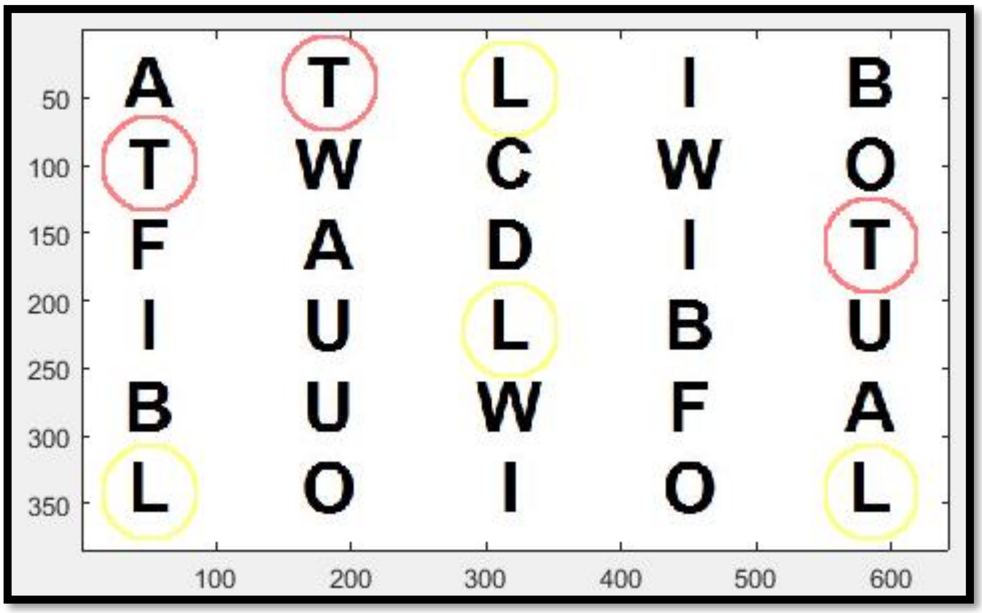


d) Develop a new enhanced programme that is capable of detecting and highlighting both the letters "T" and "L" (Figures 3 and 4) in the source image (Figure 2).

Pseudo-code:

1. Part a has already located the letter T
2. The code will replicate the process undertaken in locating the letter T with the exception that the template letter will be L.
3. After finding the result the result of image output a is added to output of part d.
4. Result is then displayed.

Result:



Matlab-code:

```
%% EE425/EE453 Assignment Part 2
%
% binary Convolution.m
%%
clear all
clc
close all

% read in the image
letters=openimage('letters.jpg');

% load T image
temp = openimage('template_T.jpg');
temp2=vsg('MidThresh',temp);
letter_t =vsg('NOT',temp2);

% Convert to white text on a black background
[out_img1]=vsg('MidThresh',letters);
[out_img2]=vsg('NOT',out_img1);

% Convolve the text image with the N-tuple
[out_img3]=vsg('Convolution',out_img2,letter_t);

% Threshold the result of the convolution. A high output from the
% convolution indicates that the N-tuple is very similar to the region in
% the input image at the location of the high response.
thresh=vsg('HighestGrey',out_img3);
[out_img4]=vsg('Threshold',out_img3,thresh);
detect_bw = im2bw(out_img4);

[x, y] = find(detect_bw == 1);
image_output = letters;

for i=1:length(x)
```

```
    image_output = vsg('DrawCircle',image_output,[y(i,1),x(i,1)],35,
[255,0,0],4);
end

% Display the images
h=figure; image(uint8(image_output)); colormap(gray); set(h,'Name','T
Found');axis image;

%% part b repeat for a rotated image
rotated90 = vsg('RotateImg',out_img2,180);

[out_rotated]=vsg('Convolution',rotated90,letter_t);

threshrotate=vsg('HighestGrey',out_rotated);
[out_part2]=vsg('Threshold',out_rotated,threshrotate);
detect_bwr = im2bw(out_part2);

[xr, yr] = find(detect_bwr == 1);
image_outputrotate = vsg('NOT',rotated90);

for i=1:length(xr)
    image_outputrotate =
vsg('DrawCircle',image_outputrotate,[yr(i,1),xr(i,1)],35, [255,0,0],4);
end
h=figure; image(uint8(image_outputrotate)); colormap(gray); set(h,'Name','T
Found with rotated image');axis image;

%% part c impulse noise added
value = 0.2;
noise = vsg('S+PNoise',out_img2,value);
[out_noise]=vsg('Convolution',noise,letter_t);

threshnoise=(vsg('HighestGrey',out_noise) - round(255*value));
[out_partc]=vsg('Threshold',out_noise,threshnoise);
detect_bwn = im2bw(out_partc);

[xn, yn] = find(detect_bwn == 1);
image_outputnoise = vsg('NOT',noise);

for i=1:length(xn)
    image_outputnoise =
vsg('DrawCircle',image_outputnoise,[yn(i,1),xn(i,1)],35, [255,0,0],4);
end
h=figure; image(uint8(image_outputnoise)); colormap(gray); set(h,'Name','T
Found with noise image');axis image;

%% repeat the same for L
temp_l = openimage('template_L.jpg');
temp_l2=vsg('MidThresh',temp_l);
letter_l =vsg('NOT',temp_l2);

% Convolve the text image with the N-tuple
[out_convl]=vsg('Convolution',out_img2,letter_l);

% the input image at the location of the high response.
thresh2=vsg('HighestGrey',out_convl);
[out_result]=vsg('Threshold',out_convl,thresh2);
detect_bw2 = im2bw(out_result);
```

```
[x2, y2] = find(detect_bw2 == 1);
image_output2 = letters;

for i=1:length(x2)
    image_output2 = vsg('DrawCircle',image_output2,[y2(i,1),x2(i,1)],35,
[255,255,0],4);
end

%h=figure; image(uint8(out_convl)); colormap(gray); set(h,'Name','Letter
Locations');axis image;
h=figure; image(uint8(image_output2)); colormap(gray); set(h,'Name','L
Found');axis image;

%% final result

result = vsg('Add',image_output,image_output2);
h=figure; image(uint8(result)); colormap(gray); set(h,'Name','Circle around L
& T');axis image;
```

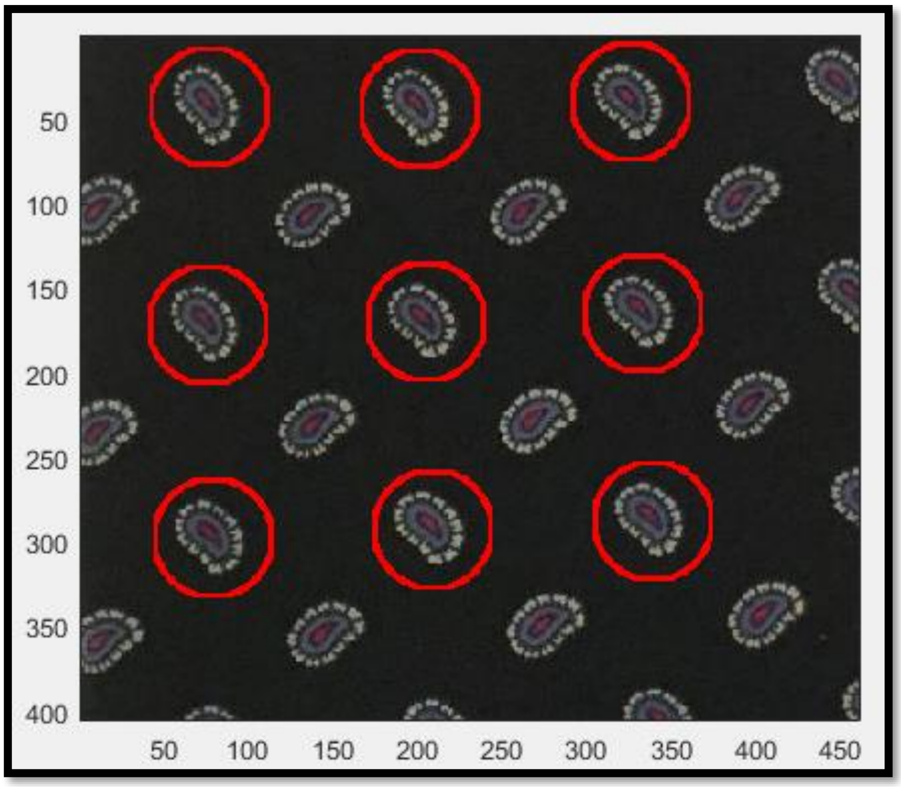
Part 3: Grey scale Convolution based Template Matching

a) Develop a fully automated, data driven programme that will find the grey scale template pattern Figure 7 (template_tie.jpg) in the grey scale source image Figure 6 (tie.jpg) using grey scale based convolution.

Pseudo-code:

1. Two images are loaded and converted to grey scale.
2. A threshold is applied to find the median threshold in each image and then the threshold is applied to each image.
3. The two results of the threshold of the grey scale images are convolved.
4. A threshold is applied to the convolved image.
5. The centroid is used to find the centre of each region that contains a high intensity.
6. The coordinates of the regions are found, and a circle is drawn on the inputted tie image where there the coordinates point to.
7. Result is then displayed.

Result:



b) Why are some patterns not found when using this single template?

Some patterns are not found because of two reasons.

1. They are at the border and are missing a section which results in a lower intensity in that region when the convolution of the two images is completed.
2. Some of the ties are in a different orientation which results in a lower intensity when the convolution is undertaken.

c) Comment on what happens at the image borders?

As mentioned above even though some of the tie images at the border may be at the right orientation they still are missing some sections which is cut off by the border which results in a lower value when the convolution and thresholding of the convoluted image is completed.

d) Experiment with different levels of Gaussian (with zero mean) noise. Comment on the effect Gaussian noise has on performance of the convolutional template matching approach. At what level of Gaussian noise does the programme fail?

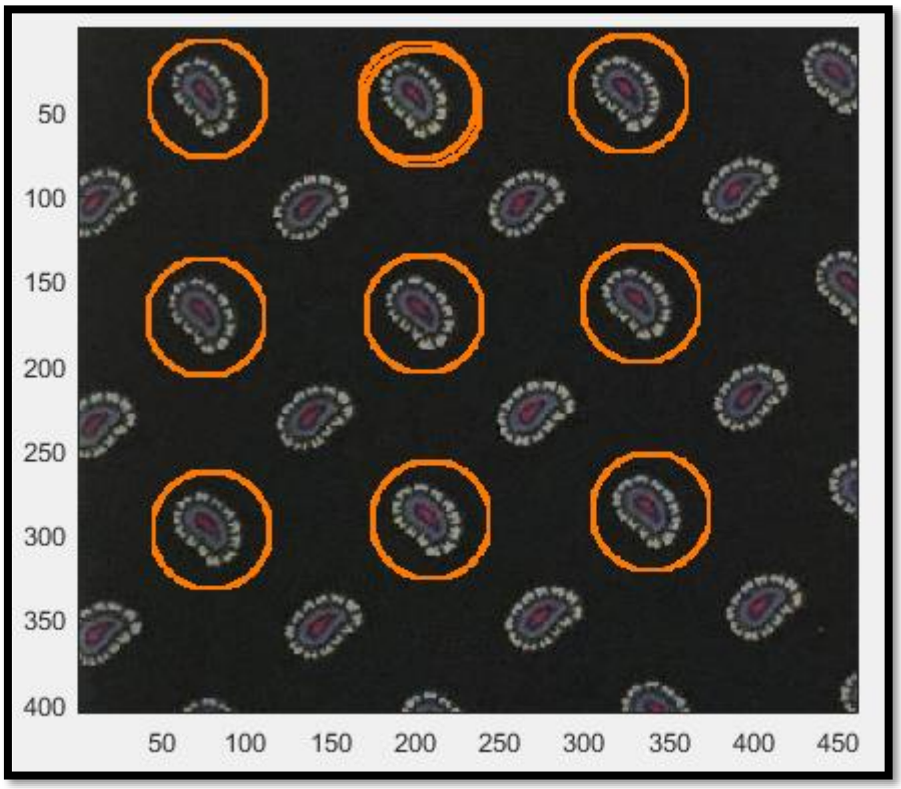
Pseudo-code:

1. To complete an accurate test gaussian noise was applied to the image from 0% up to 73%.
2. Because of how the program was implemented no noise cancelation or alteration to the program is needed.

3. Therefore, the program was tested manually by increasing the 10% until the program failed. And then increased in increments of one percent to find the exact breaking point of the program.

Result:

Below is an image of the program functioning correctly with noise at 73%.



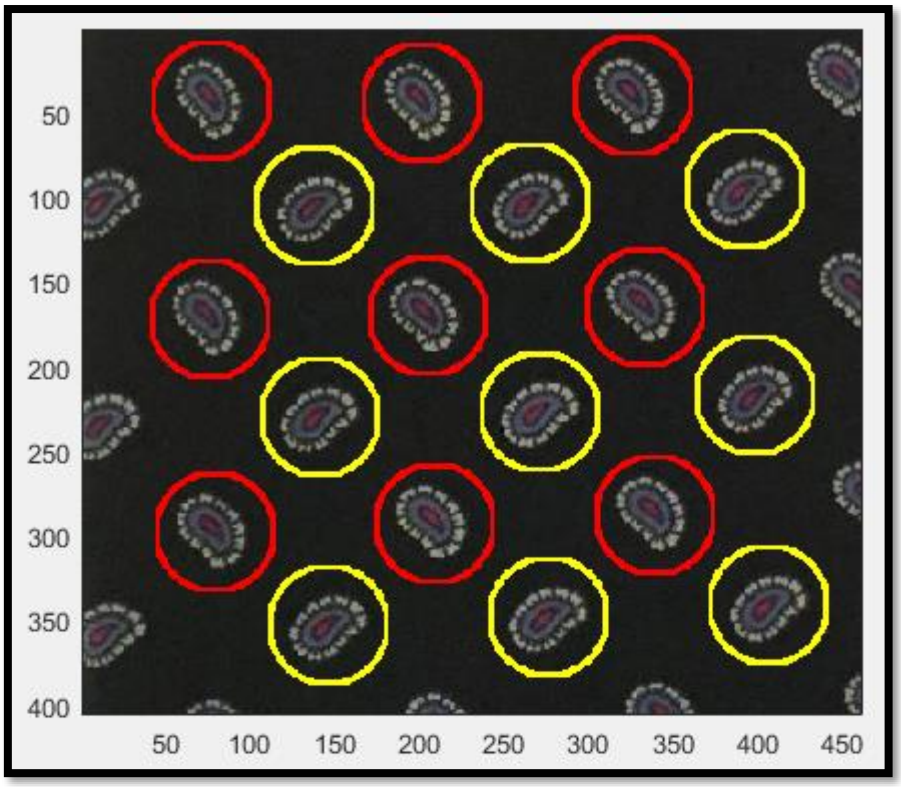
As can be seen the result is still accurate even at 73%

e) Develop a new enhanced programme to detect and highlight all the tie pattern elements which are not touching the edge of the image.

Pseudo-code:

1. To enhance the program all was done was to rotate the template tie image by 270-dgrees which results in the tie image matching the other images in the original tie.jpg image.
2. The same method is applied, and the new template and the tie image are convolved.
3. The result is displayed then the result of part A’s image is added to the result of image calculated above.
4. Result is then displayed and can be seen below.

Result:



Matlab-code:

```
        %% EE425/EE453 Assignment Part 3
%
% part3.m
%%
clear all
clc
close all

%read in the image
tie = openimage('tie.jpg');
%convert to grayscale
tiegrey=vsg('GreyScaler',tie);
[hi_g]=vsg('HighestGrey',tiegrey);
[lo_g]=vsg('LowestGrey',tiegrey);
threshtie=uint8((hi_g+lo_g)/2);
[out_img3]=vsg('Threshold',tiegrey,threshtie);

% read in the image
Template = openimage('template_tie.jpg');
% convert to greyscale
tempgrey=vsg('GreyScaler',Template);
[hi_g2]=vsg('HighestGrey',tempgrey);
[lo_g2]=vsg('LowestGrey',tempgrey);
threshtemp=uint8((hi_g2+lo_g2)/2);
[out_img4]=vsg('Threshold',tempgrey,threshtemp);



[out_img5]=vsg('Convolution',out_img3,out_img4);
```

```
% threshold for convolution of given tie temp
threshhi = vsg('HighestGrey',out_img5);
threshlo=vsg('LowestGrey',out_img5);
thresh=uint8((threshhi+threshlo)/2);


[out_img6]=vsg('Threshold',out_img5,thresh);
[out_img8]=vsg('Centroid',out_img6);
detect_bw = im2bw(out_img8);
[x, y] = find(detect_bw == 1);
image_output = tie;

for i=1:length(x)
    image_output =
vsg('DrawCircle',image_output,[y(i,1),x(i,1)],35, [255,0,0],4);
end
% Display the image
h=figure; image(uint8(image_output)); colormap(gray);
set(h,'Name','Grey scale convolution');axis image;

%% part 3 d Gaussian noise
Template2 = vsg('GaussianNoise',Template,73);
%find threshold of template
tempgrey2=vsg('GreyScaler',Template2);
[hi_g2]=vsg('HighestGrey',tempgrey2);
[lo_g2]=vsg('LowestGrey',tempgrey2);
threshtemp2=uint8((hi_g2+lo_g2)/2);
[out_img3d]=vsg('Threshold',tempgrey2,threshtemp2);

%convolve template and tie images using thrshhold images
[out_img5d]=vsg('Convolution',out_img3,out_img3d);
%threshold result of convolution
threshhi3d = vsg('HighestGrey',out_img5d);
threshlo3d=vsg('LowestGrey',out_img5d);
thresh=uint8((threshhi3d+threshlo3d)/2);
[out_img6d]=vsg('Threshold',out_img5d,thresh);

[out_img8d]=vsg('Centroid',out_img6d);
detect_bwd = im2bw(out_img8d);
[xd, yd] = find(detect_bwd == 1);
image_outputd = tie;

for i=1:length(xd)
    image_outputd =
vsg('DrawCircle',image_outputd,[yd(i,1),xd(i,1)],35,
[255,120,0],4);
end
% Display the image
h=figure; image(uint8(image_outputd)); colormap(gray);
set(h,'Name','Convolution with noise image template');axis image;
h=figure; image(uint8(Template2)); colormap(gray);
set(h,'Name','Noise image');axis image;
h=figure; image(uint8(out_img3d)); colormap(gray);
set(h,'Name','Threshold of Noise image');axis image;



%% part 3 e
tietemp270 = vsg('RotateImg',out_img4,270);
```

```
[out_img10]=vsg('Convolution',out_img3,tietemp270);

threshhi2 = vsg('HighestGrey',out_img10);
threshlo2=vsg('LowestGrey',out_img10);
thresh2=uint8((threshhi2+threshlo2)/2);

[out_img11]=vsg('Threshold',out_img10,thresh2);
[out_img12]=vsg('Centroid',out_img11);
detect_bw2 = im2bw(out_img12);

[x2, y2] = find(detect_bw2 == 1);
image_output2 = image_output;

for i=1:length(x2)
    image_output2 =
vsg('DrawCircle',image_output2,[y2(i,1),x2(i,1)],35,
[255,255,0],4);
end
% Display the image
h=figure; image(uint8(image_output2)); colormap(gray);
set(h,'Name','Enhanced programme');axis image;
```